

Papers on graph hyperbolicity

APPENDIX B

Applying clique-decomposition for computing graph hyperbolicity

Applying clique-decomposition for computing Gromov hyperbolicity

Nathann Cohen¹, David Coudert², Guillaume Ducoffe², and Aurélien Lancin³

¹LRI, Univ Paris Sud, Université Paris-Saclay, 91405 Orsay, France

²Université Côte d'Azur, Inria, CNRS, I3S, France

³LAMIA Laboratory, University of the French West Indies and Guiana, France

Abstract

Given a graph, its *hyperbolicity* is a measure of how close its distance distribution is to the one of a tree. This parameter has gained recent attention in the analysis of some graph algorithms and the classification of complex networks. We study on practical improvements for the computation of hyperbolicity in large graphs. Precisely, we investigate on relations between the hyperbolicity of a graph G and the hyperbolicity of its *atoms*, that are the subgraphs output by the clique-decomposition invented by Tarjan [44, 56]. We prove that the maximum hyperbolicity taken over the atoms is at most one unit off from the hyperbolicity of G and the bound is sharp. We also give an algorithm to slightly modify the atoms, which is at no extra cost than computing the clique-decomposition, and so that the maximum hyperbolicity taken over the resulting graphs is *exactly* the hyperbolicity of G . An experimental evaluation of our method for computing the hyperbolicity of a given graph from its atoms is provided for large collaboration networks. Finally, on a more theoretical side we deduce from our results the first *linear-time* algorithm for computing the hyperbolicity of an outerplanar graph.

Keywords: Gromov hyperbolicity; graph algorithms; clique-decomposition; outerplanar graphs.

1 Introduction

In this paper we aim at improving the computation of hyperbolicity in large graphs, whose size ranges from thousands to tens of thousands of nodes. To this end, we will establish new relations between hyperbolicity and some graph decomposition. Roughly, the hyperbolicity of a metric space is an estimate of how close it is to a metric tree (formal definitions are postponed to the technical sections of this paper). This parameter was first introduced by Gromov in the context of automatic groups [35]. Later on, it was extended to more general metric spaces including graphs equipped with their shortest-path metric. Graph hyperbolicity is now part of the parameters in use to classify complex networks [1, 2, 40]. Furthermore, the study of graphs with bounded hyperbolicity has found applications in the design and the analysis of approximation algorithms [19, 24] and geometric routing schemes [11], as well as in network security [38] and bioinformatics [18, 30], to name a few. As a result, hyperbolicity and its relations to other metric graph parameters has received growing attention over the last decades. The reader may refer to [1, 29] for a recent survey.

Computing the hyperbolicity is useful in some of the above applications. For instance, it allows to compute an embedding of the graph into the Hyperbolic plane with quasi-optimal distortion

of the distances in linear-time [58]. The latter is a prerequisite to many algorithms on negatively curved spaces [42]. However, the computational cost of hyperbolicity has only recently received a bit more attention. So far, the best-known algorithm to compute the hyperbolicity [31], though it runs in polynomial-time, is impractical for large-scale graphs such as the graph of the Autonomous Systems of the Internet, road maps, etc. This comes from its challenging implementation, relying on fast square matrix multiplications, and its time complexity which is supercubic. On a more positive side, there have been recent attempts that are much more efficient than the state-of-the-art algorithm in practice, with running time dominated by the computation of the all-pairs-shortest-paths [13, 21]. But on the negative side, it is unlikely that graph hyperbolicity can be computed in subquadratic-time, even for sparse graphs [14, 22, 31]. This motivates us to study which structural properties can help to speed-up the computation of hyperbolicity in large graphs.

Our approach. We build on *graphs decompositions* in order to gain more insights on the internal structure of graphs with bounded hyperbolicity. More precisely, we are given a decomposition of a graph in some of its subgraphs and we aim at computing its hyperbolicity from the hyperbolicity of the subgraphs. This way, for computing the hyperbolicity of a graph, we can preprocess it and replace it with the subgraphs of the decomposition, thereby decreasing the size of the inputs. Although such techniques are unlikely to improve the calculation of hyperbolicity in general (for they may be graphs that are “prime” w.r.t. the decomposition), they are expected to work well on some graph classes of interest, including some classes of real-life graphs. A first step toward this direction was made by Soto in his PHD thesis [51]. He proved that the hyperbolicity of a graph is the maximum hyperbolicity taken over the subgraphs from the modular decomposition [33] or the split-decomposition [23]. Examples of complex networks whose underlying graph has a nontrivial modular decomposition are the protein-protein interaction networks [32].

We here address a similar question for the subgraphs from the *clique-decomposition*, also known as *atoms*. The clique-decomposition was introduced by Tarjan in [56] then made unique by Leimer in [44]. Roughly, it consists in disconnecting the graph using clique-separators, and it is easy to implement. Note that there are studies supporting the existence of clique-separators in real-life graphs, such as the underlying graphs of social and biological networks [1, 7, 26, 39], that makes our approach practical for large graphs. Furthermore, clique-decomposition has been proved useful to preprocess the graphs in the computation of many optimization problems [56] – including the computation of treelength [28], related to hyperbolicity. Therefore, at first glance, it should not come as a surprise if clique-decomposition can be applied to preprocess the graphs in the computation of hyperbolicity. This being said, hyperbolicity is less robust than other metric invariants to graph modifications (for instance, it may increase after an edge-contraction [17]), so, a careful analysis is needed to prove whether it is the case.

Main contributions. In fact, our first result is that, somewhat counter-intuitively, the hyperbolicity $\delta(G)$ of a graph G *cannot* be deduced directly from the hyperbolicity of its atoms (Section 3). We will prove nonetheless that it can be approximated with additive constant 1 by taking the maximum hyperbolicity $\delta^*(G)$ over the atoms (Section 4). This result requires an in-depth analysis of clique-decomposition in order to be proved. Additionally, we characterize when it is the case that $\delta(G) > \delta^*(G)$.

Based on this characterization, we will show in Section 5 how each atom can be “modified” (augmented with few simplicial vertices) in order to compute exactly the hyperbolicity, and provide

a complexity analysis of the procedure. Experiments in Section 7 show the benefit of our method in terms of size of the graph, when applied to some real collaboration networks.

Finally, we will apply clique-decomposition for improving the best-known complexity to compute hyperbolicity in the class of outerplanar graphs. We will detail in Section 6 the first linear-time algorithm for computing the hyperbolicity of these graphs. We find the latter result all the more interesting that under classical complexity assumptions, the hyperbolicity of sparse graphs cannot be computed in subquadratic-time [14].

Definitions and notations used in this paper will be introduced in Section 2.

2 Definitions and notations

We use the graph terminology of [12, 27]. All graphs considered in this paper are finite, unweighted and simple. Given $G = (V, E)$, let $n = |V|$, $m = |E|$. For every subset of vertices $S \subseteq V$, the open neighborhood of S is the set of all vertices in $V \setminus S$ with at least one neighbor in S . We denote it by $N_G(S)$, or by $N(S)$ when G is clear from the context. The close neighborhood of S is the set $S \cup N(S)$, denoted by $N[S]$.

Given two vertices u and v , a uv -path of length $l \geq 0$ is a sequence of vertices ($u = v_0 v_1 \dots v_l = v$), such that $\{v_i, v_{i+1}\}$ is an edge for every i . In particular, a graph G is *connected* if there exists a uv -path for all pair $u, v \in V$, and in such a case the *distance* $d_G(u, v)$ is defined as the minimum length of a uv -path in G . Note that it yields a discrete metric space (V, d_G) , also known as the shortest-path metric space of G . We will write d instead of d_G whenever G is clear from the context, and we denote by $d(u, X) = \min_{x \in X} d(u, x)$ the distance between a vertex u and a set X of vertices.

Our proofs use the notions of subgraphs, *induced* subgraphs, as well as *isometric subgraphs*, the latter denoting a subgraph H of a graph G such that $d_H(u, v) = d_G(u, v)$ for any two vertices $u, v \in H$.

2.1 Gromov hyperbolicity

The space (V, d_G) is a tree metric if there exists a distance-preserving mapping from V to the nodes of an edge-weighted tree. In this case, the graph G is called 0-hyperbolic. Several characterizations exist for 0-hyperbolic graphs. Informally, a graph is called δ -hyperbolic if it satisfies one of these characterizations up to a defect at most δ . Different characterizations will lead to different values for δ , but they may differ only by a small constant-factor [6, 25, 35]. We here consider the following 4-point definition for hyperbolicity.

Definition 1 (4-points Condition, [35]). *Let G be a connected graph. For every 4-tuple u, x, v, y of vertices of G , we define $\delta(u, v, x, y)$ as half of the difference between the two largest sums among*

$$S_1 = d(u, v) + d(x, y), S_2 = d(u, x) + d(v, y) \text{ and } S_3 = d(u, y) + d(v, x).$$

The hyperbolicity of G , denoted by $\delta(G)$, is equal to $\max_{u, x, v, y \in V(G)} \delta(u, v, x, y)$. Moreover, we say that G is δ -hyperbolic, for every $\delta \geq \delta(G)$.

It is straightforward, by the above definition, to compute graph hyperbolicity in $\theta(n^4)$ -time. In theory, it can be decreased to $\mathcal{O}(n^{3.69})$ by using clever (max, min) matrix product [31]. But in practice, the best-known algorithms still run in $\mathcal{O}(n^4)$ -time [13, 21]. Graphs with small hyperbolicity

can be recognized faster. In fact, 0-hyperbolic graphs coincide with *block graphs*, that are graphs whose all biconnected components are complete subgraphs [5, 37]. Hence it can be decided in linear $\mathcal{O}(n + m)$ -time whether a graph is 0-hyperbolic. The latter characterization of 0-hyperbolic graphs follows from a more general result saying that the hyperbolicity of a graph is the maximum hyperbolicity from its biconnected components (our work will give a new proof of this well-known result). More recently, it was proved that the recognition of $\frac{1}{2}$ -hyperbolic graphs is computationally equivalent to decide whether there is a chordless cycle of length 4 in a graph [22]. The latter problem can be solved in deterministic $\mathcal{O}(n^{3.26})$ -time [43] and in randomized $\mathcal{O}(n^{2.373})$ -time [57] by using fast matrix multiplication.

2.2 Clique-decomposition

Given $G = (V, E)$, we name *separator* a subset of vertices $X \subset V$ such that the removal of X disconnects the graph. We call X a *clique-separator* when the induced subgraph $G[X]$ is a complete graph. A graph is *prime* if it does not contain a clique-separator. Examples of prime graphs are complete graphs and cycles. Finally, the *clique-decomposition* of G is the collection of its maximal sets of vertices that induce prime subgraphs of G (we will call them *atoms*). See Figure 1 for an illustration. The decomposition is unique and it can be computed in $\mathcal{O}(nm)$ -time [44, 56]. We refer to [8] for a survey on clique-decomposition.

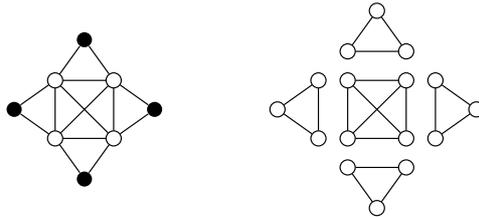


Figure 1: Clique-decomposition of a graph in five atoms. A 4-tuple with hyperbolicity 1 is drawn in bold.

Notations. Let us fix some notations for the proofs. Given $G = (V, E)$, let X be a separator of G . Let A, B denote two sets of vertices such that $A \cap B \subseteq X$ and $A \setminus X$, resp., $B \setminus X$, is nonempty. We will call X a $(A|B)$ -separator. Let us denote by $(a|b_1, b_2, b_3)$ a 4-tuple such that $a \in A$ and $b_1, b_2, b_3 \in B$. In the same way, let us denote by $(a_1, a_2|b_1, b_2)$ a 4-tuple such that $a_1, a_2 \in A$ and $b_1, b_2 \in B$. Note that we allow some vertices of the 4-tuple to be in X with this notation.

3 Hyperbolicity and clique-separators

It happens that every atom of the graph is δ -hyperbolic whereas it has hyperbolicity strictly greater than δ . As example, consider the chordal graph of Figure 1. It is 1-hyperbolic, with a 4-tuple of maximum hyperbolicity being drawn in black. However, its five atoms are complete graphs, hence they are 0-hyperbolic.

The purpose of the next two sections is to upper-bound the gap between $\delta(G)$ and $\delta^*(G)$ for every graph G , where $\delta^*(G)$ denotes the maximum hyperbolicity from the atoms of G . To this end, we analyze in this section the relationship between the hyperbolicity of a graph and a given

clique-separator, leading to the approximation with additive constant of Theorem 12. It begins with an observation about $(a_1, a_2|b_1, b_2)$ 4-tuples and the diameter $\text{diam}(X) = \max_{u,v \in X} d_G(u, v)$ of a $(A|B)$ -separator X .

3.1 Hyperbolicity of $(a_1, a_2|b_1, b_2)$ 4-tuples

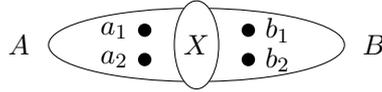


Figure 2: Illustration of a $(A|B)$ -separator.

Lemma 2. *Let X be a $(A|B)$ -separator of a connected graph G . For every $(a_1, a_2|b_1, b_2)$ 4-tuple, we have $\delta(a_1, a_2, b_1, b_2) \leq \text{diam}(X)$.*

Proof. By Definition 1, we have $\delta(a_1, a_2, b_1, b_2) = (L - M)/2$ where L and M are the two biggest sums among the following:

$$S_1 = d(a_1, a_2) + d(b_1, b_2), \quad S_2 = d(a_1, b_1) + d(a_2, b_2), \quad S_3 = d(a_1, b_2) + d(a_2, b_1).$$

Let us upper-bound L . By the triangular inequality we have that for every $u, v \in \{a_1, a_2, b_1, b_2\}$, $d(u, v) \leq d(u, X) + d(v, X) + \text{diam}(X)$. Thus, for every $i \in \{1, 2, 3\}$ we have (by applying twice the triangular inequality) that $S_i \leq d(a_1, X) + d(a_2, X) + d(b_1, X) + d(b_2, X) + 2 \cdot \text{diam}(X)$. In particular, $L \leq d(a_1, X) + d(a_2, X) + d(b_1, X) + d(b_2, X) + 2 \cdot \text{diam}(X)$.

Furthermore, since X is assumed to be a $(A|B)$ -separator, we have that for every $i, j \in \{1, 2\}$, all $a_i b_j$ -paths in G must intersect X , and so, $d(a_i, b_j) \geq d(a_i, X) + d(b_j, X)$. Hence, $d(a_1, X) + d(a_2, X) + d(b_1, X) + d(b_2, X) \leq d(a_1, b_1) + d(a_2, b_2) = S_2$, and in the same way $d(a_1, X) + d(a_2, X) + d(b_1, X) + d(b_2, X) \leq S_3$. Altogether it implies that $L \leq \min\{S_2, S_3\} + 2 \cdot \text{diam}(X)$. By noticing that $\min\{S_2, S_3\} \leq M$, one finally obtains that $L \leq M + 2 \cdot \text{diam}(X)$, and so, $\delta(a_1, a_2, b_1, b_2) \leq \text{diam}(X)$, as desired. \square

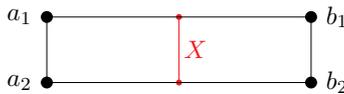


Figure 3: An $(a_1, a_2|b_1, b_2)$ 4-tuple with hyperbolicity 1.

Corollary 3. *Let X be a $(A|B)$ -clique-separator of a connected graph G . For every $(a_1, a_2|b_1, b_2)$ 4-tuple, we have $\delta(a_1, a_2, b_1, b_2) \leq 1$.*

The upper-bound of Corollary 3 is sharp, as shown with the grid of Figure 3. By taking larger grids, it can also be shown that the upper-bound of Lemma 2 is sharp.

3.2 Hyperbolicity of $(a|b_1, b_2, b_3)$ 4-tuples

In contrast to $(a_1, a_2|b_1, b_2)$ 4-tuples, the hyperbolicity of a $(a|b_1, b_2, b_3)$ 4-tuple can be arbitrarily large. We will relate $(a|b_1, b_2, b_3)$ 4-tuples with some 4-tuples of $B \cup X$ in order to upper-bound their hyperbolicity. Precisely, note that X being a clique, each vertex $a \in A$ is at distance at least $d(a, X)$ and at most $d(a, X) + 1$ from any vertex of X . We now show how this can be used with respect to the hyperbolicity.

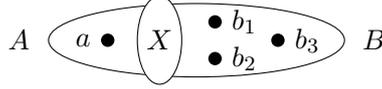


Figure 4: Illustration of a $(a|b_1, b_2, b_3)$ -separator.

Lemma 4. *Let X be a $(A|B)$ -clique-separator of a connected graph G , and let $a \in A$. We consider the graph G' obtained from G by adding a vertex a^* adjacent to $\{x \in X : d_G(a, x) = d_G(a, X)\}$. Then for every $b_1, b_2, b_3 \in B$ we have $\delta(a, b_1, b_2, b_3) = \delta(a^*, b_1, b_2, b_3)$.*

Proof. By construction G is an isometric subgraph of G' and so, $\forall u, v \in V(G)$, $d_{G'}(u, v) = d_G(u, v)$. In particular, the value $\delta(a, b_1, b_2, b_3)$ is not modified by the construction.

Let us relate $d(a^*, b)$ with $d(a, b)$ for every $b \in B$. Precisely, let us prove that $d(a, b) - d(a^*, b)$ is a constant (*i.e.*, not depending on b), that will prove by Definition 1 that $\delta(a, b_1, b_2, b_3) = \delta(a^*, b_1, b_2, b_3)$ for every $b_1, b_2, b_3 \in B$. In order to prove it, first observe that $\forall x \in X$, $d(a, x) \in \{d(a, X), d(a, X) + 1\}$ holds as X is a clique. Since a^* is adjacent to $\{x \in X : d(a, x) = d(a, X)\}$, this implies that $\forall x \in X$, $d(a, x) = d(a^*, x) + (d(a, X) - 1)$. Furthermore, X is a $(A \cup \{a^*\}|B)$ -separator of G' . Hence $\forall b \in B$, all a^*b -paths of G' , resp. all ab -paths of G' , intersect X . As a result, we have that for every $b \in B$, $d(a, b) = d(a^*, b) + d(a, X) - 1$ and replacing a with a^* does not change the hyperbolicity of the 4-tuple a, b_1, b_2, b_3 . \square

Note that it may be the case that $\delta(G') > \delta(G)$, where G' is the graph defined in Lemma 4. However this is no big deal at this step of the proof, for we are only interested in upper-bounding $\delta(a, b_1, b_2, b_3)$. We will come back to the difference between $\delta(G)$ and $\delta(G')$ later on, in Section 5.

Lemma 5. *Let X be a $(A|B)$ -clique-separator of a connected graph G . Given a $(a|b_1, b_2, b_3)$ 4-tuple, let $x \in X$ be such that $d(a, x) = d(a, X)$. We have $\delta(a, b_1, b_2, b_3) \leq \delta(x, b_1, b_2, b_3) + 1/2$.*

Proof. Let G' be obtained from G by adding a vertex a^* adjacent to $\{x \in X : d(a, x) = d(a, X)\}$. By construction, G is an isometric subgraph of G' , and so, the respective values of $\delta(a, b_1, b_2, b_3)$ and $\delta(x, b_1, b_2, b_3)$ are not modified by the addition of a^* . However, $\delta(G') \geq \delta(G)$ with the inequality possibly being strict, but we don't use $\delta(G')$ in the proof.

By Lemma 4 we have $\delta(a^*, b_1, b_2, b_3) = \delta(a, b_1, b_2, b_3)$. Furthermore, we claim that $\forall b \in B$, $d(b, x) \leq d(b, a^*) \leq d(b, x) + 1$. Indeed, since a^* and x are adjacent in G' , by the triangular inequality $d(b, a^*) \leq d(b, x) + 1$. Since in addition X is a $(A \cup \{a^*\}|B)$ -separator of G' , all a^*b -paths of G' must intersect X . As a result, and since $x \in X$ and X is a clique, $d(b, x) \leq d(b, a^*)$, that proves the claim.

Let us assume w.l.o.g. that $S_1 \geq S_2 \geq S_3$, where $S_1 = d(a^*, b_1) + d(b_2, b_3)$, $S_2 = d(a^*, b_2) + d(b_1, b_3)$ and $S_3 = d(a^*, b_3) + d(b_1, b_2)$. Since $\forall i \in \{1, 2, 3\}$, by the above claim $d(b_i, x) \leq d(b_i, a^*) \leq$

$d(b_i, x) + 1$, any sum $S'_i = d(x, b_i) + d(b_j, b_k)$, where $\{j, k\} = \{1, 2, 3\} \setminus \{i\}$, satisfies $S'_i \leq S_i \leq S'_i + 1$. Therefore, we have that for every $i \in \{2, 3\}$:

$$\begin{aligned} \delta(a^*, b_1, b_2, b_3) &\leq (S_1 - S_i)/2 \\ &\leq (S'_1 + 1 - S'_i)/2 \\ &\leq (S'_1 - S'_i)/2 + 1/2. \end{aligned}$$

In particular, if $S'_1 \neq \max\{S'_1, S'_2, S'_3\}$, then we have $\delta(a^*, b_1, b_2, b_3) \leq \frac{1}{2}$ by the choice of $S'_i = \max\{S'_1, S'_2, S'_3\}$. Otherwise, we have $\delta(a^*, b_1, b_2, b_3) \leq \delta(x, b_1, b_2, b_3) + \frac{1}{2}$ by the choice of $S'_i = \max\{S'_2, S'_3\}$. \square

3.3 Disconnection by a clique-separator

Summing up the two previous Sections 3.1 and 3.2, we can upper-bound the difference between the respective hyperbolicity of a graph and of its atoms in the particular case when there is a unique clique-separator. The following Theorem 6 was proved independently in [51, 60] for separators of any diameter. We prove it here for self-containment of the paper. Lemmas 2 and 5 will be reused in the following sections.

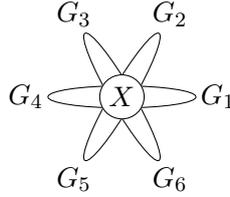


Figure 5: Illustration of a clique-separator X with the connected components of $G \setminus X$.

Theorem 6. *Let X be a clique-separator of a connected graph G , and let C_1, \dots, C_l be the connected components of $G \setminus X$. We define $G_i = G[C_i \cup X]$. We have :*

$$\max\{\delta(G_1), \dots, \delta(G_l)\} \leq \delta(G) \leq \max\{1/2, \delta(G_1), \dots, \delta(G_l)\} + 1/2.$$

Proof. Since X is a clique, and so, $G[X]$ is an isometric subgraph, every subgraph G_i is isometric as well. Hence, the lower-bound follows from the 4-point definition (Definition 1).

Let us now prove that $\delta(a, b, c, d) \leq \max\{1/2, \delta(G_1), \dots, \delta(G_l)\} + 1/2$ holds for any $a, b, c, d \in V$. We consider a connected component C_i minimizing the number of vertices in the 4-tuple a, b, c, d that are *not* in the block $C_i \cup X$. There are three cases to be considered.

- If $a, b, c, d \in C_i \cup X$ we are done as $\delta(a, b, c, d) \leq \delta(G_i)$.
- If all of a, b, c, d but one vertex are in $C_i \cup X$ let us assume w.l.o.g. that $a \notin C_i \cup X$. Then a, b, c, d is a $(a|b_1, b_2, b_3)$ 4-tuple, for the choices of $B = C_i \cup X$ and $A = V \setminus C_i$. By Lemma 5 it follows that $\delta(a, b, c, d) \leq \delta(G_i) + 1/2$.

- Else, there are no more than two vertices among a, b, c, d that are in $C_i \cup X$. Suppose w.l.o.g. $a, c \notin C_i \cup X$. Let j, k satisfy $a \in C_j$ and $c \in C_k$. By minimality of $|\{a, b, c, d\} \setminus (C_i \cup X)|$ we have $b, d \notin C_j \cup C_k$. Therefore, a, b, c, d is a $(a_1, a_2 | b_1, b_2)$ 4-tuple, for the choices of $A = C_j \cup C_k \cup X$ and $B = V \setminus (C_j \cup C_k)$, and we conclude by Corollary 3 that $\delta(a_1, a_2, b_1, b_2) \leq 1$ in this case. \square

The upper-bound of Theorem 6 is sharp. It can be shown using the graph in Figure 6, constructed from a cycle C_7 of length 7 to which we add a triangle.

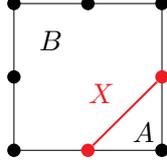


Figure 6: X is a $(A|B)$ -clique-separator: we have $\delta(G) = 3/2$, while $\delta(G[B]) = 1$, and $\delta(G[A]) = 0$.

4 Hyperbolicity and clique-decomposition

In Section 3, we gave a sharp upper-bound on the distortion of hyperbolicity when the graph is disconnected by a single clique-separator. The atoms of the graph result from its disconnection by some clique-separators [44]. However, Theorem 6 does not apply to a whole clique-decomposition as the successive approximations would add up. We thus need to find additional properties to approximate the hyperbolicity of a graph from computations on its atoms in order to prove Theorem 12.

Our proofs in this section is based on the property that the atoms of a graph can be organized into a tree (sometimes called an atom tree [9] or a maximal prime subgraph junction tree [48]). Using this tree, any 4-tuple with large hyperbolicity can be related to an atom that is most “central” to it (this will be made more precise in the following). We can then upper-bound the difference between the hyperbolicity of the 4-tuple and the hyperbolicity of this atom. For the latter, a delicate technical argument will be needed in order to obtain the sharp upper-bound on the difference.

4.1 Relating atoms and 4-tuples with large hyperbolicity

We aim at relating every 4-tuple a, b, c, d with a sufficiently large hyperbolicity to some atom by which all the paths between a, b, c, d go through. The difference between $\delta(a, b, c, d)$ and the hyperbolicity of this atom will be studied next. Our result in this section involves basic knowledge about *tree-decomposition* (see [10]). A *tree-decomposition* (T, \mathcal{X}) of a graph $G = (V, E)$ is a pair consisting of a tree T and of a family $\mathcal{X} = (X_t)_{t \in V(T)}$ of subsets of V indexed by the nodes of T and satisfying:

- $\bigcup_{t \in V(T)} X_t = V$;
- for any edge $e = \{u, v\} \in E$, there exists $t \in V(T)$ such that $u, v \in X_t$;
- for any $v \in V$, $\{t \in V(T) \mid v \in X_t\}$ induces a subtree of T , denoted by T_v .

The sets X_t are called *the bags* of the decomposition. In the following, we will use the property that there exists a tree-decomposition where the bags are exactly the atoms [9, 48].

Lemma 7. *Let $a, b, c, d \in V$ be a 4-tuple satisfying $\delta(a, b, c, d) \geq \frac{3}{2}$ in a connected graph $G = (V, E)$. There exists an atom A_0 such that $\forall u \in \{a, b, c, d\} \setminus A_0$, there is a clique-separator $X_u \subseteq A_0$ which separates u from $\{a, b, c, d\} \setminus \{u\}$.*

Proof. Let (T, \mathcal{X}) be a tree-decomposition of G where the bags are the atoms of G . Such a tree-decomposition was proved to exist in [9, 48]. In order to prove the lemma, we shall seek for an atom A_0 with the property that no more than two vertices among $\{a, b, c, d\} \setminus A_0$ are in the same connected component of $G \setminus A_0$. To find this atom, we will weight the bags of \mathcal{X} and then we will choose the atom A_0 in the weighted centroid of T .

Precisely, for every of a, b, c, d pick an atom containing the vertex. The weight of an atom is the number of times it has been picked. In particular, an atom has weight between 0 and 4, and the sum of weight of the atoms is equal to $\mathcal{W} = 4$. It is well-known that for any node-weighted tree with sum of weights \mathcal{W} , there is a node whose removal splits the tree into connected components where the sum of weight of the nodes is at most $\mathcal{W}/2$ [34]. So, let A_0 be an atom of G such that no component of $T \setminus \{A_0\}$ has sum of weight of its bags greater than 2. We claim that $\forall u \in \{a, b, c, d\} \setminus A_0$, there is a clique-separator $X_u \subseteq A_0$ which separates u from $\{a, b, c, d\} \setminus \{u\}$, that will prove the lemma.

Indeed, let $u \in \{a, b, c, d\} \setminus A_0$ be arbitrary. By the properties of a tree-decomposition, T_u (induced by the atoms containing u) is the subtree of a component C_u of $T \setminus \{A_0\}$. Let $V_u \subseteq V$ be the subset of vertices that are contained in an atom in C_u , and let $A_u \in C_u$ be the atom that is adjacent to A_0 in T . Since A_u and A_0 are atoms of G , their intersection, denoted by $X_u = A_u \cap A_0$, is a clique [8]. Furthermore, by the properties of a tree-decomposition, X_u is a $(V_u | V \setminus V_u)$ -separator of G . Therefore, we are left to prove that no vertex of $\{a, b, c, d\} \setminus \{u\}$ is in V_u , for the latter will prove that X_u is a clique-separator which separates u from $\{a, b, c, d\} \setminus \{u\}$. Assume for the sake of contradiction the existence of a vertex $v \in \{a, b, c, d\} \setminus \{u\}$ that is contained in V_u . We distinguish between two cases.

- Suppose that $v \notin X_u$. In this situation, T_u, T_v are subtrees of C_u . It implies that the sum of weight of the atoms in C_u is at least 2, and so, by the choice of atom A_0 , it is equal to 2. In particular, u, v are the only two vertices of the 4-tuple that are in $V_u \setminus X_u$ (else, the sum of weights of the atoms in C_u should be at least 3). Let $X = X_u$, $A = V_u$, $B = (V \setminus V_u) \cup X_u$. The 4-tuple a, b, c, d is a $(a_1, a_2 | b_1, b_2)$ 4-tuple with $a_1 = u$, $a_2 = v$. Therefore, $\delta(a, b, c, d) \leq 1$ by Corollary 3, that contradicts the hypothesis that $\delta(a, b, c, d) \geq \frac{3}{2}$.
- Else, $v \in X_u$ and we can assume w.l.o.g. that no vertex of $\{a, b, c, d\} \setminus \{u\}$ is in $V_u \setminus X_u$ (else, we go back to the previous case). In this situation, let $X = X_u$, $A = V_u$, $B = (V \setminus V_u) \cup X_u$ as before, the 4-tuple a, b, c, d is a $(a_1, a_2 | b_1, b_2)$ 4-tuple with $a_1 = u$, $a_2 = v$. Therefore, similarly as for the previous case, we have that $\delta(a, b, c, d) \leq 1$ by Corollary 3, that contradicts the hypothesis that $\delta(a, b, c, d) \geq \frac{3}{2}$.

As a result, no vertex of $\{a, b, c, d\} \setminus \{u\}$ is in V_u , and so, X_u is a clique-separator which separates u from $\{a, b, c, d\} \setminus \{u\}$. Since $X_u \subseteq A_0$, the latter proves the claim on A_0 , hence the lemma. \square

As an illustration, one may notice that the central atom in Figure 7 satisfies the property of Lemma 7 with respect to the 4-tuple v_0, v_1, v_2, v_3 . Indeed, none of the four vertices is contained in this atom, but each of them is simplicial and can be separated from the three others by its two neighbors.

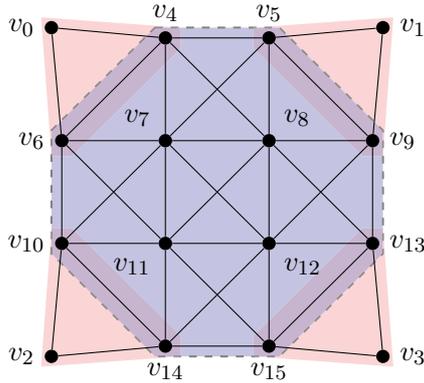


Figure 7: A 2-hyperbolic graph with five atoms: four are 0-hyperbolic, one is 1-hyperbolic. The 4-tuple v_0, v_1, v_2, v_3 has maximum hyperbolicity 2.

Discussion. It is tempting to attempt a generalization of Lemma 7 to some other graph decompositions. In particular, we may wish to consider a decomposition of the graph by separators of diameter at most k , where k is a small constant (the case $k = 1$ corresponds to the clique-decomposition). If we assume in addition that the subgraphs are organized into a tree (*i.e.*, they are the bags of a tree-decomposition), the generalization of Lemma 7 to that case is easy. However, in general there is no such a tree-decomposition, which kills all the arguments in our proof.

We can go one step further and prove the following claim.

Proposition 8. *For every $G = (V, E)$, let G_1, G_2, \dots, G_l denote the isometric subgraphs of G that do not contain any isometric separator of diameter at most two¹. The difference between $\delta(G)$ and $\max_i \delta(G_i)$ can be arbitrarily large.*

Proof. We shall prove that the result follows from some properties of bridged graphs. Indeed, we recall that a vertex is dominated if its close neighborhood is included in the close neighborhood of another vertex, and a graph is bridged if every of its isometric subgraphs contains a dominated vertex [3]. We claim that for every bridged graph G , all the subgraphs G_i have hyperbolicity at most one. Since the hyperbolicity of bridged graphs can be arbitrarily large [41], the result follows.

In order to prove the claim, let G be a bridged graph and let G_i be an isometric subgraph of G where there is no isometric separator of diameter at most two. If G_i has order at most 1, then we are done as $\delta(G_i) = 0$ in this case. So, let us assume G_i contains at least two vertices. Since G is bridged, G_i contains a dominated vertex v_i . Let $u_i \in V(G_i)$ be a vertex dominating v_i . In this situation, u_i is universal in G_i , or else $N_G[u_i] \setminus v_i$ would be an isometric separator of G_i of diameter at most two. Therefore, G_i has diameter at most two, and so, $\delta(G_i) \leq 1$ [21], that proves the claim, hence the result. \square

4.2 Substitution method

From Lemma 7 we can associate a specific atom to a 4-tuple of large hyperbolicity. Four applications of Lemma 5 are then sufficient to prove that the hyperbolicity of this 4-tuple and the hyperbolicity

¹We name a separator isometric when it induces an isometric subgraph.

of the atom differ by at most 2. The purpose of this section is to prove that this difference is in fact at most 1. To do this, we refine the results of Section 3.2.

We recall that our substitution method in Section 3.2 consists in relating a $(a|b_1, b_2, b_3)$ 4-tuple to a 4-tuple x, b_1, b_2, b_3 , with $x \in X$, such that $\delta(a, b_1, b_2, b_3) - \delta(x, b_1, b_2, b_3) \leq 1/2$. The difference between the hyperbolicity of the 4-tuples depends on the choice of x and on some properties of the $(a|b_1, b_2, b_3)$ 4-tuple. So, we will first deepen our analysis of the worst-case when it is equal to $1/2$. We will finally prove that when we apply twice the substitution method to a 4-tuple with a large hyperbolicity, this maximum difference of $1/2$ can occur at most once.

We will need the following lemma, that is a technical generalization of Lemma 5.

Lemma 9. *Let X be a $(A|B)$ -clique-separator of a connected graph G . Given a $(a|b_1, b_2, b_3)$ 4-tuple, write:*

$$S_1 = d(a, b_1) + d(b_2, b_3), \quad S_2 = d(a, b_2) + d(b_1, b_3), \quad S_3 = d(a, b_3) + d(b_1, b_2).$$

Assume w.l.o.g. that $S_1 \geq S_2 \geq S_3$, and let $x_2 \in X$ be such that $d(a, b_2) = d(a, x_2) + d(x_2, b_2) = d(a, X) + d(x_2, b_2)$. If $\delta(a, b_1, b_2, b_3) > \delta(x_2, b_1, b_2, b_3)$, then we have:

- $S_1 > S_2 = S_3$.
- $d(a, b_1) = d(a, x_2) + d(x_2, b_1)$.

Proof. For ease of calculation, we first reduce to the case when $d(a, X) = 1$. Let G' be obtained from G by adding a vertex a^* with neighbors $\{x \in X \mid d(a, x) = d(a, X)\}$. By construction, G is an isometric subgraph of G' , and so, the hyperbolicity of 4-tuples of $V(G)$ is not modified by the construction. Furthermore, by Lemma 4 we have $\delta(a^*, b_1, b_2, b_3) = \delta(a, b_1, b_2, b_3)$. In this situation, we can safely replace a with a^* in the 4-tuple, hence we may assume w.l.o.g. that $d(a, X) = 1$ for the remaining of the proof.

For every i , let $x_i \in X$ denote a vertex on a shortest ab_i -path such that $d(a, x_i) = d(a, X) = 1$. In this situation, $d(a, b_i) = d(b_i, x_i) + 1$. Let us introduce the indicator $\varepsilon_i = d(x_2, b_i) - d(x_i, b_i)$. We claim that $\varepsilon_i \in \{0, 1\}$, and $\varepsilon_i = 0$ if and only if x_2 is on a shortest ab_i -path. Indeed, since by the triangular inequality, $1 + d(x_i, b_i) = d(a, b_i) \leq d(a, x_2) + d(x_2, b_i) = 1 + d(x_2, b_i)$, we have that $\varepsilon_i \geq 0$. Furthermore, since $x_2, x_i \in X$ and X is a clique, we also have by the triangular inequality that $\varepsilon_i \leq d(x_2, x_i) \leq 1$. Altogether, $\varepsilon_i \in \{0, 1\}$, and we have $\varepsilon_i = 0$ if and only if $d(a, b_i) = d(b_i, x_2) + d(a, x_2)$, that proves the claim. In particular $\varepsilon_2 = 0$.

Let us denote by S'_i the sum $d(x_2, b_i) + d(b_j, b_k)$, where $\{j, k\} = \{1, 2, 3\} \setminus \{i\}$. We aim to exhibit a relation between S_i and S'_i , that would yield in turn a relation between the values $\delta(a, b_1, b_2, b_3)$ and $\delta(x_2, b_1, b_2, b_3)$. At first we notice that $d(a, b_i) = d(x_i, b_i) + 1 = d(x_2, b_i) + 1 - \varepsilon_i$. So, we have:

$$\begin{aligned} S_i &= d(a, b_i) + d(b_j, b_k) \\ &= d(x_2, b_i) + d(b_j, b_k) + 1 - \varepsilon_i \\ &= S'_i + 1 - \varepsilon_i. \end{aligned}$$

Furthermore, by the hypothesis $S_1 \geq S_2 \geq S_3$ and $\delta(a, b_1, b_2, b_3) > 0$. Thus, by Definition 1 we have $S_1 = S_2 + 2\delta(a, b_1, b_2, b_3)$, and so, it holds that $S_1 > \max\{S_2, S_3\}$. The latter implies $S'_1 \geq S_1 - 1 \geq \max\{S_2, S_3\} \geq \max\{S'_2, S'_3\}$. More precisely:

- Suppose $S'_2 \geq S'_3$. In this situation, $\delta(a, b_1, b_2, b_3) = \frac{S_1 - S_2}{2} = \frac{(S'_1 + 1 - \varepsilon_1) - (S'_2 + 1 - \varepsilon_2)}{2}$. Since $\varepsilon_2 = 0$, we have $\delta(a, b_1, b_2, b_3) = (S'_1 - S'_2)/2 - \varepsilon_1/2 = \delta(x_2, b_1, b_2, b_3) - \varepsilon_1/2 \leq \delta(x_2, b_1, b_2, b_3)$. However, this case contradicts the hypothesis that $\delta(a, b_1, b_2, b_3) > \delta(x_2, b_1, b_2, b_3)$.
 - Else, $S'_3 > S'_2$. Since $S_2 \geq S_3$, we have $\varepsilon_3 = 1$, which implies $S_2 = S_3$. This, in turn, implies that $\delta(x_2, b_1, b_2, b_3) = (S'_1 - S'_3)/2 = (S_1 - 1 + \varepsilon_1 - S_3)/2 = (S_1 - S_2)/2 - (1 - \varepsilon_1)/2 = \delta(a, b_1, b_2, b_3) - (1 - \varepsilon_1)/2 \leq \delta(a, b_1, b_2, b_3)$.
- In such a case, $\delta(x_2, b_1, b_2, b_3) < \delta(a, b_1, b_2, b_3)$ if and only if we have $\varepsilon_1 = 0$, *i.e.*, x_2 is on a shortest ab_1 -path. □

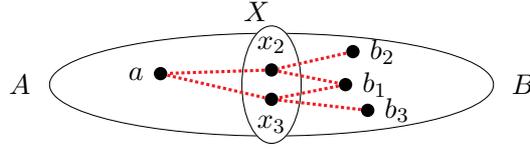


Figure 8: An illustration of the metric property of Lemma 9. The dashed lines represent shortest paths.

The metric property of Lemma 9 is illustrated with Figure 8. We use it to strengthen our results in Section 3.2 as follows.

Lemma 10. *Let a, b, c, d be a 4-tuple of a connected graph G , and X_a, X_d be two cliques of G satisfying:*

- X_a is a $(a|b, c, d)$ -separator;
- X_d is a $(d|a, b, c)$ -separator;
- all vertices of $X_a \setminus X_d$ and a, b, c are in the same connected component of $G \setminus X_d$.

Then there exist $x_a \in X_a$ and $x_d \in X_d$ such that

$$\delta(a, b, c, d) \leq \delta(x_a, b, c, x_d) + 1/2.$$

Proof. Consider the three sums S_1, S_2, S_3 from Definition 1. For ease of reasoning, let us order the sums by decreasing value, *i.e.*, let T_1, T_2, T_3 be such that $\{T_1, T_2, T_3\} = \{S_1, S_2, S_3\}$ and $T_1 \geq T_2 \geq T_3$. Accordingly, let u_1, u_2, u_3 be such that $\{u_1, u_2, u_3\} = \{b, c, d\}$ and for every $i \in \{1, 2, 3\}$, $T_i = d(a, u_i) + d(u_j, u_k)$ where $\{j, k\} = \{1, 2, 3\} \setminus \{i\}$. We distinguish between two cases:

- Suppose there is $x_a \in X_a$ satisfying $\delta(a, b, c, d) \leq \delta(x_a, b, c, d)$. By the hypothesis, the clique X_d is a $(d|x_a, b, c)$ -separator. Hence by Lemma 5 there exists $x_d \in X_d$ such that $\delta(x_a, b, c, d) \leq \delta(x_a, b, c, x_d) + \frac{1}{2}$. Altogether, $\delta(a, b, c, d) \leq \delta(x_a, b, c, d) \leq \delta(x_a, b, c, x_d) + \frac{1}{2}$, so, the lemma holds true in this case.
- Else, no vertex of X_a satisfies the above property. In particular, let $x_a \in X_a$ be such that $d(a, x_a) = d(a, X_a)$, and x_a is on a shortest au_2 -path. By the hypothesis, the clique X_a is a $(a|b, c, d)$ -separator. Hence, we can deduce the following information on the 4-tuple:

- by Lemma 5 $\delta(a, b, c, d) \leq \delta(x_a, b, c, d) + \frac{1}{2}$, and so, $\delta(a, b, c, d) = \delta(x_a, b, c, d) + \frac{1}{2}$;
- moreover, by Lemma 9 we have that $T_1 > T_2 = T_3$ and x_a is also on a shortest au_1 -path.

We shall prove that there exists $x_d \in X_d$ such that $\delta(x_a, b, c, d) \leq \delta(x_a, b, c, x_d)$, that will prove the lemma in this case.

For every i , let $T'_i = d(x_a, u_i) + d(u_j, u_k)$ where $\{j, k\} = \{1, 2, 3\} \setminus \{i\}$. Observe that when x_a is on a shortest au_i -path, we have that $T'_i = T_i - d(a, x_a) = T_i - d(a, X_a)$. As a result,

$$T'_1 = T_1 - d(a, X_a), \quad T'_2 = T_2 - d(a, X_a) \quad \text{and} \quad T'_3 = T_3 - d(a, X_a) + 1.$$

Indeed, the two first equalities follow from the fact that x_a is on a shortest au_1 -path, resp. on a shortest au_2 -path. The third equality follows from the fact that $\delta(x_a, b, c, d) = (T'_1 - \max\{T'_2, T'_3\})/2 = \delta(a, b, c, d) - 1/2$. In particular, we have $T'_1 \geq T'_3 > T'_2$.

Furthermore, by the hypothesis X_d is a $(d|x_a, b, c)$ -separator. Let v_1, v_2, v_3 be such that $\{v_1, v_2, v_3\} = \{x_a, b, c\}$ and for every i , $T'_i = d(v_i, d) + d(v_j, v_k)$ where $\{j, k\} = \{1, 2, 3\} \setminus \{i\}$. Finally, let $x_d \in X_d$ be a vertex satisfying $d(d, x_d) = d(d, X_d)$ and x_d is on a shortest v_3d -path. We claim that $\delta(x_a, b, c, d) \leq \delta(x_a, b, c, x_d)$, that will prove the lemma in this case. Indeed, suppose for the sake of contradiction that $\delta(x_a, b, c, x_d) < \delta(x_a, b, c, d)$. By Lemma 9, it implies that $T'_1 > T'_3 = T'_2$, which contradicts the fact that $T'_3 > T'_2$.

As a result, in both cases there exist $x_a \in X_a$ and $x_d \in X_d$ such that $\delta(a, b, c, d) \leq \delta(x_a, b, c, x_d) + 1/2$. \square

Corollary 11. *Let a, b, c, d be a 4-tuple of a connected graph G satisfying $\delta(a, b, c, d) \geq 3/2$. There exists an atom A of G such that $\delta(a, b, c, d) \leq \delta(G[A]) + 1$.*

Proof. The atom A is obtained by applying Lemma 7. For every vertex $u \in \{a, b, c, d\} \setminus A$, let $X_u \subseteq A$ be a clique-separator disconnecting u from $\{a, b, c, d\} \setminus \{u\}$.

We claim that all vertices of $A \setminus X_u$ and $\{a, b, c, d\} \setminus \{u\}$ are in the same connected component of $G \setminus X_u$. Indeed, since A is an atom, all vertices of $A \setminus X_u$ are in the same connected component C of $G \setminus X_u$. Suppose for the sake of contradiction that there exists $v \in \{a, b, c, d\} \setminus \{u\}$ such that $v \notin C$. Since $v \notin A$, there exists a clique $X_v \subseteq A$ which separates v from $\{a, b, c, d\} \setminus \{v\}$, and so, no vertex of $\{a, b, c, d\} \setminus \{v\}$ is in the same connected component of $G \setminus X_u$ as v . However, in this situation let C_u, C_v be the respective components of u and v in $G \setminus X_u$ and let $X = X_u$, $A = C_u \cup C_v \cup X$, $B = V \setminus (C_u \cup C_v)$. The 4-tuple a, b, c, d is a $(a_1, a_2|b_1, b_2)$ 4-tuple with $a_1 = u$, $a_2 = v$, and so, it implies by Corollary 3 that $\delta(a, b, c, d) \leq 1$, a contradiction. As a result, all vertices of $\{a, b, c, d\} \setminus \{u\}$ are in C , that proves the claim.

In order to prove the corollary, we can then apply our substitution method (at most four times). Precisely, consider the two vertices a, b of the 4-tuple.

- If $a, b \in A$ then we set $a' = a$, $b' = b$. In this situation $\delta(a, b, c, d) \leq \delta(a', b', c, d)$.
- Else, if $a \in A$ and $b \notin A$ then let $x_b \in X_b$ be a vertex satisfying $d(b, x_b) = d(b, X_b)$. Let $a' = a$, $b' = x_b$. By Lemma 5, we have $\delta(a, b, c, d) \leq \delta(a', b', c, d) + 1/2$. In the same way, if $a \notin A$, $b \in A$ then we set $a' = x_a$, $b' = b$ where $x_a \in X_a$ is a vertex satisfying $d(a, x_a) = d(a, X_a)$.

- Else, $a, b \notin A$. By the claim above, all vertices of $X_a \setminus X_b \subseteq A \setminus X_b$ and a, c, d are in the same connected component of $G \setminus X_b$. Hence, by Lemma 10 there exist $x_a \in X_a$ and $x_b \in X_b$ such that $\delta(a, b, c, d) \leq \delta(x_a, x_b, c, d) + 1/2$. We set $a' = x_a$, $b' = x_b$ in this case.

Overall, in all cases there exist $a', b' \in A$ such that $\delta(a, b, c, d) \leq \delta(a', b', c, d) + 1/2$. We then proceed similarly with c, d in order to find two vertices $c', d' \in A$ such that $\delta(a', b', c, d) \leq \delta(a', b', c', d') + 1/2$. Altogether, $\delta(a, b, c, d) \leq \delta(a', b', c', d') + 1 \leq \delta(G[A]) + 1$. \square

4.3 Additive approximation for hyperbolicity

Theorem 12. *Let A_1, \dots, A_l be the atoms of a connected graph G . Then:*

$$\max_i \delta(G[A_i]) \leq \delta(G) \leq \max_i \delta(G[A_i]) + 1.$$

Proof. As for Theorem 6, the lower-bound of Theorem 12 follows from the fact that the subgraphs $G_i = G[A_i]$ are isometric subgraphs of G . The upper-bound trivially holds when $\delta(G) \leq 1$. We can thus suppose that $\delta(G) \geq 3/2$ and so, that there exist four vertices a, b, c, d such that $\delta(a, b, c, d) = \delta(G) \geq 3/2$. Corollary 11 then yields an atom A such that $\delta(G) \leq \delta(G[A]) + 1$, which proves the second part of our claim. \square

Note that the upper-bound is reached by the graph of Figure 7, and by the 1-hyperbolic chordal graph from Figure 1 whose atoms have hyperbolicity 0.

5 Substitution method for an exact computation

As shown with Theorem 12, the hyperbolicity of a graph cannot be deduced from the computation of its clique-decomposition directly. We next introduce a new graph decomposition in order to compute the hyperbolicity of a graph exactly. It bases on clique-decomposition, and it outputs supergraphs of the atoms (we call them *substitute graphs*).

For the remaining of the section, we will consider graphs with hyperbolicity at least 1 (the case of 1/2-hyperbolic graphs will be discussed in the following sections). We will show that under this assumption on the hyperbolicity of the initial graph, it can be computed from the hyperbolicity of the atoms' substitutes.

Outline of the method. We build upon Lemma 4 and other results from the two previous sections. We recall that a *simplicial* vertex is a vertex whose neighborhood induces a complete subgraph. Given a $(A|B)$ -clique separator, we add simplicial vertices to the induced subgraphs $G[A]$ and $G[B]$ in order to mimic the maximum $(a|b_1, b_2, b_3)$ 4-tuples that may result from the disconnection (Section 5.1.1). Since the atoms result from the disconnection of the graph by some of its clique-separators, we can repeatedly apply this method and so obtain the atom's substitutes. We finally focus on technical details and additional data structures related to the implementation.

5.1 Substitute graphs

5.1.1 Basic step: single disconnection

Following [8] the clique-decomposition of $G = (V, E)$ can be computed by repeatedly applying the following decomposition step, until none of the subgraphs considered contains a clique-separator.

At first, G is the only subgraph. We find a clique-separator X with some properties² in one of the subgraphs G' . The vertex-set of G' is partitioned into two sets A, B so that $A \cap B = X$ and X is a $(A|B)$ -separator of G' . Then the subgraph G' is replaced with the two subgraphs $G'[A]$ and $G'[B]$. So, let us first consider the case of a *single* disconnection, of a given graph G' with a $(A|B)$ -clique-separator X .

- Let $G_A = G'[A]$. For every $b \in B \setminus X$, we consider the set of vertices $X_b \subseteq X$ which are at distance $d_{G'}(b, X)$ from b . For every X_b , we add in G_A a (simplicial) vertex s_{X_b} whose neighborhood is X_b . The resulting graph is named G_A^* .
- Let $G_B = G'[B]$. For every $a \in A \setminus X$, we consider the set of vertices $X_a \subseteq X$ which are at distance $d_{G'}(a, X)$ from a . For every X_a , we add in G_B a (simplicial) vertex s_{X_a} whose neighborhood is X_a . The resulting graph is named G_B^* .

More formally, the *substitute graphs* (or *substitutes* for short) G_A^* and G_B^* of the graphs G_A and G_B with respect to the $(A|B)$ -separator X are defined as follows:

Definition 13. Let X be a $(A|B)$ -clique-separator of a connected graph G' , where $A \cap B = X$ and $A \cup B = V(G')$. The substitute graphs G_A^*, G_B^* are defined as:

$$V(G_A^*) = A \cup \{s_{X_b} : \exists b \in B \text{ s.t. } X_b = \arg \min_{x \in X} d(b, x)\} \text{ and } E(G_A^*) = E(A) \cup \{\{s_{X_b}, x\} : x \in X_b\};$$

$$V(G_B^*) = B \cup \{s_{X_a} : \exists a \in A \text{ s.t. } X_a = \arg \min_{x \in X} d(a, x)\} \text{ and } E(G_B^*) = E(B) \cup \{\{s_{X_a}, x\} : x \in X_a\}.$$

Lemma 14. Let X be a $(A|B)$ -clique-separator of a connected graph G' , where $A \cap B = X$ and $A \cup B = V(G')$. Suppose $\delta(G') \geq 1$. We have:

$$\delta(G') = \max\{1, \delta(G_A^*), \delta(G_B^*)\}.$$

Proof. Let $G_A = G'[A]$, $G_B = G'[B]$. By construction (Definition 13), G_A is an isometric subgraph of G_A^* , resp. G_B is an isometric subgraph of G_B^* .

We claim $\delta(G') \leq \max\{1, \delta(G_A^*), \delta(G_B^*)\}$. In order to prove the claim, let a, b, c, d be a 4-tuple of vertices of G' such that $\delta(a, b, c, d) = \delta(G')$. We distinguish between three cases.

1. **Case** $a, b, c, d \in A$. In this situation, $\delta(G') \leq \delta(G_A) \leq \delta(G_A^*)$.

Similarly, when $a, b, c, d \in B$ we have $\delta(G') \leq \delta(G_B) \leq \delta(G_B^*)$.

2. **Case** $a \in A, b, c, d \in B$. The 4-tuple is a $(a|b_1, b_2, b_3)$ 4-tuple. In such case, there exists by construction a simplicial vertex a^* of $V(G_B^*) \setminus B$ that is adjacent to $\{x \in X : d(a, x) = d(a, X)\}$. Therefore, by Lemma 4 $\delta(G') \leq \delta(a^*, b_1, b_2, b_3) \leq \delta(G_B^*)$.

In the same way, if $b \in B$ and $a, c, d \in A$ then $\delta(G') \leq \delta(G_A^*)$.

3. Else, it can be assumed w.l.o.g. that the 4-tuple is a $(a_1, a_2|b_1, b_2)$ 4-tuple. By Corollary 3, $\delta(G') \leq 1$ in this case.

²The additional properties on the clique-separator ensure the unicity of the decomposition.

Altogether, $\delta(G') \leq \max\{1, \delta(G_A^*), \delta(G_B^*)\}$, as desired.

Conversely, we claim $\delta(G') \geq \max\{1, \delta(G_A^*), \delta(G_B^*)\}$. By the hypothesis, $\delta(G') \geq 1$. Furthermore, let a, b, c, d be a 4-tuple of G_B^* such that $\delta(a, b, c, d) = \delta(G_B^*)$ (the proof for G_A^* is symmetrical to this one). We distinguish between three cases.

1. **Case** $a, b, c, d \in B$. In this situation, $\delta(G_B^*) \leq \delta(G_B) \leq \delta(G')$.
2. **Case** $a = a^* \notin B$ and $b, c, d \in B$. By construction, a^* substitutes to some vertex of A , *i.e.*, there exists $a' \in A$ such that $N(a^*) = \{x \in X : d(a', x) = d(a', X)\}$. Furthermore, by Lemma 4 $\delta(a^*, b, c, d) = \delta(a', b, c, d)$. Hence, $\delta(G_B^*) \leq \delta(a', b, c, d) \leq \delta(G')$.
3. Else, there are at least two vertices of the 4-tuple that are not in B . Say w.l.o.g. $a, b \notin B$ and let $A^* = X \cup \{a, b\}$, $B^* = V(G_B^*) \setminus \{a, b\}$. In this situation, the 4-tuple is a $(a_1, a_2 | b_1, b_2)$ 4-tuple with $a_1 = a$, $a_2 = b$. As a result, by Corollary 3 $\delta(G_B^*) \leq 1 \leq \delta(G')$ in this case.

Altogether, $\delta(G_B^*) \leq \delta(G')$. □

We emphasize that some simple rules can be applied to reduce the size of the substitute graphs. In particular, we can remove the pendant vertices which may be added in the construction (we postpone a short analysis of the size of substitutes to Section 5.2.2).

5.1.2 Extension to the atoms

The substitution operation can be naturally extended to the whole clique-decomposition, by mimicking each step of it and applying the basic substitution operation that we describe above at each of these steps. We formalize it by first introducing the following definition of an atom tree.

Definition 15 ([7, 8, 44]). *Let G be a connected graph. An atom tree of G is a labeled binary rooted tree T , satisfying the following recursive definition:*

- if G is prime w.r.t. clique-decomposition, then T is reduced to a node labeled with V ;
- otherwise, the root of T is labeled with a clique-separator X , and there exists two connected components C_1, C_2 of $G \setminus X$ satisfying:
 - $N_G(C_1) = N_G(C_2) = X$;
 - the left child of the root is labeled with $A_1 = C_1 \cup X$, which does not contain any clique-separator;
 - and the right child of the root is an atom tree of $G \setminus C_1$.

In order to prevent any confusion, the reader has to notice that an atom tree is *not* a tree-decomposition (as defined in Section 4.1). In fact, an atom tree can be seen as the trace of some execution of the algorithm of [44, 56] for computing the clique-decomposition. Indeed, it is proved in [44] that in an atom tree, the leaves are in bijective correspondance with the atoms of the graph. Given a *fixed* atom tree, this yields a natural total ordering of the atoms by increasing depth. We now follow this ordering to construct the substitutes of the atoms from the atom tree. There are as many steps for our substitution method as there are atoms in the graph.

- Starting from $H_1 = G$, we disconnect the first atom A_1 by using the clique-separator X_1 from the atom tree. Applying the substitution operation of Definition 13 to $A = A_1$ and $B = V(G) \setminus (A_1 \setminus X_1)$, we obtain two substitute graphs: $G_{A_1}^*$ which substitutes A_1 , and another one denoted by $H_2 = G_B^*$.
- After $i-1$ steps, $i \in \{2, \dots, l-1\}$, we constructed the substitute graphs of atoms A_1, \dots, A_{i-1} , plus an additional graph H_i . The graph H_i contains $G[\bigcup_{j \geq i} A_j]$, to which were added simplicial vertices during the previous steps. By using the clique-separator X_i from the atom tree we disconnect the graph H_i and we apply the substitution operation of Definition 13, this time to the set A equal to $C_A \cup X_i$ where C_A is the connected component of $H_i \setminus X_i$ which intersects A_i , and to $B = V(H_i) \setminus (A \setminus X_i)$. We replace H_i with the two substitute graphs, one containing the atom A_i and being its substitute, the other being denoted by $H_{i+1} = G_B^*$.
- We finally stop at the l^{th} step, and we set H_l as the substitute graph of the *last* atom A_l .

Figure 9 illustrates this process. The numbers reported in Tab. 9i illustrate the interest of our pre-processing method for the computation of the hyperbolicity. Indeed, the graph G of Figure 9a has 28 nodes and so, 20 475 4-tuples, while the sum of the numbers of 4-tuples in the graphs G_i^* (Figs. 9c–9h) is 1 800. We thus significantly reduce the size of the search space. Moreover, a simple cutting rule allows us to reduce the number of 4-tuples to consider to 1 575. To do so, we first order the graphs G_i^* by decreasing diameters, then we iteratively compute the hyperbolicity of these graphs in this order, and we stop exploration as soon as the diameter of a graph G_j^* is smaller than twice the largest value of δ computed so far.

5.2 Implementation and complexity analysis

In order to stay competitive with clique-decomposition, the complexity of computing the substitutes needs to be of the same order of magnitude as the one of computing the atoms. A straightforward calculation would require the computation of distances in the graphs H_1, H_2, \dots, H_l (defined in the previous Section 5.1.2), hence the computation of $\mathcal{O}(n)$ all-pairs shortest-paths. Furthermore, the addition of simplicial vertices at each step causes an increase of the size of the graphs, which further complicates the complexity analysis.

The purpose of this section is to show how to compute the atom’s substitutes in $\mathcal{O}(nm)$ -time (Corollary 20) We achieve the goal by using standard partition refinement techniques and additional properties of clique-decomposition. On the way, we introduce a few rules in order to reduce the size of the substitutes.

5.2.1 Precomputation step and updates

We first focus on some computational tasks that have to be repeatedly executed at each step of our substitution method. In this section, we provide a high-level description for their implementation.

Computation of the distances. Given a $(A|B)$ -separator X , we need to compute all distances $d(v, x)$, $v \in V \setminus X, x \in X$, in order to compute the substitutes. If the distance matrix of the graph is precomputed then each distance can be accessed to in constant-time, hence an $\mathcal{O}(n|X|)$ -time complexity. However, new vertices are added by our construction. We wish to avoid recomputing

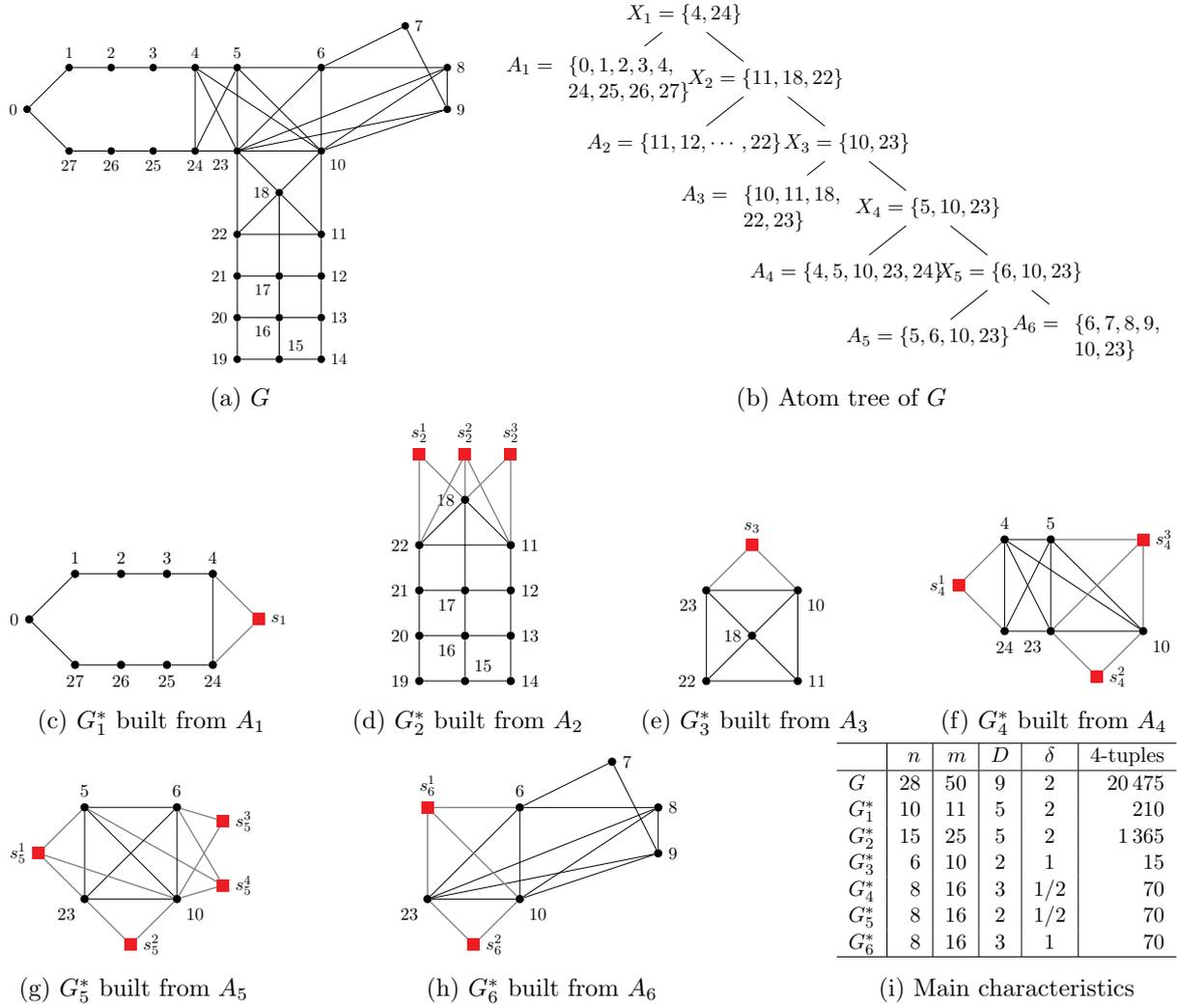


Figure 9: A connected graph G (Figure 9a), an atom tree of the graph (Figure 9b), the substitute of the atoms of G (Figs. 9c–9h), and the characteristics of these graphs (Tab. 9i).

the distances from scratch at each step of the substitution method (that would result in an $\mathcal{O}(n^2m)$ -time complexity). In what follows, we base on Lemma 4 (substitution of a vertex a by a simplicial vertex a^*) in order to reach the goal.

Lemma 16. *Let G be a connected graph. We can embed in quadratic time the distance matrix of G into a data structure, supporting:*

- $\mathcal{O}(1)$ access to the distance between a non-simplicial vertex and any other;
- $\mathcal{O}(|A|)$ updates when G is replaced with the substitute G_B^* (w.r.t. a $(A|B)$ -clique-separator X).

Proof. The gist of such a structure is Lemma 4. Let X be a $(A|B)$ -clique separator of G , and s be a simplicial vertex added in the substitute graph of $G[B]$. Let $a \in A \setminus X$ satisfy $N(s) = \{x \in X :$

$d_G(a, x) = d_G(a, X)$. Then we have for every $b \in B$, $d(b, s) = d_G(a, b) - d_G(a, X) + 1$.

It thus follows that once the substitution of a with s has been completed, we only need to remember the association of s with a and an offset, so that we can compute the distances in the substitute graphs. The offset can be computed in constant time by picking a neighbor of the simplicial vertex s , as it is the distance $d(a, X)$ between this neighbor and the vertex a which s substitutes. Finally, since there are $l = O(n)$ steps for our substitution method, and that no more than $O(n)$ new simplicial vertices are added at each step, a quadratic-size array is sufficient to store all the pairs $(a, d(a, X_a))$. \square

Note that the data structure of Lemma 16 does not support the computation of distances between two vertices added by our construction. We can safely ignore this drawback, as we do not need to compute such distances in our method.

Computation of connected components. Other complexity bottlenecks arise from the computation of connected components in graphs with a superlinear number of edges (up to $\Omega(nm)$ edges). Indeed, at each step i , $1 \leq i < l$, we need to compute the connected component C_A containing the next atom A_i to deal with. Determining the connected components of a graph is linear-time computable. However, as we detailed in Section 5.1.2, here we have to extract the component from a graph $H_i \neq G$, possibly containing more edges than G due to the addition of simplicial vertices at previous steps. Thus it may result in an $\Omega(m)$ -time complexity by using the classical algorithm for this problem. Instead, we propose a method to construct the component incrementally, starting from A_i and adding simplicial vertices at every step $1 \leq j \leq i - 1$.

Lemma 17. *Let G be a connected graph, T be an atom tree of G , and A_1, \dots, A_l be its atoms ordered according to their depth in T . We denote by $H_1 = G, H_2, \dots, H_l$ the sequence of l graphs that are computed by our process, each H_i being decomposed into H_{i+1} and the substitute graph of the i^{th} atom by applying the substitution method of Definition 13. For every (simplicial) vertex $s_i \in H_{i+1} \setminus H_i$, we can compute the index j such that s_i belongs to the substitute graph of the j^{th} atom, in total $O(n|X_i|)$ -time.*

Proof. Let $s_i \in H_{i+1} \setminus H_i$ be a simplicial vertex. By construction (Section 5.1.2), we have $N(s_i) \subseteq X_i \subseteq V(G)$. Therefore, if s_i belongs to the substitute graph of the j^{th} atom, $j > i$, then it holds that $N(s_i)$ intersects the connected component containing $A_j \setminus X_j$ —in the graph $H_j \setminus X_j$ —. In such case since every vertex in the component either belongs to the atom or is simplicial and not in $V(G)$, then it follows that s_i has a neighbor in $A_j \setminus X_j$. Conversely, if $s_i \in V(H_j)$ and $N(s_i) \cap (A_j \setminus X_j) \neq \emptyset$, then s_i is in the same connected component as $A_j \setminus X_j$ in the graph $H_j \setminus X_j$, hence s_i belongs to the substitute graph of the j^{th} atom. So, at every step of our substitution method, if a simplicial vertex is added by our construction, we consider the minimum index j such that $A_j \setminus X_j$ contains a neighbor of the vertex, and we update the vertex set of the substitute graph of the j^{th} atom by adding this new vertex into it. Since only $O(n)$ vertices are added at step i , and that their neighborhood is contained into X_i , the $O(n|X_i|)$ -time complexity follows. \square

The two above routines (Lemmas 16 and 17) will be combined in what follows in order to obtain the desired $\mathcal{O}(nm)$ -time complexity for our substitution method. Before this, we will show how to reduce the number of simplicial vertices to be added at each step (Section 5.1.1).

5.2.2 Applying simplification rules

The purpose of this section is to reduce the size of the substitutes. Precisely, given $G = (V, E)$ and X a $(A|B)$ -clique-separator, we wish to construct a substitute G_B^* from $G[B]$ by adding as few simplicial vertices as possible. A naive implementation would consist in computing the subsets $X_a = \{x \in X : d_G(a, x) = d_G(a, X)\}$, for every $a \in A \setminus X$, then adding a simplicial vertex adjacent to X_a . However, by doing so we would add $|A \setminus X|$ new vertices in the substitute, and so, we would lose all the benefit of the separation in terms of size of the graphs. We now define rules in order to avoid this worst-case in some situations. The goal of this section is to give hints on an efficient way to implement these rules.

Partition refinement techniques. We remove pendant and *twin vertices*. Indeed, it may happen that $X_a = \{x_a\}$ for some $a \in A$, and in such a case we needn't add a simplicial vertex for a since the removal of a pendant vertex does not affect the value of hyperbolicity. Furthermore, it may also happen that $X_a = X_{a'}$ for some pair $a, a' \in A$, and in such a case we wish to add only one simplicial vertex in G_B^* .

To do that efficiently, we will use the well-known *partition refinement* techniques (see *e.g.* [36, 49]). Given a partition \mathcal{P} of a set V , and a subset $S \subseteq V$ called the *pivot*, the partition refinement of \mathcal{P} w.r.t. S consists in replacing every group V_i of \mathcal{P} by the non-empty groups among $V_i \cap S$ and $V_i \cap \bar{S}$. This can be achieved in $O(|S|)$ -time, up to the precomputation of an appropriate data structure in linear $O(|V|)$ -time.

We deduce from this standard technique the following result:

Lemma 18. *Let G be a connected graph given by its distance matrix, and $X \subseteq V(G)$. We define the relation \equiv_X over the set $V(G) \setminus X$ as*

$$u \equiv_X v \iff^{def} \{x \in X : d_G(u, x) = d_G(u, X)\} = \{x \in X : d_G(v, x) = d_G(v, X)\}.$$

The equivalence classes of \equiv_X can be computed in $O(n|X|)$ -time.

Proof. Since the distance matrix is given, we can compute $X_u = \{x \in X \mid d_G(u, x) = d_G(u, X)\}$ for every vertex u , that takes $\mathcal{O}(n|X|)$ -time. Then, we start from the partition $\mathcal{P} = \{V \setminus X\}$ which we refine successively for every $x \in X$ with the set $\{u : u \in V \setminus X \text{ s.t. } d_G(u, x) = d_G(u, X)\}$. The total cost is $O(\sum_{x \in X} |N_{G_X}(x)|) = O(n|X|)$. \square

There are at most $2^{|X|}$ equivalence classes of \equiv_X , and we can remove the $|X| + 1$ classes corresponding to the subsets of X of size at most one. Altogether, the substitute G_B^* has at most $|B| + \min\{|A \setminus X|, 2^{|X|} - |X| - 1\}$ vertices when we apply our simplification rules. This bound is sharp, as shown with Figure 10.

Overall, the substitution method will add at most $\sum_{i=1}^{l-1} \min\{n, 2^{|X_i|} - |X_i| - 1\}$ new vertices to the vertex-set of the atoms. In particular, consider the special case when all but at most c clique-separators X_i have size at most k , for some universal constants c and k . In this situation, there are at most $(c + 2^k - k - 1) \cdot n$ new vertices added to the atoms. Hence, the total number of vertices is only increased by a constant-factor. This property will hold for the outerplanar graphs and some real-life graphs that we will study in the next two sections.

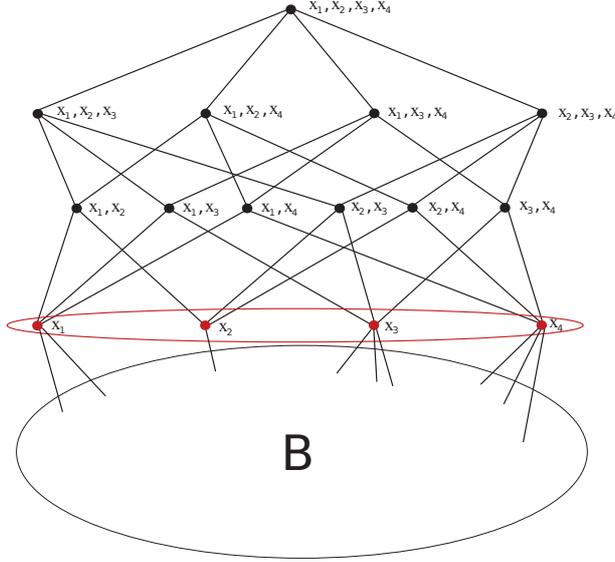


Figure 10: A case when $|A \setminus X| = 2^{|X|} - |X| - 1$ vertices need to be added in G_B^* . The graph $G[A]$ is obtained from the subset lattice of the set $X = \{x_1, x_2, x_3, x_4\}$. Every vertex $a \in A$ is labeled with $\{x \in X \mid d(a, x) = d(a, X)\}$.

5.2.3 Complexity analysis

Finally, to determine the time complexity of our substitution method, we will use the following result:

Lemma 19 ([7]). *Let G be a connected graph, and A_1, \dots, A_l be its atoms. Then $\sum_i |A_i| \leq n + m$.*

Corollary 20. *The substitute of the atoms of a connected graph G can be computed in $O(nm)$ -time.*

Proof. The notations are the same as for Section 5.1.2. That is, fix any atom tree T of G and let A_1, \dots, A_l be the atoms ordered by increasing depth. For every $i < l$, let X_i be the clique-separator of G labeling the father node of leaf A_i in T . By Definition 15, $X_i \subseteq A_i$.

We first precompute the distance matrix of G in $O(nm)$ -time, then we embed it in quadratic-time into the data structure of Lemma 16. Also, for every i , we initialize the vertex set of the i^{th} substitute graph with the atom A_i . We then apply each step of our substitution method sequentially.

Precisely, at each step i we are given the vertex set V_i of the graph H_i to be considered. Initially, $V_1 = V(G)$.

- Let us partition the vertices in A and B , where A is the set of all vertices in V_i that are in the i^{th} substitute graph. Since A is known (initialized with A_i then updated at each previous step), it can be done in $\mathcal{O}(|V_i|)$ -time.
- Then, we compute the simplicial vertices that result from the substitution operation, and we add them in A and B , respectively. Since we have constant-time access to the distances, we have by Lemma 18 that it can be done in $\mathcal{O}(|V_i||X_i|)$ -time.
- For every vertex newly added at this step, we next compute the index of the atom's substitute to which it belongs to. By Lemma 17, it can also be done in $\mathcal{O}(|V_i||X_i|)$ -time.

- We set $V_{i+1} = B$ and then we update the distances accordingly. By Lemma 17, it can be done in $\mathcal{O}(|A|)$ -time, that is $\mathcal{O}(|V_i|)$.

We can easily show by induction that $|V_i| = \mathcal{O}(n)$. Hence, the i^{th} step can be executed in $\mathcal{O}(n|X_i|)$ -time. Overall, our modified clique-decomposition can be computed in $\mathcal{O}(n \sum_i |X_i|)$ -time, that is in $\mathcal{O}(n \sum_i |A_i|) = \mathcal{O}(nm)$ -time by Lemma 19. \square

6 Hyperbolicity of outerplanar graphs

Equipped with the substitution method of Section 5 and our in-depth analysis of clique-decomposition (Sections 3 and 4), we aim at applying these results in order to speed-up the computation of hyperbolicity in some graph classes. In the final two sections, we will review theoretical and practical cases when it is possible to do so. We start with a linear-time algorithm for computing the hyperbolicity of a given outerplanar graph (Theorem 28).

The outerplanar graphs can be characterized in several ways (see [53]). We will use the following ones.

A *planar graph* is a graph drawable in the Euclidean plane so that edges may only intersect at their endpoints. It is *outerplanar* if it stays planar whenever one adds a universal vertex to it. Equivalently, a graph is outerplanar if it is drawable in the Euclidean plane so that edges may only intersect at their endpoints, and all the vertices lie on a common face which is called the *outerface*. Such a drawing is furthermore called an outerplanar embedding, and it can be computed in linear-time [54]. Note that it easily follows from this definition that all cycles are outerplanar graphs.

The class of outerplanar graphs is minor-closed, and a graph is outerplanar if and only if it is K_4 -minor-free and $K_{2,3}$ -minor-free [53].

We will exploit nontrivial properties of the clique-decomposition of outerplanar graphs to prove Theorem 28. In particular, clique-separators of an outerplanar graph have size at most two, *i.e.*, they are either cut-vertices or edge-separators. More precisely, the atoms of an outerplanar graph are cycles. We will base on the property that the hyperbolicity of a given cycle can be computed in linear time, by using the following formula:

Lemma 21 ([59, 20]). *Cycles of order $4p + \varepsilon \geq 3$, with $p \geq 0$ and $\varepsilon \in \{0, 1, 2, 3\}$, are $(p - 1/2)$ -hyperbolic when $\varepsilon = 1$, and p -hyperbolic otherwise.*

The main purpose of this section is to obtain a similar characterization for the substitute of cycles (Lemma 26). So, we will analyze the properties of this class of graphs in Section 6.2. We will also need to compute the substitutes of atoms of a given outerplanar graph in linear time. For this purpose, we will rely on the notion of *weak dual* [4].

Definition 22. *Let G be a biconnected outerplanar graph. The weak dual of G is a tree T_G equal to the intersection graph of the atoms of G . Two adjacent nodes of T_G correspond to atoms which share a single edge.*

Note that a weak dual is nothing else than a tree-decomposition whose bags are the atoms of an outerplanar graph. If G is biconnected, then starting from an outerplanar embedding of the graph, we construct it by removing from the dual of the graph the universal vertex corresponding to the outerface (see Figure 13 for an example).

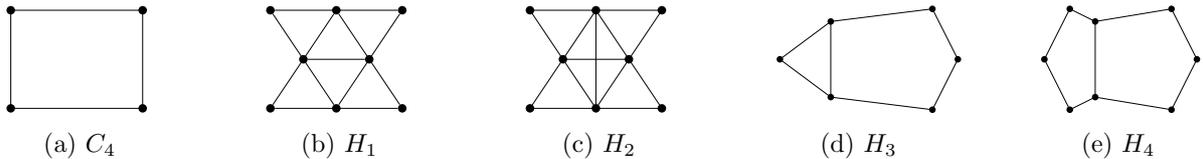


Figure 11: Characterization of 5-chordal $\frac{1}{2}$ -hyperbolic graphs, in terms of forbidden isometric subgraphs.

We will show that the atoms' substitutes can be computed by dynamic programming on the weak dual (Lemma 27). This combined with Lemma 26 and a characterization of $1/2$ -hyperbolic outerplanar graphs (Proposition 23) will achieve proving Theorem 28.

6.1 Outerplanar graphs with small hyperbolicity

As observed in Section 5, our substitution method for an exact computation of hyperbolicity requires the hyperbolicity of the graph to be at least 1. To overcome this drawback, we first characterize in this section outerplanar graphs that are $\frac{1}{2}$ -hyperbolic. Note that we only consider biconnected graphs, as the hyperbolicity of a graph is the maximum hyperbolicity taken over all its biconnected components, and the biconnected components of a graph are computable in linear-time [55].

Proposition 23. *A biconnected outerplanar graph is $\frac{1}{2}$ -hyperbolic if, and only if, either it is isomorphic to C_5 , or it is chordal and it does not contain the graph of Figure 11b as a subgraph. Furthermore, these conditions can be checked in linear-time.*

Proof. Let G be a $\frac{1}{2}$ -hyperbolic outerplanar biconnected graph. By Lemma 21, the graph C_5 is $\frac{1}{2}$ -hyperbolic, and we now assume that G is not isomorphic to C_5 . The induced cycles of G are exactly its atoms. As a result, we have by Lemma 21 that G only has induced cycles of length 3 or 5 (else, $\delta(G) \geq 1$). Moreover, Wu and Zhang prove in [59] that a 5-chordal graph is $\frac{1}{2}$ -hyperbolic if, and only if, it does not contain any graph of Figure 11 as an isometric subgraph³.

Since we consider graphs with induced cycles of length 3 or 5, G is C_4 -free and so, it does not contain the graph of Figure 11a as an isometric subgraph. Moreover, we claim that G is C_5 -free, as otherwise it would contain the graph of Figure 11d, or the graph of Figure 11e, as an isometric subgraph. Thus G has to be chordal. In addition, we claim that G cannot contain the graph of Figure 11c, since it is not outerplanar and being outerplanar is a hereditary property. Consequently, G is $\frac{1}{2}$ -hyperbolic if, and only if, it is chordal and it does not contain the graph of Figure 11b as an isometric subgraph.

Let H_1 be the graph of Figure 11b. To complete the proof of the proposition, we are left to prove that every subgraph of G that is isomorphic to H_1 is isometric. Indeed, the latter will prove that G is $\frac{1}{2}$ -hyperbolic if, and only if, it is chordal and it does not contain H_1 as a subgraph.

We observe that H_1 is an edge-maximal outerplanar graph, hence every subgraph isomorphic to H_1 must be induced. Suppose for the sake of contradiction that there is an induced subgraph of G that is isomorphic to H_1 but not isometric. In this situation, there is a vertex $x \in V(G \setminus H_1)$ connecting two vertices at distance 3 in H_1 . As shown in Figure 12, it implies the existence of a

³The characterization of [59] is composed of six forbidden isometric subgraphs, but the sixth one is actually 6-chordal.

$K_{2,3}$ -minor in G , thereby contradicting that it is outerplanar. Therefore, the claim is proved, and so, G is $\frac{1}{2}$ -hyperbolic if, and only if, it is chordal and it does not contain H_1 as a subgraph.

Being chordal can be checked in linear-time [50]. Furthermore, when G is chordal, all its induced cycles have length three, hence it contains H_1 as a subgraph if and only if there are two adjacent vertices of degree 3 in its weak dual (see Figure 13 for an illustration). Overall, since the weak dual can be computed in linear time, deciding whether a chordal outerplanar graph G is $1/2$ -hyperbolic can be done in linear time. \square

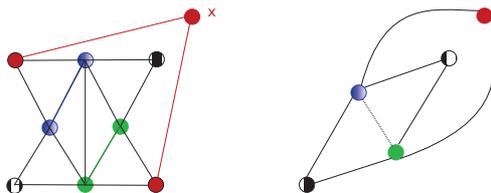


Figure 12: Existence of a $K_{2,3}$ -minor in G .

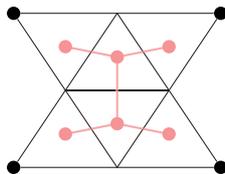


Figure 13: The forbidden subgraph of Figure 11b, and its characterization in the weak dual.

6.2 Substitute graphs of cycles

As we constrain ourselves to outerplanar graphs, recall that the atoms are exactly the induced cycles of the graph. Clearly, a clique-separator contained in a cycle is either a cut-vertex or an edge-separator. Since we never add pendant vertices with our substitution method (cf. Section 5.2.2), we never add a simplicial vertex in the first case, and in the second case we might only add a single vertex which has to be adjacent to both ends of the edge-separator. Substitute graphs of cycles thus fall into the following subclass of outerplanar graphs:

Definition 24. *A biconnected outerplanar graph is called a sunshine graph if it can be obtained from a cycle C by adding, for every edge of C , at most one simplicial vertex that is adjacent to that edge.*

Trivially, all cycles are sunshine graphs. Two other examples of sunshine graphs are given in Figure 14.

We can derive some useful properties of sunshine graphs from their definition. First is that vertices in the cycle C form a dominating set of the graph. Furthermore, the choice of C is unique, except for the particular case of the diamond graph (obtained from two triangles sharing an edge).

Finally, since every sunshine graph G is obtained by adding simplicial vertices to a cycle, it has at most one induced cycle of length at least four, and if it exists this cycle must be C . Thus, we have by Theorem 12 that $\delta(C) \leq \delta(G) \leq \delta(C) + 1$. This difference can be decreased by $\frac{1}{2}$ as follows.

Lemma 25. *Let G be a sunshine graph, and C be a dominating cycle of G . Then we have:*

$$\delta(C) \leq \delta(G) \leq \delta(C) + \frac{1}{2}.$$

Proof. By Theorem 12, we have $\delta(C) \leq \delta(G) \leq \delta(C) + 1$. So, our aim is to prove that no 4-tuple of G has a hyperbolicity greater than $\delta(C) + \frac{1}{2}$. By contradiction, let a, b, c, d be such that $\delta(a, b, c, d) = \delta(C) + 1$.

We arbitrarily orient the cycle C . For every $u \in \{a, b, c, d\} \setminus C$, we denote by $e_u = \{h_u, t_u\}$ the edge of C induced by its neighbors, where h_u denotes the head of the edge w.r.t. the orientation. Observe that for every $u, v \in \{a, b, c, d\} \setminus C$, we have $d(u, v) = 2 + \min\{d(h_u, t_v), d(h_v, t_u)\} = 1 + d(h_u, h_v) = 1 + d(t_u, t_v)$.

We then claim that there is exactly one vertex among a, b, c, d which belongs to the cycle C . We prove the claim with a case-by-case analysis.

- Since we assume $\delta(a, b, c, d) > \delta(C)$, not all of a, b, c, d belong to the cycle;
- Furthermore, suppose for the sake of contradiction that $a \notin C$ but $b, c, d \in C$. Let $X = N(a)$, $A = N[a]$, and $B = V(G) \setminus a$. The 4-tuple is a $(a|b_1, b_2, b_3)$ 4-tuple, and so, by Lemma 5, there exists $x \in N(a)$ such that $\delta(a, b, c, d) \leq \delta(x, b, c, d) + 1/2$. In this situation, $x, b, c, d \in C$, hence $\delta(a, b, c, d) \leq \delta(C) + 1/2$, thereby contradicting our assumption that $\delta(a, b, c, d) = \delta(C) + 1$.
- In the same way, suppose for the sake of contradiction that $a, d \notin C$ but $b, c \in C$. Let $X_a = N(a)$ and $X_d = N(d)$. These two clique-separators satisfy the conditions of Lemma 10, so, there exist $x_a \in X_a$, $x_d \in X_d$ such that $\delta(a, b, c, d) \leq \delta(x_a, b, c, x_d) + 1/2$. In this situation, $x_a, b, c, x_d \in C$, hence $\delta(a, b, c, d) \leq \delta(C) + 1/2$, thereby contradicting our assumption that $\delta(a, b, c, d) = \delta(C) + 1$.
- So, there are at least three vertices of a, b, c, d that are not in C . Finally, suppose for the sake of contradiction that $a, b, c, d \notin C$. In this situation, for every $u, v \in \{a, b, c, d\}$, we have $d(u, v) = 1 + d(h_u, h_v)$. Therefore by the 4-point condition we have $\delta(a, b, c, d) = \delta(h_a, h_b, h_c, h_d) \leq \delta(C)$. The latter contradicts our assumption that $\delta(a, b, c, d) = \delta(C) + 1$.

As a result, there is exactly one vertex of the 4-tuple that is in C , which proves the claim. Assume w.l.o.g. that $a \in C$. In such a case, for every $u \in \{b, c, d\}$ we have $d(a, h_u) \leq d(a, u) \leq d(a, h_u) + 1$.

Let S_1, S_2, S_3 satisfy $\{S_1, S_2, S_3\} = \{d(a, b) + d(c, d), d(a, c) + d(b, d), d(a, d) + d(b, c)\}$ and $S_1 \geq S_2 \geq S_3$. Accordingly, let u_1, u_2, u_3 satisfy $\{u_1, u_2, u_3\} = \{b, c, d\}$ and for every i , $S_i = d(a, u_i) + d(u_j, u_k)$ where $\{j, k\} = \{1, 2, 3\} \setminus i$. Finally, let $S'_i = d(a, h_{u_i}) + d(h_{u_j}, h_{u_k})$.

We have $d(a, h_{u_i}) \leq d(a, u_i) \leq d(a, h_{u_i}) + 1$ and $d(u_j, u_k) = d(h_{u_j}, h_{u_k}) + 1$. Consequently, we have $S'_i + 1 \leq S_i \leq S'_i + 2$ for every i . By the 4-point condition, it implies $\delta(a, h_b, h_c, h_d) \leq \delta(a, b, c, d) + 1/2$. Since in addition $a, h_b, h_c, h_d \in C$, we have $\delta(a, b, c, d) \leq \delta(C) + 1/2$, thereby contradicting our assumption that $\delta(a, b, c, d) = \delta(C) + 1$. Altogether, this proves $\delta(G) \leq \delta(C) + 1/2$. \square

We now present a characterization for the hyperbolicity of sunshine graphs, from which it can be easily derived a linear-time algorithm in order to compute it.

Lemma 26. *Let G be a sunshine graph, and C be a dominating cycle for G of length $4p + \varepsilon \geq 3$, with $p \geq 0$ and $\varepsilon \in \{0, 1, 2, 3\}$. Assuming $G \setminus C$ is nonempty we have:*

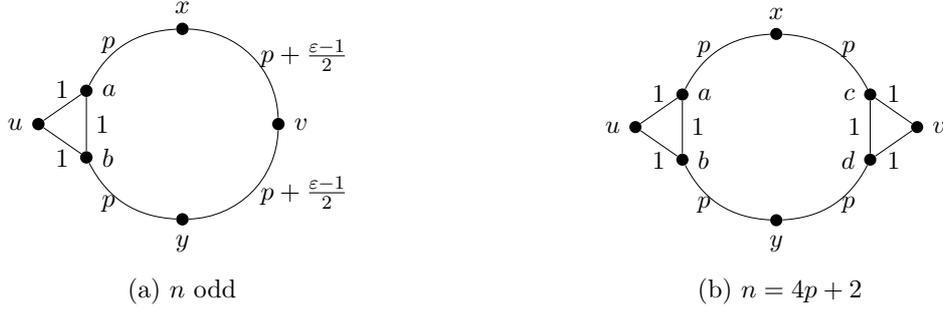


Figure 14: Substitute graphs of the atoms of an outerplanar graph.

- if ε is odd, then $\delta(G) = \delta(C) + \frac{1}{2}$;
- if $\varepsilon = 2$, then $\delta(G) = \delta(C) + \frac{1}{2}$ if, and only if, there is a diametral pair made of two simplicial vertices not in C (otherwise, $\delta(G) = \delta(C)$);
- finally, if $\varepsilon = 0$, then $\delta(G) = \delta(C)$.

Proof. Recall that by the previous Lemma 26, we have $\delta(G) \leq \delta(C) + \frac{1}{2}$. Thus we only focus on finding 4-tuples u, v, x, y of hyperbolicity (at least) this value, and we choose one, if any, maximizing $|C \cap \{u, x, v, y\}|$. In what follows, write $S_1 = d(u, v) + d(x, y)$, $S_2 = d(u, x) + d(v, y)$ and $S_3 = d(u, y) + d(v, x)$. We will assume in addition that $S_1 \geq S_2 \geq S_3$.

Case ε odd Equivalently, we have $\varepsilon \in \{1, 3\}$. In such a case, we have $\delta(C) = p + \frac{\min\{0, \varepsilon - 2\}}{2}$ by Lemma 21. Figure 14a exhibits a 4-tuple u, v, x, y satisfying:

$$\begin{aligned} S_1 &= \left(2p + \frac{\varepsilon + 1}{2}\right) + (2p + \min\{1, \varepsilon - 1\}) = 4p + \frac{\varepsilon + 1}{2} + \min\{1, \varepsilon - 1\}; \\ S_2 &= (p + 1) + \left(p + \frac{\varepsilon - 1}{2}\right) = 2p + \frac{\varepsilon + 1}{2}; \\ S_3 &= S_2. \end{aligned}$$

Hence, this 4-tuple has hyperbolicity $p + \frac{\min\{1, \varepsilon - 1\}}{2} = \delta(C) + \frac{1}{2}$.

Case $\varepsilon = 2$ In such a case, we have $\delta(C) = p$ by Lemma 21. We assume w.l.o.g. that $u \notin C$, and we claim that it implies $v \notin C$. Indeed, by the metric property of Lemma 9, and noticing that $S_1 \geq S_2 \geq S_3$, the vertex v has to be at equal distance l of both neighbors of u , as otherwise u could be replaced with one of its two neighbors, contradicting the maximality of $|C \cap \{u, x, v, y\}|$. Hence $v \in C$ is impossible, as it would yield the length of C is $2l + 1 = 4p + 2$. It thus follows that $v \notin C$, and the length of C is in fact $2(l - 1) + 2 = 2l$, yielding $l = 2p + 1$.

Conversely, assume that there exist two simplicial vertices u, v that are diametrically opposed in G . We choose the 4-tuple u, x, v, y as in Figure 14b, and it satisfies:

$$\begin{aligned} S_1 &= (2p + 2) + (2p + 1) = 4p + 3; \\ S_2 &= 2(p + 1) = 2p + 2; \\ S_3 &= S_2. \end{aligned}$$

So, we have $\delta(u, v, x, y) = p + \frac{1}{2} = \delta(C) + \frac{1}{2}$.

Case $\varepsilon = 0$ Another application of Lemma 21 yields $\delta(C) = p$. Assuming $u \notin C$, we deduce as for the previous case that $v \notin C$, and v is at equal distance $l = 2p$ from both neighbors of u . Thus, C is partitioned by the neighborhoods of u and v in two paths of length $l - 1 = 2p - 1$, that is in the same way as in Figure 14b. Furthermore, since the diameter of C is $2p$, those paths are geodesics of the cycle.

We recall that on the way to prove Lemma 25, we showed that $u, v, x, y \notin C$ implies $\delta(u, v, x, y) \leq \delta(C)$. Since we assume $\delta(u, v, x, y) > \delta(C)$, it implies the existence of one vertex $z \in \{x, y\}$ among the 4-tuple that must be in C . Furthermore, in this situation we obtain by considering the geodesic containing z that $d(u, z) + d(v, z) = 2 + (l - 1) = 2p + 1$. In particular, we have $\min\{d(u, z), d(v, z)\} \leq p$. The latter contradicts the assumption that $\delta(u, v, x, y) > \delta(C)$, since we have by [51] that $\delta(u, v, x, y) \leq \min\{d(u, z), d(v, z)\}$. Altogether, we always have $\delta(G) = \delta(C)$ if $\varepsilon = 0$. \square

6.3 Applying the substitution method in linear-time

Given an outerplanar graph G , we recall that we aim at computing $\delta(G)$ from the substitute of its atoms. From Lemma 26, the hyperbolicity of the substitutes can be computed in linear time. So, we are left to prove that the atom's substitutes can also be computed in linear time.

Lemma 27. *Let G be an outerplanar biconnected graph. The substitute graphs of the atoms of G can be computed in linear-time.*

Proof. We construct the weak dual T_G of G from an outerplanar embedding, that is linear-time computable. Let C_1, \dots, C_l be the atoms of G . We root T_G on an atom C_1 , which is an induced cycle. Then, we claim that the following algorithm for computing the atom's substitutes is correct.

- For every i , we initialize C_i^* with C_i .
- We start a depth-first search from the root, and so obtain a postordering of the nodes of T_G . Then, we visit the atoms following this ordering, and we proceed as follows. For every C_i , we name $e_{ij} = C_i \cap C_j$ an edge shared with a child in the rooted tree. If there is a vertex of C_j^* that is at equal distance to both ends of e_{ij} then we add in C_i^* a new simplicial vertex that is adjacent to e_{ij} . That is, we add such new vertex if either C_j is odd, or there is a simplicial vertex of $C_j^* \setminus C_j$ that is adjacent to the edge opposed to e_{ij} .
- Finally, we start a breadth-first search from the root and for every visited atom $C_i \neq C_1$, we consider its parent atom, denoted by C_k , naming $e_{i,k}$ the edge-separator that it shares with it. As before, we add in C_i^* a simplicial vertex whose neighborhood is $e_{i,k}$ if, and only if, either the length of C_k is odd, or there is a simplicial vertex in $C_k^* \setminus C_k$ whose neighborhood is the edge diametrically opposed to $e_{i,k}$ in the atom C_k .

This algorithm runs in linear time. Furthermore, we note that an atom tree can be obtained from $(T_G; C_1)$ (as defined in Definition 15) by disconnecting at every step an atom that is a leaf of T_G until the clique-decomposition is obtained. Following this atom tree, the atom's substitutes so obtained are isomorphic to the output $C_1^*, C_2^*, \dots, C_l^*$ of the above algorithm. In particular, for every atom, we use the depth-first search to compute the simplicial vertices resulting from the disconnection of its sons, whereas the breadth-first search is used to compute the single vertex resulting from its own disconnection, if any. Hence, the above algorithm for computing the atom's

substitutes is correct, and so, the resulting C_1^*, \dots, C_l^* are the substitute graphs of the atoms of G . \square

Figure 15 shows the substitute graphs resulting from the application of the substitution method to a biconnected outerplanar graph.

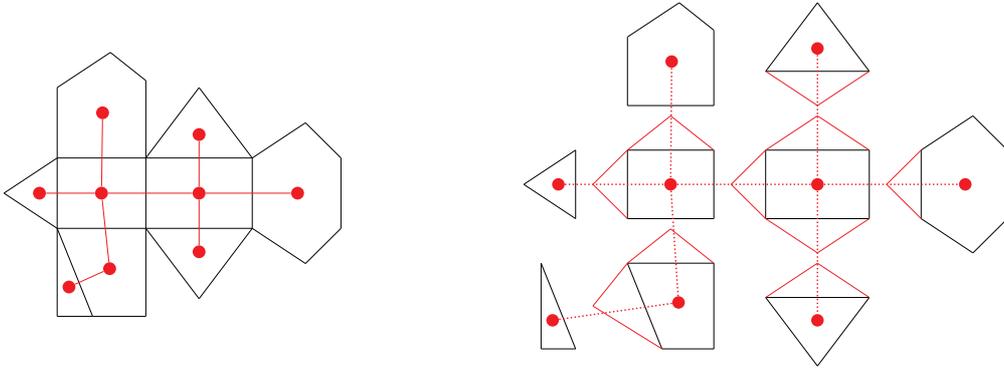


Figure 15: An application of the substitution method to an outerplanar graph.

We finally conclude with the following theorem.

Theorem 28. *The hyperbolicity of a given connected outerplanar graph G is computable in linear time.*

Proof. We can safely assume G to be biconnected by [55]. By [5, 37], G is 0-hyperbolic if, and only if, G is a clique. If it is not, then by Proposition 23, we can check whether it is $\frac{1}{2}$ -hyperbolic in linear time.

From now on, assume $\delta(G) \geq 1$. By Lemma 27, we can compute the substitute graphs of the atoms of G in linear time. We can thus conclude by Lemma 14 (*i.e.* the correctness of our substitution method), as these substitute graphs are sunshine graphs and their hyperbolicity is linear-time computable by Lemma 26. \square

7 Experimental evaluation

Before concluding the paper, we shall apply the substitution method of Section 5 to some real-life graphs with a large number of vertices. We report in this section on experiments performed with our substitution methodology on the graphs of five collaboration networks. This way, we aim to evaluate the computation time of the substitutes on some empirical graphs, and to better understand the factors impacting their size (compared with the upper-bound of Section 5.2.2).

The section is subdivided as follows. In Section 7.1, we present the graphs from the dataset and we motivate our choice to test the method on these graphs. We report on the reduction on the size of the subgraphs (biconnected components, atoms and substitutes) in Section 7.2. In spite of strong similarities between the graphs from the dataset, the results obtained vary from one graph to the other. So, we conduct a deeper analysis of their clique-decomposition in Section 7.3, reporting on the structure of their atom tree and on the size of the clique-separators, in order to justify the variations in the results. We complete our experiments with a numerical analysis of the time needed

for computing the hyperbolicity of these graphs, with and without the clique-decomposition and the substitute decomposition of Section 5. On the way, we report on the hyperbolicity of all graphs in the dataset (Section 7.4).

7.1 Datasets

We apply the algorithm presented in Section 5 to the collaboration networks of five different scientific communities [45], namely:

- **ca-AstroPh**, for the astrophysics community;
- **ca-CondMat**, for the condensed matter physics community;
- **ca-GrQc**, for the general relativity and quantum cosmology community;
- **ca-HepPh**, for the high energy physics-phenomenology community;
- and **ca-HepTh**, for the high energy physics-theory community.

In the **ca-*** graphs, nodes represent scientists and edges represent collaborations (i.e., co-authoring a paper). These graphs are interesting to analyze the behavior of our algorithm, and the size of their substitute graphs, because they have many cliques of various sizes. Indeed, a paper co-authored by k scientists induces a clique of size k in the graph. Furthermore, the number of co-authors per papers varies from one community to another. As noted in Section 5.2.2, there are $\mathcal{O}(2^k)$ vertices newly added in the substitutes for any clique-separator of size k . Therefore, we expect to observe different results in terms of the size of the substitutes, despite the graphs from the dataset share many properties (see [45]).

7.2 Empirical results

We modified the clique-decomposition algorithm of [8] to implement the substitution method that we presented in Section 5. We used it here to compute, for every graph, the substitute of each atom of the decomposition.

Below, we report on the size of the substitutes. We compare it with the size of the atoms and the biconnected components (see Figure 16 and Table 1). This preliminary analysis also explains why we can ignore almost all substitutes in the computation of hyperbolicity (precisely, all but one substitute), that will further reduce the time of computation.

Decomposition into biconnected components We observed that all of the five graphs are composed of one largest biconnected component, that we call LBC, that includes from 50% to 84.85% of all the vertices. This can be observed from the cumulative distribution of the size of the biconnected components in Figure 16a. The cumulative number of components is given as a percentage of the total number of biconnected components, and the size of the components as a percentage of the total number of vertices in the graph. We noticed that all the biconnected components but the LBC are small: only covering at most 1% of the vertices.

Clearly, the smallest biconnected components can be safely ignored for the computation of hyperbolicity, provided that their diameter is smaller than two times the hyperbolicity of the LBC, which is always the case for these graphs (see [20]). Thus, we now focus on the clique-decomposition of the LBC, and on its resulting substitute graphs.

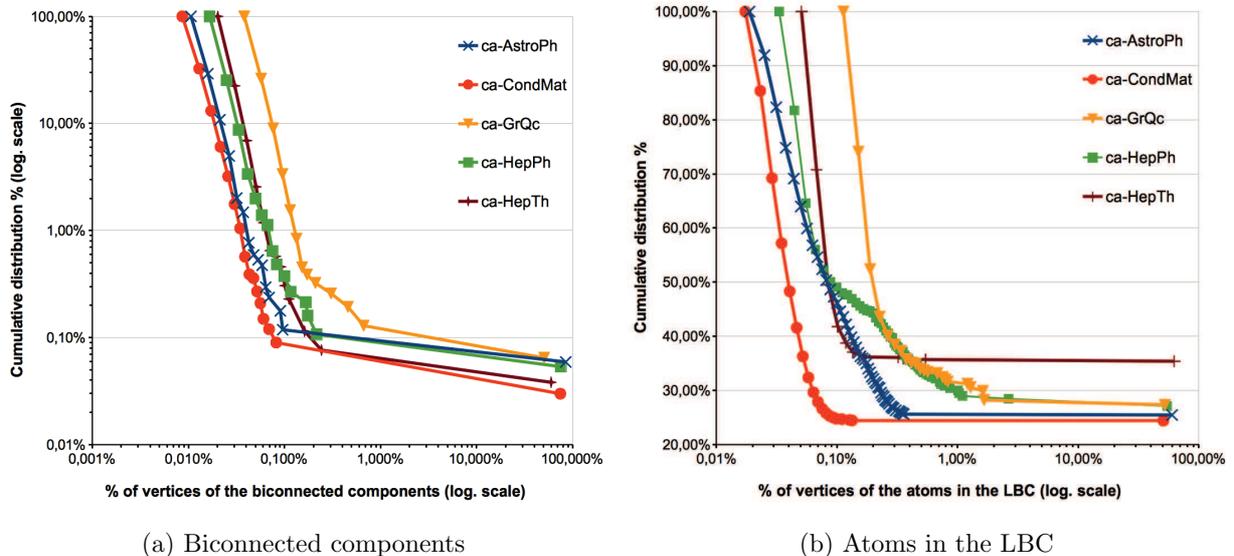


Figure 16: Cumulative distributions of the size of the biconnected components (Figure 16a) and of the atoms in the LBC of each graph (Figure 16b).

Clique-decomposition We plotted in Figure 16b the cumulative distribution of the size of the atoms of the LBC. The cumulative number of atoms is given as a percentage of the total number of atoms and the size of the atoms as a percentage of the total number of vertices in the LBC. Again, for all of the graphs, we observed one largest atom, that we call the LA, that includes from 50% to 60% of all the vertices, and all the other atoms only represent a small fraction of the overall vertices. In the worst case (ca-HepPh), all the atoms but the LA solely cover 2.65% of the vertices of the graph.

Moreover, like for the smallest biconnected components and as reported in [20], the substitute graphs of the smallest atoms can be safely ignored for the computation of hyperbolicity. As a result, the only component of the graphs to deal with for computing their hyperbolicity is the substitute graph of the LA. We will denote it by LS in what follows.

Size of the substitute graphs As explained in Section 5, the size of the LS depends on both the initial size of the LA and the number of added simplicial vertices. We have reported in Table 1 the original size n of each graph, the size n_B of its LBC, the size n_{LA} of the LA, and the size n_{LS} of the largest substitute. We have then computed the percentage R_{LA} of vertices that have been removed from the LBC to obtain the LA, that is $R_{LA} = \frac{n_B - n_{LA}}{n_B}$. We observe a significant reduction rate R_{LA} , varying from 37.40% to 49.22%. We have also computed the reduction rate R_{LS} of the LS with respect to the LBC, that is $R_{LS} = \frac{n_B - n_{LS}}{n_B}$. We observe that this reduction rate falls between 11.22% and 20.84%. It indicates that in spite of the simplification rules presented in Section 5.2.2, the substitution method adds many simplicial vertices to the LA when constructing the LS.

We reported in Table 1 as *Cost* the percentage of vertices in the LBC representing the addition of new simplicial vertices.

We first analyze the ca-CondMat graph. This graph has the largest reduction rates R_{LS} and

Instance name	n	n_B	n_{LA}	n_{LS}	R_{LA}	R_{LS}	Cost	Time (in sec.)
ca-CondMat	23 133	17 234	8 751	13 643	49.22%	20.84%	28.39%	672
ca-GrQc	5 242	2 651	1 386	2 107	47.72%	20.52%	27.20%	5
ca-HepPh	12 008	9 025	4 925	7 170	45.43%	20.55%	24.88%	167
ca-AstroPh	18 772	15 929	9 561	13 407	39.98%	15.83%	24.14%	679
ca-HepTh	9 877	5 898	3 692	5 236	37.40%	11.22%	26.18%	53

Table 1: Characteristics of the collaboration networks. The size of the graph is given as n , the size of the LBC as n_b , the size of the LA as n_{LA} and the size of the LS as n_{LS} . The percentage of vertices removed from the LBC to obtain the LA is given as R_{LA} , the reduction rate is $R_{LS} = \frac{n_B - n_{LS}}{n_B}$, and the percentage of vertices in the LBC, representing the addition of simplicial vertices, is given as $Cost$. Finally, the computation time of the substitution method, denoted by $Time$, is given in seconds.

R_{LA} from the dataset. However, despite a R_{LA} of 49.22%, it has almost the same reduction rate R_{LS} as **ca-HepPh** and **ca-GrQc** — ranging from 20.55% to 20.84%. This is the consequence of more simplicial vertices added with our substitution methodology. The new simplicial vertices represent 28.39% of the size of its LBC, whereas for the **ca-HepPh** graph it goes up to only 24.88%.

A similar behavior is observed between **ca-AstoPh** and **ca-CondMat**: even though their R_{LA} differ on 9.25%, the difference of their reduction rate R_{LS} finally falls to 5%. This results from the addition of 4.24% less simplicial vertices in **ca-AstoPh** than in **ca-CondMat**. As an extremal case, the R_{LA} and R_{LS} of the **ca-HepPh** graph are respectively bigger and smaller than the R_{LA} and R_{LS} of **ca-GrQc**.

We thus conclude that the impact of n_{LA} and of the number of new simplicial vertices on the final size n_{LS} differs greatly depending on the graph.

7.3 Decomposition analysis

Having noticed the heterogeneous results of our empirical section, we are now analyzing in more details the properties causing the asymmetry between the various **ca-*** graphs. To do so, we report on the structure of the intersection graph of the atoms (sometimes call the atom graph) and on the size of the clique-separators.

We support through our experiments that most clique-separators are small (with no more than two or three nodes), and they are responsible for the largest part of new simplicial vertices. One plausible explanation for small-size separators are the student interns, publishing one paper with their supervisors before changing their lab or leaving the community.

Clique-decomposition We first analyzed the neighborhood of the LA in the *atom graph*, as it is defined in [7]. That is, we consider the set of atoms $\mathcal{A}_{LA} = \{A_1, \dots, A_l\}$ that intersect the LA, naming $\mathcal{X}_{LA} = \{X_1 = A_1 \cap LA, \dots, X_l = A_l \cap LA\}$ the clique-separators at their intersection. We emphasize that there might be other atoms in the graph than the LA and those in \mathcal{A}_{LA} . But such atoms, if any, do not overlap the LA.

We plotted in Figure 17a the cumulative distribution of the size of the clique-separators in the

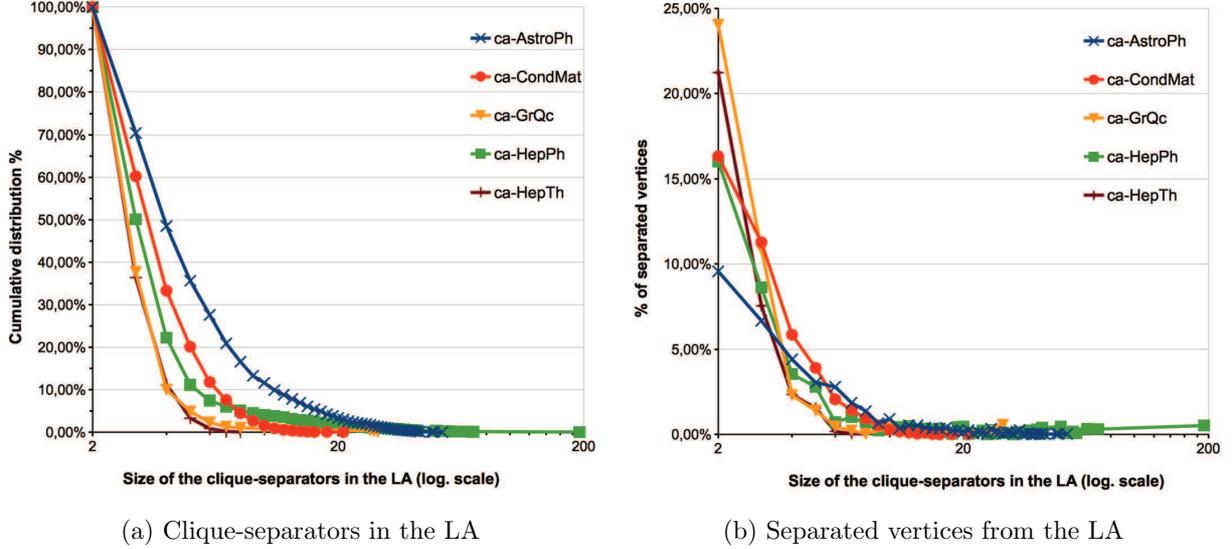


Figure 17: Cumulative distribution of the size of the clique-separators in the LA (Figure 17a) and percentage of separated vertices as a function of the size of the clique-separators in the LA (Figure 17b).

LA as a percentage of the total number of clique-separators. By doing so, we observed smaller clique-separators for the `ca-HepTh` and `ca-CondMat` graphs, with a maximum size of 8 and 21, respectively, than for the three other graphs `ca-GrQc`, `ca-AstroPh` and `ca-HepPh`, having clique-separators of maximum size 42, 53 and 192, respectively. Also, we reported in Table 2 that the ratio $R_{|\mathcal{X}_{LA}|} = |\mathcal{X}_{LA}|/n_{LA}$ varies from 0.39 for `ca-AstroPh` to 0.54 for `ca-CondMat`. To sum up, there are more clique-separators in `ca-CondMat` than in `ca-AstroPh`, but there are larger clique-separators in `ca-AstroPh` than in `ca-CondMat`.

Recall that our substitution methodology never adds more simplicial vertices than the number of nodes disconnected by the clique-separator. So, to complete our measurements, we related the size of clique-separators with the proportion of vertices that are disconnected by them from the LA. We reported in Table 2 as $\alpha_1 = n_B - n_{LA}$ the total number of vertices separated from the LA in the LBC, and as $\alpha_2 = |\bigcup_{i=1}^l A_i \setminus X_i|$ the number of vertices of $LBC \setminus LA$ present in an atom of \mathcal{A}_{LA} . Finally, we computed the fraction $\Delta_1 = \frac{\alpha_1 - \alpha_2}{n_B}$, quantifying the percentage of vertices that are neither contained into the LA, nor in any of the atoms in \mathcal{A}_{LA} . We reported as $\Delta_2 = R_{LA} - \Delta_1$ the fraction of vertices in some atom of \mathcal{A}_{LA} , hence those that are *directly* separated from the LA.

Our results put in evidence that most of the vertices are either contained in the LA, or in some other atom intersecting the LA. Other vertices comprise around 2.88% and 7.03% of the overall vertices. Moreover, as shown with Figure 17b, where we plotted the percentage of separated vertices as a function of the size of clique-separators, smaller clique-separators of size ≤ 5 are responsible for a significant part (w.r.t. Δ_2) of the vertices disconnected from the LA in `ca-CondMat` (37.34% of vertices over 49.22%), whereas in `ca-AstroPh` they solely disconnect 23.67% over 39.98% of vertices. This difference is not balanced with clique-separators of larger size, even though these ones disconnect 13.43% of vertices in `ca-AstroPh`, while only 5.67% in `ca-CondMat`. Comparing `ca-CondMat` with `ca-HepPh` does not change the picture. In contrast, for the graphs `ca-GrQc`

and **ca-HepTh**, we notice that 6.71% and 4.91% more vertices, respectively, than in **CondMat**, are disconnected by edge-separators. But the rest of the clique-separators only disconnect 16.67% and 11.70% of the vertices from the LA, respectively, whereas 26.70% of them are separated from the LA in **ca-CondMat**. Therefore, most of the difference for the final size of the substitute graph LS comes from the number of vertices that are disconnected by clique-separators of *small* size.

Instance name	α_1	α_2	$ \mathcal{X}_{LA} $	$R_{\mathcal{X}_{LA}}$	$\Delta_1\%$	$\Delta_2\%$
ca-CondMat	8 483	7 413	4 702	0.54	6.21%	43.01%
ca-GrQc	1 265	1 079	698	0.5	7.03%	40.69%
ca-HepPh	4 100	3 727	2 166	0.44	4.13%	41.3%
ca-AstroPh	6 368	5 910	3 715	0.39	2.88%	37.1%
ca-HepTh	2 206	1 942	1 506	0.41	4.47%	32.93%

Table 2: Distribution of clique-minimal separators, and of the vertices disconnected from the LA. The total number of vertices separated from the LA in the LBC is given as $\alpha_1 = n_B - n_{LA}$, and the number of disconnected vertices being present in the subset of neighboring atoms \mathcal{A}_{LA} as $\alpha_2 = |V(\mathcal{A}_{LA} \setminus \mathcal{X}_{LA})|$.

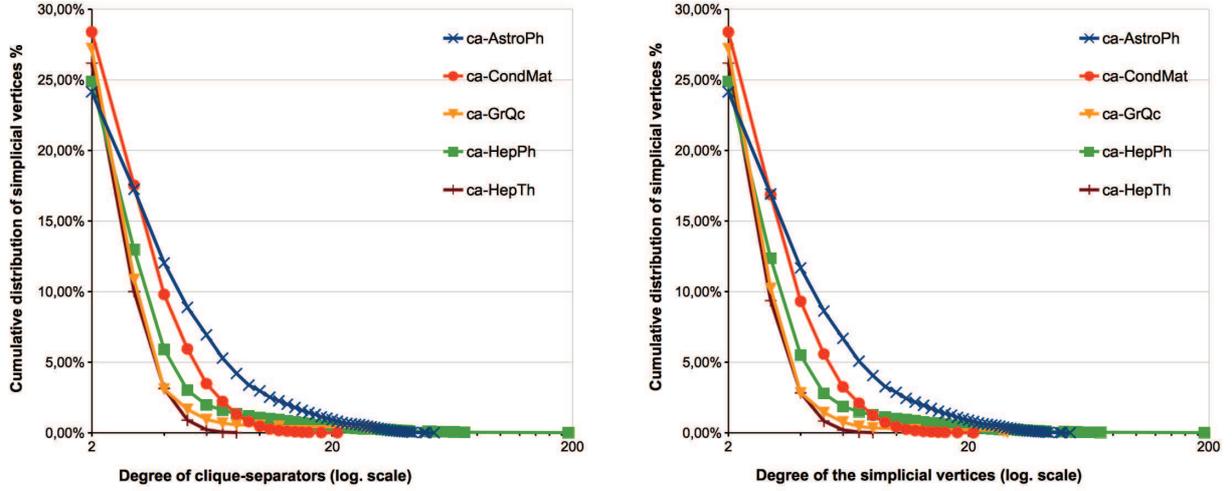
Also, the number of clique-minimal separators in the LA is given as $|\mathcal{X}_{LA}|$.

We quantify the percentage of vertices that are neither contained into the LA nor in any of the atoms in \mathcal{A}_{LA} as $\Delta_1 = \frac{\alpha_1 - \alpha_2}{n_B}$; the fraction of vertices in some atom of \mathcal{A}_{LA} that are *directly* separated from the LA is equal to $\Delta_2 = R_{LA} - \Delta_1$.

Substitute construction Recall that we assume that the largest number of simplicial vertices are connected to the smallest clique-separators. In order to validate the assumption, we plotted in Figure 18a the cumulative number of simplicial vertices connected to the LA, normalized by the size of the LBC, as a function of the size of the clique-separators. In particular, note that for each graph, such a summation is equal to the value given as **Cost** in Table 1. By looking only at clique-separators of size two and three, the proportions of simplicial vertices for the graphs **ca-CondMat**, **ca-GrQc**, **ca-HepPh**, **ca-AstroPh** and **ca-HepTh** respectively, represent 65.49%, 88.63%, 76.26%, 50.16% and 88.02% respectively, of the total number of simplicial vertices connected to the LA. Thus it highlights the importance of clique-separators of small size, to which a large proportion of simplicial vertices are connected to.

Let us also remark by comparing Figure 18a to Figure 18b that almost all simplicial vertices have *same degree*. In the worst case (**ca-GrQc**), there are no more than 0.75% of the simplicial vertices whose degree differs from the others. Most of these simplicial vertices have degree two. Hence, the final proportion of simplicial vertices, given in Table 1 as **Cost**, mostly depends on the size distribution of the clique-separators in the graphs. Also, since there is a worst-case variation of only 4.25% in the proportion of simplicial vertices in our graphs – that is reached with **ca-CondMat** and **ca-AstroPh** –, that allows us to make relative comparisons between the graphs from the dataset. Especially we are interested in comparing the proportion of simplicial vertices of small degree (less than four). Such a proportion represents, for **ca-CondMat**, **ca-GrQc**, **ca-HepPh**, **ca-AstroPh** and **ca-HepTh** respectively, a percentage of 18.59%, 24.1%, 18.97%, 12.11% and 23.04% respectively, of the simplicial vertices in total. To sum up:

- when comparing **ca-AstroPh** to **ca-CondMat**: even if the former has 2.23% more simplicial vertices with degree at least three, this is compensated by its 6.48% less simplicial vertices of



(a) Simplicial vertices connected to the LA as a function of the size of the clique-separators

(b) Degree distribution of the simplicial vertices connected to the LA

Figure 18: Cumulative number of simplicial vertices connected to the LA normalized by the size of the LBC as a function of the size of the clique-separators to which they are connected (Figure 18a), cumulative degree distribution of the simplicial vertices connected to the LA normalized by the size of the LBC (Figure 18b)

degree at most four, which results in overall to 4.25% less simplicial vertices in *ca-AstroPh* than in *ca-CondMat*. The same happens when comparing *ca-AstroPh* to the remaining graphs. Indeed, the lower number of simplicial vertices with degree at most four *always* compensates its larger number of simplicial vertices of higher degree.

- when comparing *ca-CondMat* to *ca-GrQc* and *ca-HepTh*: the two latter graphs respectively have 5.51% and 4.45% more simplicial vertices with degree at most four. However, they respectively have 6.7% and 6.66% less simplicial vertices with degree at least three. As a result, there are 1.19% less simplicial vertices in *ca-GrQc*, and 2.21% less simplicial vertices in *ca-HepTh*, respectively, than in *ca-CondMat*.
- finally, when comparing *ca-CondMat* to *ca-HepPh*: we observe quite similar numbers of simplicial vertices of degree smaller than four. They respectively represent 18.59% and 18.97% of the simplicial vertices in total. Again, the main difference comes from the proportion of simplicial vertices with degree higher than three, with 5.91% more simplicial vertices in *ca-CondMat* than in *ca-HepPh*, resulting in 3.51% less vertices in *ca-HepPh*.

7.4 Computation times

In order to complete the empirical section, we present in Table 3 the computation times of the hyperbolicity on the LS and on the LBC of the *ca-** graphs. Of course, we expect the computation time to decrease proportionally to the size of the graphs. However, we use the algorithm that we proposed in [21] for computing the hyperbolicity, and so, the computation time may be impacted by other factors.

On the way, we also give in Table 3 the values of the hyperbolicity we obtained and the computation time of the LS using the algorithm given in Section 5. Interestingly, for all graphs from the dataset, the hyperbolicity of the graph always equals the hyperbolicity of its largest atom.

Instance name	n	n_B	n_{LA}	n_{LS}	δ	T_{LS}	$T_{\delta_{LS}}$	$T_{\delta_{LBC}}$	$R1$	$R2$
ca-GrQc	5 242	2 651	1 386	2 107	3.5	6s.	1s.	8.8s.	8.8	1.26
ca-HepTh	9 877	5 898	3 692	5 236	4	52s.	0.3s.	2.4s.	8	0.046
ca-HepPh	12 008	9 025	4 925	7 170	3	162s.	50s.	677s.	13.5	3.19
ca-AstroPh	18 772	15 929	9 561	13 407	3	762s.	22s.	202s.	9.2	0.26
ca-CondMat	23 133	17 234	8 751	13 643	3.5	1180s.	101s.	2498s.	24.7	1.95

Table 3: Hyperbolicity and computation times on the **ca-*** graphs. The size of the graph is given as n , the size of the LBC as n_B , the size of the LA as n_{LA} and the size of the LS as n_{LS} . The value of the hyperbolicity computed on the LS is given as δ . The computation time in second of the largest substitute is given as T_{LS} . The computation times in second of the hyperbolicity, on the LS and on the LBC respectively, is given as $T_{\delta_{LS}}$ and $T_{\delta_{LBC}}$ respectively. Finally, the ratio $T_{\delta_{LS}}/T_{\delta_{LBC}}$ is given as $R1$ and $(T_{LS} + T_{\delta_{LS}})/T_{\delta_{LBC}}$ is given as $R2$.

We observe that the hyperbolicity can be computed from 8 to 24 times faster on the LS than on the LBC. However, computing the hyperbolicity on the LS also comes at the cost of the time to construct this graph. By combining the two, one improves the time of computation by a smaller factor between 1.26 and 3.19 (for the graphs **ca-GrQc**, **ca-HepPh** and **ca-CondMat**). In some cases (**ca-AstroPh** and **ca-HepTh**), our method even increases the time of computation. However, these cases happen only for small graphs where the hyperbolicity can be computed in a few seconds. For larger graphs, up to 50 000 nodes, we have been able to reduce the times by two, saving from hours to days of computation.

8 Conclusion

We proved a tight relationship between the hyperbolicity of a given graph and the maximum hyperbolicity from its atoms. This gives a new proof that chordal graphs (and other related graph classes such as 2-chordal graphs [46, 47], nearly chordal graphs [15] or quasi-triangulated graph [52]) have a bounded hyperbolicity [16]. Our results also cover some class with unbounded hyperbolicity, namely the outerplanar graphs, for which we give a complete characterization of their hyperbolicity. This extends to a linear-time algorithm for computing the hyperbolicity of these graphs. To the best of our knowledge, this is the first linear-time algorithm for computing the hyperbolicity in these graphs. We let open whether the same can be done for other classes of graphs. Especially, can it be taken advantage of the linear-time algorithm for outerplanar graphs in order to compute the hyperbolicity of planar graphs more efficiently ?

Furthermore, we deduced from our proofs a general substitution method, allowing us to modify the atoms at no extra-cost than the clique-decomposition. For graphs with hyperbolicity at least one, the maximum hyperbolicity from the resulting graphs is exactly the hyperbolicity of the graphs. However, the graphs to be considered may have a larger size than the atoms. Experiments suggest that the final size of the substitute graphs is mostly related with the number of clique-separators of small size, and the disconnections resulting from them. Part of our future work will consist in finding other graph decompositions which are applicable to the computation of the hyperbolicity.

References

- [1] M. Abu-Ata and F. F. Dragan. Metric tree-like structures in real-life networks: an empirical study. *Networks*, 67(1):49–69, 2016.
- [2] H. Alrasheed and F. F. Dragan. Core-periphery models for graphs based on their δ -hyperbolicity: An example using biological networks. In *CompleNet VI*, pages 65–77. 2015.
- [3] R. Anstee and M. Farber. On bridged graphs and cop-win graphs. *Journal of Combinatorial Theory, Series B*, 44(1):22–28, 1988.
- [4] B. Baker. Approximation algorithms for NP-complete problems on planar graphs. *Journal of the ACM (JACM)*, 41(1):153–180, 1994.
- [5] H.-J. Bandelt and H. Mulder. Distance-hereditary graphs. *Journal of Combinatorial Theory, Series B*, 41(2):182–208, 1986.
- [6] S. Bermudo, J. Rodríguez, J. Sigarreta, and J.-M. Vilaire. Gromov hyperbolic graphs. *Discrete Mathematics*, 313(15):1575–1585, 2013.
- [7] A. Berry, R. Pogorelcnik, and G. Simonet. Efficient clique decomposition of a graph into its atom graph. Technical Report RR-10-07, LIMOS, Aubière, France, Mar. 2010.
- [8] A. Berry, R. Pogorelcnik, and G. Simonet. An introduction to clique minimal separator decomposition. *Algorithms*, 3(2):197–215, 2010.
- [9] A. Berry, R. Pogorelcnik, and G. Simonet. Organizing the atoms of the clique separator decomposition into an atom tree. *Discrete Applied Mathematics*, 177:1 – 13, 2014.
- [10] H. L. Bodlaender. Discovering treewidth. In *31st Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)*, volume 3381 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2005.
- [11] M. Boguñá, F. Papadopoulos, and D. V. Krioukov. Sustaining the Internet with hyperbolic mapping. *Nature Communications*, 1(62):1–18, Oct. 2010.
- [12] J. A. Bondy and U. Murty. *Graph theory with applications*, volume 290. Macmillan London, 1976.
- [13] M. Borassi, D. Coudert, P. Crescenzi, and A. Marino. On computing the hyperbolicity of real-world graphs. In *23rd Annual European Symposium on Algorithms (ESA)*, volume 9294 of *Lecture Notes in Computer Science*, pages 215–226, Patras, Greece, Sept. 2015. Springer.
- [14] M. Borassi, P. Crescenzi, and M. Habib. Into the square: On the complexity of some quadratic-time solvable problems. *Electronic Notes in Theoretical Computer Science*, 322:51–67, 2016.
- [15] A. Brandstädt and C. T. Hoàng. On clique separators, nearly chordal graphs, and the maximum weight stable set problem. *Theoretical Computer Science*, 389(1):295–306, 2007.
- [16] G. Brinkmann, J. H. Koolen, and V. Moulton. On the hyperbolicity of chordal graphs. *Annals of Combinatorics*, 5(1):61–69, 2001.

- [17] W. Carballosa, D. Pestana, J. Rodríguez, and J. Sigarreta. Distortion of the hyperbolicity constant of a graph. *the Electronic Journal of Combinatorics*, 19(1):P67, 2012.
- [18] J. Chakerian and S. Holmes. Computational tools for evaluating phylogenetic and hierarchical clustering trees. *Journal of Computational and Graphical Statistics*, 21(3):581–599, 2012.
- [19] V. Chepoi, F. F. Dragan, B. Estellon, M. Habib, and Y. Vaxès. Diameters, centers, and approximating trees of delta-hyperbolic geodesic spaces and graphs. In *24th Symposium on Computational Geometry (SCG)*, pages 59–68. ACM, 2008.
- [20] N. Cohen, D. Coudert, and A. Lancin. Exact and approximate algorithms for computing the hyperbolicity of large-scale graphs. Rapport de recherche RR-8074, Inria, Sept. 2012.
- [21] N. Cohen, D. Coudert, and A. Lancin. On computing the gromov hyperbolicity. *Journal of Experimental Algorithmics (JEA)*, 20(1):1–6, 2015.
- [22] D. Coudert and G. Ducoffe. Recognition of C_4 -free and $1/2$ -hyperbolic graphs. *SIAM Journal on Discrete Mathematics*, 28(3):1601–1617, Sept. 2014.
- [23] W. H. Cunningham. Decomposition of directed graphs. *SIAM Journal on Algebraic Discrete Methods*, 3(2):214–228, 1982.
- [24] B. DasGupta, M. Karpinski, N. Mobasher, and F. Yahyanejad. Effect of gromov-hyperbolicity parameter on cuts and expansions in graphs and some algorithmic implications. Technical Report arXiv:1510.08779, ArXiv, 2015.
- [25] P. de La Harpe and E. Ghys. *Sur les groupes hyperboliques d’après Mikhael Gromov*, volume 83. Progress in Mathematics, 1990.
- [26] M. Didi Biha, B. Kaba, M.-J. Meurs, and E. SanJuan. Graph decomposition approaches for terminology graphs. In *MICAI 2007: Advances in Artificial Intelligence*, volume 4827 of *Lecture Notes in Computer Science*, pages 883–893. Springer, 2007.
- [27] R. Diestel. *Graph theory, graduate texts in mathematics, vol. 173*. Springer, Heidelberg, 1997.
- [28] Y. Dourisboure and C. Gavaille. Tree-decompositions with bags of small diameter. *Discrete Mathematics*, 307(16):2008–2029, 2007.
- [29] F. Dragan. Tree-like structures in graphs: A metric point of view. In *39th International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, volume 8165 of *Lecture Notes in Computer Science*, pages 1–4. Springer, 2013.
- [30] A. Dress, K. Huber, J. Koolen, V. Moulton, and A. Spillner. *Basic Phylogenetic Combinatorics*. Cambridge University Press, Cambridge, UK, Dec. 2011.
- [31] H. Fournier, A. Ismail, and A. Vigneron. Computing the gromov hyperbolicity of a discrete metric space. *Information Processing Letters*, 115(6):576–579, 2015.
- [32] J. Gagneur, R. Krause, T. Bouwmeester, and G. Casari. Modular decomposition of protein-protein interaction networks. *Genome Biology*, 5(8):R57, 2004.

- [33] T. Gallai. Transitiv orientierbare graphen. *Acta Mathematica Hungarica*, 18(1):25–66, 1967.
- [34] A. Goldman. Optimal center location in simple networks. *Transportation science*, 5(2):212–221, 1971.
- [35] M. Gromov. Hyperbolic groups. In S. Gersten, editor, *Essays in Group Theory*, volume 8 of *Mathematical Sciences Research Institute Publications*, pages 75–263. Springer, New York, 1987.
- [36] M. Habib, C. Paul, and L. Viennot. Partition refinement techniques: An interesting algorithmic tool kit. *International Journal of Foundations of Computer Science*, 10(02):147–170, 1999.
- [37] E. Howorka. On metric properties of certain clique graphs. *Journal of Combinatorial Theory, Series B*, 27(1):67–74, 1979.
- [38] E. A. Jonckheere and P. Lohsoonthorn. Geometry of network security. In *American Control Conference*, volume 2, pages 976–981, Boston, MA, USA, 2004. IEEE.
- [39] B. Kaba, N. Pinet, G. Lelandais, A. Sigayret, and A. Berry. Clustering gene expression data using graph separators. *In Silico Biology*, 7(4):433–452, 2007.
- [40] W. S. Kennedy, O. Narayan, and I. Saniee. On the hyperbolicity of large-scale networks. Technical Report arXiv:1307.0031, ArXiv, 2013.
- [41] J. H. Koolen and V. Moulton. Hyperbolic bridged graphs. *European Journal of Combinatorics*, 23(6):683–699, 2002.
- [42] R. Krauthgamer and J. Lee. Algorithms on negatively curved spaces. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 119–132. IEEE, 2006.
- [43] F. Le Gall. Faster algorithms for rectangular matrix multiplication. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 514–523, New Brunswick, NJ, USA, 2012. IEEE.
- [44] H.-G. Leimer. Optimal decomposition by clique separators. *Discrete Mathematics*, 113(1):99–123, 1993.
- [45] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data*, 1(1):1–41, Mar. 2007.
- [46] S. McCullough. 2-chordal graphs. In *Contributions to Operator Theory and its Applications*, pages 143–192. Springer, 1988.
- [47] S. McCullough. Minimal separators of 2-chordal graphs. *Linear algebra and its applications*, 184:187–199, 1993.
- [48] K. Olesen and A. Madsen. Maximal prime subgraph decomposition of bayesian networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 32(1):21–31, 2002.
- [49] R. Paige and R. Tarjan. Three partition refinement algorithms. *SIAM Journal on Computing*, 16(6):973–989, 1987.

- [50] D. J. Rose, R. E. Tarjan, and G. S. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM Journal on computing*, 5(2):266–283, 1976.
- [51] M. A. Soto Gómez. *Quelques propriétés topologiques des graphes et applications à internet et aux réseaux*. PhD thesis, Univ. Paris Diderot (Paris 7), 2011.
- [52] J. P. Spinrad. Recognizing quasi-triangulated graphs. *Discrete applied mathematics*, 138(1):203–213, 2004.
- [53] M. Sysło. Characterizations of outerplanar graphs. *Discrete Mathematics*, 26(1):47–53, 1979.
- [54] R. Tamassia and I. Tollis. Planar grid embedding in linear time. *IEEE Transactions on Circuits and Systems*, 36(9):1230–1234, 1989.
- [55] R. E. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.
- [56] R. E. Tarjan. Decomposition by clique separators. *Discrete Mathematics*, 55(2):221–232, 1985.
- [57] V. Vassilevska Williams, J. Wang, R. Williams, and H. Yu. Finding four-node subgraphs in triangle time. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '15, pages 1671–1680. SIAM, 2015.
- [58] K. Verbeek and S. Suri. Metric embedding, hyperbolic space, and social networks. In *Annual Symposium on Computational Geometry (SCG)*, pages 501–510, New York, NY, USA, 2014. ACM.
- [59] Y. Wu and C. Zhang. Hyperbolicity and chordality of a graph. *The Electronic Journal of Combinatorics*, 18(1):P43, 2011.
- [60] M. Yancey. An investigation into graph curvature’s ability to measure congestion in network flow. Technical Report arXiv:1512.01281, ArXiv, 2015.

APPENDIX C

On the recognition of C_4 -free and 1/2-hyperbolic graphs

RECOGNITION OF C_4 -FREE AND 1/2-HYPERBOLIC GRAPHS*

DAVID COUDERT^{†‡} AND GUILLAUME DUCOFFE^{†‡§}

Abstract. The shortest-path metric d of a connected graph G is *1/2-hyperbolic* if, and only if, it satisfies $d(u, v) + d(x, y) \leq \max\{d(u, x) + d(v, y), d(u, y) + d(v, x)\} + 1$, for every 4-tuple u, x, v, y of G . We show that the problem of deciding whether an unweighted graph is 1/2-hyperbolic is *subcubic equivalent* to the problem of determining whether there is a chordless cycle of length 4 in a graph. An improved algorithm is also given for both problems, taking advantage of fast rectangular matrix multiplication. In the worst case it runs in $O(n^{3.26})$ -time.

Key words. Hyperbolicity, discrete metric space, graph algorithms, C_4 -free graphs, rectangular matrix multiplication

AMS subject classifications. 05C85, 65F30, 68Q17, 68Q25, 68R10

1. Introduction. The primary aim of our work is to study hyperbolicity of simple unweighted graphs. This is a metric parameter, that was first introduced by Gromov in the context of automatic groups (see [23]), then extended to more general metric spaces [5]. Roughly, the hyperbolicity of a connected graph is a measure of how far is the shortest-path metric of the graph from a *tree metric*. One can deduce from this parameter tight bounds for the (worst) additive distortion of the distances in the graph when its vertices are embedded into a weighted tree [10]. Practical applications of hyperbolicity were proposed in the domains of routing [6], network security [27], bioinformatics [18], and in the spread of information in social networks [4].

So far, the best known algorithm for determining the hyperbolicity of a graph has an $O(n^{3.69})$ -time complexity [21]. This is however prohibitive for graphs with tens of thousands of nodes such as the graph of the Autonomous Systems of the Internet, road maps, etc. An algorithm with good practical performances has been proposed in [11]. It improves the worst-case running time on certain graph classes, but it cannot be used on graphs with hyperbolicity less than one.

Related work. Our work focuses on a decision version of the problem, namely the recognition of graphs with hyperbolicity (at most) 1/2. Graphs with small hyperbolicity value have already received some attention, as a first characterization of 1/2-hyperbolic graphs was proposed in [1]. However, to the best of our knowledge, there was no known algorithmic application to it prior to this work. We are more interested in a reduction such as the recent one in [21], where the authors proved an equivalence between the problems of finding a 2-approximation for the hyperbolicity and the (max, min)-matrix multiplication. A recent work [19] further exploits the relation between both problems, yielding constant-factor approximations for the hyperbolicity in subcubic-time. We point out that a similar line of research was followed in [34, 39], where they determined the subcubic equivalence between various combinatorial problems.

Our contribution. We relate the recognition of graphs with hyperbolicity (at most) 1/2 to the search of (induced) cycles of length 4, *e.g.* C_4 , in a graph. It

*This work has been partially supported by ANR project Stint under reference ANR-13-BS02-0007, ANR program "Investments for the Future" under reference ANR-11-LABX-0031-01, and by European project FP7 EULER (Grant No.258307).

[†]Inria, France

[‡]Univ. Nice Sophia Antipolis, CNRS, I3S, UMR 7271, 06900 Sophia Antipolis, France.

[§]ENS Cachan, 61 Avenue du Président Wilson, 94230 Cachan, France.

actually follows from our work that either both problems are solvable in subcubic-time, or none of them is. We first present a linear-time reduction from the C_4 -free graph recognition problem to the recognition of 1/2-hyperbolic graphs (§3.1). Then we prove a new characterization of 1/2-hyperbolic graphs, which is based on graph powers [30], and from which it follows that, conversely, deciding whether a graph is 1/2-hyperbolic can be reduced in subcubic-time to the C_4 -free graph recognition problem (§3.2). In §4, we finally reduce both problems to the problem of the rectangular matrix product that was defined in [31]. This allows us to solve both of them in $O(n^{3.26})$ -time, which beats the previous records established in [21, 37].

We give the notations used in this paper in §2, along with definitions for graph hyperbolicity and C_4 -free graphs.

2. Definitions and Notations. A graph G is a pair (V, E) , whose n vertices are the elements of the set V , and whose m edges are the elements of E ; every edge is a set of two distinct vertices of G . The *neighborhood* $N(u)$ of a vertex $u \in V$, is the (possibly empty) set of vertices $v \in V$ such that $\{u, v\}$ is an edge. Alternatively, we say that the elements of $N(u)$ are *adjacent* to u . A *clique* is a set of pairwise adjacent vertices. Note that the adjacency relation is clearly symmetric; we also define the (symmetric) adjacency matrix $A = (\mathbb{I}_{\{u \in N(v)\}})_{u, v \in V}$, where \mathbb{I} denotes the Kronecker delta¹.

Finally, an *induced subgraph* of G is a graph $G[X] = (X, F)$ such that $X \subseteq V$ and $F = \{\{u, v\} \in E : u, v \in X\}$. In particular if X is a clique, then it is called a *complete* subgraph. The induced subgraph is a *path of length* $l \geq 0$ if $|X| = l + 1$ and the vertices of X can be linearly ordered into a sequence (v_0, v_1, \dots, v_l) such that for every $0 \leq i, j \leq l$, the vertex v_i is adjacent to v_j if, and only if, $|j - i| = 1$. In such a case, the vertices v_0 and v_l are called the *endpoints* of the path, and the path is a $v_0 v_l$ -path. The graph G is *connected* if, for every pair $u, v \in V$, there exists a uv -path. Also, a cycle is a graph such that the deletion of any edge $\{v_0, v_l\} \in F$ yields a $v_0 v_l$ -path, and a tree is a connected graph which does not contain any cycle as a subgraph.

Further standard graph terminology can be found in [7, 14].

2.1. 1/2-hyperbolic graphs. Given a connected graph $G = (V, E)$, we define the distance $d_G(u, v)$ between two vertices $u, v \in V$ as the minimum length of a uv -path in the graph. This yields a (discrete) metric space (V, d_G) . For a survey of metric graph theory, the reader may refer to [2]. We define in the space (V, d_G) an interval $[u, v]$ between any two vertices $u, v \in V$, as the set of vertices “in between” u and v , e.g. $[u, v] = \{x \in V : d_G(u, v) = d_G(u, x) + d_G(x, v)\}$.

In the sequel, we will call a uv -path of minimum length a *uv-shortest path*, and we will denote the distance function by d instead of d_G whenever G is clear from the context. The graph hyperbolicity of G can now be defined as follows:

DEFINITION 2.1 (4-points Condition, [23]). *Let G be a connected graph. For every 4-tuple u, x, v, y of G , we define $\delta(u, v, x, y)$ as half of the difference between the two largest sums amongst*

$$\begin{aligned} S_1 &= d(u, v) + d(x, y), \\ S_2 &= d(u, x) + d(v, y), \text{ and} \\ S_3 &= d(u, y) + d(v, x). \end{aligned}$$

The graph hyperbolicity, denoted by $\delta(G)$, is equal to $\max_{u, x, v, y} \delta(u, v, x, y)$.

¹We use the symbol \mathbb{I} instead of the classical symbol δ for the Kronecker delta in order to prevent confusion with hyperbolicity.

Moreover, we say that G is δ -hyperbolic, for every $\delta \geq \delta(G)$.

Unlike many well-known graph properties, it is very important to note that the hyperbolicity of an *induced subgraph* of G does not yield any information in general about the hyperbolicity of G . For example, the wheel W_n is 1-hyperbolic, whereas it contains as an induced subgraph the cycle C_n whose hyperbolicity grows linearly with n [11]. A way to deal with this difficulty is to constrain ourselves to distance-preserving, or *isometric* subgraphs. Formally, an induced subgraph H of G is isometric if, and only if, it is connected and for every pair of vertices $u, v \in H$, we have that $d_H(u, v) = d_G(u, v)$. We also say that a subgraph which is not isometric is a *bridged* subgraph.

Lower and upper bounds on the hyperbolicity can be deduced from classical parameters such as the girth [33], the circumference [8], the domination number [35], and the chordality [29, 40]. In particular, we have that $\delta(G) \leq \lfloor \text{diam}(G)/2 \rfloor$, where $\text{diam}(G) = \max_{u, v \in V} d_G(u, v)$ is the *diameter* of the graph [11, 40].

We inform the reader that Definition 2.1 is not a universal definition for the hyperbolicity of a graph. Some authors actually proposed and studied other definitions (see, for instance [13, 23]). Though the value of $\delta(G)$ may vary depending on the choice of the definition, any δ -hyperbolic graph with respect to (w.r.t.) any of the definitions is $f(\delta)$ -hyperbolic w.r.t. any other definition of the hyperbolicity. The function f is linear in δ in most cases. Moreover, the class of trees is always contained into the class of 0-hyperbolic graphs, which makes the graph hyperbolicity a tree-likeness parameter.

We here restrict our study to Definition 2.1, as it has algorithmic applications. Indeed, it is straightforward by using Definition 2.1 to compute the graph hyperbolicity $\delta(G)$ in $\Theta(n^4)$ -time (see [11] and [21] for practical and theoretical improvements of the complexity). Also, note that $\delta(G)$ is always a half-integer (w.r.t. Definition 2.1). Our work focuses on graphs with small hyperbolicity, that is hyperbolicity at most 1/2. Those graphs thus satisfy either $\delta(G) = 0$ or $\delta(G) = 1/2$. We address the problem of recognizing those graphs, that we formulate as follows.

PROBLEM 2.2. *Given a connected graph G , is G a 1/2-hyperbolic graph ?*

In [1], Bandelt and Chepoi characterized the 1/2-hyperbolic graphs as the connected graphs that simultaneously satisfy the three following conditions:

CONDITION 2.3. *Every cycle of length at least 6 in G is bridged.*

CONDITION 2.4. *For every pair $u, v \in G$, $N(u) \cap [u, v]$ is a clique.*

CONDITION 2.5. *No graph in Figure 1 is an isometric subgraph of G .*

A simpler characterization was previously given for 0-hyperbolic graphs [3, 24]. In fact, 0-hyperbolic graphs are *block-graphs*, that are graphs in which every biconnected component (block) is a clique (possibly reduced to a single vertex). This class includes cliques and trees, and a block-graph can be recognized in $O(n + m)$ -time.

2.2. C_4 -free graphs. The C_4 -free graph recognition problem asks whether a given graph G contains an induced cycle of length 4. In the sequel, such a cycle, if any, is called a C_4 , or a *quadrangle*. A graph G which does not contain any C_4 as an induced subgraph is a *C_4 -free* graph. Let us define our decision problem in the following way.

PROBLEM 2.6. *Given a graph G , does G contain a C_4 as an induced subgraph ?*

We now remind a well-known, local characterization of those graphs:

FACT 2.7. *A graph $G = (V, E)$ is C_4 -free if, and only if, for every pair of non-adjacent vertices u, v , the set $N(u) \cap N(v)$ is a (possibly empty) clique.*

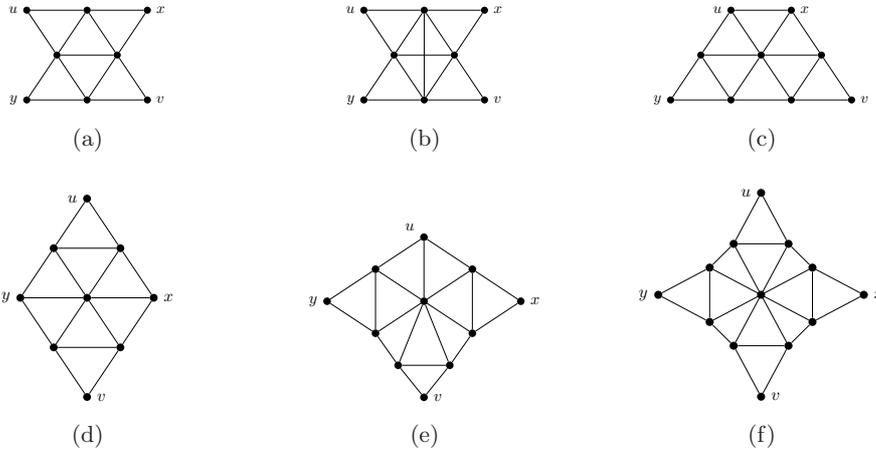


Figure 1: The six forbidden isometric subgraphs.

To see the relation between Problem 2.6 and Problem 2.2, one can observe that the condition of Fact 2.7 is equivalent to Condition 2.4 when considering vertices u, v at distance 2. As a consequence, every $1/2$ -hyperbolic graph is also C_4 -free. In fact, a more direct way to see this is to note that every induced subgraph which is a quadrangle is *isometric* (this comes from the fact that a C_4 is connected and it has diameter 2). Since one can easily check that $\delta(C_4) = 1$, then it indeed follows that a $1/2$ -hyperbolic graph cannot contain a quadrangle as an induced subgraph.

So far, the best-known algorithm we are aware of to detect an induced C_4 in a graph has $O(n^{\omega(1)+1}) = O(n^{3.3727})$ -time complexity [37], with $O(n^{\omega(1)})$ being the complexity of multiplying two $n \times n$ matrices (see §4.1 for details). We will improve this result in §4.

3. The subcubic equivalence. It is straightforward by the definitions that both the $1/2$ -hyperbolic graph recognition problem (Problem 2.2), and the C_4 -free graph recognition problem (Problem 2.6), are polynomial-time solvable [21, 37]. On the other hand, the best-known upper-bound on their time complexity is strictly more than cubic. Thus it motivates the search for *subcubic reductions* between these problems, as they are defined in [39]. Formally, a subcubic reduction from a problem A to a problem B is a subcubic-time Turing reduction, which verifies the following additional properties on the oracle access to problem B. For every positive real μ , there has to exist a positive real ε such that:

- (i) the reduction runs in $\tilde{O}(n^{3-\varepsilon})$ -time², where n denotes the size of the input;
- (ii) and given an instance of size n of problem A, $\sum_i \tilde{O}(n_i^{3-\mu}) = \tilde{O}(n^{3-\varepsilon})$, where n_i denotes the size of the i^{th} oracle call to problem B in the reduction.

In particular, a linear-time reduction is a subcubic reduction. More generally, any subcubic-time reduction which satisfies $\sum_i n_i = \tilde{O}(n)$ is also a subcubic reduction. We will only consider subcubic reductions of this kind in the sequel.

²The notation $\tilde{O}(f(n))$ is for a complexity $f(n) \cdot \log^{O(1)} n$.

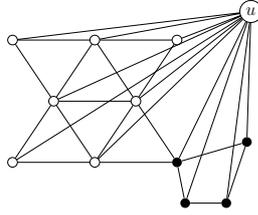


Figure 2: An illustration of the linear-time reduction of Proposition 3.1.

Subcubic reductions are of specific interest in the study of subcubic-time algorithms, because if there exists a subcubic reduction from problem A to problem B, and there is a subcubic-time algorithm which solves problem B, then there also exists a subcubic-time algorithm which solves problem A. In particular, if problems A and B are *subcubic equivalent*, then either both of them are solvable in subcubic-time, or none of them is. In this section, we will show that Problem 2.2 and Problem 2.6 are subcubic equivalent. We first present a linear-time reduction from Problem 2.6 to Problem 2.2 in §3.1 (Proposition 3.1). Then we present a subcubic reduction from Problem 2.2 to Problem 2.6 in §3.2 (Theorem 3.13).

3.1. Reducing the detection of a C_4 to the recognition of a 1/2-hyperbolic graph. PROPOSITION 3.1. *There is a linear-time reduction from the C_4 -free graph recognition problem to the problem of deciding whether a graph is 1/2-hyperbolic.*

Proof. Let $G = (V, E)$ be an instance of the C_4 -free graph recognition problem. Let $u \notin V$, and let $G' = (V \cup \{u\}, E \cup \{\{u, v\} : v \in V\})$. By construction, G' is connected, and it has diameter at most 2. Thus, we have that $\delta(G') \leq 1$ (e.g. see [11]), and all of its isometric subgraphs also have diameter at most 2. Moreover we remind that a cycle of length l has diameter $\lfloor l/2 \rfloor$. In particular, every cycle of length at least 6 is a graph of diameter at least 3 and as such, it cannot be an isometric cycle of G' . Consequently, G' always satisfies Condition 2.3. We can prove that it always satisfies Condition 2.5 in the same way. Finally, since Condition 2.4 is satisfied for every pair $u, v \in V$ of *adjacent* vertices, then G' satisfies Condition 2.4 if, and only if, for every pair $u, v \in V$ of *non-adjacent* vertices, we have that $N_{G'}(u) \cap [u, v]$ is a clique; since $d_{G'}(u, v) = 2$ in such a case, then it is equivalent to have that $N_{G'}(u) \cap N_{G'}(v)$ is a clique, e.g. the graph G' is C_4 -free by Fact 2.7. Furthermore, we have by construction that any induced C_4 in G' is an induced quadrangle in G , and vice-versa. Consequently, G is C_4 -free if, and only if, the graph G' is 1/2-hyperbolic. \square

We want to highlight that our reduction from Problem 2.6 to Problem 2.2 here is linear-time, whereas the converse reduction from Problem 2.2 to Problem 2.6, presented in §3.2, is super-linear. It might be of interest to determine whether a linear-time reduction from Problem 2.2 to Problem 2.6 exists.

3.2. Finding quadrangles to recognize non-1/2-hyperbolic graphs. Our aim is now to prove that there exists a subcubic reduction from Problem 2.2 to Problem 2.6 (Theorem 3.13). Ideally, we would ask for a subcubic-time routine for checking whether Conditions 2.3, 2.4 and 2.5 are satisfied. Our reduction is however more complex, as we actually introduce and verify different conditions in subsequent

sections. Thus we have to prove that these conditions imply Conditions 2.3, 2.4 and 2.5, and also that they are satisfied by 1/2-hyperbolic graphs.

3.2.1. Quickly excluding long isometric cycles. Let us first deal for our reduction with a tool that we will use later to verify whether Condition 2.3 is satisfied. We recall that Condition 2.3 requires that every cycle of length at least 6 in G is a bridged subgraph. A first naive approach to deal with this condition is to compute the length of a longest isometric cycle of G . This can be done in polynomial-time, but the best-known algorithm runs in $O(n^{2\omega(1)} \log n) = O(n^{4.752} \log n)$ -time, which is super-cubic [32]. Instead, we propose in this section a way to weaken Condition 2.3, by only having to consider isometric cycles of *polylogarithmically-bounded* length.

Given a connected graph G , we recall that a c -factor approximation of the hyperbolicity of G is a half-integer $\delta_c(G)$ such that $\delta(G) \leq \delta_c(G) \leq c \cdot \delta(G)$. In this section, we will assume we have a subcubic-time algorithm computing a c -factor approximation of the hyperbolicity, for some fixed choice of $c = \log^{O(1)} n$. Below, we remind possible ways to achieve such a result:

LEMMA 3.2 ([9, 19, 21]). *Let $G = (V, E)$ be a connected graph.*

(i) *There exists an algorithm computing a 2-factor approximation of the hyperbolicity in $O(n^{3-\omega(1)/2}) = O(n^{2.69})$ -time [21].*

(ii) *For every $\varepsilon > 0$, there exists an algorithm computing a $(2 + \varepsilon)$ -factor approximation of the hyperbolicity in $\tilde{O}(\varepsilon^{-1} n^{\omega(1)}) = \tilde{O}(\varepsilon^{-1} n^{2.3727})$ -time [19].*

(iii) *There exists an algorithm computing a 1569-factor approximation of the hyperbolicity in $O(\min\{nm, n^{\omega(1)} \log n\} + n^2) = \tilde{O}(n^{2.3727})$ -time³ [9].*

Combining results from [10, 15, 21], there also exists a $\theta(\log n)$ -factor approximation algorithm of the hyperbolicity in $\tilde{O}(n^2)$ -time⁴. In addition, one can deduce from the results from [16, 17, 22] an algorithm which computes a $\theta(\log^2 n)$ -factor approximation of the hyperbolicity in $O(m \log n)$ -time.

It is well-known that the hyperbolicity of the cycle C_n grows linearly with n . Formally:

LEMMA 3.3 ([11, 40]). *Cycles of order $4p + \varepsilon \geq 3$, with $p \geq 0$ and $\varepsilon \in \{0, 1, 2, 3\}$, are $(p - 1/2)$ -hyperbolic when $\varepsilon = 1$, and p -hyperbolic otherwise.*

As a result, it follows from Lemma 3.3 that a (polylogarithmic) upper-bound on the hyperbolicity of G yields a (polylogarithmic) upper-bound on the length of a longest isometric cycle of G ; more accurately:

COROLLARY 3.4. *Let $G = (V, E)$ be a connected graph. Then all the isometric cycles of G have length upper-bounded by $4\delta(G) + 3$.*

Proof. First assume $\delta(G)$ is an integer. By Lemma 3.3, the longest $\delta(G)$ -hyperbolic cycle has length at most $4\delta(G) + 3$. Otherwise, $\delta(G)$ is a half-integer by Definition 2.1 and so, again by Lemma 3.3, the longest $\delta(G)$ -hyperbolic cycle has length at most $4(\delta(G) + 1/2) + 1 = 4\delta(G) + 3$. \square

Since we are interested in 1/2-hyperbolic graphs, then Corollary 3.4 implies that every isometric cycle must have length upper-bounded by 5 *i.e.*, Condition 2.3. We introduce in the next section a second tool so that we can detect isometric cycles of polylogarithmically-bounded length.

³The term $\min\{nm, n^{\omega(1)} \log n\}$ in the complexity comes from the computation of the all-pairs shortest-paths in the graph (see [36] for an algorithm in $\tilde{O}(n^{\omega(1)})$ -time for the problem).

⁴The authors in [21] actually claim a $O(n^2)$ -time complexity for their algorithm, but it takes as inputs discrete metric spaces and so, it does not apply to graphs directly. To apply their results on graphs, we can use a $\theta(\log n)$ -additive approximation of the all-pairs shortest-paths, which can be computed in $\tilde{O}(n^2)$ -time (see [15]).

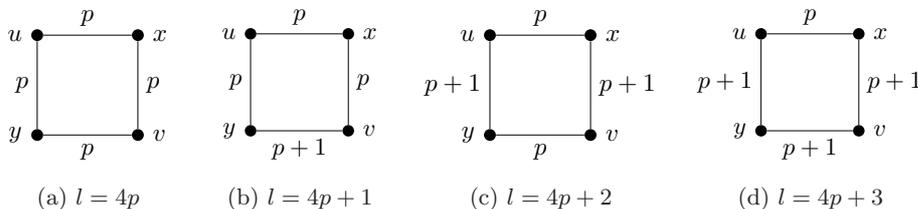


Figure 3: Shrinking long isometric cycles into quadrangles.

3.2.2. Using graph powers. Let us now present the main tool for our reduction, namely graph powers.

DEFINITION 3.5. Given a connected graph $G = (V, E)$, let i be a positive integer. The i^{th} -power of G , denoted by $G^i = (V, E_i)$, is a graph whose set of vertices is the same as for G ; two vertices $u, v \in V$ are adjacent in G^i if, and only if, there exists a uv -path of length at most i in G . Formally, $E_i = \{\{u, v\} : 0 < d_G(u, v) \leq i\}$.

In particular, the graph power G^1 is G , and the graph power $G^{\text{diam}(G)}$ is the complete graph K_n , where $\text{diam}(G) = \max_{u, v \in V} d(u, v)$ is the diameter of G . It is folklore that the graph power G^i can be computed in $O(n^{\omega(1)} \log i)$ -time, using fast square matrix multiplication [36].

Recall that we said in §2.2 that every 1/2-hyperbolic graph G is C_4 -free. Roughly, most of our reduction will consist in checking whether a *polylogarithmic* number of graph powers of G are C_4 -free as well. This is a necessary condition so that G is 1/2-hyperbolic, as stated below.

LEMMA 3.6. If G is a 1/2-hyperbolic graph, then for every positive integer i , the graph G^i is C_4 -free.

Proof. By contradiction, let u, v, x, y be the vertices of an induced quadrangle in G^i , for some positive integer $i \geq 1$. Without loss of generality, assume that $x, y \in N_{G^i}(u) \cap N_{G^i}(v)$. It follows by Definition 3.5 that:

$$\begin{aligned} \max\{d_G(u, x), d_G(u, y), d_G(v, x), d_G(v, y)\} &\leq i; \\ \text{and} \quad \min\{d_G(u, v), d_G(x, y)\} &\geq i + 1. \end{aligned}$$

As a consequence, we have by Definition 2.1 that:

$$\begin{aligned} \delta(u, v, x, y) &= \frac{1}{2}[(d_G(u, v) + d_G(x, y)) \\ &\quad - \max\{d_G(u, x) + d_G(v, y), d_G(u, y) + d_G(v, x)\}] \\ &\geq \frac{1}{2}[2(i + 1) - 2i] \\ &\geq 1 \end{aligned}$$

which contradicts the fact that G is 1/2-hyperbolic. \square

Intuitively, the existence of isometric cycles of length l in G yields the existence of induced quadrangles in some graph power $G^{\theta(l)}$; this shrinking effect is illustrated with Figure 3. As a result, it may be more efficient to search for induced cycles of length 4 in the graph powers rather than computing the length of a longest isometric cycle of G directly.

We emphasize that the converse does not hold: not all the induced quadrangles in G^i yield an isometric cycle in the original graph G . For instance, the square graph H_1^2 of the graph H_1 in Figure 1a contains a quadrangle as an induced subgraph (the vertices of which are u, x, v, y), yet H_1 does not contain an isometric cycle of length more than 3. However, we remind that every graph power has to be C_4 -free by Lemma 3.6 and so, *any* induced quadrangle that we detect in some graph power is a certificate to prove that the graph is not 1/2-hyperbolic.

We formalize it as follows.

LEMMA 3.7. *Let $G = (V, E)$ be a connected graph, and let C_l be an isometric cycle of length $l = 4p + \varepsilon$, $p \geq 1$, $\varepsilon \in \{0, 1, 2, 3\}$ and $l \neq 5$. There is an integer $i \in [p, 2p]$ such that the graph power G^i contains a C_4 as an induced subgraph.*

Proof. Let us fix an arbitrary orientation for C_l , and choose a 4-tuple u, x, v, y (in clockwise order) such that:

- (i) if $\varepsilon = 0$: $d_G(u, x) = d_G(u, y) = d_G(v, x) = d_G(v, y) = p$;
- (ii) if $\varepsilon = 1$: $d_G(u, x) = d_G(u, y) = d_G(v, x) = p$, and $d_G(v, y) = p + 1$;
- (iii) if $\varepsilon = 2$: $d_G(u, x) = d_G(v, y) = p$, and $d_G(u, y) = d_G(v, x) = p + 1$;
- (iv) if $\varepsilon = 3$: $d_G(u, x) = d_G(u, y) = d_G(v, x) = p + 1$, and $d_G(v, y) = p$.

An example of such a choice is given in Figure 3. Note that all the above distances are upper-bounded by $p + \lceil \varepsilon/4 \rceil$. Furthermore, recall that C_l is an isometric cycle by the hypothesis. So, we have:

$$d_G(u, v) = d_G(x, y) = \begin{cases} 2p & \text{when } \varepsilon \in \{0, 1\} \\ 2p + 1 & \text{when } \varepsilon \in \{2, 3\} \end{cases}$$

Equivalently, we have $d_G(u, v) = d_G(x, y) = 2p + \lfloor \varepsilon/2 \rfloor$. As a consequence, by Definition 3.5, we have that $G^i[\{u, x, v, y\}]$ is an induced C_4 , for every $p + \lceil \varepsilon/4 \rceil \leq i \leq 2p + \lfloor \varepsilon/2 \rfloor - 1$. To prove that such a value of i always exists, it now remains to prove that:

$$p + \left\lceil \frac{\varepsilon}{4} \right\rceil \leq 2p + \left\lfloor \frac{\varepsilon}{2} \right\rfloor - 1,$$

that is:

$$\left(2p + \left\lfloor \frac{\varepsilon}{2} \right\rfloor - 1\right) - \left(p + \left\lceil \frac{\varepsilon}{4} \right\rceil\right) = p + \left(\left\lfloor \frac{\varepsilon}{2} \right\rfloor - \left\lceil \frac{\varepsilon}{4} \right\rceil\right) - 1 \geq 0.$$

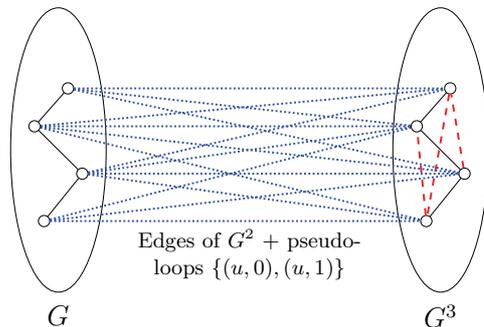
A straightforward calculation shows that:

$$\left\lfloor \frac{\varepsilon}{2} \right\rfloor - \left\lceil \frac{\varepsilon}{4} \right\rceil = \begin{cases} -1 & \text{if } \varepsilon = 1 \\ 0 & \text{otherwise} \end{cases}$$

As a consequence, if $\varepsilon \neq 1$ then we are done. Otherwise, since $l = 4p + 1 > 5$ by the hypothesis, then $p \geq 2$ and so:

$$p + \left(\left\lfloor \frac{\varepsilon}{2} \right\rfloor - \left\lceil \frac{\varepsilon}{4} \right\rceil\right) - 1 \geq 2 - 1 - 1 \geq 0. \quad \square$$

Note that the only two possible lengths for an isometric cycle of the graph that Lemma 3.7 does not take into account are 3 and 5. This is not a coincidence, as C_3 and C_5 are the only cycles that are 1/2-hyperbolic by Lemma 3.3.


 Figure 4: The construction of the graph $G^{[2]}$.

3.2.3. Transforming some obstructions into quadrangles. One of the most fundamental steps for our reduction was to prove with Lemma 3.6 that every graph power G^i , $i \geq 1$, has to be C_4 -free so that a connected graph G is 1/2-hyperbolic. An interesting question on its own is whether it is also a sufficient condition.

We give a negative answer to this conjecture, using the graph H in Figure 5. The graph H is the union of a C_5 and a C_3 that both share a single edge; using the 4-tuple in bold in Figure 5, one can verify that $\delta(H) \geq 1$. Clearly, H does not contain a quadrangle as an induced subgraph. Moreover, $\text{diam}(H) = 3$ and so, for every $i \geq 3$, the graph power H^i is a complete graph. Finally, as there is only one couple x, y of H such that $d_H(x, y) = 3$, it follows that the square graph H^2 of H only lacks a single edge to be complete; as a result, H^2 is C_4 -free as well.

The primary aim of this section is now to show that in order to complete our reduction, we solely need to decide whether only *one* additional graph is C_4 -free:

DEFINITION 3.8. *Let $G = (V, E)$ be a connected graph. The graph $G^{[2]} = (V^{[2]}, E^{[2]})$ is defined as follows:*

- (i) $V^{[2]} \simeq V \times \{0, 1\}$;
- (ii) $G^{[2]}[V \times \{0\}] \simeq G$;
- (iii) $G^{[2]}[V \times \{1\}] \simeq G^3$;
- (iv) $\forall u, v \in V$, the vertices $(u, 0)$ and $(v, 1)$ are adjacent in $G^{[2]}$ if, and only if, $d_G(u, v) \leq 2$.

In particular, $\forall u \in V$, there is an edge $\{(u, 0), (u, 1)\} \in E^{[2]}$.

An illustration of the construction of $G^{[2]}$ is presented in Figure 4. It might help to observe that for every edge of G^2 *i.e.*, for every two distinct vertices u, v such that $d_G(u, v) \leq 2$, there are exactly two corresponding edges in $G^{[2]}$, denoted by $\{(u, 0), (v, 1)\}$ and $\{(u, 1), (v, 0)\}$, that connect the sets $V \times \{0\}$ and $V \times \{1\}$. Every other connecting edge of $G^{[2]}$ is a pseudo-loop $\{(u, 0), (u, 1)\}$, for some vertex $u \in V$.

We interpret the role of $G^{[2]}$ as the role of an "intermediate power" between the square graph G^2 and the cube graph G^3 . First, let us prove that it is necessary for $G^{[2]}$ to be C_4 -free so that G is 1/2-hyperbolic.

LEMMA 3.9. *If G is a 1/2-hyperbolic graph, then the graph $G^{[2]}$ is C_4 -free.*

Proof. By contradiction, let a, b, c, d be the vertices of an induced quadrangle in $G^{[2]}$. Without loss of generality, we assume that $d_{G^{[2]}}(a, b) = d_{G^{[2]}}(c, d) = 2$. Several cases have to be considered, that we illustrate with Figure 6.

If $a, b, c, d \in V \times \{0\}$, then there is an induced C_4 in G by Definition 3.8. Similarly,

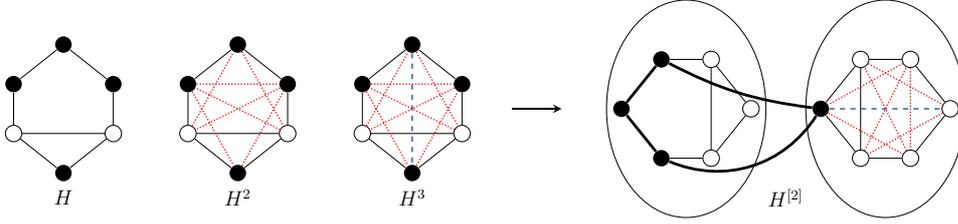


Figure 5: A 1-hyperbolic graph whose powers are C_4 -free. A quadrangle in $H^{[2]}$ is drawn in bold.

if $a, b, c, d \in V \times \{1\}$, then there is an induced C_4 in G^3 by Definition 3.8. In both cases, this implies that G is *not* 1/2-hyperbolic, by Lemma 3.6.

For all the remaining cases, we claim that there is no vertex $u \in V$ such that $\{(u, 0), (u, 1)\} \subseteq \{a, b, c, d\}$. It easily follows from the fact that $N_{G^{[2]}}((u, 0)) \cup \{(u, 0)\} \subseteq N_{G^{[2]}}((u, 1)) \cup \{(u, 1)\}$. Thus we can write $(a, b, c, d) = ((u, k), (v, k'), (x, j), (y, j'))$, with $\{k, k', j, j'\} = \{0, 1\}$ and vertices u, v, x, y are pairwise distinct.

Case 1: $k = 0, k' = j = j' = 1$. Here it comes that:

$$\begin{aligned} \max\{d_G(u, x) + d_G(v, y), d_G(u, y) + d_G(v, x)\} &\leq 2 + 3 = 5, \\ \text{whereas } d_G(u, v) + d_G(x, y) &\geq 3 + 4 = 7. \end{aligned}$$

In other words, $\delta(G) \geq \delta(u, v, x, y) \geq 1$.

Case 2: $k = 1, k' = j = j' = 0$. In such a case:

$$\begin{aligned} \max\{d_G(u, x) + d_G(v, y), d_G(u, y) + d_G(v, x)\} &\leq 2 + 1 = 3, \\ \text{whereas } d_G(u, v) + d_G(x, y) &\geq 3 + 2 = 5. \end{aligned}$$

So, $\delta(G) \geq \delta(u, v, x, y) \geq 1$.

Case 3: $k = k' = 0, j = j' = 1$. It follows that:

$$\begin{aligned} \max\{d_G(u, x) + d_G(v, y), d_G(u, y) + d_G(v, x)\} &\leq 2 + 2 = 4, \\ \text{whereas } d_G(u, v) + d_G(x, y) &\geq 2 + 4 = 6. \end{aligned}$$

In other words, $\delta(G) \geq \delta(u, v, x, y) \geq 1$.

Case 4: $k = j = 0, k' = j' = 1$. Then we have:

$$\begin{aligned} \max\{d_G(u, x) + d_G(v, y), d_G(u, y) + d_G(v, x)\} &\leq \max\{1 + 3, 2 + 2\} = 4, \\ \text{whereas } d_G(u, v) + d_G(x, y) &\geq 3 + 3 = 6. \end{aligned}$$

So, we again conclude that $\delta(G) \geq \delta(u, v, x, y) \geq 1$. \square

Recall that in order to decide whether a graph is 1/2-hyperbolic, our aim is to check whether all of the three Conditions 2.3, 2.4 and 2.5 of [1] are satisfied, using stronger necessary conditions. So far, we only dealt with Condition 2.3, developing tools in §3.2.1 and 3.2.2 in order to determine if every cycle of length at least 6 in the graph is bridged. The following two lemmas will show a way to combine the graph $G^{[2]}$ of Definition 3.8 with the graph powers of Definition 3.5, in order to ensure that both Conditions 2.4 and 2.5 are satisfied as well.

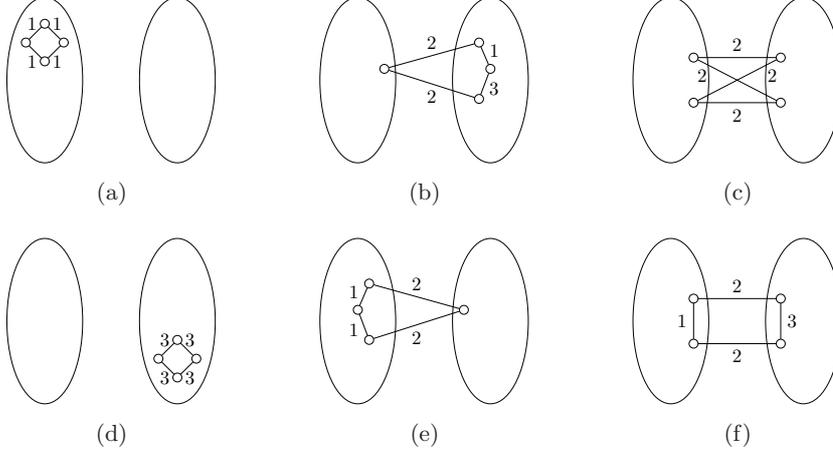


Figure 6: Possible quadrangles in $G^{[2]}$. Edge weights represent distances in G .

Let us start with Condition 2.4:

LEMMA 3.10. *Let $G = (V, E)$ be a δ -hyperbolic graph, for some $\delta \geq 1/2$. Suppose G does not satisfy Condition 2.4. Then $G^{[2]}$ is not C_4 -free, or there exists some positive integer $i \leq 2\delta$ such that G^i is not C_4 -free.*

Proof. Let $u, v \in V$ be such that $N(u) \cap [u, v]$ is not a clique, and $d(u, v)$ is minimum w.r.t. this property. Let $x_1, y_1 \in N(u) \cap [u, v]$ such that x_1 and y_1 are not adjacent in G .

- Note that if $d(u, v) = 2$, then we are done as u, v, x_1, y_1 are the vertices of an induced C_4 in G .
- Similarly, if $d(u, v) = 3$, then we have:
 - (i) $d(u, x_1) = d(u, y_1) = 1$;
 - (ii) $d(x_1, y_1) = d(v, x_1) = d(v, y_1) = 2$;
 as a consequence, $(u, 0), (x_1, 0), (y_1, 0), (v, 1)$ are the vertices of an induced quadrangle in the graph $G^{[2]}$.

In the sequel, we will assume $d(u, v) \geq 4$. Let us define the two uv -shortest paths:

$$P_1 = u, x_1, x_2, x_3, \dots, x_{d(u,v)-1}, v$$

$$\text{and } P_2 = u, y_1, y_2, y_3, \dots, y_{d(u,v)-1}, v$$

Note that for every $i \leq j$ we have $d(x_i, y_j) \geq d(y_i, y_j) = j - i$, and in the same way $d(x_j, y_i) \geq d(x_j, x_i) = j - i$. We now claim that for every $i \leq j$, the inequalities above are strict, or equivalently $d(x_i, y_j) > j - i$ and $d(x_j, y_i) > j - i$. By contradiction, suppose that there is some $i \leq j$ satisfying $d(x_i, y_j) = j - i$. Then in such a case we have $x_1, y_1 \in N(u) \cap [u, y_j]$, contradicting the minimality of $d(u, v)$. The case when $d(x_j, y_i) = j - i$ is dealt with similarly. To sum up, by the minimality of $d(u, v)$ we have that for any $i \leq j$:

$$d(x_i, y_j) > d(y_i, y_j) = j - i \text{ and } d(x_j, y_i) > d(x_j, x_i) = j - i.$$

In particular, for all i, j this yields that x_i and y_j are pairwise distinct.

Also, note that for all i , $\delta(u, v, x_i, y_i) = d(x_i, y_i)/2$ and so, we have that for every i :

$$d(x_i, y_i) \leq 2\delta.$$

Let $l \geq 3$ be the least index greater than 2 such that $d(x_l, y_l) \leq l - 1$. One has to observe that since for all i , $d(x_i, y_i) \leq 2\delta$, it holds that $l \leq \min\{2\delta + 1, d(u, v) - 1\}$. In such a case:

- (i) $d(x_1, x_l) = d(y_1, y_l) = l - 1$;
- (ii) $d(x_1, y_1), d(x_l, y_l) \leq l - 1$;
- (iii) $d(x_1, y_l), d(x_l, y_1) > l - 1$.

Consequently, x_1, y_1, x_l, y_l are the vertices of an induced quadrangle in G^{l-1} . \square

Let us finally prove with Lemma 3.11 that we can verify whether Condition 2.5 is satisfied in the same way as we verified Condition 2.4 in Lemma 3.10.

LEMMA 3.11. *Let $G = (V, E)$ be a connected graph that does not satisfy Condition 2.5. Then there is an induced C_4 in the graph $G^{[2]}$, or there is an induced C_4 in the square graph G^2 .*

Proof. We proceed by contradiction. Let us assume that one of the graphs of Figure 1 is an isometric subgraph of G . For each of the forbidden graphs of Condition 2.5, we will only consider the 4-tuple of vertices that are drawn in bold in Figure 1, denoted by u, y, v, x .

Cases 1a and 1b One can easily check that in both cases, the four vertices in bold are the vertices of an induced quadrangle in the square graph G^2 .

Case 1c Observe that $d(u, x) = 1$, $d(u, y) = d(v, x) = 2$, $d(u, v) = d(x, y) = d(v, y) = 3$. So, $(u, 0), (y, 1), (v, 1), (x, 0)$ are the vertices of an induced quadrangle in $G^{[2]}$. A contradiction.

Case 1d We have that all distances but $d(u, v)$ equal 2, and that $d(u, v) = 4$. Therefore, $(u, 1), (y, 0), (v, 1), (x, 0)$ are the vertices of an induced C_4 in $G^{[2]}$. Again, this is not possible.

Case 1e Observe that $d(u, x) = d(u, y) = 2$, $d(x, y) = 4$, and all the remaining distances are equal to 3. As a consequence, $(u, 0), (y, 1), (v, 1), (x, 1)$ are the vertices of an induced quadrangle in $G^{[2]}$, which contradicts the fact that $G^{[2]}$ is C_4 -free.

Case 1f The vertices $(u, 1), (y, 1), (v, 1), (x, 1)$ induce a C_4 in G^3 , hence in $G^{[2]}$, that is once more a contradiction. \square

To sum up, we obtain as a byproduct of our reduction, and especially of Lemmas 3.6, 3.7, 3.9, 3.10 and 3.11, the following new characterization of 1/2-hyperbolic graphs:

CHARACTERIZATION 3.12. *A connected graph G is 1/2-hyperbolic if, and only if, every graph power G^i , $i \geq 1$, is C_4 -free, and the graph $G^{[2]}$ is C_4 -free.*

The condition is necessary by Lemmas 3.6 and 3.9, and it is sufficient by Lemmas 3.7, 3.10 and 3.11.

3.2.4. The reduction. **THEOREM 3.13.** *There is a subcubic reduction from the 1/2-hyperbolic graph recognition problem to the problem of detecting an induced quadrangle in a graph.*

Proof. Let $G = (V, E)$ be a connected graph. Since there exists a linear-time algorithm to recognize 0-hyperbolic graphs⁵, we will assume for the proof $\delta(G) >$

⁵Recall that a graph is 0-hyperbolic if, and only if, it is a *block-graph* i.e., a graph whose biconnected components are complete subgraphs [3, 24].

0, or equivalently $\delta(G) \geq 1/2$. Let us fix any $c = \log^{O(1)} n, c \geq 1$, such that we can compute a c -factor approximation of the hyperbolicity in subcubic-time. Three possible choices for c are given in Lemma 3.2, and two others ones are discussed in §3.2.1.⁶ In the sequel, let $\delta_c(G)$ be a c -factor approximation of the hyperbolicity. Recall that we have $\delta(G) \leq \delta_c(G) \leq c\delta(G)$ by the hypothesis. So, if $\delta_c(G) > c/2$, then we are done as the graph G is not 1/2-hyperbolic. Let us now assume that $\delta(G) \leq \delta_c(G) \leq c/2$. By Corollary 3.4, every isometric cycle of G has length upper-bounded by $4\delta_c(G) + 3 \leq 2c + 3$, which is polylogarithmically upper-bounded.

We then compute all the graph powers G^i , for $1 \leq i \leq 2\delta_c(G) + 1$. This can be done in subcubic-time, by first computing the distance-matrix of G in $O(n^{\omega(1)} \log n)$ -time (see [36]). Moreover every G^i has to be C_4 -free by Lemma 3.6. If so, then by Lemma 3.7, there is no isometric cycle C_l of length $6 \leq l \leq 4\delta_c(G) + 3$. Consequently, the graph G satisfies Condition 2.3. Finally, let us build $G^{[2]}$, which can also be done in subcubic-time using the distance-matrix of G . By Lemma 3.9, the graph $G^{[2]}$ has to be C_4 -free so that G is 1/2-hyperbolic. If it is indeed the case, then we have:

(i) $G^{[2]}$ and all of $G^i, 1 \leq i \leq 2\delta_c(G)$ are C_4 -free, hence G satisfies Condition 2.4 by Lemma 3.10;

(ii) $G^{[2]}$ and the square graph G^2 are both C_4 -free and so, G satisfies Condition 2.5 by Lemma 3.11.

Thus we can conclude by [1] that G is a 1/2-hyperbolic graph. \square

COROLLARY 3.14. *There is a subcubic equivalence between the 1/2-hyperbolic graph recognition problem and the C_4 -free graph recognition problem.*

4. Finding a quadrangle. We will conclude this paper with an improved algorithm for the C_4 -free graph recognition problem, that hence improves the best-known upper-bound on the time complexity of both Problem 2.2 and Problem 2.6 by the subcubic equivalence of Corollary 3.14. While the algorithm proposed in [37] relies on transitive orientation, our algorithm merely reduces the whole Problem 2.6 to a fast rectangular matrix multiplication.

A quick reminding of the problem of matrix multiplication is given in §4.1, before we present our algorithm to detect an induced quadrangle in §4.2.

4.1. Fast rectangular matrix multiplication. The study of fast matrix multiplication mainly focuses on the $O(n^{\omega(1)})$ time complexity of multiplying two $n \times n$ matrices, also known as the square matrix product. Currently, $\omega(1)$ is known to be less than 2.3727 [38]. The *rectangular* matrix multiplication problem has received less attention, maybe because the product of an $n \times m$ matrix with an $m \times p$ matrix is known to be reducible to square matrix multiplications, yielding an $O(q^{\omega(1)-2} \cdot \max\{mn, mp, np\})$ -time complexity, for $q = \min\{m, n, p\}$ [25].

On the other hand, there is evidence that faster methods for the rectangular matrix multiplication which do not rely on the square matrix product may exist. This is known to be the case even for truly practical improvements of the matrix multiplication such as the Strassen algorithm and its variations [26, 28]. As stated below, the conjecture is (numerically) true w.r.t. the best known algorithms for square and rectangular matrix multiplications.

LEMMA 4.1 ([12, 25, 31, 38]). *Let $r \geq 1$ be a rational number. There exists a non-decreasing function $\omega : [1; +\infty[\rightarrow [2.3727; +\infty[$ such that multiplying an $n \times n^r$*

⁶A careful reader will remark that the choice of c determines the maximum number of calls in the reduction to the algorithm for detecting an induced quadrangle. There is a trade-off between this number and the running-time of the c -factor approximation algorithm.

matrix with an $n^r \times n$ matrix can be done in $O(n^{\omega(r)})$ -time.

Furthermore, $\omega(2) \leq 3.26$, and for every $r \geq 1$ we have $\omega(r) \leq r + \omega(1) - 1$.

Note that reducing the rectangular matrix product to square matrix multiplications would have only yielded $\omega(2) \leq 3.3727$.

A more efficient method is known for sparse matrices (e.g. see [41]).

4.2. An algorithm to count quadrangles in a graph. In this section, we essentially apply the local characterization of Fact 2.7. There are two main steps of computation in our algorithm, that are described below.

FACT 4.2 ([36]). *Given a graph $G = (V, E)$, let $A = (A_{u,v})_{u,v \in V}$ be the adjacency matrix of G .*

For every pair $u, v \in V$, we have $A_{u,v}^2 = |N(u) \cap N(v)|$.

Hence, $d(u, v) = 2$ if, and only if, $u \neq v$, $A_{u,v} = 0$ and $A_{u,v}^2 \neq 0$.

Proof. By the definition of matrix multiplication, we have for every pair $u, v \in V$ that:

$$\begin{aligned} A_{u,v}^2 &= \sum_{x \in V} A_{u,x} A_{x,v} = \sum_{x \in V} \mathbb{I}_{\{\{x,u\} \in E\}} \mathbb{I}_{\{\{x,v\} \in E\}} \\ &= \sum_{x \in V} \mathbb{I}_{\{x \in N(u)\}} \mathbb{I}_{\{x \in N(v)\}} = \sum_{x \in V} \mathbb{I}_{\{x \in N(u) \cap N(v)\}} \\ &= |N(u) \cap N(v)|. \end{aligned}$$

Moreover, $d(u, v) = 2$ if, and only if, $u \neq v$, u and v are not adjacent in G (e.g. $A_{u,v} = 0$), and $N(u) \cap N(v) \neq \emptyset$. Clearly, we have that $N(u) \cap N(v) \neq \emptyset$ if, and only if, $A_{u,v}^2 \neq 0$. \square

LEMMA 4.3. *Given a graph $G = (V, E)$, let $T = (T_{u,e})_{u \in V, e \in E}$ be such that $T_{u,e} = \mathbb{I}_{\{e \subseteq N(u)\}}$ for every $u \in V, e \in E$.*

For every pair $u, v \in V$, we have $TT_{u,v}^\top = |\{e \in E : e \subseteq N(u) \cap N(v)\}|$.

Proof. Similarly to the proof of Fact 4.2, we have by the definition of matrix multiplication that for every pair $u, v \in V$:

$$\begin{aligned} TT_{u,v}^\top &= \sum_{e \in E} T_{u,e} T_{e,v}^\top = \sum_{e \in E} T_{u,e} T_{v,e} = \sum_{e \in E} \mathbb{I}_{\{e \subseteq N(u)\}} \mathbb{I}_{\{e \subseteq N(v)\}} \\ &= \sum_{e \in E} \mathbb{I}_{\{e \subseteq N(u) \cap N(v)\}} = |\{e \in E : e \subseteq N(u) \cap N(v)\}|. \quad \square \end{aligned}$$

Combining Fact 4.2 and Lemma 4.3, we can now rely on Fact 2.7 in order to detect, to count and to output induced quadrangles in the graph as follows.

PROPOSITION 4.4. *Counting the number of induced quadrangles in a graph G , and returning an induced C_4 of G if any, can be done in $O(n^{\omega(\log_n m)}) = O(n^{3.26})$ -time.*

Proof. First, it is straightforward that we can compute the adjacency matrix A of G in quadratic-time. Using A , we can compute the matrix $T = (\mathbb{I}_{\{e \subseteq N(u)\}})_{u \in V, e \in E}$, hence the transpose matrix T^\top as well, as they are defined in Lemma 4.3, in $O(nm)$ -time.

Let us now compute A^2 and TT^\top . This can be done, respectively, in $O(n^{\omega(1)}) = O(n^{2.3727})$ -time and in $O(n^{\omega(\log_n m)}) = O(n^{\omega(2)}) = O(n^{3.26})$ -time by Lemma 4.1.

By Fact 2.7, G is C_4 -free if, and only if, for every pair $u, v \in V$ of non-adjacent vertices, we have that $N(u) \cap N(v)$ is a clique, e.g. that:

$$|\{e \in E : e \subseteq N(u) \cap N(v)\}| = \frac{|N(u) \cap N(v)|(|N(u) \cap N(v)| - 1)}{2}.$$

This is equivalent to have that $TT_{u,v}^\top = A_{u,v}^2(A_{u,v}^2 - 1)/2$ by Fact 4.2 and Lemma 4.3. Hence, it can be checked with an enumeration of all the possible pairs $u, v \in V$, in quadratic-time.

We actually note that if there is some pair $u, v \in V$ of non-adjacent vertices such that $TT_{u,v}^\top < A_{u,v}^2(A_{u,v}^2 - 1)/2$, then there are exactly $A_{u,v}^2(A_{u,v}^2 - 1)/2 - TT_{u,v}^\top$ induced quadrangles of G which contain this pair of vertices (that is one quadrangle for each of the missing edges in $G[N(u) \cap N(v)]$). Consequently, there are exactly $1/4 \sum_{u,v \in V} (1 - A_{u,v})(A_{u,v}^2(A_{u,v}^2 - 1)/2 - TT_{u,v}^\top)$ induced quadrangles in the graph, which can also be computed in quadratic-time, by an enumeration of all the possible pairs $u, v \in V$.

To conclude, observe that if $TT_{u,v}^\top \neq A_{u,v}^2(A_{u,v}^2 - 1)/2$ for some pair $u, v \in V$ of non-adjacent vertices, then it is straightforward to compute an induced quadrangle of G containing u, v in quadratic-time. \square

We remind the reader that there is only evidence that a fast rectangular matrix product can be computed faster than up to the reduction to fast square matrix multiplications. However, we want to highlight that, from a *theoretical* point of view, our algorithm for detecting an induced quadrangle is never slower than the algorithm of [37]. The dominant term for the complexity of our algorithm is indeed the fast rectangular matrix multiplication TT^\top (e.g. Lemma 4.3), which can be computed in $O(mn^{\omega(1)-1})$ -time in the worst-case, using the reduction to fast square matrix products that we described earlier in §4.1. In comparison, the time complexity of the algorithm of [37] is $O(n^{\omega(1)+1})$.

Also, we emphasize that for graphs with few C_3 's, a speed-up for the computation of TT^\top can be achieved using the results from [41] for sparse matrix multiplication. Indeed, the number of non-zero elements in the matrix T is exactly $3.t(G)$, where $t(G)$ denotes the number of C_3 in the graph.

5. Conclusion. In this work, we proved an interesting equivalence between the complexity of the purely *metric* problem of recognizing $1/2$ -hyperbolic graphs, and the purely *structural* problem of detecting an induced quadrangle in a graph. This shows a surprising gap in the complexity for recognizing graphs with small hyperbolicity, as in comparison there is a linear-time algorithm to decide whether a graph is 0-hyperbolic.

Our reduction being subcubic, it remains open whether $1/2$ -hyperbolic graphs can be recognized in linear-time, for some classes of graphs for which detecting an induced quadrangle is easy, like for instance planar graphs [20]. Also, it would be nice to extend our results to find a better upper-bound on the complexity of the problem of deciding if a graph is 1-hyperbolic. Note that this latter problem might be easier than the recognition of $1/2$ -hyperbolic graphs, as the true difficulty may only lie in the distinction between graphs with hyperbolicity *exactly* 1 and *exactly* $1/2$. Any recognition algorithm in $O(f(n))$ -time for 1-hyperbolic graphs would furthermore yield a 4-factor approximation algorithm for the hyperbolicity that runs in $\tilde{O}(f(n) + n^{\omega(1)})$ -time.

Acknowledgments. We wish to thank the referees for their reading of the first version of this manuscript, and their useful comments. We would also like to thank Frédéric Havet and Fatima Zahra Moataz for helpful comments on this work.

REFERENCES

- [1] HANS-JÜRGEN BANDELT AND VICTOR CHEPOI, *1-hyperbolic graphs*, SIAM Journal on Discrete Mathematics, 16 (2003), pp. 323–334.

- [2] ———, *Metric graph theory and geometry: a survey*, Contemporary Mathematics, 453 (2008), pp. 49–86.
- [3] H.-J. BANDELT AND H.M. MULDER, *Distance-hereditary graphs*, Journal of Combinatorial Theory, Series B, 41 (1986), pp. 182–208.
- [4] J. BARAS, *Hyperbolic embedding to the rescue in communication and social networks*, in Bell Labs-NIST Workshop on Large-Scale Networks, 2013.
- [5] S. BERMUDO, J.M. RODRÍGUEZ, J.M. SIGARRETA, AND J.-M. VILAIRE, *Gromov hyperbolic graphs*, Discrete Mathematics, 313 (2013), pp. 1575–1585.
- [6] MARIÁN BOGUÑÁ, FRAGKISKOS PAPADOPOULOS, AND DMITRI V. KRIOUKOV, *Sustaining the Internet with hyperbolic mapping*, Nature Communications, 1 (2010), pp. 1–18.
- [7] J. A. BONDY AND U. MURTY, *Graph theory with applications*, vol. 290, Macmillan London, 1976.
- [8] W. CARBALLOSA, J.M. RODRÍGUEZ, J.M. SIGARRETA, AND M. VILLETÀ, *Gromov hyperbolicity of line graphs*, The Electronic Journal of Combinatorics, 18 (2011), pp. 1–18.
- [9] J. CHALOPIN, V. CHEPOI, P. PAPASOGLU, AND T. PECATTE, *Cop and robber game and hyperbolicity*, Tech. Report arXiv:1308.3987, ArXiv, 2013.
- [10] V. CHEPOI AND F. DRAGAN, *A note on distance approximating trees in graphs*, European Journal of Combinatorics, 21 (2000), pp. 761–766.
- [11] NATHANN COHEN, DAVID COUDERT, AND AURÉLIEN LANCIN, *Exact and approximate algorithms for computing the hyperbolicity of large-scale graphs*, Rapport de recherche RR-8074, Inria, Sept. 2012.
- [12] D. COPPERSMITH, *Rectangular matrix multiplication revisited*, Journal of Complexity, 13 (1997), pp. 42–49.
- [13] P. DE LA HARPE AND E. GHYS, *Sur les groupes hyperboliques d’après Mikhael Gromov*, vol. 83, Progress in Mathematics, 1990.
- [14] R. DIESTEL, *Graph theory, graduate texts in mathematics, vol. 173*, Springer, Heidelberg, 1997.
- [15] D. DOR, S. HALPERIN, AND U. ZWICK, *All-pairs almost shortest paths*, SIAM Journal on Computing, 29 (2000), pp. 1740–1759.
- [16] F.F. DRAGAN, *Tree-like structures in graphs: A metric point of view*, in 39th International Workshop on Graph-Theoretic Concepts in Computer Science (WG), vol. 8165 of Lecture Notes in Computer Science, Springer, 2013, pp. 1–4.
- [17] F.F. DRAGAN AND E. KÖHLER, *An approximation algorithm for the tree t -spanner problem on unweighted graphs via generalized chordal graphs*, in Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, vol. 6845 of Lecture Notes in Computer Science, Springer, 2011, pp. 171–183.
- [18] ANDREAS DRESS, KATHARINA HUBER, JACOBUS KOOLEN, VINCENT MOULTON, AND ANDREAS SPILLNER, *Basic Phylogenetic Combinatorics*, Cambridge University Press, Cambridge, UK, Dec. 2011.
- [19] RAN DUAN, *Approximation algorithms for the gromov hyperbolicity of discrete metric spaces*, in 11th Latin American Theoretical Informatics Symposium (LATIN), vol. 8392 of Lecture Notes in Computer Science, Montevideo, Uruguay, 2014, Springer, pp. 285–293.
- [20] D. EPPSTEIN, *Subgraph isomorphism in planar graphs and related problems*, in 6th annual ACM-SIAM symposium on Discrete Algorithms (SODA), Society for Industrial and Applied Mathematics, 1995, pp. 632–640.
- [21] HERVÉ FOURNIER, ANAS ISMAIL, AND ANTOINE VIGNERON, *Computing the gromov hyperbolicity of a discrete metric space*, Tech. Report arXiv:1210.3323, ArXiv, Oct. 2012.
- [22] C. GAVOILLE, D. PELEG, S. PÉRENNES, AND R. RAZ, *Distance labeling in graphs*, in 12th annual ACM-SIAM symposium on Discrete algorithms (SODA), Society for Industrial and Applied Mathematics, 2001, pp. 210–219.
- [23] MICHA GROMOV, *Hyperbolic groups*, in Essays in Group Theory, S.M. Gersten, ed., vol. 8 of Mathematical Sciences Research Institute Publications, Springer, New York, 1987, pp. 75–263.
- [24] E. HOWORKA, *On metric properties of certain clique graphs*, Journal of Combinatorial Theory, Series B, 27 (1979), pp. 67–74.
- [25] X. HUANG AND V.Y. PAN, *Fast rectangular matrix multiplication and applications*, Journal of complexity, 14 (1998), pp. 257–299.
- [26] S. HUSS-LEDERMAN, E.M. JACOBSON, J.R. JOHNSON, A. TSAO, AND T. TURNBULL, *Implementation of strassen’s algorithm for matrix multiplication*, in ACM/IEEE Conference on Supercomputing, IEEE, 1996, pp. 32–32.
- [27] EDMOND A. JONCKHEERE AND POONSUK LOHSONTHORN, *A hyperbolic geometric approach to multipath routing*, in In Proc. 10th Mediterranean Conference on Control and Automation (MED 2002), Lisbon, Portugal, 2002.

- [28] P.A. KNIGHT, *Fast rectangular matrix multiplication and QR decomposition*, Linear algebra and its applications, 221 (1995), pp. 69–81.
- [29] A. KOSOWSKI, B. LI, N. NISSE, AND K. SUCHAN, *k-chordal graphs: From cops and robber to compact routing via treewidth*, in International Conference on Automata, Languages, and Programming (ICALP), vol. 7392 of Lecture Notes in Computer Science, Springer, 2012, pp. 610–622.
- [30] R. LASKAR AND D. SHIER, *On powers and centers of chordal graphs*, Discrete Applied Mathematics, 6 (1983), pp. 139–147.
- [31] FRANÇOIS LE GALL, *Faster algorithms for rectangular matrix multiplication*, in IEEE 53rd Annual Symposium on Foundations of Computer Science, New Brunswick, NJ, USA, 2012, IEEE, pp. 514–523.
- [32] DANIEL LOKSHTANOV, *Finding the longest isometric cycle in a graph*, Discrete Applied Mathematics, 157 (2009), pp. 2670–2674.
- [33] J. MICHEL, J.M. RODRÍGUEZ, J.M. SIGARRETA, AND M. VILLETA, *Hyperbolicity and parameters of graphs*, Ars Combinatoria, 100 (2011), pp. 43–63.
- [34] L. RODITY AND V. VASSILEVSKA WILLIAMS, *Minimum weight cycles and triangles: Equivalences and algorithms*, in 52nd IEEE Annual Symposium on Foundations of Computer Science (FOCS), IEEE, 2011, pp. 180–189.
- [35] J.M. RODRÍGUEZ AND J.M. SIGARRETA, *Bounds on Gromov hyperbolicity constant in graphs*, in Proceedings Indian Acad. Sci. (Mathematical Sciences), vol. 122, 2012, pp. 53–65.
- [36] R. SEIDEL, *On the all-pairs-shortest-path problem in unweighted undirected graphs*, Journal of Computer and System Sciences, 51 (1995), pp. 400–403.
- [37] J.P. SPINRAD, *Finding large holes*, Information Processing Letters, 39 (1991), pp. 227–229.
- [38] V. VASSILEVSKA WILLIAMS, *Multiplying matrices faster than coppersmith-Winograd*, in 44th symposium on Theory of Computing (STOC), ACM, 2012, pp. 887–898.
- [39] V. VASSILEVSKA WILLIAMS AND R. WILLIAMS, *Subcubic equivalences between path, matrix and triangle problems*, in 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS), IEEE, 2010, pp. 645–654.
- [40] YAOKUN WU AND CHENGPENG ZHANG, *Hyperbolicity and chordality of a graph*, The Electronic Journal of Combinatorics, 18 (2011), p. P43.
- [41] R. YUSTER AND U. ZWICK, *Fast sparse matrix multiplication*, ACM Transactions on Algorithms (TALG), 1 (2005), pp. 2–13.

APPENDIX D

On the hyperbolicity of bipartite and intersection graphs

On the hyperbolicity of bipartite graphs and intersection graphs

David Coudert^{a,b}, Guillaume Ducoffe^{b,a}

^a*Inria, France*

^b*Univ. Nice Sophia Antipolis, CNRS, I3S, UMR 7271, 06900 Sophia Antipolis, France*

Abstract

Hyperbolicity is a measure of the tree-likeness of a graph from a metric perspective. Recently, it has been used to classify complex networks depending on their underlying geometry. Motivated by a better understanding of the structure of graphs with bounded hyperbolicity, we here investigate on the hyperbolicity of bipartite graphs. More precisely, given a bipartite graph $B = (V_0 \cup V_1, E)$ we prove it is enough to consider any one side V_i of the bipartition of B to obtain a close approximate of its hyperbolicity $\delta(B)$ — up to an additive constant 2. We obtain from this result the sharp bounds $\delta(G) - 1 \leq \delta(L(G)) \leq \delta(G) + 1$ and $\delta(G) - 1 \leq \delta(K(G)) \leq \delta(G) + 1$ for every graph G , with $L(G)$ and $K(G)$ being respectively the line graph and the clique graph of G . Finally, promising extensions of our techniques to a broader class of intersection graphs are discussed and illustrated with the case of the biclique graph $BK(G)$, for which we prove $(\delta(G) - 3)/2 \leq \delta(BK(G)) \leq (\delta(G) + 3)/2$.

Keywords: Gromov hyperbolicity; bipartite graph; intersection graph; graph power; line graph; clique graph; biclique graph.

1. Introduction

The purpose of this paper is to bound the hyperbolicity of some classes of graphs that are defined in terms of graph operators. Roughly, hyperbolicity is a tree-likeness parameter that measures how close the shortest-path

^{*}This work is partially supported by ANR project Stint under reference ANR-13-BS02-0007 and ANR program “Investments for the Future” under reference ANR-11-LABX-0031-01.

metric of a graph is to a tree metric (the smaller the hyperbolicity the closer the graph is to a metric tree). It has thus been proposed to take hyperbolicity into account to better classify complex networks [22]. For instance, it has been experimentally shown in [22] that social networks and protein interaction networks have bounded hyperbolicity while it is not the case for road networks. Another interest for hyperbolicity is that it helps analyzing some graph heuristics on large-scale networks. A good example to this is the *2-sweep* heuristic for computing the diameter, that provides very good results in practice [23]; such good results can be explained assuming a bounded hyperbolicity [11].

Relating the structural properties of graphs with hyperbolicity can be useful in this context, and it has become a growing line of research (*e.g.*, see [12, 15, 24, 31]). Indeed, we argue that one can obtain from such relations a comprehensive overview of the reasons why some complex networks are hyperbolic and some others are not. Following this line, we proved in [15] that most data center interconnection networks are *not* hyperbolic because they are symmetric graphs. As an attempt to go further in this direction, we here investigate on the hyperbolicity of bipartite graphs. In fact, we were motivated at first to bound the hyperbolicity of *line graphs* [30], that are intersection graphs of edges in a graph and have already received some attention in the literature of graph hyperbolicity [8, 9]. In this paper, we fully characterize what can be the defect between the hyperbolicity of a given graph and the hyperbolicity of its line graph, using an original connection with bipartite graphs. To better depict our novel approach, let us first recall that intersection graphs over a ground set S have for vertices a family of subsets of S with an edge between every two intersecting subsets. Therefore, they can be naturally represented as a bipartite graph — with vertices of the graph on one side, the ground set S on the other side, and an edge between every element of S and the subsets that contain it. Our main contribution is to show how we can use this representation so as to bound the hyperbolicity of intersection graphs—. This simple framework does not only apply to line graphs. We can use it to bound the hyperbolicity of *clique graphs* [20] and (with slightly more work) *biclique graphs* [19]. Overall, our main results can be expressed as follows.

- Given a bipartite graph $B = (V_0 \cup V_1, E)$, for every $i \in \{0, 1\}$ let G_i be the graph with vertex-set the side V_i and with an edge between every two vertices that share a common neighbor in B . We prove that $2\delta(G_i) \leq \delta(B) \leq 2\delta(G_i) + 2$ and the bounds are sharp (Theorem 4).
- We deduce from the above inequalities that $\delta(G) - 1 \leq \delta(L(G)) \leq$

$\delta(G) + 1$ for every graph G , with $L(G)$ being the line graph of G (Theorem 6). Furthermore we show that all possible cases (between $\delta(G) - 1$ and $\delta(G) + 1$) can happen. This complements the bounds in [8, 9] that are proved to be sharp only for cycles (but with an alternative definition of hyperbolicity).

- By applying the same technique as for line graphs, we are the first to bound the hyperbolicity of clique graphs, *a.k.a.*, the intersection graphs of maximal cliques. More precisely, we prove that $\delta(G) - 1 \leq \delta(K(G)) \leq \delta(G) + 1$ for every graph G , with $K(G)$ being the clique graph of G , and all possible cases between $\delta(G) - 1$ and $\delta(G) + 1$ can happen (Theorem 8).
- We introduce graph powers [3] in our framework to obtain bounds on the hyperbolicity of other graphs. As example we prove that $(\delta(G) - 3)/2 \leq \delta(BK(G)) \leq (\delta(G) + 3)/2$ for every graph G , with $BK(G)$ being the biclique graph of G (Theorem 13). Bicliques are maximal induced complete bipartite subgraphs and they have gained recent attention in graph theory and graph algorithms. We refer to [19] and the papers cited therein for details.
- Finally, we bound the hyperbolicity of some other extensions of line graphs using our framework (Section 4.4), namely the *incidence graph*, the *total graph* [5], the *middle graph* [27], and the *k-edge graph* [26] of G .

Definitions and useful notations are given in Section 2.

2. Definitions and notations

We will follow the graph terminology in [6, 17]. Graphs in this study are connected, unweighted and finite (although part of the results extend to infinite weighted graphs). Given a graph $G = (V, E)$, the distance between every two vertices u, v in V equals the minimum number of edges on an uv -path. We will denote the distance between u and v by $d_G(u, v)$, or simply $d(u, v)$ when G is clear from the context. Informally, we are interested in this paper in embedding the vertices of G into a tree T (possibly, edge-weighted) while minimizing the additive distortion of the distances in G . Hyperbolicity is both a lower-bound and an $\mathcal{O}(\log |V|)$ -approximation for the minimum possible distortion [18]. Finer-grained relations between hyperbolicity and the minimum possible distortion for graphs are discussed in [16].

Definition 1 (4-points Condition, [18]). Let $G = (V, E)$ be a connected graph.

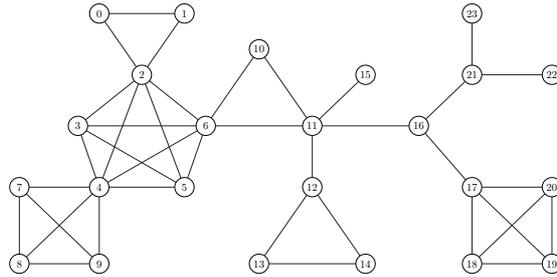
For every 4-tuple u, v, x, y of V , we define $\delta(u, v, x, y)$ as half of the difference between the two largest sums amongst:

$$S_1 = d(u, v) + d(x, y), S_2 = d(u, x) + d(v, y), \text{ and } S_3 = d(u, y) + d(v, x).$$

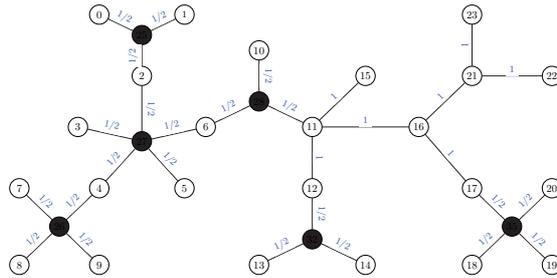
The graph hyperbolicity, denoted by $\delta(G)$, is equal to $\max_{u, v, x, y \in V} \delta(u, v, x, y)$.

Moreover, we say that G is δ -hyperbolic for every $\delta \geq \delta(G)$.

It is well-known that 0-hyperbolic graphs are exactly those that can be embedded into a tree without any distortion, including trees and complete graphs. In fact, 0-hyperbolic graphs coincide with the *block graphs*, that are graphs whose all biconnected components are cliques (see Figure 1 for an illustration) [4, 21]. The class of 1/2-hyperbolic graphs has also been characterized in [2, 14].



(a) A block graph G .



(b) An embedding of G into an edge-weighted tree with null distortion.

Figure 1: Block graphs are exactly the 0-hyperbolic graphs.

Furthermore, it turns out that not all 4-tuples in the graph need to be

considered for the computation of hyperbolicity. This crucial point is the cornerstone of the most efficient algorithms so far to compute this parameter [7, 13]. Here we will use this observation to gain more insights on 4-tuples with maximum hyperbolicity in our proofs. This will require us to introduce the central notion of far-apart pairs.

Definition 2 (Far-apart pair [25, 28]). *Given $G = (V, E)$, the pair (u, v) is far-apart if for every $w \in V \setminus \{u, v\}$, we have $d(w, u) + d(u, v) > d(w, v)$ and $d(w, v) + d(u, v) > d(w, u)$.*

Said differently, far-apart pairs are the ends of maximal shortest-paths in the graph. Their key property is that there always exists a 4-tuple with maximum hyperbolicity which contains two far-apart pairs.

Lemma 3 ([25, 28]). *Given $G = (V, E)$, there exist two far-apart pairs (u, v) and (x, y) satisfying:*

- i) $d_G(u, v) + d_G(x, y) \geq \max\{d_G(u, x) + d_G(v, y), d_G(u, y) + d_G(v, x)\}$;
- ii) $\delta(u, v, x, y) = \delta(G)$.

3. New bounds on the hyperbolicity of bipartite graphs

Let us start proving our main tool for the remaining of the paper, that is Theorem 4. Informally, we will consider a bipartite graph $B = (V_0 \cup V_1, E)$ as obtained from two smaller intersection graphs G_0 and G_1 , each having one side of the bipartition as its vertex-set. Our goal is to bound $\delta(B)$ depending on $\delta(G_i)$, for any $i \in \{0, 1\}$. In fact, since any side V_i is a dominating set of B , then it is not hard to prove that $\delta(B) \leq 2\delta(G_i) + 4$ (using the the 4-point Condition of Definition 1). The main difficulty is to obtain the sharp upper-bound $\delta(B) \leq 2\delta(G_i) + 2$, for which we will need far-apart pairs.

Theorem 4. *Let $B = (V_0 \cup V_1, E)$ be a bipartite graph. We have $\delta_B(V_i) \leq \delta(B) \leq \delta_B(V_i) + 2$, where $\delta_B(V_i) = \max_{u, v, x, y \in V_i} \delta_B(u, v, x, y)$ for every $i \in \{0, 1\}$, and these bounds are sharp.*

Proof. We will only need to consider the upper-bound $\delta(B) \leq \delta_B(V_i) + 2$, for the lower-bound $\delta_B(V_i) \leq \delta(B)$ trivially follows from the 4-points Condition of Definition 1. To prove the upper-bound, let (u, v) and (x, y) be two far-apart pairs of B such that $S_1 = d(u, v) + d(x, y) \geq \max\{d(u, x) + d(v, y), d(u, y) + d(v, x)\} = S_2$ and $\delta(u, v, x, y) = \delta(B)$, that exist by Lemma 3. Note that $\delta(B) = (S_1 - S_2)/2$.

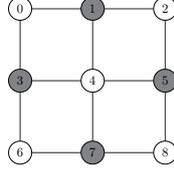


Figure 2: The bipartite graph $G_{3,3}$ with each side of the bipartition colored differently.

We claim that there are $u', v' \in V_i$ such that $\delta(u, v, x, y) \leq \delta(u', v', x, y) + 1$. To prove the claim assume $\delta(u, v, x, y) > 0$ (or else, it is trivial). The latter implies (by Definition 1) that u, v, x, y are pairwise different. There are three cases to be considered.

- If $u, v \in V_i$, then we are done by setting $u' = u$ and $v' = v$.
- If $u \in V_i$ and $v \notin V_i$ (resp., $u \notin V_i$ and $v \in V_i$), let us set $u' = u$ and $v' \in N(v)$ (resp., $u' \in N(u)$ and $v' = v$). In such case let $S'_1 = d(u', v') + d(x, y)$ and let $S'_2 = \max\{d(u', x) + d(v', y), d(u', y) + d(v', x)\}$. By the triangular inequality $|S_1 - S'_1| \leq 1$ and similarly $|S_2 - S'_2| \leq 1$. Therefore, either $S'_1 < S'_2$ and so, $\delta(u, v, x, y) = (S_1 - S_2)/2 \leq (S'_1 - S'_2 + 2)/2 < 1 \leq \delta(u', v', x, y) + 1$, or $S'_1 \geq S'_2$ and so, $\delta(u, v, x, y) = (S_1 - S_2)/2 \geq (S'_1 - S'_2)/2 \geq (S_1 - S_2 - 2)/2 = \delta(u', v', x, y) - 1$.
- Else, $u, v \notin V_i$. In particular, $N(u) \subseteq V_i$ and $N(v) \subseteq V_i$ because B is bipartite by the hypothesis. We will prove as an intermediate subclaim that for every pair (u', v') with $u' \in N(u)$ and $v' \in N(v)$, we have either $d(u', v') = d(u, v)$ or $d(u', v') = d(u, v) - 2$. Indeed, $d(u, v) - 2 \leq d(u', v') \leq d(u, v) + 2$ by the triangular inequality, and so, since the pairs (u, v) and (u', v') are in distinct sides of the bipartition of B , either $d(u', v') = d(u, v) - 2$ or $d(u', v') = d(u, v)$ or $d(u', v') = d(u, v) + 2$. The latter case, $d(u', v') = d(u, v) + 2$, would contradict the fact that (u, v) is far-apart. Hence either $d(u', v') = d(u, v)$ or $d(u', v') = d(u, v) - 2$, which proves the subclaim. Now there are two subcases to be considered.
 - Suppose there are $u' \in N(u)$ and $v' \in N(v)$ such that $d(u', v') = d(u, v)$. Let $S'_1 = d(u', v') + d(x, y)$ and let $S'_2 = \max\{d(u', x) + d(v', y), d(u', y) + d(v', x)\}$. By the choice of u' and v' , we have $S'_1 = S_1$ while $|S_2 - S'_2| \leq 2$ by the triangular inequality. As

- a result, either $S'_1 < S'_2$ and so, $\delta(u, v, x, y) = (S_1 - S_2)/2 \leq (S'_1 - S'_2 + 2)/2 < 1 \leq \delta(u', v', x, y) + 1$, or $S'_1 \geq S'_2$ and so, $\delta(u', v', x, y) = (S'_1 - S'_2)/2 \geq (S_1 - S_2 - 2)/2 = \delta(u, v, x, y) - 1$.
- Else, for every $u' \in N(u)$ and $v' \in N(v)$, we have $d(u', v') = d(u, v) - 2$. Let $u' \in N(u)$ and $v' \in N(v)$ satisfy $d(u, x) = 1 + d(u', x)$ and $d(v, x) = 1 + d(v', x)$. We set as before $S'_1 = d(u', v') + d(x, y)$ and $S'_2 = \max\{d(u', x) + d(v', y), d(u', y) + d(v', x)\}$. By the choice of u' and v' , we have $S'_1 = S_1 - 2$. Furthermore, $S_2 - 2 \leq S'_2 \leq S_2$ because $d(u', x) = d(u, x) - 1$, $d(v', x) = d(v, x) - 1$, and $|d(u', y) - d(u, y)| \leq 1$ and $|d(v', y) - d(v, y)| \leq 1$ by the triangular inequality. It follows that either $S'_1 < S'_2$ and so, $\delta(u, v, x, y) = (S_1 - S_2)/2 \leq (S'_1 - S'_2 + 2)/2 < 1 \leq \delta(u', v', x, y) + 1$, or $S'_1 \geq S'_2$, and so $\delta(u', v', x, y) = (S'_1 - S'_2)/2 \geq (S_1 - S_2 - 2)/2 \geq \delta(u, v, x, y) - 1$, that achieves proving the claim.

Finally, since the pair (x, y) is also far-apart, there exist $x', y' \in V_i$ such that $\delta(u', v', x, y) \leq \delta(u', v', x', y') + 1$. As a result, $\delta(B) = \delta(u, v, x, y) \leq \delta(u', v', x, y) + 1 \leq \delta(u', v', x', y') + 2 \leq \delta_B(V_i) + 2$.

To show that the bounds are sharp, let us consider the square grid $G_{3,3}$ of side length two as drawn in Figure 2. This bipartite graph has vertex-set $V = V_0 \cup V_1$, with $V_0 = \{0, 2, 4, 6, 8\}$ and $V_1 = \{1, 3, 5, 7\}$. We have $\delta(G_{3,3}) = 2$, that is reached with the four corners 0, 2, 6 and 8 (i.e., $\delta(0, 2, 6, 8) = 2$). On the one hand, side V_0 contains the four corners and so $\delta_{G_{3,3}}(V_0) = \delta(G_{3,3}) = 2$. On the other hand, vertices on the other side V_1 are exactly the four neighbors of vertex 4, and so $\delta_{G_{3,3}}(V_1) = 0 = \delta(G_{3,3}) - 2$. \square

4. Applications to intersection graphs

Our main results in this section are (sharp) lower and upper-bounds on the hyperbolicity of intersection graphs that have been considered in the literature. These comprise the two well-known families of line graphs and clique graphs (we refer to [1, 29] for surveys), along with biclique graphs that have been introduced more recently as extensions of line graphs.

4.1. Line graph

Definition 5. Given $G = (V, E)$, the line-graph of G , denoted by $L(G)$, is the intersection graph of E . That is, it has vertex-set E and for every $e, e' \in E$ there is an edge $\{e, e'\}$ in $L(G)$ if and only if e and e' share an end in G .

Theorem 6. For every graph G , $\delta(G) - 1 \leq \delta(L(G)) \leq \delta(G) + 1$, and these bounds are sharp.

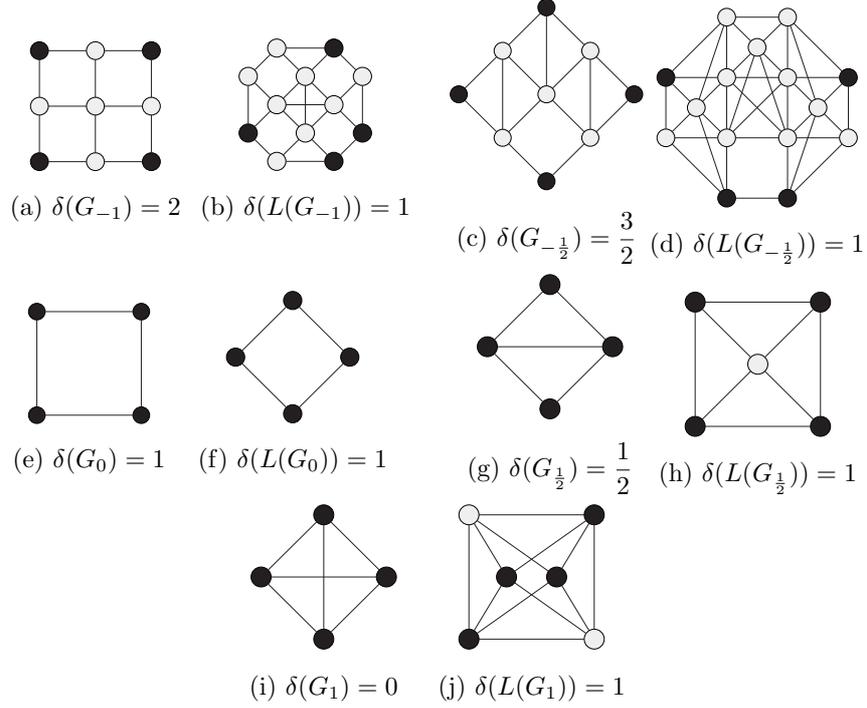


Figure 3: Examples of graphs G_i with $\delta(L(G_i)) = \delta(G_i) + i$ for every $i \in \{-1, -1/2, 0, +1/2, +1\}$. A 4-tuple with maximum hyperbolicity is drawn in bold on each graph.

Proof. Let B be the incidence graph of G , that is, it has vertex-set $V \cup E$ and there is an edge in B between $u \in V$ and $e \in E$ if and only if u is an end of e in G . By Theorem 4, $\delta_B(V) \leq \delta(B) \leq \delta_B(V) + 2$ and similarly $\delta_B(E) \leq \delta(B) \leq \delta_B(E) + 2$. Furthermore by construction $d_B(u, v) = 2d_G(u, v)$ for every $u, v \in V$ and in the same way $d_B(e, e') = 2d_{L(G)}(e, e')$ for every $e, e' \in E$. As a result, $\delta_B(V) = 2\delta(G)$, similarly $\delta_B(E) = 2\delta(L(G))$, and so,

$$2\delta(G) \leq \delta(B) \leq 2\delta(G) + 2,$$

$$2\delta(L(G)) \leq \delta(B) \leq 2\delta(L(G)) + 2.$$

By mixing up the two chains of inequality one obtains $2\delta(G) \leq 2\delta(L(G)) + 2$ and $2\delta(L(G)) \leq 2\delta(G) + 2$, whence $\delta(G) \leq \delta(L(G)) + 1$ and $\delta(L(G)) \leq \delta(G) + 1$, as desired.

To show that the bounds are sharp, consider the graphs G_{-1} and G_1 as drawn respectively in Figures 3a and 3i. We have $\delta(L(G_{-1})) = \delta(G_{-1}) - 1$ and $\delta(L(G_1)) = \delta(G_1) + 1$. \square

In Figure 3 we show that all possible cases of Theorem 6 (with defect between -1 and $+1$) are realized by some graphs. By taking the incidence graphs of G_{-1} and G_1 , one obtains a new proof that the bounds of Theorem 4 are sharp.

4.2. Clique graph

Definition 7. Given $G = (V, E)$, let Ω be the set of all maximal cliques of G . The clique-graph of G , denoted by $K(G)$, is the intersection graph of Ω . That is, it has vertex-set Ω and for every $S, S' \in \Omega$ there is an edge $\{S, S'\}$ in $K(G)$ if and only if the two cliques S and S' intersect.

Theorem 8. For every graph G , $\delta(G) - 1 \leq \delta(K(G)) \leq \delta(G) + 1$, and these bounds are sharp.

Proof. Let B be the bipartite graph defined as follows. It has vertex-set $V \cup \Omega$ and there is an edge between $u \in V$ and $S \in \Omega$ if and only if $u \in S$. By Theorem 4, $\delta_B(V) \leq \delta(B) \leq \delta_B(V) + 2$ and similarly $\delta_B(\Omega) \leq \delta(B) \leq \delta_B(\Omega) + 2$. Furthermore, $d_B(S, S') = 2d_{K(G)}(S, S')$ for every $S, S' \in \Omega$ by construction, so, $\delta_B(\Omega) = 2\delta(K(G))$. We claim in addition that $d_B(u, v) = 2d_G(u, v)$ for every $u, v \in V$. To prove the claim it is enough to prove $d_B(u, v) = 2$ if and only if u and v are adjacent in G . By construction, $d_B(u, v) = 2$ if and only if there is $S \in \Omega$ such that $u, v \in S$. If $u, v \in S$ for some $S \in \Omega$ then u and v are adjacent in G because S is a clique of G , conversely if u and v are adjacent in G then $u, v \in S$ with S being any maximal clique containing the edge $\{u, v\}$. Therefore, the claim is proved, and so, since $d_B(u, v) = 2d_G(u, v)$ for every $u, v \in V$, $\delta_B(V) = 2\delta(G)$. As a result:

$$2\delta(G) \leq \delta(B) \leq 2\delta(G) + 2,$$

$$2\delta(K(G)) \leq \delta(B) \leq 2\delta(K(G)) + 2.$$

By mixing up the two chains of inequality one obtains $2\delta(G) \leq 2\delta(K(G)) + 2$ and $2\delta(K(G)) \leq 2\delta(G) + 2$, whence $\delta(G) \leq \delta(K(G)) + 1$ and $\delta(K(G)) \leq \delta(G) + 1$, as desired.

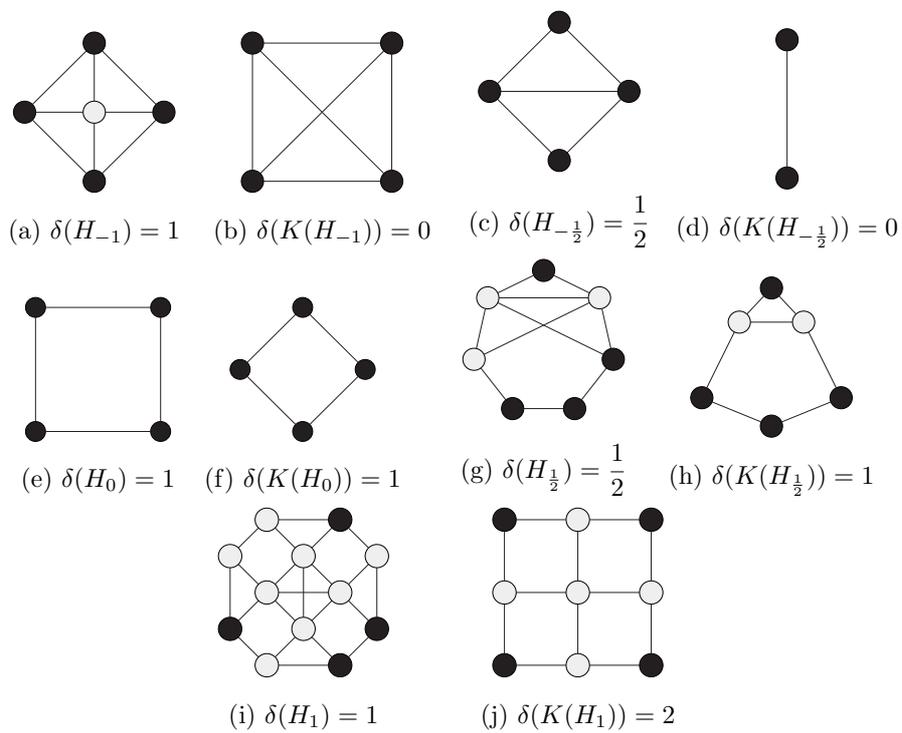


Figure 4: Examples of graphs H_i with $\delta(K(H_i)) = \delta(H_i) + i$ for every $i \in \{-1, -1/2, 0, +1/2, +1\}$. A 4-tuple with maximum hyperbolicity is drawn in bold on each graph.

To show that the bounds are sharp, consider the graphs H_{-1} and H_1 as drawn respectively in Figures 4a and 4i. We have $\delta(K(H_{-1})) = \delta(H_{-1}) - 1$ and $\delta(K(H_1)) = \delta(H_1) + 1$. \square

In Figure 4 we show that all possible cases of Theorem 8 (with defect between -1 and $+1$) are realized by some graphs. Note that $H_{-1} = L(G_{\frac{1}{2}})$, $H_{-\frac{1}{2}} = G_{\frac{1}{2}}$, $H_0 = G_0$ and $H_1 = L(G_{-1})$.

4.3. Biclique graph

The above two examples of line graphs and clique graphs are intersection graphs of *cliques*. However, there are interesting graph families that are defined as the intersection graphs of some subgraphs of diameter larger than one. As a general method to overcome this difficulty, we now introduce graph powers in our framework.

Definition 9. Given $G = (V, E)$ and $k \geq 1$, the k^{th} -power of G , denoted by G^k , is defined as follows. It has vertex-set V and for every $u, v \in V$ there is an edge $\{u, v\}$ in G^k if and only if $d_G(u, v) \leq k$.

Pushing further a previous result from [14], let us bound the hyperbolicity of graph powers (Proposition 11). We will need the following intermediate lemma.

Lemma 10 ([3]). Given $G = (V, E)$ and $k \geq 1$, $d_{G^k}(u, v) = \left\lceil \frac{d_G(u, v)}{k} \right\rceil$ for every $u, v \in V$.

Proposition 11. For every graph G and $k \geq 2$, $\frac{\delta(G) + 1}{k} - 1 \leq \delta(G^k) \leq \frac{\delta(G) - 1}{k} + 1$, and these bounds are sharp.

Proof. Let $u, v, x, y \in V$ be arbitrary. Assume w.l.o.g. $S_1 = d_G(u, v) + d_G(x, y) \geq S_2 = d_G(u, x) + d_G(v, y) \geq S_3 = d_G(u, y) + d_G(v, x)$. In order to prove Proposition 11, we will need to prove some relations between the hyperbolicity $\delta_G(u, v, x, y)$ of the 4-tuple in G and the hyperbolicity $\delta_{G^k}(u, v, x, y)$ of the 4-tuple in G^k . Let $S'_1 = d_{G^k}(u, v) + d_{G^k}(x, y)$, $S'_2 = d_{G^k}(u, x) + d_{G^k}(v, y)$ and $S'_3 = d_{G^k}(u, y) + d_{G^k}(v, x)$. By Lemma 10, we have:

$$S'_1 = \left\lceil \frac{d_G(u, v)}{k} \right\rceil + \left\lceil \frac{d_G(x, y)}{k} \right\rceil, \quad S'_2 = \left\lceil \frac{d_G(u, x)}{k} \right\rceil + \left\lceil \frac{d_G(v, y)}{k} \right\rceil \quad \text{and}$$

$$S'_3 = \left\lceil \frac{d_G(u, y)}{k} \right\rceil + \left\lceil \frac{d_G(v, x)}{k} \right\rceil.$$

In particular $S_i/k \leq S'_i \leq S_i/k + 2(1 - 1/k)$ for every $1 \leq i \leq 3$. Since $2(1 - 1/k) < 2$, there can be no more than two integers between S_i/k and $S_i/k + 2(1 - 1/k)$, that implies either $S'_i = \lceil S_i/k \rceil$ or $S'_i = \lceil S_i/k \rceil + 1$. Now there are two cases to be considered.

- Suppose $S'_1 < S'_j$ with $S'_j = \max\{S'_2, S'_3\}$. Then it must be the case that $S'_1 = \lceil S_1/k \rceil$ and $S'_j = \lceil S_j/k \rceil + 1$ because $\lceil S_1/k \rceil \geq \lceil S_j/k \rceil$. The latter implies

$$\begin{aligned} \delta_{G^k}(u, v, x, y) &\leq \frac{S'_j - S'_1}{2} \leq \frac{\lceil \frac{S_j}{k} \rceil + 1 - \lceil \frac{S_1}{k} \rceil}{2} \\ &\leq \frac{1}{2} \leq 1 - \frac{1}{k} \leq \frac{\delta_G(u, v, x, y) - 1}{k} + 1 \end{aligned} \quad (1)$$

$$\begin{aligned} \delta_G(u, v, x, y) &\leq \frac{S_1 - S_j}{2} \leq k \frac{\frac{S_1}{k} - \frac{S_j}{k}}{2} \leq k \left[\frac{S'_1 - S'_j}{2} + 1 - \frac{1}{k} \right] \\ &\leq -\frac{k}{2} + k - 1 \leq \frac{k}{2} - 1 \end{aligned} \quad (2)$$

Furthermore if $\delta_G(u, v, x, y) \leq k/2 - 1$ then $(\delta_G(u, v, x, y) + 1)/k - 1 < 0 \leq \delta_{G^k}(u, v, x, y)$.

- Else, $S'_1 \geq \max\{S'_2, S'_3\}$. In such case $\delta_{G^k}(u, v, x, y) = (S'_1 - \max\{S'_2, S'_3\})/2$. Moreover, $S_2/k \leq \max\{S'_2, S'_3\} \leq S_2/k + 2(1 - 1/k)$ because $S_2 \geq S_3$. Therefore,

$$\delta_{G^k}(u, v, x, y) \geq \frac{\frac{S_1}{k} - \frac{S_2}{k} - 2(1 - \frac{1}{k})}{2} \geq \frac{\delta_G(u, v, x, y)}{k} - 1 + \frac{1}{k} \quad (3)$$

$$\delta_{G^k}(u, v, x, y) \leq \frac{\frac{S_1}{k} + 2(1 - \frac{1}{k}) - \frac{S_2}{k}}{2} \leq \frac{\delta_G(u, v, x, y)}{k} + 1 - \frac{1}{k} \quad (4)$$

It follows that $(\delta_G(u, v, x, y) + 1)/k - 1 \leq \delta_{G^k}(u, v, x, y) \leq (\delta_G(u, v, x, y) - 1)/k + 1$ in both cases.

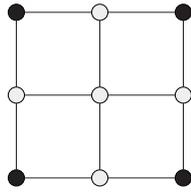
- When u, v, x, y maximizes δ_G the first inequality leads to $(\delta(G) + 1)/k - 1 \leq \delta_{G^k}(u, v, x, y) \leq \delta(G^k)$.
- When it maximizes δ_{G^k} the second inequality leads to $\delta(G^k) \leq (\delta_G(u, v, x, y) - 1)/k + 1 \leq (\delta(G) - 1)/k + 1$.

Let us finally show that the bounds of Proposition 11 are sharp. Indeed, on the one hand the cycle C_4 with four vertices satisfies $\delta(C_4) = 1$ and $C_4^2 =$

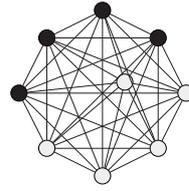
K_4 , the clique with four vertices. Therefore, $\delta(C_4^2) = 0 = (\delta(C_4) + 1)/2 - 1$. On the other hand the rectangular grid $G_{2,3}$ (obtained from two C_4 's sharing exactly one edge) satisfies $\delta(G_{2,3}) = 1$, and its four borders induce a C_4 in $G_{2,3}^2$. Consequently, $\delta(G_{2,3}^2) \geq 1 \geq (\delta(G_{2,3}) - 1)/2 + 1$. \square

We will illustrate the benefit of using graph powers within our framework through the case of *biclique graphs*, that are defined as follows.

Definition 12. Given $G = (V, E)$, the set $S \subseteq V$ is a *biclique* of G if it induces a complete bipartite subgraph of G . Let Σ be all maximal bicliques of G . The *biclique graph* of G , denoted by $BK(G)$, is the intersection graph of Σ . That is, it has vertex-set Σ and for every $S, S' \in \Sigma$ there is an edge $\{S, S'\}$ in $BK(G)$ if and only if the two bicliques S and S' intersect.



(a) $\delta(G_{3,3}) = 2$



(b) $\delta(BK(G_{3,3})) = 0$

Figure 5: The grid graph $G_{3,3}$ along with its biclique graph, that is the complete graph K_9 with nine vertices. A 4-tuple with maximum hyperbolicity is drawn in bold on each graph.

For instance, the biclique graph of a complete graph is exactly its line graph. In Figure 5 we consider the biclique graph of the grid $G_{3,3}$. The maximal bicliques of this grid comprise four cycles of length four, four stars with three branches each and one star with four branches. All of these pairwise intersect at the central vertex of the grid, therefore, $BK(G_{3,3}) = K_9$, the complete graph with nine vertices.

Theorem 13. For every graph G , $(\delta(G) - 3)/2 \leq \delta(BK(G)) \leq (\delta(G) + 3)/2$.

Proof. Let B be the bipartite graph defined as follows. It has vertex-set $V \cup \Sigma$ and there is an edge between $u \in V$ and $S \in \Sigma$ if and only if $u \in S$. By Theorem 4, $\delta_B(V) \leq \delta(B) \leq \delta_B(V) + 2$ and similarly $\delta_B(\Sigma) \leq \delta(B) \leq \delta_B(\Sigma) + 2$. Furthermore, $d_B(S, S') = 2d_{BK(G)}(S, S')$ for every $S, S' \in \Sigma$ by construction, so, $\delta_B(\Sigma) = 2\delta(BK(G))$. We claim in addition that $d_B(u, v) = 2d_{G^2}(u, v)$ for every $u, v \in V$, with G^2 be defined as in

Definition 9. To prove the claim it is enough to prove $d_B(u, v) = 2$ if and only if $d_G(u, v) \leq 2$. By construction, $d_B(u, v) = 2$ if and only if there is $S \in \Sigma$ such that $u, v \in S$. If $u, v \in S$ for some $S \in \Sigma$ then $d_G(u, v) \leq 2$ because S induces a complete bipartite subgraph of G , conversely if $d_G(u, v) \leq 2$ then every uv -shortest-path P in G induces a complete bipartite subgraph of G (with one side containing one vertex and the other side containing one or two vertices) and so, $u, v \in S$ with S being any maximal biclique containing P . Therefore, the claim is proved, and since $d_B(u, v) = 2 d_{G^2}(u, v)$ for every $u, v \in V$, we obtain $\delta_B(V) = 2\delta(G^2)$. As a result:

$$2\delta(G^2) \leq \delta(B) \leq 2\delta(G^2) + 2,$$

$$2\delta(BK(G)) \leq \delta(B) \leq 2\delta(BK(G)) + 2.$$

By mixing up the two chains of inequality one obtains $2\delta(G^2) \leq 2\delta(BK(G)) + 2$ and $2\delta(BK(G)) \leq 2\delta(G^2) + 2$, whence $\delta(G^2) \leq \delta(BK(G)) + 1$ and $\delta(BK(G)) \leq \delta(G^2) + 1$. Since by Proposition 11 $(\delta(G) - 1)/2 \leq \delta(G^2) \leq (\delta(G) + 1)/2$, one obtains $\delta(BK(G)) \geq \delta(G^2) - 1 \geq (\delta(G) - 3)/2$ and $\delta(BK(G)) \leq (\delta(G) + 1)/2 + 1 \leq (\delta(G) + 3)/2$, as desired. \square

In fact, we prove the more precise inequalities $\delta(G^2) - 1 \leq \delta(BK(G)) \leq \delta(G^2) + 1$ for every graph G , and we claim these bounds are sharp.

Corollary 14. *For every graph G , $\delta(G^2) - 1 \leq \delta(BK(G)) \leq \delta(G^2) + 1$, and these bounds are sharp.*

Proof. The bounds are given by Theorem 13. Also, let us now show that they are sharp.

Consider first the grid graph $G_{3,3}$. We have $\delta(BK(G_{3,3})) = \delta(K_9) = 0$, while $\delta(G_{3,3}^2) \geq 1$ because the four corners of the grid induce a cycle of length four in the square graph $G_{3,3}^2$. As a result, $\delta(BK(G_{3,3})) = \delta(G_{3,3}^2) - 1$, and so the lower-bound is reached.

Now, recall that the biclique graph of a complete graph K_n is exactly the line graph $L(K_n)$. Therefore, consider the graph K_4 and its line graph (G_1 and $L(G_1)$ in Figures 3i and 3j). Since $K_4^2 = K_4$ then it is indeed the case that $\delta(BK(K_4)) = \delta(L(K_4)) = \delta(K_4) + 1 = \delta(K_4^2) + 1$, and so the upper-bound is also reached. \square

4.4. Additional bounds

Before we conclude this paper, let us present a few other results that are obtained within our framework. More precisely, given a graph $G = (V, E)$

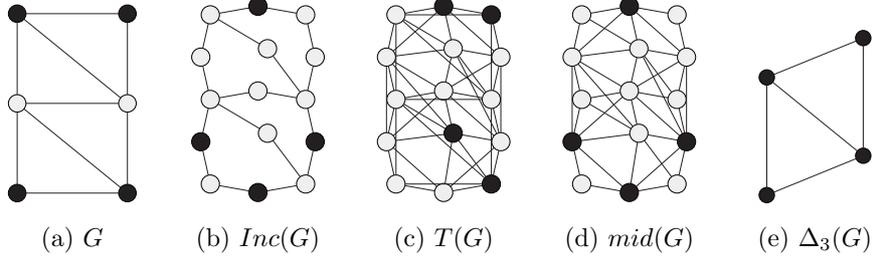


Figure 6: Some intersection graphs obtained from a triangular grid graph G . We have $\delta(G) = 1/2$, $\delta(Inc(G)) = 2$, $\delta(T(G)) = 1$, $\delta(mid(G)) = 1$, and $\delta(\Delta_3(G)) = 1/2$.

we consider the following extensions of line graphs (illustrations for each case are given in Figure 6).

The *incidence graph* of G , denoted by $Inc(G)$, has vertex set $V \cup E$ with an edge between every $u \in V$ and every $e \in E$ such that u is an end of e in G (see Figure 6b). It follows from the proof of Theorem 6 (for line graphs) that $2\delta(G) \leq \delta(Inc(G)) \leq 2\delta(G) + 2$ and the bounds are sharp.

The *total graph* of G [5], denoted by $T(G)$, is constructed from G and $L(G)$ by adding an edge between every $u \in V$ and every $e \in E$ such that u is an end of e in G (see Figure 6c). In fact, this implies $T(G) = (Inc(G))^2$, hence by Proposition 11 we have $(\delta(Inc(G)) - 1)/2 \leq \delta(T(G)) \leq (\delta(Inc(G)) + 1)/2$, and so, $\delta(G) - 1/2 \leq \delta(T(G)) \leq \delta(G) + 3/2$. One can sharpen the lower-bound and write $\delta(G) \leq \delta(T(G)) \leq \delta(G) + 3/2$ after noticing that G is an isometric subgraph (*i.e.*, a distance-preserving subgraph) of $T(G)$.

The *middle graph* of G [27], denoted by $mid(G)$, is constructed from $Inc(G)$ by adding an edge between every two “edge-vertices” $e, e' \in E$ sharing an end in G (see Figure 6d). Said differently, it is the intersection graph of all cliques of size two or less in G . Using a bipartite representation that is similar in spirit with those for line graphs (Theorem 6) and clique graphs (Theorem 8), one obtains $\delta(G) - 1 \leq \delta(mid(G)) \leq \delta(G) + 1$.

Last, the *k-edge graph* of G [26], denoted by $\Delta_k(G)$, is the intersection graph of all cliques of size k and maximal cliques of size at most $k - 1$ in G (see Figure 6e). Note that if $k = 2$ then it is exactly the line graph, and if $k = n$ then it is exactly the clique graph. Again using a bipartite representation with similar properties as those for line graphs and clique graphs, one obtains $\delta(G) - 1 \leq \delta(\Delta_k(G)) \leq \delta(G) + 1$.

5. Conclusion

We have proved that the hyperbolicity of any bipartite graph can be approximated up to a small additive constant by only considering the smallest side of its bipartition. This means a decrease by half of the number of vertices to be considered, hence a speed-up in the computation of hyperbolicity. On a more theoretical side, we detailed a simple framework so as to bound the hyperbolicity of line graphs and several other intersection graphs. We let open the question whether our techniques could also be applied to more “exotic” generalizations of line graphs – say, edge clique graphs [10].

References

- [1] Bagga, J., 2004. Old and new generalizations of line graphs. *International Journal of Mathematics and Mathematical Sciences* 2004 (29), 1509–1521.
- [2] Bandelt, H.-J., Chepoi, V., 2003. 1-hyperbolic graphs. *SIAM Journal on Discrete Mathematics* 16 (2), 323–334.
- [3] Bandelt, H.-J., Henkmann, A., Nicolai, F., 1995. Powers of distance-hereditary graphs. *Discrete Mathematics* 145 (1), 37–60.
- [4] Bandelt, H.-J., Mulder, H., 1986. Distance-hereditary graphs. *Journal of Combinatorial Theory, Series B* 41 (2), 182–208.
URL [http://dx.doi.org/10.1016/0095-8956\(86\)90043-2](http://dx.doi.org/10.1016/0095-8956(86)90043-2)
- [5] Behzad, M., Chartrand, G., 1966. Total graphs and traversability. *Proceedings of the Edinburgh Mathematical Society (Series 2)* 15 (02), 117–120.
- [6] Bondy, J., Murty, U., 1976. *Graph theory with applications*. Vol. 290. Macmillan London.
- [7] Borassi, M., Coudert, D., Crescenzi, P., Marino, A., Sep. 2015. On computing the hyperbolicity of real-world graphs. In: *23rd Annual European Symposium on Algorithms (ESA)*. Vol. 9294 of *Lecture Notes in Computer Science*. Springer, Patras, Greece, pp. 215–226.
- [8] Carballosa, W., Rodríguez, J., Sigarreta, J., 2014. New inequalities on the hyperbolicity constant of line graphs. *arXiv preprint arXiv:1410.2941*.

- [9] Carballosa, W., Rodríguez, J., Sigarreta, J., Villeta, M., 2011. On the hyperbolicity constant of line graphs. *Electronic journal of combinatorics* 18 (1), P210.
- [10] Chartrand, G., Kapoor, S., McKee, T., Saba, F., 1991. Edge-clique graphs. *Graphs and Combinatorics* 7 (3), 253–264.
- [11] Chepoi, V., Dragan, F., Estellon, B., Habib, M., Vaxès, Y., 2008. Diameters, centers, and approximating trees of delta-hyperbolic geodesic spaces and graphs. In: *24th Symposium on Computational Geometry (SCG)*. ACM, pp. 59–68.
URL <http://dx.doi.org/10.1145/1377676.1377687>
- [12] Cohen, N., Coudert, D., Ducoffe, G., Lancin, A., Jun. 2014. Applying clique-decomposition for computing Gromov hyperbolicity. Research Report hal-00989024, Inria Sophia Antipolis; I3S; Université Nice Sophia Antipolis; CNRS.
URL <http://hal.inria.fr/hal-00989024>
- [13] Cohen, N., Coudert, D., Lancin, A., 2015. On computing the gromov hyperbolicity. *ACM Journal of Experimental Algorithmics* 20 (1), 18.
- [14] Coudert, D., Ducoffe, G., Sep. 2014. Recognition of C_4 -free and $1/2$ -hyperbolic graphs. *SIAM Journal on Discrete Mathematics* 28 (3), 1601–1617.
URL <https://hal.inria.fr/hal-01070768>
- [15] Coudert, D., Ducoffe, G., 2016. Data center interconnection networks are not hyperbolic. *Theoretical Computer Science*.
URL <https://hal.inria.fr/hal-01323301>
- [16] Coudert, D., Ducoffe, G., Nisse, N., 2016, to appear. To approximate treewidth, use treelength! *SIAM Journal on Discrete Mathematics*.
- [17] Diestel, R., 1997. *Graph theory*. Vol. 173 of Graduate texts in mathematics. Springer, Heidelberg.
- [18] Gromov, M., 1987. Hyperbolic groups. *Essays in Group Theory* 8, 75–263.
- [19] Groshaus, M., Szwarcfiter, J., 2010. Biclique graphs and biclique matrices. *Journal of Graph Theory* 63 (1), 1–16.

- [20] Hamelink, R., 1968. A partial characterization of clique graphs. *Journal of Combinatorial Theory* 5 (2), 192–197.
- [21] Howorka, E., 1979. On metric properties of certain clique graphs. *Journal of Combinatorial Theory, Series B* 27 (1), 67–74.
URL [http://dx.doi.org/10.1016/0095-8956\(79\)90069-8](http://dx.doi.org/10.1016/0095-8956(79)90069-8)
- [22] Kennedy, W., Narayan, O., Saniee, I., 2013. On the hyperbolicity of large-scale networks. Tech. Rep. arXiv:1307.0031, ArXiv.
URL <http://arxiv.org/abs/1307.0031>
- [23] Magnien, C., Latapy, M., Habib, M., 2008. Fast computation of empirically tight bounds for the diameter of massive graphs. *ACM Journal of Experimental Algorithmics* 13.
URL <http://doi.acm.org/10.1145/1412228.1455266>
- [24] Martínez-Pérez, A., 2015. Chordality properties and hyperbolicity on graphs. arXiv preprint arXiv:1505.05675.
- [25] Noguès, D., 2009. δ -hyperbolicité et graphes. Master’s thesis, MPRI, Université Paris 7.
- [26] Prisner, E., 1994. A common generalization of line graphs and clique graphs. *Journal of Graph Theory* 18 (3), 301–313.
- [27] Prisner, E., 1995. *Graph dynamics*. Vol. 338. CRC Press.
- [28] Soto Gómez, M. A., 2011. Quelques propriétés topologiques des graphes et applications à internet et aux réseaux. Ph.D. thesis, Univ. Paris Diderot (Paris 7).
URL <http://www.dim.uchile.cl/~mausoto/memoires/these.pdf>
- [29] Szwarcfiter, J., 2003. A survey on clique graphs. In: *Recent advances in algorithms and combinatorics*. Springer, pp. 109–136.
- [30] Whitney, H., 1992. Congruent graphs and the connectivity of graphs. In: *Hassler Whitney Collected Papers*. Springer, pp. 61–79.
- [31] Wu, Y., Zhang, C., 2011. Hyperbolicity and chordality of a graph. *The Electronic Journal of Combinatorics* 18 (1), P43.
URL <http://www.combinatorics.org/ojs/index.php/eljc/article/view/v18i1p43>

APPENDIX E

Data center interconnection networks are not hyperbolic

Data center interconnection networks are not hyperbolic*

David Coudert^{1,2} and Guillaume Ducoffe^{2,1}

¹Inria, France

²Univ. Nice Sophia Antipolis, CNRS, I3S, UMR 7271, 06900 Sophia Antipolis, France

May 30, 2016

Abstract

Topologies for data center interconnection networks have been proposed in the literature through various graph classes and operations. A common trait to most existing designs is that they enhance the symmetric properties of the underlying graphs. Indeed, symmetry is a desirable property for interconnection networks because it minimizes congestion problems and it allows each entity to run the same routing protocol. However, despite sharing similarities these topologies all come with their own routing protocol. Recently, generic routing schemes have been introduced which can be implemented for any interconnection network. The performances of such universal routing schemes are intimately related to the *hyperbolicity* of the topology. Roughly, graph hyperbolicity is a metric parameter which measures how close is the shortest-path metric of a graph from a tree metric (the smaller the gap the better). Motivated by the good performances in practice of these new routing schemes, we propose the first general study of the hyperbolicity of data center interconnection networks. Our findings are disappointingly negative: we prove that the hyperbolicity of most data center interconnection topologies scales linearly with their diameter, that it is the worst-case possible for hyperbolicity. To obtain these results, we introduce original connection between hyperbolicity and the properties of the endomorphism monoid of a graph. In particular, our results extend to all vertex and edge-transitive graphs. Additional results are obtained for de Bruijn and Kautz graphs, grid-like graphs and networks from the so-called Cayley model.

Keywords. greedy routing scheme; metric embedding; graph endomorphism; Gromov hyperbolicity; Cayley graph; data center; interconnection network

1 Introduction

The network topologies that are used to interconnect the computing units of large-scale facilities (e.g., super computers, data centers hosting cloud applications, etc.) are designed to optimize various constraints such as equipment cost, deployment time, capacity and bandwidth, routing functionalities, reliability to equipment failures, power consumption, etc. This large variety of (conflicting) criteria has yielded numerous proposals of interconnection networks. See for instance [11, 12, 20, 47, 81] and [36, 38–40, 59] for the most recent ones. A common feature of the

*This work is partially supported by ANR project Stint under reference ANR-13-BS02-0007 and ANR program “Investments for the Future” under reference ANR-11-LABX-0031-01.

proposed constructions is to design network topologies offering a high-level of *symmetries*. Indeed, it is easier to balance the traffic load, and hence to minimize the congestion, on network topologies with a high-level of symmetry. Furthermore, it simplifies the initial wiring of the physical infrastructure and it ensures that each router node can run the protocol.

However, despite sharing properties, interconnection networks rely on specific routing algorithms that are optimized for each topology. As a novel step toward efficient and topology agnostic routing schemes, the authors in [64–66] proposed to use greedy routing schemes based on an embedding of the topology into certain metric space such as the hyperbolic metric space, and more recently the word metric space. This approach has been shown particularly efficient for Internet-like graphs [33, 42] where routes with low stretch are obtained. One explanation of this good behavior is that Internet-like graphs have low *hyperbolicity* [10, 74], a graph parameter providing sharp bounds on the stretch (or distortion) of the distances in a graph when it is embedded into an edge-weighted tree.

In this paper, we characterize or give upper and lower-bounds on the hyperbolicity of a broad range of interconnection network topologies. These bounds can be used to analyze the worst-case behavior of greedy routing schemes in these topologies. Before we present our results, let us further put in context the role they play in routing and in other distance-related problems.

Related work. Greedy routing schemes based on an embedding into the hyperbolic space have been introduced by Kleinberg in [33]. Since then, various authors explored further this approach [42, 45, 48]). In particular, they showed that the graphs of the Autonomous Systems of the Internet embed better into a hyperbolic space than into an Euclidean space¹. It was only recently in [70] that a formal relationship between the performances of hyperbolic embeddings and the hyperbolicity was proved. Namely, the authors proved that the over-delay for such routing schemes, or equivalently the stretch of the routing, depends on the hyperbolicity. In [73], the authors proved that similar results hold for greedy routing schemes based on an embedding of the topology into some word metric space (*e.g.*, see [26] for more information). More precisely, they use hyperbolicity to upper-bound the complexity of their routings, as well as to bound the size of the automata that are involved in their routing schemes.

Their results add up to prior worst-case analysis of graph heuristics that already pointed out the important role played by the hyperbolicity. For instance, there are approximation algorithms for problems related to distances in graphs —like diameter and radius computation [37], and minimum ball covering [32]— whose approximation constant depends on the hyperbolicity. Sometimes the approximation factor is a universal constant but the algorithm relies on a data-structure whose size is proportional to the hyperbolicity of the network topology [31]. Geometric routing schemes in [42, 45, 48] do not make exception and so have a stretch *lower-bounded* by the hyperbolicity (the bound is reached by some of them).

There have been measurements to confirm that complex networks such as the graphs of the Autonomous Systems of the Internet, social networks and phylogenetic networks all have a low hyperbolicity. We refer to [60, 62, 63, 71, 74, 77] for the most important studies in this area. Additional related work in [50, 57] shows that the low hyperbolicity of complex networks may be a consequence

¹In fact, it follows from [23] that for any n -vertex graph G there is an embedding φ of G into the Euclidean space (with unbounded dimension) such that for every $u, v \in V(G)$ we have $d(\varphi(u), \varphi(v)) \leq \mathcal{O}(\sqrt{\log \log n}) \cdot d_G(u, v) + \mathcal{O}(\delta(G) \cdot \log n)$, with $\delta(G)$ being the hyperbolicity (the \mathcal{O} -notation suppresses the polyloglog factors). However, it does not seem that hyperbolicity is the most relevant parameter in the study of Euclidean embeddings.

of some preferential attachment mechanisms. However, we are not informed of any study on the hyperbolicity of data center interconnection networks. In this paper, we aim to fill in this gap through a theoretical study of their underlying graphs.

Our contributions. In an attempt to confront with the diversity of interconnection network topologies proposed in the literature, we relate hyperbolicity with a few graph properties that are frequently encountered in these topologies. Indeed, we do not aim to provide a —long and non-exhaustive— listing of unrelated results for each network, but rather to exhibit a small number of their characteristics that are strongly related with their metric invariants. In particular, we relate hyperbolicity with the symmetries of a graph.

- We prove in Section 3 that for graphs whose center is a k -distance dominating set for some small value of k , the hyperbolicity scales linearly with the diameter. This class of graphs strictly contains graphs whose diameter equals the radius, *a.k.a.* the *self-centered graphs* [9, 14]. In particular, it comprises all vertex-transitive graphs (a strict subclass of self-centered graphs), as well as edge-transitive graphs. A main consequence of our result is that every interconnection network whose topology is based on a *Cayley graph* has large hyperbolicity².
- In addition, we prove that similar results hold for graphs admitting an *endomorphism* such that the distance between any vertex and its symmetric image is large. On the way to prove these results, we define a new graph invariant which is called *weak mobility*, that generalizes the so-called graph mobility (*e.g.*, see [22, 34]). We use these new results to improve our lower-bounds on the hyperbolicity of several interconnection networks.
- For completeness, we also characterize the hyperbolicity of other “symmetric” networks such as de Bruijn, Kautz and grid-like graphs. More precisely, we apply different techniques that are based on their shortest-paths distribution so that we can prove in Section 4 that they also have a large hyperbolicity. The techniques that are involved in the proofs have been introduced in previous papers [16, 41], but to the best of our knowledge the way we use them in this work is new.

All of the above results are summarized in Table 1.

- Last, we extend our results in Section 5 to *heterogenous* data center interconnection networks. That is, we relate hyperbolicity with several graph operations, most of them being introduced in the Cayley model of [59] in order to enhance some desirable properties of data center interconnection networks.

Our main message is that existing designs in the literature yield graphs with the highest possible value for the hyperbolicity —w.r.t. their diameter. On the negative side, it means that any greedy routing scheme whose stretch depends on the hyperbolicity is not scalable enough to cope with large data centers. But on a more positive side, it also implies that any routing scheme relying on

² Independently from this work, the authors in [54] proved that for any vertex-transitive graph, the hyperbolicity scales linearly with the diameter. However, their proof relies on another definition of hyperbolicity, and it is unclear whether the proof can be extended to other graph classes. By contrast, our proof yields a tighter lower-bound for hyperbolicity, and it relies on a much simpler and more general argument (*i.e.*, see Theorem 4). Especially, it also applies to edge-transitive graphs.

a data-structure with size proportional to the hyperbolicity solely requires *sublogarithmic* space in the number of servers. Indeed, it is well-known that the data center interconnection networks often have a diameter that is logarithmic or sublogarithmic in their size.

We start this paper providing useful notations and definitions in Section 2, and we conclude it in Section 6 with open questions. Especially, can we infer a formal relationship between network congestion and graph hyperbolicity ?

2 Preliminaries

A data center is a facility that is used to house resources such as computer systems, servers, etc. Data center resources are interconnected using communication networks, that are called data center interconnection networks. They are modeled as a graph where the vertices are the data center resources (*e.g.*, computing units) and there is an edge between two resources if they are directly connected in the network. Different graph classes have been proposed in order to design data center interconnection networks [11, 12, 20, 36, 38–40, 47, 59, 81]. In what follows, our results apply to general graphs, but they are aimed at providing good lower-bounds on the hyperbolicity for these specific topologies.

We refer to [80, 82] for the usual graph terminology. Graphs in this study are finite, simple (hence, without loop nor multiple edges), connected and unweighted.

2.1 Metric graph theory

Given a connected graph $G = (V, E)$, the *distance* between any two vertices $u, v \in V$ is defined as the minimum number of edges on a uv -path. We will denote it by $d_G(u, v)$, or by $d(u, v)$ whenever G is clear from the context. For any subset $S \subseteq V$, the *eccentricity* of vertex $v \in S$, denoted $\text{ecc}_G(v, S)$, is defined as the maximum distance in G between v and any other vertex in S . The *radius* of S is defined as the least eccentricity of vertices in S and is denoted by $\text{rad}_G(S)$, while the *diameter* of S is defined as the largest eccentricity of vertices in S and is denoted by $\text{diam}_G(S)$. Observe that it always holds $\text{rad}_G(S) \leq \text{diam}_G(S) \leq 2 \cdot \text{rad}_G(S)$. In particular, for any vertex $v \in V$, we denote by $\text{ecc}(v) = \text{ecc}_G(v, V)$, $\text{rad}(G) = \text{rad}_G(V)$ and $\text{diam}(G) = \text{diam}_G(V)$. The *center* $\mathcal{C}(G)$ of the graph is the subset of all vertices with minimum eccentricity $\text{rad}(G)$. We call the graph G *self-centered* if it holds $\text{diam}(G) = \text{rad}(G)$ *i.e.*, every vertex of G is in the center.

Last, we define graph hyperbolicity as follows.

Definition 1 (4-points Condition, [10]). Let G be a connected graph.

For every 4-tuple u, v, x, y of G , we define $\delta(u, v, x, y)$ as half of the difference between the two largest sums amongst:

$$S_1 = d(u, v) + d(x, y), S_2 = d(u, x) + d(v, y), \text{ and } S_3 = d(u, y) + d(v, x).$$

The graph hyperbolicity, denoted by $\delta(G)$, is equal to $\max_{u,v,x,y} \delta(u, v, x, y)$.

Moreover, we say that G is δ -*hyperbolic*, for every $\delta \geq \delta(G)$.

Other definitions exist for the hyperbolicity, but they are pairwise equivalent up to a constant-factor (*e.g.*, see [10] for details). So far, the hyperbolicity of a few graph classes has been characterized such as: random graphs [56, 61, 69], chordal graphs [25], k -chordal graphs [53], outerplanar

Table 1: Summary of results

Name	Degree max.	Diameter	Order	δ	Proof
de Bruijn graph, $UB(d, D)$	$2d$	D	d^D	$\frac{1}{2} \lfloor \frac{D}{2} \rfloor \leq \delta \leq \lfloor \frac{D}{2} \rfloor$	Prop. 37
Kautz graph, $UK(d, D)$	$2d$	D	$d^D(d+1)$	$\lfloor \frac{D}{4} \rfloor + \varepsilon \leq \delta \leq \lfloor \frac{D}{2} \rfloor, \varepsilon \in \{0, 1\}$	Prop. 40
Shuffle exchange, $SE(n)$	3	$2n - 1$	2^n	$\frac{1}{2} \lfloor \frac{n}{2} \rfloor \leq \delta \leq n - 1$	Prop. 42
(n, m) -grid	4	$n + m - 2$	nm	$\min\{n, m\} - 1$	Cor. 48
d -dimensional grid of size s	$2d$	$d(s - 1)$	s^d	$(s - 1) \lfloor \frac{d}{2} \rfloor$	Cor. 49
Triangular (n, m) -grid	6	$n + m - 2$	nm	$\frac{\min\{n, m\} - 1}{2}$	Lem. 51
Hexagonal (n, m) -grid	6	$\begin{cases} n - 1 + \lceil \frac{m-1}{2} \rceil & \text{when } m \leq 2n - 1 \\ m - 1 & \text{otherwise} \end{cases}$	nm	$\frac{\min\{n, m\} - 1}{2}$	Lem. 53
Cylinder (n, m) -grid	4	$\lfloor \frac{n}{2} \rfloor + m - 1$	nm	$\min\{\lfloor \frac{n}{2} \rfloor, \frac{1}{2}(\lfloor \frac{n}{2} \rfloor + m) - \varepsilon\}, \varepsilon \in \{\frac{1}{2}, 1\}$	Lem. 55
Torus (n, m) -grid	4	$\lfloor \frac{n}{2} \rfloor + \lfloor \frac{m}{2} \rfloor$	nm	$\lfloor \frac{1}{2}(\lfloor \frac{n}{2} \rfloor + \lfloor \frac{m}{2} \rfloor) \rfloor - 1 \leq \delta \leq \lfloor \frac{1}{2}(\lfloor \frac{n}{2} \rfloor + \lfloor \frac{m}{2} \rfloor) \rfloor$	Lem. 11
Gen. hypercube, $G(m_1, \dots, m_r)$	$\sum_{i=1}^r m_i - r$	r	$\prod_{i=1}^r m_i$	$\lfloor \frac{r}{2} \rfloor$	Lem. 13
Cube Connected Cycle, $CCC(n)$	3	$2n - 2 + \max\{2, \lfloor \frac{n}{2} \rfloor\}$	$n2^n$	$n \leq \delta \leq n - 1 + \lfloor \frac{\max\{2, \lfloor \frac{n}{2} \rfloor\}}{2} \rfloor$	Lem. 15
BCube $_k(n)$	$\max\{n, k + 1\}$	$2(k + 1)$	$2^k(n + k + 1)$	$k + 1$	Lem. 17
Fat-Tree $_k$	k	6	$\frac{k^2}{4}(k + 5)$	2	Lem. 19
Butterfly graph, $BF(n)$	4	$2n$	$2^n(n + 1)$	n	Lem. 21
Wrapped Butterfly graph, $WBF(n)$	4	$n + \lfloor \frac{n}{2} \rfloor$	$2^n(n + 1)$	$\lfloor \frac{n}{2} \rfloor \leq \delta \leq \lfloor \frac{1}{2}(n + \lfloor \frac{n}{2} \rfloor) \rfloor$	Lem. 21
k -ary n -fly	$2k$	$2n$	$k^n(n + 1)$	n	Lem. 23
k -ary n -tree	$3k$	$2n$	$k^{n-1}(n + k)$	$n - 1$	Lem. 25
d -ary tree grid, $MT(d, h)$	$d + 1$	$4h$	$d^h(d^h + 2\frac{d^h-1}{d-1})$	$2h$	Lem. 27
Bubble-sort graph, $BS(n)$	$n - 1$	$\binom{n}{2}$	$n!$	$\lfloor \frac{n(n-1)}{4} \rfloor$	Lem. 30
Transposition graph, $T(n)$	$\binom{n}{2}$	$n - 1$	$n!$	$\frac{1}{2} \lfloor \frac{n-1}{2} \rfloor \leq \delta \leq \lfloor \frac{n-1}{2} \rfloor$	Lem. 32
Star graph, $S(n)$	$n - 1$	$\lfloor \frac{3(n-1)}{2} \rfloor$	$n!$	$\lfloor \frac{1}{2} \lfloor \frac{3(n-1)}{2} \rfloor - \frac{1}{2} \rfloor \rfloor \leq \delta \leq \lfloor \frac{1}{2} \lfloor \frac{3(n-1)}{2} \rfloor \rfloor$	Lem. 34
Cayley graph, $G(\Gamma, S)$	$2 S $	$\text{diam}(G(\Gamma, S))$	$ \Gamma $	$\frac{1}{2} \lfloor \frac{\text{diam}(G(\Gamma, S))}{2} \rfloor \leq \delta \leq \lfloor \frac{\text{diam}(G(\Gamma, S))}{2} \rfloor$	Cor. 5

graphs [67] and other geometrical graph classes [37]. Lower and upper-bounds for the hyperbolicity are obtained in [58] using graph invariants, and also in [52, 67] using graph decompositions. We refer to [44] for a compelling of many well-known facts about hyperbolicity. In particular, we will make use of the following upper-bound for hyperbolicity:

Lemma 2 ([10, 44, 74]). *For every connected graph G , it holds that $\delta(G) \leq \left\lfloor \frac{\text{diam}(G)}{2} \right\rfloor$.*

Based on Lemma 2, the authors in [71] have proposed the following classification of finite graphs. A graph G is *strongly hyperbolic* if $\delta(G) = \mathcal{O}(\log(\log(\text{diam}(G))))$, *hyperbolic* if $\delta(G) = \mathcal{O}(\log(\text{diam}(G)))$, and *non hyperbolic* otherwise. We follow their terminology and we aim at proving that some graph classes are non hyperbolic. This is in contrast with many graph classes in the literature that have a constant upper-bound on their hyperbolicity, and so, that are strongly hyperbolic [53]. By Lemma 2, in order to prove that a graph is non hyperbolic, and more precisely that its hyperbolicity scales linearly with its diameter, it suffices to prove that one can *lower-bound* the hyperbolicity with the diameter —up to a constant-factor. This line of work was followed in [21, 76] to prove that expander graphs are non hyperbolic. Our proofs will make use of the notion of *isometric subgraphs*, the latter denoting a subgraph H of a graph G such that $d_H(u, v) = d_G(u, v)$ for any two vertices $u, v \in H$.

2.2 Algebraic graph theory

A graph *endomorphism* is a mapping σ from the vertex-set of a graph G to itself which preserves the adjacency relations, *i.e.*, for every $\{u, v\} \in E(G)$ we have that $\{\sigma(u), \sigma(v)\} \in E(G)$.

Definition 3. Let $G = (V, E)$ be a graph. Given an endomorphism σ of G , the *mobility* of σ is equal to $\min_{v \in V} d(v, \sigma(v))$. The *weak mobility* of G is the largest integer l such that it admits an endomorphism with mobility l .

We note that a graph endomorphism might fail to preserve the *non-adjacency* relations, but it does so if it is a graph *automorphism*, *i.e.*, a one-to-one endomorphism. In particular a graph endomorphism σ is called *idempotent* if for every $v \in V(G)$ it holds that $\sigma^2(v) = v$, and in such a case it is an automorphism.

A graph is called *vertex-transitive* if for every $u, v \in V(G)$, there is an automorphism σ such that $\sigma(u) = v$. Similarly, we call a graph *edge-transitive* if for every $e = \{u, v\}, e' = \{u', v'\} \in E(G)$, there is an automorphism σ such that $\{\sigma(u), \sigma(v)\} = \{u', v'\}$. We emphasize that every vertex-transitive graph is self-centered. We will use this property in the following sections. Finally, let (Γ, \cdot) be a group and let S be a generating set of Γ that is symmetric and that does not contain the neutral element of group Γ , *i.e.*, $S = S^{-1}$ and $S \cap S^{-1} = \emptyset$. The *Cayley graph* $G(\Gamma, S)$ of group Γ w.r.t. S has vertex-set Γ and edge-set $\{\{g, g \cdot s\} \mid g \in \Gamma, s \in S\}$. It is well-known that every Cayley graph is vertex-transitive [12].

3 The metric properties of the endomorphism monoid of a graph

Our belief is that any method to lower-bound the value of hyperbolicity needs to rely *as few as possible* on the shortest-path distribution of the graphs so as to be of practical use. Indeed, in most cases there is no good characterization of this distribution. There even exist interconnection

networks topologies the diameter of which is still unknown [3,46]. In a need of more robust methods, we introduce new lower-bounds on the hyperbolicity that are based on non-trivial symmetries of the graphs. For clarity, our results are presented separately from their applications to interconnection networks topologies.

3.1 Main results

We first introduce a very generic argument to obtain lower-bounds on the hyperbolicity. In particular, we will show that it applies to highly symmetric graphs such as transitive graphs.

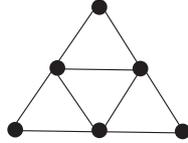


Figure 1: A self-centered graph G with $\text{diam}(G) = \text{rad}(G) = 2$, while $\delta(G) = 1/2 = \text{diam}(G)/4$.

Theorem 4. *Let G be a connected graph, and let $k \geq 0$ be such that all vertices are at distance at most k from the center of G . Then, $\delta(G) \geq \frac{1}{2} \cdot \left\lfloor \frac{\text{diam}(G)}{2} \right\rfloor - \frac{k}{2}$ and this bound is sharp.*

Proof. Let $\mathcal{C}(G)$ be the center of G . By the hypothesis every node in G is at distance at most k from $\mathcal{C}(G)$, therefore $\text{diam}_G(\mathcal{C}(G)) \geq \text{diam}(G) - 2k$. Moreover, by [37, Proposition 5] $\text{diam}_G(\mathcal{C}(G)) \leq 4\delta(G) + 1$. Consequently, it holds $\delta(G) \geq \lfloor \text{diam}_G(\mathcal{C}(G)) / 2 \rfloor / 2 \geq \lfloor \text{diam}(G) / 2 \rfloor / 2 - k/2$.

The lower-bound is sharp, as shown with the example of Figure 1 where $\text{diam}(G) = \text{rad}(G) = 2$ while $\delta(G) = 1/2 = \text{diam}(G)/4$. \square

Unlike all other techniques that we will discuss next, we can use the lower-bound of Theorem 4 to prove that all graphs studied in this work are non hyperbolic. However, the bounds obtained with this first method are usually loose, and they never outmatch the bounds obtained with the other techniques — when they apply. We will illustrate this point in what follows.

It is straightforward that Theorem 4 applies to self-centered graphs (with $k = 0$). Especially, it applies to vertex-transitive graphs.

Corollary 5. *Let G be a connected vertex-transitive graph. Then, $\delta(G) \geq \frac{1}{2} \cdot \left\lfloor \frac{\text{diam}(G)}{2} \right\rfloor$ and this bound is sharp.*

The lower-bound of Corollary 5 is sharp, as shown by any clique (that has diameter one and null hyperbolicity).

On the practical side, most of the interconnection networks topologies are based on vertex-transitive graphs. This comprises hypercube-based networks [8], generalized Petersen graphs [1,18], generalized Heawood graphs [30,68] and Cayley graphs [12]. For some of these topologies such as the Pancake graph [3], a well-known Cayley graph, Corollary 5 is the best lower-bound on the hyperbolicity we know so far.

Corollary 6. *Let G be a connected edge-transitive graph. Then, $\delta(G) \geq \frac{1}{2} \cdot \left\lfloor \frac{\text{diam}(G)}{2} \right\rfloor - \frac{1}{2}$ and this bound is sharp.*

Proof. We first claim that the center $\mathcal{C}(G)$ is a dominating set of G . Indeed, let $u \in V(G)$ and $v \in \mathcal{C}(G)$, and let $x \in N_G(u)$ and $y \in N_G(v)$. Since G is edge-symmetric by the hypothesis, there exists an automorphism σ such that $\{\sigma(v), \sigma(y)\} = \{u, x\}$. Furthermore $\sigma(v) \in \mathcal{C}(G)$ because σ is an automorphism and so, $d_G(u, \mathcal{C}(G)) \leq d_G(u, \sigma(v)) \leq 1$ which proves the claim. As a result, we can apply Theorem 4 by setting $k = 1$.

The lower-bound is sharp, as shown by any star (that has diameter two and null hyperbolicity). \square

3.1.1 Improved lower-bounds using graph endomorphisms

However, despite its wide applicability to interconnection networks, the above Corollaries 5 and 6 require graphs to have an automorphism group with constrained properties. A natural question is whether we can weaken the requirements by considering endomorphisms instead of automorphisms. To answer this question, we use weakly vertex-transitive graphs that have been defined in [29] in a similar fashion to vertex-transitive graphs. Namely, a graph G is weakly vertex-transitive if, for any two vertices $u, v \in V(G)$ there exists a graph *endomorphism* σ satisfying $\sigma(u) = v$. Unlike vertex-transitive graphs, the gap between hyperbolicity and diameter may be arbitrarily large for weakly vertex-transitive graphs. Indeed, on the one hand it was proved in [29] that bipartite graphs are weakly vertex-transitive. On the other hand, trees are bipartite 0-hyperbolic graphs, whereas they may have a diameter that is arbitrarily large. We now show that surprisingly, some lower-bounds on the hyperbolicity can still be deduced from graph endomorphisms.

Theorem 7. *Let G be a connected graph of weak mobility $l \geq 2$. Then it holds $\delta(G) \geq \frac{1}{2} \cdot \lceil \frac{l}{2} \rceil$.*

Proof. We will consider a graph game which is a slight variation of the well-known 'Cop and Robber' game (e.g. see [5–7]). There are two players in this game that are playing alternatively on a (connected) graph, by moving along a path of length at most s , for some positive integer s . The first player to position herself on the graph is the Cop, and the second player is called the Robber. Last a graph is said *Cop-win* for this game if the Cop always has a winning-strategy i.e., she can always reach the position of the Robber in a finite number of moves, and hence eventually catch the Robber. In [49] the authors proved that every connected graph G is Cop-win whenever $s \geq 4\delta(G)$. So, to prove the theorem we claim that it suffices to show that G is not Cop-win if $s \leq l - 1$. Indeed, in such a case it holds $4\delta(G) \geq l$, hence $2\delta(G) \geq l/2$ that implies $2\delta(G) \geq \lceil l/2 \rceil$ and so, $\delta(G) \geq \lceil l/2 \rceil / 2$. Equivalently, we will exhibit a winning-strategy for the Robber in such a case.

Let σ be an endomorphism of G with mobility l , that exists by the hypothesis. One can observe that if at each turn of the Cop the Robber is onto the image by σ of her current position, then it is a winning strategy for the Robber because by the hypothesis, both vertices are at distance at least l , and the maximum speed of the Cop is $l - 1$. To achieve the result, let us proceed as follows. First if the Cop picks vertex u as her initial position then the Robber starts the game at vertex $\sigma(u)$. Then, if the Cop moves along a path $(u = x_0, x_1, \dots, x_i, \dots, x_k = v)$, $k \leq l - 1$, then the Robber moves along the path $(\sigma(u), \sigma(x_1), \dots, \sigma(x_i), \dots, \sigma(v))$ which exists because σ is a graph endomorphism. Such a move for the Robber is valid as long as $v \notin \{\sigma(u), \sigma(x_1), \dots, \sigma(x_i), \dots, \sigma(v)\}$, and that is always the case since $\sigma(x_i) = v$ would imply $d(x_i, \sigma(x_i)) \leq l - 1$. \square

We are particularly interested in the special case of the graphs G with weak mobility equal to their diameter $diam(G)$. These graphs are self-centered, and so, their hyperbolicity is at least

$\lfloor \text{diam}(G)/2 \rfloor / 2$ by Theorem 4. The lower-bound is slightly improved by Theorem 7 in this situation. However, not all self-centered graphs have their weak mobility equal to the diameter [34].

In what follows, we will mostly rely upon the below refinement of Theorem 7 in our proofs. This way, we will obtain almost tight bounds on the hyperbolicity of data center interconnection networks. However, note that the following results require stronger constrictions on the endomorphism monoid than Theorem 7.

Theorem 8. *Let G be a connected graph, and l, l' be two non-negative integers. Suppose there exists an endomorphism σ of G with mobility l and such that for every $v \in V(G)$, $d(v, \sigma^2(v)) \leq l'$. Then, it holds $\delta(G) \geq \lfloor \frac{l}{2} \rfloor - \frac{l'}{2}$.*

Proof. Clearly, if $l \leq l'$ then $\delta(G) \geq 0 \geq \lfloor l/2 \rfloor - l'/2$. Therefore, we will assume w.l.o.g. that $l \geq l' + 1$. Let $u \in V(G)$ minimizing $d_G(u, \sigma(u))$ and let v be on a $u\sigma(u)$ -shortest-path such that $d_G(u, v) = \lfloor d_G(u, \sigma(u))/2 \rfloor$. Then, we deduce from the endomorphism σ the following inequalities:

$$\begin{aligned} S_1 &= d(u, \sigma(u)) + d(v, \sigma(v)) \geq 2 \cdot d(u, \sigma(u)) \geq 2l; \\ S_2 &= d(u, v) + d(\sigma(u), \sigma(v)) \leq 2 \cdot d(u, v) \leq 2 \lfloor d(u, \sigma(u))/2 \rfloor; \\ S_3 &= d(u, \sigma(v)) + d(v, \sigma(u)) \leq d(u, \sigma^2(u)) + d(\sigma^2(u), \sigma(v)) + d(v, \sigma(u)) \leq l' + 2 \cdot d(v, \sigma(u)) \\ &\leq 2 \lfloor d(u, \sigma(u))/2 \rfloor + l' \leq d(u, \sigma(u)) + 1 + l'. \end{aligned}$$

In such a case, $S_1 \geq \max\{S_2, S_3\}$ and as a result:

$$\delta(G) \geq \delta(u, v, \sigma(u), \sigma(v)) \geq \min \left(\left\lceil \frac{d(u, \sigma(u))}{2} \right\rceil, \left\lfloor \frac{d(u, \sigma(u))}{2} \right\rfloor - \frac{l'}{2} \right) \geq \left\lfloor \frac{l}{2} \right\rfloor - \frac{l'}{2}.$$

□

The lower-bound of Theorem 8 outmatches the one of Theorem 7 when $l' \leq \lfloor l/2 \rfloor - 1$. Furthermore, in practice, we will use Theorem 8 with $l = \text{diam}(G)$ and $l' \in \{0, 1\}$. This way, we will improve by a factor two all previous lower-bounds.

It can be noticed that the lower-bound of Theorem 8 is sharp for almost every cycle. Indeed, let \mathbb{Z}_n be the vertex set of the n -cycle C_n , and let σ be the automorphism mapping any vertex i to the vertex $i + \lfloor n/2 \rfloor \pmod{n}$. Applying Theorem 8 to σ , we obtain a lower-bound $\lfloor n/4 \rfloor$ for the hyperbolicity of even-length cycles, which is exact, and a lower-bound $\lfloor n/4 \rfloor - 1/2$ for odd-length cycles, which is exact when $n \equiv 1 \pmod{4}$ and below $1/2$ of the true hyperbolicity when $n \equiv 3 \pmod{4}$ [28, 53].

We emphasize on the following consequence of Theorem 8.

Corollary 9. *Let G be a connected graph and σ be an idempotent endomorphism with mobility l . Then, it holds $\delta(G) \geq \lfloor \frac{l}{2} \rfloor$.*

Proof. By the hypothesis, the endomorphism σ is idempotent and so, we can apply Theorem 8 by setting $l' = 0$. □

In the special case when $l = \text{diam}(G)$, the lower-bound of Corollary 9 is best possible. Indeed, it coincides with the upper-bound of Lemma 2, thereby giving the exact value for hyperbolicity.

It is natural to ask whether Theorems 7 and 8 can be further improved by using bounds on the distances $d(v, \sigma^3(v)), d(v, \sigma^4(v))$ and so on. However, answering this question is nontrivial since the techniques used for Theorems 7 and 8 are already quite different. We leave it as an interesting open question.

3.2 Applications

Equipped with Theorems 7, 8 and Corollary 9, we subsequently apply them on a broad range of topologies studied in the literature. We will combine the lower-bounds that we obtain with a slight variation of the well-known upper-bound of Lemma 2. Indeed it is folklore that the hyperbolicity of a graph is the maximum hyperbolicity taken over all of its biconnected components. So, $\delta(G) \leq \lfloor \text{effdiam}(G)/2 \rfloor$, where the so-called *efficient diameter* $\text{effdiam}(G)$ denotes the largest diameter amongst the biconnected components of the graph. This way, we will show that for most graphs found in the literature, their hyperbolicity scales linearly with the efficient diameter—that is the worst-case possible for hyperbolicity.

3.2.1 Torus

Let us first consider the torus, a well-known grid-like graph which is highly symmetrical. Other grid-like graphs will be considered in Section 4.2 using a different approach.

Definition 10. The *torus* (n, m) -grid has vertex-set $\mathbb{Z}_n \times \mathbb{Z}_m$; any two vertices $(i, j), (i', j')$ are adjacent if either $i' = i, j' \equiv j + 1 \pmod{m}$, or $i' \equiv i + 1 \pmod{n}, j' = j$.

Lemma 11. Let $n = 2p + r, m = 2q + s$, with $r, s \in \{0, 1\}$. Then, the hyperbolicity $\delta_{n,m}$ of the torus (n, m) -grid satisfies:

$$\left\lfloor \frac{1}{2} \cdot \left(\left\lfloor \frac{n}{2} \right\rfloor + \left\lfloor \frac{m}{2} \right\rfloor \right) \right\rfloor - \frac{r+s}{2} \leq \delta_{n,m} \leq \left\lfloor \frac{1}{2} \cdot \left(\left\lfloor \frac{n}{2} \right\rfloor + \left\lfloor \frac{m}{2} \right\rfloor \right) \right\rfloor.$$

Proof. For any two vertices $u = (i_u, j_u), v = (i_v, j_v)$:

$$d(u, v) = \min\{|i_u - i_v|, n - |i_u - i_v|\} + \min\{|j_u - j_v|, m - |j_u - j_v|\}.$$

It implies that the diameter of the torus grid is $\lfloor n/2 \rfloor + \lfloor m/2 \rfloor$ and so, $\delta_{n,m} \leq \lfloor (\lfloor n/2 \rfloor + \lfloor m/2 \rfloor) / 2 \rfloor$ by Lemma 2. Finally, let σ be the automorphism of the torus grid which maps any vertex (i, j) to the vertex $(i + \lfloor n/2 \rfloor \pmod{n}, j + \lfloor m/2 \rfloor \pmod{m})$. Since for any vertex v , $d(v, \sigma(v)) = \lfloor n/2 \rfloor + \lfloor m/2 \rfloor$ and $d(v, \sigma^2(v)) = r + s$, then it follows from Theorem 8 that $\delta_{n,m} \geq \lfloor (\lfloor n/2 \rfloor + \lfloor m/2 \rfloor) / 2 \rfloor - \frac{r+s}{2} \geq \lfloor (\lfloor n/2 \rfloor + \lfloor m/2 \rfloor) / 2 \rfloor - 1$. \square

3.2.2 Hypercube-like networks

Definition 12 ([8, 11]). Let m_1, m_2, \dots, m_r be positive integers with for every $i, m_i \geq 2$ and $r \geq 1$. The generalized hypercube $G(m_1, m_2, \dots, m_r)$ has vertex-set $\{(x_1, x_2, \dots, x_r) \mid \forall i, 0 \leq x_i \leq m_i - 1\}$, and two vertices $(x_1, x_2, \dots, x_r), (y_1, y_2, \dots, y_r)$ are adjacent in the graph if and only if their Hamming distance $\sum_i \mathbb{I}_{\{x_i \neq y_i\}}$ is equal to 1.

In particular, the k -ary hypercube $H_k(n)$ is the generalized hypercube $G(m_1, m_2, \dots, m_n)$ with for every $i, m_i = k$.

Lemma 13. $\delta(G(m_1, m_2, \dots, m_r)) = \lfloor \frac{r}{2} \rfloor$.

Proof. The diameter of $G(m_1, m_2, \dots, m_r)$ is r and so, $\delta(G(m_1, m_2, \dots, m_r)) \leq \lfloor r/2 \rfloor$ by Lemma 2. To prove the lower-bound, we first make the observation that the binary hypercube $H_2(r)$ is an isometric subgraph of $G(m_1, m_2, \dots, m_r)$. Let σ be the automorphism mapping any vertex

$(x_1, x_2, \dots, x_r) \in V(H_2(r))$ to its *complementary* vertex $(1-x_1, 1-x_2, \dots, 1-x_r)$. Note that σ has mobility r and it is idempotent. As a result, we conclude by Corollary 9 that $\delta(G(m_1, m_2, \dots, m_r)) \geq \delta(H_2(r)) \geq \lfloor r/2 \rfloor$. \square

As we will show later, Lemma 13 also follows from Corollary 49 and the fact that the n -dimensional grid of size 2 is exactly the hypercube $H_2(n)$.

Definition 14 ([4]). The cube-connected-cycle $CCC(n)$ has vertex-set the pairs $\langle i, w \rangle$, for $0 \leq i \leq n-1$ and for w any binary word of length n ; two vertices $\langle i, x_1x_2\dots x_n \rangle$ and $\langle j, y_1y_2\dots y_n \rangle$ are adjacent in the graph if and only if either $i = j$, $x_i = 1 - y_i$ and for every $k \neq i$, $x_k = y_k$; or $i \equiv j + 1 \pmod{n}$ and for every k , $x_k = y_k$.

Lemma 15. $n \leq \delta(CCC(n)) \leq n - 1 + \lfloor \max\{1, \frac{1}{2} \cdot \lfloor \frac{n}{2} \rfloor\} \rfloor$.

Proof. By [19], $\text{diam}(CCC(n)) = 2n - 2 + \max\{2, \lfloor n/2 \rfloor\}$ and so, $\delta(CCC(n)) \leq n - 1 + \lfloor (\max\{2, \lfloor n/2 \rfloor\}) / 2 \rfloor$ by Lemma 2. Furthermore, the mapping $\sigma : \langle i, w \rangle \rightarrow \langle i, \bar{w} \rangle$ is an idempotent endomorphism and it has mobility $2n$ by [19]. We conclude by Corollary 9 that $\delta(CCC(n)) \geq n$. \square

Definition 16 ([39]). Let \mathbb{Z}_n^l be the set of words of length l over the alphabet $\{0, 1, \dots, n-1\}$. The graph $\text{BCube}_k(n)$ has vertex-set $\mathbb{Z}_n^{k+1} \cup (\{0, 1, \dots, k\} \times \mathbb{Z}_n^k)$ and edge-set $\{\langle l, s_k s_{k-1} \dots s_{l+1} s_{l-1} \dots s_0 \rangle, s_k s_{k-1} \dots s_{l+1} s_{l-1} \dots s_0 \rangle \mid 0 \leq l \leq k \text{ and for every } i, 0 \leq s_i \leq n-1\}$.

Lemma 17. $\delta(\text{BCube}_k(n)) = k + 1$.

Proof. By [43] $\text{diam}(\text{BCube}_k(n)) = 2(k+1)$ and so, $\delta(\text{BCube}_k(n)) \leq k+1$ by Lemma 2. Then, let us assume that $n = 2$ because we have by [43] that $\text{BCube}_k(2)$ is an isometric subgraph of $\text{BCube}_k(n)$. We define the automorphism σ satisfying that for all binary word $w \in \mathbb{Z}_2^{k+1}$, $\sigma(w) = \bar{w}$, and for every pair $\langle l, w \rangle \in \{0, 1, \dots, k\} \times \mathbb{Z}_2^k$, $\sigma(\langle l, w \rangle) = \langle l, \bar{w} \rangle$. By [39, 43] σ has mobility $2(k+1)$ and so, by noticing that σ is idempotent we can conclude by Corollary 9 that $\delta(\text{BCube}_k(n)) \geq \delta(\text{BCube}_k(2)) \geq k+1$. \square

3.2.3 Tree-like networks

Definition 18 ([36]). Let $k \geq 4$ be even. The Fat-Tree_k is a graph with vertex-set that is partitioned into four layers:

1. a *core layer*, labeled with $\{0\} \times \mathbb{Z}_{(k/2)^2}$;
2. an *aggregation layer*, labeled with $\{1\} \times \mathbb{Z}_k \times \mathbb{Z}_{k/2}$. For every $0 \leq i \leq (k/2)^2 - 1$ the vertex labeled $\langle 0, i \rangle$ in the core layer is adjacent to all the vertices labeled $\langle 1, j, i \pmod{k}/2 \rangle$ in the aggregation layer, with $0 \leq j \leq k-1$;
3. an *edge layer*, labeled with $\{2\} \times \mathbb{Z}_k \times \mathbb{Z}_{k/2}$. For every $0 \leq i \leq k-1$ there is a complete join between the subsets of vertices $\{\langle 1, i, j \rangle \mid 0 \leq j \leq k/2 - 1\}$ and $\{\langle 2, i, j \rangle \mid 0 \leq j \leq k/2 - 1\}$;
4. finally, a *server layer* labeled with $\{3\} \times \mathbb{Z}_k \times \mathbb{Z}_{(k/2)^2}$. For any $0 \leq q, r < k/2$ the vertex labeled $\langle 3, k, (k/2)q + r \rangle$ in the server layer is adjacent to the vertex labeled $\langle 2, k, q \rangle$ in the edge layer.

An example of a Fat-Tree_4 is given in Figure 2.

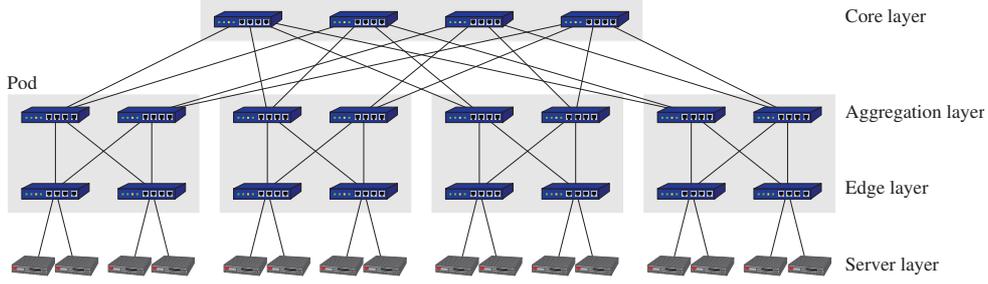


Figure 2: The graph Fat-Tree_4 .

Lemma 19. $\delta(\text{Fat-Tree}_k) = 2$.

Proof. By construction, every vertex in the server layer is a pending vertex, that is a vertex of degree one. As a result, it can be ignored for the computation of hyperbolicity because the hyperbolicity of a graph is equal to the maximum hyperbolicity taken over all its biconnected components. It follows that the efficient diameter of Fat-Tree_k is 4, hence $\delta(\text{Fat-Tree}_k) \leq 2$.

Furthermore, by construction Fat-Tree_4 is an isometric subgraph of Fat-Tree_k . So, let σ be the idempotent endomorphism of Fat-Tree_4 mapping: any vertex $\langle 0, i \rangle$ to the vertex $\langle 0, 3 - i \rangle$ in the core layer; any vertex $\langle 1, i, j \rangle$ to the vertex $\langle 1, 3 - i, 1 - j \rangle$ in the aggregation layer, and in the same way any vertex $\langle 2, i, j \rangle$ to the vertex $\langle 2, 3 - i, 1 - j \rangle$ in the edge layer; last, any vertex $\langle 3, i, j \rangle$ to the vertex $\langle 3, 3 - i, 3 - j \rangle$ in the server layer. It can be hand-checked that σ has mobility 4 and so, by Corollary 9 $\delta(\text{Fat-Tree}_k) \geq \delta(\text{Fat-Tree}_4) \geq 2$. \square

Definition 20 ([47]). The Butterfly graph $BF(n)$ has vertex-set $\{0, 1, \dots, n\} \times \mathbb{Z}_2^n$; two vertices $\langle i, w \rangle, \langle i', w' \rangle$ are adjacent if $i' = i + 1$ and for every $j \neq i, w_j = w'_j$.

Lemma 21. $\delta(BF(n)) = n$.

Proof. Let w and w' be two binary words of length n and let i_1 and i_l be respectively the least and the largest index in which they differ. Then, it can be checked that for every integer i , $d_{BF(n)}(\langle i, w \rangle, \langle i, w' \rangle) = 2(i_l - i_1)$. As a result, the endomorphism σ mapping any vertex $\langle i, w \rangle$ to the vertex $\langle i, \bar{w} \rangle$ has mobility $2n$. Since σ is idempotent then it follows from Corollary 9 that $\delta(BF(n)) \geq n$. Last, we also have that $\text{diam}(BF(n)) = 2n$, hence $\delta(BF(n)) \leq n$ by Lemma 2. \square

In the literature, the edge-set of the Butterfly network is sometimes defined as $\{\{\langle i, w \rangle, \langle i + 1 \pmod{n}, w' \rangle\} \mid 0 \leq i \leq n \text{ and for every } j \neq i, w_j = w'_j\}$ [81], and this definition is also known as the wrapped Butterfly network. It modifies the diameter of the topology from $2n$ to $n + \lfloor n/2 \rfloor$, and the distance between any two vertices $\langle i, w \rangle, \langle i, \bar{w} \rangle$ from $2n$ to n . As a result, using the same arguments as for Lemma 21 one obtains that the hyperbolicity of the wrapped Butterfly graph is comprised between $\lfloor n/2 \rfloor$ and $\lfloor (n + \lfloor n/2 \rfloor) / 2 \rfloor$.

Definition 22 ([20]). The k -ary n -fly has vertex-set $\{0, 1, \dots, n\} \times \mathbb{Z}_k^n$; two vertices $\langle i, w \rangle, \langle i', w' \rangle$ are adjacent if $i' = i + 1$ and for every $j \neq i, w_j = w'_j$.

Observe that the Butterfly graph $BF(n)$ is isomorphic to the 2-ary n -fly.

Lemma 23. *The k -ary n -fly is n -hyperbolic.*

Proof. By [20], the diameter of the k -ary n -fly is $2n$ and so, it has hyperbolicity bounded from above by n by Lemma 2. Moreover, by construction it contains the Butterfly graph $BF(n)$ as an isometric subgraph and so, it has hyperbolicity at least n by Lemma 21. \square

Definition 24 ([20]). The k -ary n -tree is the graph with vertex-set $\mathbb{Z}_k^n \cup (\{0, 1, \dots, n-1\} \times \mathbb{Z}_k^{n-1})$ such that any two vertices $\langle i, w \rangle, \langle i', w' \rangle$ are adjacent if $i' = i + 1$ and for every $j \neq i$, $w_j = w'_j$; any two vertices $\langle i, w \rangle, w'$ are adjacent if $i = n - 1$ and $w' = w \cdot b$ for some $b \in \mathbb{Z}_k$.

Lemma 25. *The k -ary n -tree is $(n - 1)$ -hyperbolic.*

Proof. By construction, the biconnected components of the k -ary n -tree are composed of one single-vertex graph for each vertex $w \in \mathbb{Z}_k^n$, and of the k -ary $(n - 1)$ -fly. Since the hyperbolicity of the graph is equal to the maximum hyperbolicity taken over its biconnected components, then it follows from Lemma 23 that the k -ary n -tree is $(n - 1)$ -hyperbolic. \square

Definition 26 ([81]). The d -ary tree grid $MT(d, h)$ is a graph whose vertices are labeled with the pairs of words $\langle u, v \rangle$ over an alphabet of size d and such that $\max\{|u|, |v|\} = h$. Any two vertices $\langle u, v \rangle$ and $\langle u', v' \rangle$ are adjacent in $MT(d, h)$ if and only if there is some letter λ such that: either $|u| = h$, $u = u'$ and $v = v' \cdot \lambda$; or $|v| = h$, $v = v'$ and $u' = u \cdot \lambda$.

Lemma 27. $\delta(MT(d, h)) = 2h$.

Proof. By [81] $\text{diam}(MT(d, h)) = 4h$ and so, $\delta(MT(d, h)) \leq 2h$. Furthermore, $MT(2, h)$ is an isometric subgraph of $MT(d, h)$ by construction. Let σ be the idempotent endomorphism of $MT(2, h)$ mapping any vertex $\langle u, v \rangle$ to the vertex $\langle \bar{u}, \bar{v} \rangle$. By construction σ has mobility $4h$ and so, we conclude by Corollary 9 that $\delta(MT(d, h)) \geq \delta(MT(2, h)) \geq 2h$. \square

3.2.4 Symmetric networks and Cayley graphs

Let (Γ, \cdot) be a group and let S be a generating set of Γ that is symmetric and that does not contain the neutral element of Γ . We remind that the *Cayley graph* $G(\Gamma, S)$ —of group Γ w.r.t. S — has vertex-set Γ and edge-set $\{\{g, g \cdot s\} \mid g \in \Gamma, s \in S\}$. It is well-known that every Cayley graph is vertex-transitive [12]. Furthermore, it has been shown (see for instance Exercise 2.4.14 in [81]) that the cube connected cycle $CCC(n)$ and the Butterfly graph $BF(n)$ are Cayley graphs.

Lemma 28. *Let (Γ, \cdot) be a commutative group and S be a symmetric generating set that does not contain the neutral element of Γ . If $G(\Gamma, S)$ is not a clique, then $\delta(G(\Gamma, S)) \geq \frac{1}{2} \left\lceil \frac{\text{diam}(G(\Gamma, S))}{2} \right\rceil$.*

Proof. Let $id_\Gamma, g \in \Gamma$ be such that id_Γ is the neutral element of group Γ and $d(id_\Gamma, g) = \text{diam}(G(\Gamma, S)) = D > 1$. The mapping $\sigma : v \rightarrow g \cdot v$ is an automorphism satisfying that for every $v \in \Gamma$, $d(v, \sigma(v)) = d(id_\Gamma, v^{-1} \cdot g \cdot v) = d(id_\Gamma, g) = D$. Therefore, we can conclude by Theorem 7 that $\delta(G(\Gamma, S)) \geq \lceil D/2 \rceil / 2$. \square

Definition 29 ([12]). The Bubble-sort graph $BS(n)$ has vertex-set the n -element permutations, that is $\{\phi_1 \phi_2 \dots \phi_i \dots \phi_n \mid \{\phi_1, \dots, \phi_n\} = \{1, \dots, n\}\}$. Any two vertices ϕ, ψ are adjacent if and only if there is some index $i < n$ such that $\phi_i = \psi_{i+1}$, $\phi_{i+1} = \psi_i$ and for every $j \notin \{i, i+1\}$, $\phi_j = \psi_j$.

Lemma 30. $\delta(BS(n)) = \left\lfloor \frac{n(n-1)}{4} \right\rfloor$.

Proof. By [12] $\text{diam}(BS(n)) = \binom{n}{2}$, hence $\delta(BS(n)) \leq \lfloor \text{diam}(BS(n))/2 \rfloor$ by Lemma 2. Now, let σ be the idempotent endomorphism mapping any vertex $\phi_1\phi_2\dots\phi_i\dots\phi_n$ to $\phi_n\dots\phi_{n-i+1}\dots\phi_2\phi_1$. By [12] all pairs $(u, \sigma(u))$ are diametral pairs and so, we can conclude by Corollary 9 that $\delta(BS(n)) \geq \lfloor \text{diam}(BS(n))/2 \rfloor$. \square

Definition 31 ([17]). The Transposition graph $T(n)$ has vertex-set the n -element permutations. Any two vertices ϕ, ψ are adjacent if and only if there are i, j , $i \neq j$ such that $\phi_i = \psi_j, \phi_j = \psi_i$ and for every $k \notin \{i, j\}$, $\phi_k = \psi_k$.

Lemma 32. $\frac{1}{2} \lceil \frac{n-1}{2} \rceil \leq \delta(T(n)) \leq \lfloor \frac{n-1}{2} \rfloor$.

Proof. By [17] the diameter of $T(n)$ is $n-1$ and so, by Lemma 2 $\delta(T(n)) \leq \lfloor (n-1)/2 \rfloor$. Moreover, let σ be the endomorphism mapping any vertex $\phi_1\phi_2\dots\phi_i\dots\phi_{n-1}\phi_n$ to $\phi_2\phi_3\dots\phi_{i+1}\dots\phi_n\phi_1$. Again by [17] all pairs $(u, \sigma(u))$ are diametral pairs and so, we can conclude by Theorem 7 that $\delta(S(n)) \geq \lceil n-1/2 \rceil / 2$. \square

Definition 33 ([12]). The star graph $S(n)$ has vertex-set the n -element permutations and edge-set $\{\{\phi_1\dots\phi_{i-1}\phi_i\phi_{i+1}\dots\phi_n, \phi_i\dots\phi_{i-1}\phi_1\phi_{i+1}\dots\phi_n\} \mid 2 \leq i \leq n\}$.

Lemma 34. $\left\lfloor \frac{1}{2} \left\lfloor \frac{3(n-1)}{2} \right\rfloor - \frac{1}{2} \right\rfloor \leq \delta(S(n)) \leq \left\lfloor \frac{1}{2} \left\lfloor \frac{3(n-1)}{2} \right\rfloor \right\rfloor$.

Proof. By [12] the diameter of $S(n)$ is $\lfloor 3(n-1)/2 \rfloor$ and so, $\delta(S(n)) \leq \lfloor \lfloor 3(n-1)/2 \rfloor / 2 \rfloor$ by Lemma 2. Then, given $\phi = \phi_1\phi_2\dots\phi_i\dots\phi_{n-1}\phi_n$, let ψ be the unique n -element permutation satisfying that $\psi_{n-2j} = \phi_{n-2j-1}, \psi_{n-2j-1} = \phi_{n-2j}$, for every $0 \leq j \leq \lfloor (n-1)/2 \rfloor - 1$. Again by [12], $d(\psi, \phi) \geq \lfloor 3(n-1)/2 \rfloor - \varepsilon \geq \lfloor 3(n-1)/2 \rfloor - 1$, with $\varepsilon = n+1 \pmod{2}$. Moreover it can be checked that the mapping $\sigma : \psi \rightarrow \phi$ is an idempotent endomorphism of $S(n)$. Therefore, by Corollary 9 $\delta(S(n)) \geq \lfloor \lfloor 3(n-1)/2 \rfloor / 2 - 1/2 \rfloor$. \square

4 Using the shortest-path distribution

It turns out that for “simple” topologies that are commonly found in the literature, desirable symmetries such as those in use in Section 3 might fail to exist. For instance, the infinite rectangular grid is vertex-symmetric, but finite rectangular grids are not. As we will show next, the more generic Theorem 4 could still be applied in order to obtain loose lower-bounds in these situations. However, since the shortest-path distributions of the “simplest” topologies are well-known and characterized, that allows us to lower-bound their hyperbolicity using more involved techniques. In particular, our proofs for grid-like graphs introduce a novel way to make use of the maximal shortest-paths in the study of graph hyperbolicity.

4.1 The fellow traveler property for graphs defined on an alphabet

As a warm up, we will lower-bound the hyperbolicity of some graph classes defined on alphabets, starting with the undirected de Bruijn graph.

Definition 35 ([13]). The undirected de Bruijn graph $UB(d, D)$ has vertex-set the words of length D taken over an alphabet Σ of size d . The 2-set $\{u, v\}$ is an edge of $UB(d, D)$ if and only if $u = u_{d-1}u_{d-2}\dots u_1u_0$ and $v = u_{d-2}\dots u_1u_0v_0$ for some letters $u_{d-1}, u_{d-2}, \dots, u_1, u_0, v_0 \in \Sigma$.

De Bruijn graphs have been extensively studied in the literature [15, 24, 27, 81]. In particular, $UB(d, D)$ has diameter D , maximum degree $2d$, and d^D vertices. Shortest-path routing and shortest-path distances in $UB(d, D)$ are characterized as follows.

Lemma 36 ([24]). *Let u, v be two words of length D taken over some alphabet Σ of size d , and write $u = u_L \cdot x \cdot u_R$ and $v = v_L \cdot x \cdot v_R$ so that $D - |x| + \min\{|u_L| + |v_R|, |v_L| + |u_R|\}$ is minimized. Then it holds $d_{UB(d, D)}(u, v) = D - |x| + \min\{|u_L| + |v_R|, |v_L| + |u_R|\}$.*

We say that a graph G falsifies the k -fellow traveler property if there are two shortest-paths $\mathcal{P}_1, \mathcal{P}_2$ with same endpoints $u, v \in V(G)$, and there are two vertices $x \in \mathcal{P}_1, y \in \mathcal{P}_2$ such that $d_G(u, x) = d_G(u, y)$ and $d_G(x, y) > k$. By a straightforward calculation we obtain that in such a case $\delta(u, v, x, y) = d_G(x, y)/2 > k/2$. So, we can lower-bound the hyperbolicity of G with the least k such that it satisfies the $2k$ -fellow traveler property. This standard argument will be the one in use throughout the remaining of Section 4.1.

Proposition 37. *For any positive integers d and D , $\delta(UB(d, D)) \geq \frac{1}{2} \cdot \lfloor \frac{D}{2} \rfloor$.*

Proof. We prove that $UB(d, D)$ cannot satisfy the k -fellow traveler property for some range of k . W.l.o.g. the vertices of $UB(d, D)$ are labeled with the words of length D taken over the alphabet $\Sigma = \{0, 1, \dots, d-1\}$. Let $u = 0^D$, $v = 1^D$, $x = 0^{\lfloor D/2 \rfloor} \cdot 1^{\lceil D/2 \rceil}$, and $y = 1^{\lfloor D/2 \rfloor} \cdot 0^{\lceil D/2 \rceil}$. By Lemma 36 it comes that $d(u, v) = D = \lceil D/2 \rceil + \lfloor D/2 \rfloor = d(u, x) + d(x, v) = d(u, y) + d(y, v)$. As a result, the graph $UB(d, D)$ cannot satisfy the k -fellow traveler property for $k < d(x, y) = \lfloor D/2 \rfloor$ and so, $\delta(UB(d, D)) \geq \lfloor D/2 \rfloor / 2$. \square

To compare the bounds of Theorem 4 and Proposition 37, we note that it has been proved in [79] that de Bruijn graphs with maximum degree $d \geq 3$ are self-centered. Therefore, if $d \geq 3$ then Proposition 37 follows from Theorem 4 (with $k = 0$), but it is not the case if $d = 2$. Furthermore, the lower-bound of Proposition 37 is reached for $d = D = 2$, *a.k.a.* the diamond graph. It can be computer-checked that it also holds for $d = 2, D = 4$. However, $\delta(UB(2, D)) = \lfloor \frac{D}{2} \rfloor$ for every odd $D \leq 11$. Based on computer experiments (for $d = 2, D \leq 12$), we made the following stronger conjecture:

Conjecture 38. *For every $D \geq 7$, $\delta(UB(d, D)) = \lfloor \frac{D}{2} \rfloor$.*

A closely related graph class that has been extensively studied in the literature is the class of undirected Kautz graphs $UK(d, D)$ [2, 13]. The graph $UK(d, D)$ has diameter D , maximum degree $2d$, and $d^D(d+1)$ vertices. Furthermore, it can be checked that the Kautz graph $UK(d, D)$ is an induced subgraph of the de Bruijn graph $UB(d+1, D)$.

Definition 39 ([2, 13]). The undirected Kautz graph $UK(d, D)$ has vertex-set the words of length D taken over an alphabet Σ of size $d+1$ and satisfying that no two adjacent letters are equal. The 2-set $\{u, v\}$ is an edge of $UK(d, D)$ if and only if $u = u_{d-1}u_{d-2} \dots u_1u_0$ and $v = u_{d-2} \dots u_1u_0v_0$ for some letters $u_{d-1}, u_{d-2}, \dots, u_1, u_0, v_0 \in \Sigma$.

Proposition 40. *For any positive integers d and D , $\delta(UK(d, D)) \geq \lfloor \frac{D}{4} \rfloor + \left\lfloor \frac{D \pmod{4}}{3} \right\rfloor$.*

Proof. As for the proof of Proposition 37, we prove that $UK(d, D)$ cannot satisfy the k -fellow traveler property for some range of k . W.l.o.g. the vertices of $UK(d, D)$ are labeled with the words of length D taken over the alphabet $\{0, 1, 2, \dots, d\}$. Let $D = 2D' + r$, $r \in \{0, 1\}$, let

$u = (01)^{D'} \cdot 0^r, v = (21)^{D'} \cdot 2^r$. Note that 0^r (resp. 2^r) is either the empty word or it is equal to 0 (resp. to 2). By Lemma 36 $d_{UK(d,D)}(u, v) \geq d_{UB(d+1,D)}(u, v) = D$ and so, $d_{UK(d,D)}(u, v) = D$ because $\text{diam}(UK(d, D)) = D$. In particular, let \mathcal{P}_1 be the uv -shortest-path in $UK(d, D)$ that one obtains by applying ‘‘right shiftings’’ on u until one obtains vertex v *i.e.*,

$$\mathcal{P}_1 = (01)^{D'} \cdot 0^r \rightarrow 1 \cdot (01)^{D'-1} \cdot 0^r \cdot 2 \rightarrow (01)^{D'-1} \cdot 0^r \cdot 21 \rightarrow \dots \rightarrow (21)^{D'} \cdot 2^r$$

Similarly, let \mathcal{P}_2 be the vu -shortest-path in $UK(d, D)$ that one obtains by applying ‘‘right shiftings’’ on v until one obtains vertex u . That is,

$$\mathcal{P}_2 = (21)^{D'} \cdot 2^r \rightarrow 1 \cdot (21)^{D'-1} \cdot 2^r \cdot 0 \rightarrow (21)^{D'-1} \cdot 2^r \cdot 01 \rightarrow \dots \rightarrow (01)^{D'} \cdot 0^r$$

Let now

$$x = (01)^{\lfloor D'/2 \rfloor} \cdot 0^r \cdot (21)^{\lceil D'/2 \rceil} \in \mathcal{P}_1$$

$$\text{and } y = 1^r \cdot (21)^{\lceil D'/2 \rceil - r} \cdot 2^r \cdot (01)^{\lfloor D'/2 \rfloor} \cdot 0^r \in \mathcal{P}_2$$

be such that $d(u, x) = d(u, y)$.

The graph $UK(d, D)$ falsifies the k -fellow traveler property for all $k < d_{UK(d,D)}(x, y)$, and we have by Lemma 36 that $d_{UK(d,D)}(x, y) \geq d_{UB(d+1,D)}(x, y) \geq 2(\lfloor D/4 \rfloor + \lfloor (D \pmod{4})/3 \rfloor)$.

As a result, it holds $\delta(UK(d, D)) \geq \lfloor D/4 \rfloor + \lfloor (D \pmod{4})/3 \rfloor$. \square

The lower-bound of Proposition 40 is reached for $d = 2, D = 3$. Again to compare with Theorem 4, we note that it was also proved in [79] that Kautz graphs are self-centered, for every $d \geq 2$. Therefore, applying Theorem 4 (with $k = 0$) gives us a lower-bound $\lfloor D/2 \rfloor / 2$ for the hyperbolicity of $UK(d, D)$, that is of the same order of magnitude as the one of Proposition 40 (Proposition 40 is slightly better if $D \equiv 3 \pmod{4}$, and slightly worse if $D \equiv 2 \pmod{4}$). We last define another topology that is related to the de Bruijn graph:

Definition 41 ([81]). The shuffle-exchange graph $SE(n)$ has vertex-set the binary words of length n . The 2-set $\{u, v\}$ is an edge of $SE(n)$ if and only if $u = u_{n-1}u_{n-2} \dots u_1u_0$ and: either $v = u_0u_{n-1}u_{n-2} \dots u_1$, or $v = u_{n-2} \dots u_1u_0u_{n-1}$, or $v = u_{n-1}u_{n-2} \dots u_1\bar{u}_0$, for some booleans $u_{n-1}, u_{n-2}, \dots, u_1, u_0$.

It was proved in [81] that the diameter of $SE(n)$ is $2n-1$, and that the pair of vertices $(0^n, 1^n)$ is a diametral pair. Furthermore, it can be checked that one can obtain the de Bruijn graph $UB(2, n-1)$ from $SE(n)$ as follows: for each edge $\{u_{n-1}u_{n-2} \dots u_1u_0, u_{n-1}u_{n-2} \dots u_1\bar{u}_0\}$, we contract the edge and we label $u_{n-1}u_{n-2} \dots u_1$ the resulting vertex. This defines a *contraction mapping* σ , mapping any vertex $u_{n-1}u_{n-2} \dots u_1u_0$ of $SE(n)$ to the vertex $u_{n-1}u_{n-2} \dots u_1$ of $UB(2, n-1)$. In the following, it will be useful to observe that by construction, for every two vertices u, v of $SE(n)$ it holds $d_{SE(n)}(u, v) \geq d_{UB(2, n-1)}(\sigma(u), \sigma(v))$.

Proposition 42. For any positive integer n , $\delta(SE(n)) \geq \frac{1}{2} \cdot \lfloor \frac{n}{2} \rfloor$.

Proof. As for the proof of Proposition 37, we prove that $SE(n)$ cannot satisfy the k -fellow traveler property for some range of k . Let $u = 0^n, v = 1^n$ be a diametral pair of $SE(n)$, with $d(u, v) = 2n-1$. Let \mathcal{P}_1 be the uv -shortest-path:

$$0^n \rightarrow 0^{n-1} \cdot 1 \rightarrow 1 \cdot 0^{n-1} \rightarrow 1 \cdot 0^{n-2} \cdot 1 \rightarrow 11 \cdot 0^{n-2} \rightarrow \dots \rightarrow 1^{n-1} \cdot 0 \rightarrow 1^n$$

Similarly, let \mathcal{P}_2 be the vu -shortest-path:

$$1^n \rightarrow 1^{n-1} \cdot 0 \rightarrow 0 \cdot 1^{n-1} \rightarrow 0 \cdot 1^{n-2} \cdot 0 \rightarrow 00 \cdot 1^{n-2} \rightarrow \dots \rightarrow 0^{n-1} \cdot 1 \rightarrow 0^n.$$

Finally, let $x = 1^{\lfloor n/2 \rfloor} \cdot 0^{\lceil n/2 \rceil} \in \mathcal{P}_1$, $y = 0^{\lceil n/2 \rceil - 1} \cdot 1^{\lfloor n/2 \rfloor} \cdot 0 \in \mathcal{P}_2$ be such that $d(u, x) = d(u, y)$. By using the above contraction mapping from $SE(n)$ to $UB(2, n-1)$ one obtains $d_{UB(2, n-1)}(x', y') \leq d_{SE(n)}(x, y)$ with $x' = 1^{\lfloor n/2 \rfloor} \cdot 0^{\lceil n/2 \rceil - 1}$, $y' = 0^{\lceil n/2 \rceil - 1} \cdot 1^{\lfloor n/2 \rfloor}$. As a result, we have by Lemma 36 that the shuffle-exchange graph falsifies the k -fellow traveler property for every $k < d_{UB(2, n-1)}(x', y') = \lfloor \frac{n}{2} \rfloor$ and so, it holds $\delta(SE(n)) \geq \frac{1}{2} \cdot \lfloor \frac{n}{2} \rfloor$. \square

4.2 The maximal shortest-paths in grid-like topologies

In this section, we name grid-like graphs some slight variations of the 2-dimensional grid. As a reminder, an (n, m) -grid is the Cartesian product of the path P_n , with n vertices, with the path P_m , with m vertices. That is, the vertex-set is $\{0, \dots, n-1\} \times \{0, \dots, m-1\}$, and the edge-set is $\{(i, j), (i', j')\} \mid |i - i'| + |j - j'| = 1\}$. Grid-like networks are used for modeling interconnection networks and other computational applications. We now propose to compute their hyperbolicity. Our main tool in this section is the notion of *far-apart pairs*, first introduced in [41, 52]:

Definition 43 (Far-apart pair [41, 52]). Given $G = (V, E)$, the pair (u, v) is far-apart if for every $w \in V \setminus \{u, v\}$, $d(w, u) + d(w, v) > d(w, v)$ and $d(w, v) + d(u, v) > d(w, u)$.

Said differently, far-apart pairs are the endpoints of *maximal* shortest-paths in the graph. The main motivation for introducing far-apart pairs was to speed-up the computation of hyperbolicity, via the following pre-processing method.

Lemma 44 ([41, 52]). *Let G be a connected graph. There exist two far-apart pairs (u, v) and (x, y) satisfying:*

- $d_G(u, v) + d_G(x, y) \geq \max\{d_G(u, x) + d_G(v, y), d_G(u, y) + d_G(v, x)\}$;
- $\delta(u, v, x, y) = \delta(G)$.

We here propose a novel application of this result in order to simplify proofs for the hyperbolicity of grid-like topologies.

Definition 45. The (s_1, s_2, \dots, s_d) -grid is a graph with vertex set $\prod_{i=1}^d \{0, \dots, s_i - 1\}$ such that any two vertices $\langle u_1, u_2, \dots, u_d \rangle, \langle v_1, v_2, \dots, v_d \rangle$ are adjacent only if $\sum_{i=1}^d |u_i - v_i| = 1$.

Definition 46. The d -dimensional grid of size s is the (s_1, s_2, \dots, s_d) -grid with for every i , $s_i = s$.

Let us determine the hyperbolicity of the above graphs. By doing so, we answer an open question of the literature [44, Remark 7].

Proposition 47. *The (s_1, s_2, \dots, s_d) -grid has hyperbolicity:*

$$h_d(s_1, s_2, \dots, s_d) = \max_{\mathcal{E} \subseteq \{1, \dots, d\}} \min \left\{ \sum_{i \in \mathcal{E}} s_i - 1, \sum_{i \notin \mathcal{E}} s_i - 1 \right\}.$$

Proof. The 2^{d-1} far-apart pairs of the grid are the diametral pairs $\{(\langle u_1, \dots, u_d \rangle, \langle v_1, \dots, v_d \rangle) \mid \forall i, \{u_i, v_i\} = \{0, s_i - 1\}\}$. Let $(\langle u_1, \dots, u_d \rangle, \langle v_1, \dots, v_d \rangle)$ and $(\langle x_1, \dots, x_d \rangle, \langle y_1, \dots, y_d \rangle)$ be two such pairs, denoted with (\vec{u}, \vec{v}) and (\vec{x}, \vec{y}) for short. Finally, let $D = \sum_i s_i - 1$ be the diameter of the grid and let $l = \sum_{i|u_i \neq x_i} s_i - 1$. Then it comes:

$$\begin{aligned} S_1 &= d(\vec{u}, \vec{v}) + d(\vec{x}, \vec{y}) = 2D \\ S_2 &= d(\vec{u}, \vec{x}) + d(\vec{v}, \vec{y}) = 2l \\ S_3 &= d(\vec{u}, \vec{y}) + d(\vec{v}, \vec{x}) = 2(D - l). \end{aligned}$$

As a result, $\delta(\vec{x}, \vec{y}, \vec{u}, \vec{v}) = \min\{l, D - l\}$ which is maximum for $l = h_d(s_1, s_2, \dots, s_d)$. We conclude that $h_d(s_1, s_2, \dots, s_d)$ is the hyperbolicity by Lemma 44. \square

We highlight two particular cases of Proposition 47 that were already known in the literature.

Corollary 48 ([44, 55]). *The (n, m) -grid is $(\min\{n, m\} - 1)$ -hyperbolic.*

Corollary 49 ([55]). *The d -dimensional grid of size s is $(s - 1) \cdot \lfloor \frac{d}{2} \rfloor$ -hyperbolic.*

Similar results can be obtained for other grid-like graphs which can be found in the literature. We prove some of these results before concluding this section.

Definition 50. The triangular (n, m) -grid is a supergraph of the (n, m) -grid with same vertex-set and with additional edges $\{(i, j), (i + 1, j + 1)\}$ for every $0 \leq i \leq n - 2$ and $0 \leq j \leq m - 2$.

An example of a triangular $(6, 7)$ -grid is given in Figure 3a.

Lemma 51. *The triangular (n, m) -grid is $\frac{\min\{n, m\} - 1}{2}$ -hyperbolic.*

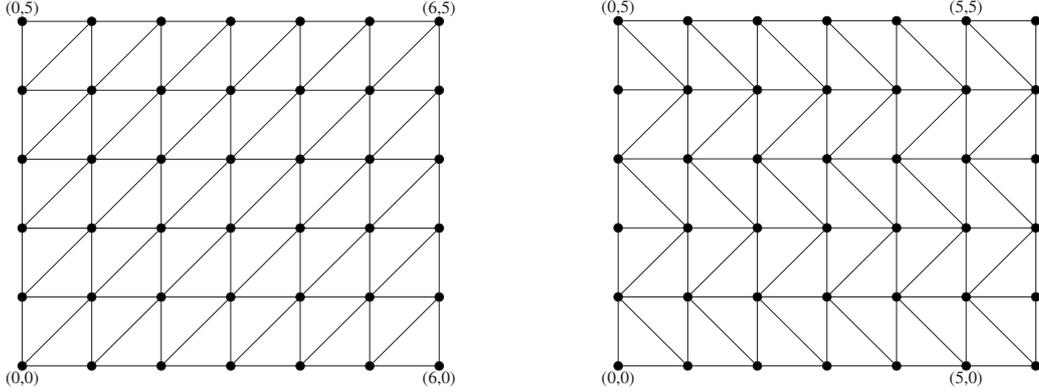
Proof. Let $u = (i_u, j_u)$ and $v = (i_v, j_v)$ be two vertices of the grid. We can assume w.l.o.g. that $i_u \geq i_v$. In such a case, either $j_u \geq j_v$ and so, $d(u, v) = \max\{i_u - i_v, j_u - j_v\}$; or $j_u < j_v$ and so, $d(u, v) = (i_u - i_v) + (j_v - j_u)$. We deduce from the above characterization that there is only one far-apart pair (u, v) such that $d(u, v) \neq \max\{|i_u - i_v|, |j_u - j_v|\}$ namely, $u = (n - 1, 0)$ and $v = (0, m - 1)$ for which $d(u, v) = n + m - 2$. Furthermore, for any other far-apart pair (x, y) either $d(x, y) = n - 1$ or $d(x, y) = m - 1$.

Let (u, v) and (x, y) be two far-apart pairs satisfying the conditions of the above Lemma 44. We assume w.l.o.g. that $d(u, v) \geq d(x, y)$, and we claim that $2\delta(u, v, x, y) \leq \min\{n, m\} - 1$. First, by [74] $2\delta(u, v, x, y) \leq \min\{d(u, v), d(x, y)\} \leq d(x, y)$. Note that $d(x, y) = k \in \{n - 1, m - 1\}$ by the above characterization of the far-apart pairs in the grid. As a result, if $n = m$ then we are done because $d(x, y) = \min\{n, m\} - 1$.

For the remaining of the proof, we will suppose that $n \neq m$ and $d(x, y) = \max\{n, m\} - 1 = k$ (else we are done because $d(x, y) = \min\{n, m\} - 1$). If $k = n - 1$, it implies that $d(u, v) \geq |i_u - i_v| = |i_x - i_y| = d(x, y) = n - 1$; else, it implies $d(u, v) \geq |j_u - j_v| = |j_x - j_y| = d(x, y) = m - 1$. Therefore, we always have that $\max\{d(u, x) + d(v, y), d(u, y) + d(v, x)\} \geq 2k$. It follows by Lemma 44 that the hyperbolicity of the triangular grid is:

$$\begin{aligned} 2\delta(u, v, x, y) &= d(u, v) + d(x, y) - \max\{d(u, x) + d(v, y), d(u, y) + d(v, x)\} \\ &\leq n + m - 2 + k - 2k = n + m - 2 - \max\{n - 1, m - 1\} = \min\{n, m\} - 1 \end{aligned}$$

The bound is reached by setting $u = (n - 1, 0)$, $v = (0, m - 1)$, $x = (0, 0)$, $y = (n - 1, m - 1)$. \square



(a) The triangular $(7,6)$ -grid has hyperbolicity $\delta = \frac{5}{2} = \delta(u, v, x, y)$ with $u = (6, 0)$, $v = (0, 5)$, $x = (0, 0)$, $y = (6, 5)$.
(b) The hexagonal $(7,6)$ -grid has hyperbolicity $\delta = \frac{5}{2} = \delta(u, v, x, y)$ with $u = (0, 5)$, $v = (5, 0)$, $x = (0, 0)$, $y = (5, 5)$.

Figure 3: Examples of grid-like graphs.

In the example of Figure 3a, the hyperbolicity of the graph is given by the 4-tuple $u = (6, 0)$, $v = (0, 5)$, $x = (0, 0)$, $y = (6, 5)$.

Definition 52. The hexagonal (n, m) -grid is a supergraph of the (n, m) -grid with same vertex-set and with additional edges $\{(i, m - 2j - 1), (i + 1, m - 2j - 2)\} \mid 0 \leq i \leq n - 2 \text{ and } 0 \leq j \leq \lfloor \frac{m}{2} \rfloor - 1\} \cup \{(i, m - 2j - 3), (i + 1, m - 2j - 2)\} \mid 0 \leq i \leq n - 2 \text{ and } 0 \leq j \leq \lfloor \frac{m-1}{2} \rfloor - 1\}$. The additional edges are called *diagonal* edges.

Informally, the difference between the triangular grid and the hexagonal grid is that in the hexagonal grid, the direction of diagonal edges alternate at each row. We refer to Figure 3b for an illustration. The hyperbolicity of hexagonal grids has already received some attention in [28]. In fact, they showed using the hexagonal grid that the gap between hyperbolicity of a graph and the length of its longest isometric cycle can be arbitrarily large (see also [78] for more explanations). However, to the best of our knowledge there was no formal bound so far established for the hyperbolicity of hexagonal grids.

Lemma 53. *The hexagonal (n, m) -grid is $\frac{\min\{n, m\} - 1}{2}$ -hyperbolic.*

Proof. We will first characterize the distances in the grid. Let $u = (i_u, j_u)$, $v = (i_v, j_v)$ be two vertices of the hexagonal grid. W.l.o.g., $i_u \geq i_v$. Let us observe that in order to obtain an uv -shortest-path, it suffices to maximize the number of *diagonal* edges used in the path, that is $\min\{k, |i_u - i_v|\}$ with:

- $k = \lfloor |j_u - j_v|/2 \rfloor$ if both $j_u - j_v$ and $2[m - j_v \pmod{2}] - 1$ have the same sign;
- $k = \lceil |j_u - j_v|/2 \rceil$ otherwise.

As a result $d(u, v) = |i_u - i_v| + |j_u - j_v| - \min\{k, |i_u - i_v|\}$ for some k depending on j_u and j_v , $k \in \{\lfloor |j_u - j_v|/2 \rfloor, \lceil |j_u - j_v|/2 \rceil\}$.

Suppose in addition that (u, v) is a far-apart pair. There are two cases. If $d(u, v) = |j_u - j_v|$ then it is monotonically increasing with $|j_u - j_v|$ and so, $|j_u - j_v| = m - 1$. Else, $d(u, v) =$

$|i_u - i_v| + |j_u - j_v| - k$ for some k *only* depending on j_u and j_v , that is monotonically increasing with $|i_u - i_v|$ and so, $|i_u - i_v| = n - 1$.

Finally, let $(u, v), (x, y)$ be two far-apart pairs satisfying the conditions of the above Lemma 44. We will prove that $2\delta(u, v, x, y) \leq \min\{n, m\} - 1$.

Case $m \leq n$. If $\min\{d(u, v), d(x, y)\} \leq m - 1$ then we are done because by [74] we have that $\delta(u, v, x, y) \leq \min\{d(u, v), d(x, y)\}/2 \leq (m - 1)/2$. Else, we must have that $|i_u - i_v| = |i_x - i_y| = n - 1$ and so, $\max\{d(u, x) + d(v, y), d(u, y) + d(v, x)\} \geq 2(n - 1)$. Since in such a case $d(u, v) + d(x, y) \leq (n - 1 + \lceil(m - 1)/2\rceil) + (n - 1 + \lfloor(m - 1)/2\rfloor) = 2(n - 1) + m - 1$ then it follows once again that $\delta(u, v, x, y) \leq (m - 1)/2$.

Case $m > n$. There are three subcases to be considered.

- Suppose $d(u, v) = |j_u - j_v| = m - 1$, $d(x, y) = |j_x - j_y| = m - 1$. Then it comes that $\max\{d(u, x) + d(v, y), d(u, y) + d(v, x)\} \geq 2(m - 1)$ and so, $\delta(u, v, x, y) = 0$.
- Suppose $d(u, v) = j_u - j_v = m - 1$ and $n - 1 + \lfloor(j_x - j_y)/2\rfloor \leq d(x, y) \leq n - 1 + \lceil(j_x - j_y)/2\rceil$. Then it holds that $d(u, y) + d(v, x) \geq (j_u - j_y) + (j_x - j_v) = (j_u - j_x) + (j_x - j_y) + (j_x - j_v) = m - 1 + (j_x - j_y)$. As a result,

$$2\delta(u, v, x, y) \leq (n - 1 + \lceil(j_x - j_y)/2\rceil + m - 1) - (m - 1 + j_x - j_y) = n - 1 - \lfloor(j_x - j_y)/2\rfloor \leq n - 1.$$

- Else, we consider the smallest hexagonal grid of dimensions (n', m') for which there exists two far-apart pairs (u', v') and (x', y') that satisfy the conditions of the above Lemma 44 and such that $\delta(u', v', x', y') \geq \delta(u, v, x, y)$. We assume w.l.o.g. that $n' < m'$ and $d(u', v') \neq |j_{u'} - j_{v'}|$, $d(x', y') \neq |j_{x'} - j_{y'}|$ (otherwise we fall in one of the above cases). Note that it implies that $|i_{u'} - i_{v'}| = |i_{x'} - i_{y'}| = n' - 1$ by our above characterization of the far-apart pairs.

If the two far-apart pairs are $((0, 0), (n' - 1, m' - 1))$ and $((0, m' - 1), (n' - 1, 0))$, then we obtain by the computation that $2\delta(u', v', x', y') = n' - 1 + (n' - m') < n' - 1 \leq n - 1$.

Else, by minimality of the subgrid there is some vertex in the 4-tuple, say u' , which is contained amongst $\{(0, 0), (n' - 1, m' - 1), (n' - 1, 0), (0, m' - 1)\}$ and such that no other vertex $z \in \{v', x', y'\}$ satisfies that $j_{u'} = j_z$. By symmetry, we will assume that $u' \in \{(0, m' - 1), (n' - 1, m' - 1)\}$. Then, using the above characterization of the distances in the hexagonal grid, it can be checked that for any $0 \leq i \leq n' - 1$ and for any $0 \leq j \leq m' - 2$:

$$\begin{aligned} d((n' - 1, m' - 2), (i, j)) &= d((n' - 1, m' - 1), (i, j)) - 1 \\ \text{and } d((1, m' - 2), (i, j)) &= d((0, m' - 1), (i, j)) - 1 \text{ unless } (i, j) = (0, m' - 2) \end{aligned}$$

Therefore, by the 4-point condition $\delta(u', v', x', y') = \delta((n' - 1, m' - 2), v', x', y')$ when $u' = (n' - 1, m' - 1)$; $\delta(u', v', x', y') \leq \max\{d((0, m' - 1), (0, m' - 2)), \delta((n' - 1, m' - 2), v', x', y')\} \leq \max\{1, \delta((n' - 1, m' - 2), v', x', y')\}$ when $u' = (0, m' - 1)$. In both cases, it contradicts the minimality of (n', m') .

To conclude, let $l = \min\{n, m\} - 1$. The upper-bound $l/2$ for the hyperbolicity is reached by setting $u = (0, m - 1), v = (l, m - 1 - l), x = (0, m - 1 - l), y = (l, m - 1)$. \square

In the example of Figure 3b for an illustration, the hyperbolicity of the graph is given by the 4-tuple $u = (0, 5)$, $v = (5, 0)$, $x = (0, 0)$, $y = (5, 5)$.

Definition 54. The cylinder (n, m) -grid is the supergraph of the (n, m) -grid with the same vertex-set and with additional edge-set $\{(0, j), (n-1, j) \mid 0 \leq j \leq m-1\}$.

In particular, when $m = 1$, then the cylinder (n, m) -grid is the n -cycle C_n . More generally, each row induces a cycle instead of inducing a path.

Lemma 55. *The cylinder (n, m) -grid is*

$$\begin{cases} \lfloor \frac{n}{2} \rfloor\text{-hyperbolic} & \text{when } m > \lfloor \frac{n}{2} \rfloor \\ \left(\frac{\lfloor \frac{n}{2} \rfloor + m}{2} - 1 \right)\text{-hyperbolic} & \text{when } m \leq \lfloor \frac{n}{2} \rfloor \text{ and } (n \text{ is odd or } \lceil \frac{n}{2} \rceil - m + 1 \text{ is odd}) \\ \left(\frac{\lfloor \frac{n}{2} \rfloor + m}{2} - \frac{1}{2} \right)\text{-hyperbolic} & \text{otherwise.} \end{cases}$$

Proof. Let $u = (i_u, j_u), v = (i_v, j_v)$ be two vertices of the grid. We have:

$$d(u, v) = \min\{|i_u - i_v|, n - |i_u - i_v|\} + |j_u - j_v|.$$

As a result, the far-apart pairs of the cylinder (n, m) -grid are exactly the pairs $\{(i, 0), (i + \lfloor n/2 \rfloor \pmod{n}, m-1)\}$, and the pairs $\{(i, 0), (i + \lceil n/2 \rceil \pmod{n}, m-1)\}$, with $0 \leq i \leq n-1$. Equivalently, these are the pairs $\{(u', 0), (v', m-1)\}$ with (u', v') an arbitrary far-apart pair of the n -cycle C_n .

Let (u, v) and (x, y) be two far-apart pairs of the cylinder (n, m) -grid satisfying the conditions of the above Lemma 44. Write $u = (u', 0)$, $v = (v', m-1)$, $x = (x', 0)$, $y = (y', m-1)$. Furthermore, let $S_1 = d(u, v) + d(x, y)$, $S_2 = d(u, x) + d(v, y)$, and $S_3 = d(u, y) + d(v, x)$. Similarly, let $S'_1 = d_{C_n}(u', v') + d_{C_n}(x', y')$, $S'_2 = d_{C_n}(u', x') + d_{C_n}(v', y')$, and $S'_3 = d_{C_n}(u', y') + d_{C_n}(v', x')$. Note that it holds: $S'_1 = 2 \lfloor n/2 \rfloor = \max\{S'_1, S'_2, S'_3\}$; $S_1 = S'_1 + 2(m-1) = 2(\lfloor n/2 \rfloor + m-1)$, $S'_2 = S_2$, and $S_3 = S'_3 + 2(m-1)$.

There are two cases to be considered.

- Suppose that $m > \lfloor n/2 \rfloor$. We have that $\delta(u, v, x, y) \leq (S_1 - S_3)/2 \leq (S'_1 - S'_3)/2 \leq S'_1/2 \leq \lfloor n/2 \rfloor$. The bound is reached by setting $u' = y'$ and $v' = x'$.
- Suppose that $m \leq \lfloor n/2 \rfloor$. If $(u', v') = (y', x')$ then we obtain by the calculation that $\delta(u, v, x, y) = (m-1)/2$. Otherwise, $S'_2 + S'_3 = n$ and hence

$$2\delta(u, v, x, y) = S'_1 - \max\{S'_3, S'_2 - 2(m-1)\} = S'_1 - \max\{S'_3, (n - 2(m-1)) - S'_3\},$$

this is maximum when $\lfloor n/2 \rfloor - (m-1) \leq S'_3 \leq \lceil n/2 \rceil - (m-1)$. In the following, let $\lfloor n/2 \rfloor - (m-1) = 2q + r$ with $0 \leq r \leq 1$. There are two subcases to be considered.

- Assume that n is odd and let us set $u' = 0$, $v' = \lfloor n/2 \rfloor$, $x' = \lfloor n/2 \rfloor - q$, and $y' = n - q - r$. In such a case, $S'_3 = (q+r) + q = 2q + r$. As a result, $\delta(u, v, x, y) = (\lfloor n/2 \rfloor + m)/2 - 1$ and so, the above upper-bound is always reached when n is odd.

- (ii) Assume that n is even. Then, $S'_3 = 2d(u', y')$ cannot be odd. It implies that the hyperbolicity is bounded from above by $n/4 + (m - 1 - r)/2$. We set $u' = 0$, $v' = n/2$, $x' = n/2 - q$, and $y' = n - q$. In such a case, $S'_3 = 2q$, hence $(n - 2(m - 1)) - S'_3 = 4q + 2r - 2q = 2q + 2r$ and so, $\delta(u, v, x, y) = n/4 + (m - 1 - r)/2$ that is maximum. \square

Before concluding this section, let us compare the techniques employed for grids with Theorem 4. It is easy to see that for all grid-like graphs considered (cf. Definitions 10, 50, 52 and 54), there is either a row or a column contained in the center. Therefore, the diameter of the center is at least $\min\{n, m\} - 1$, and so, by [37, Proposition 5] one obtains the lower-bound $\lfloor (\min\{n, m\} - 1)/2 \rfloor / 2$ on the hyperbolicity for all these graphs. In this section, we have conducted an in-depth analysis of their shortest-path distribution in order to establish the exact hyperbolicity of these grid-like graphs.

5 Relations between hyperbolicity and some graph operations

Our results so far are heavily focused on the so-called *homogeneous* data center interconnection networks. By contrast, heterogeneous data centers are based on the composition of homogeneous data center interconnection topologies through graph operations. We survey a few of these operations so that we can study the impact that they may have on the hyperbolicity of the network.

5.1 Biswap operation and biswapped networks

Definition 56 ([35]). Let G be a graph. The biswapped graph $Bsw(G)$ has vertex set $\{0, 1\} \times V(G) \times V(G)$. Two vertices (b, u, v) and (b', u', v') are adjacent if, and only if either $b = b'$, $u = u'$ and $\{v, v'\} \in E(G)$, or $b = \bar{b}' = 1 - b'$, $u = v'$, and $u' = v$.

Lemma 57. For any connected graph G , $\delta(Bsw(G)) = \text{diam}(G) + 1$.

Proof. By [35] $\text{diam}(Bsw(G)) = 2 \cdot \text{diam}(G) + 2$ and so, by Lemma 2 $\delta(Bsw(G)) \leq \text{diam}(G) + 1$. To prove the lower-bound, let $u, v \in V(G)$ be such that $\text{diam}(G) = d_G(u, v)$. We define $\vec{x}_1 = (0, u, v)$, $\vec{x}_2 = (0, v, u)$, $\vec{x}_3 = (1, u, u)$ and $\vec{x}_4 = (1, v, v)$. We deduce from [35] that:

$$\begin{aligned} S_1 &= d(\vec{x}_1, \vec{x}_2) + d(\vec{x}_3, \vec{x}_4) = 2(2d_G(u, v) + 2) \\ S_2 &= d(\vec{x}_1, \vec{x}_3) + d(\vec{x}_2, \vec{x}_4) = 2(d_G(u, v) + 1) \\ S_3 &= d(\vec{x}_1, \vec{x}_4) + d(\vec{x}_2, \vec{x}_3) = 2(d_G(u, v) + 1) \end{aligned}$$

As a result, $\delta(Bsw(G)) \geq \delta(\vec{x}_1, \vec{x}_2, \vec{x}_3, \vec{x}_4) = d_G(u, v) + 1 = \text{diam}(G) + 1$. \square

It follows from Lemma 57 that the hyperbolicity of a biswap network always scales with its diameter, *regardless of the topology that is used for the operation.*

5.2 Generic Cayley construction

Let us finally consider the following transformation of a Hamiltonian graph, and the consequences of it on the hyperbolicity.

Lemma 58. *Let G be a Hamiltonian graph and c be a positive integer. We construct a graph G' from G by replacing every edge in some Hamilton cycle of G with a path of length c . Then, it holds $\delta(G') \geq \frac{1}{2} \lceil \frac{c-1}{2} \rceil$.*

Proof. Let P be a path of length c added by the construction, let x and y be the endpoints of P , and let P' be a xy -shortest-path in $G' \setminus (P \setminus \{x, y\})$. The union of P with P' is an isometric cycle and so, it has length upper-bounded by $4 \cdot \delta(G') + 3$ by [78]. Moreover, the length of P' is at least 2 because $\{x, y\}$ is an edge of G by the hypothesis. Thus it comes that the length of the cycle is at least $c + 2$ and so, $c \leq 4 \cdot \delta(G') + 3$. \square

The Cayley model in [59] aims to apply the construction defined in Lemma 58 to some Hamiltonian graph G of order N , with $c = \Omega(\log N)$ and so that the diameter of the resulting graph G' is $O(\log N)$. Summarizing, we get.

Theorem 59. *Graphs in the Cayley model have hyperbolicity $\Theta(\log N)$, which scales linearly with their diameter.*

6 Conclusion

We proved in this work that the topologies of various interconnection networks have their hyperbolicity that scales linearly with their diameter. This property is inherent to any graph having desired properties for data centers such as a high-level of symmetry. Interestingly, symmetries are a common way to minimize network congestion whereas it was shown in [51], using a simplified model, that a bounded hyperbolicity might explain the congestion phenomenon observed in some real-life networks. This was formally proved in [75] for shortest-path routing, but to the best of our knowledge no relation is known between hyperbolicity and congestion in *general*. Therefore, we let open whether a more general relationship between congestion and hyperbolicity can be determined.

Finally, our results imply that in any greedy routing scheme based on an embedding into the hyperbolic space —and in some cases, on an embedding into some word metric space— there is a *linear* number of routing paths for which the stretch is arbitrarily bad. However, this does not preclude the possibility that for most other routing paths, the stretch is bounded by a small constant. We thus believe that it might be of interest to compute the *average hyperbolicity* [63, 72] of the data center interconnection topologies so as to verify whether it is the case.

Acknowledgments

We wish to thank Miguel Camelo, Lluís Fabrega and Pere Vilà Talleda for having suggested to us this work on the hyperbolicity of data center interconnection networks. We would also like to thank the anonymous reviewers for their valuable feedbacks.

References

- [1] H. S. M. Coxeter. Self-dual configurations and regular graphs. *Bulletin of the American Mathematical Society*, 56:413–455, 1950.

- [2] W.H. Kautz. Bounds on directed (d, k) graphs. Theory of cellular logic networks and machines. *AFCRL-68-0668, SRI Project 7258, Final report*, pages 20–28, 1968.
- [3] W. H. Gates and C. H. Papadimitriou. Bounds for sorting by prefix reversal. *Discrete Mathematics*, 27(1):47–57, 1979.
- [4] F. P. Preparata and J. Vuillemin. The cube-connected cycles: a versatile network for parallel computation. *Communications of the ACM*, 24(5):300–309, 1981.
- [5] R. Nowakowski and P. Winkler. Vertex-to-vertex pursuit in a graph. *Discrete Mathematics*, 43(2):235–239, 1983.
- [6] A. Quilliot. *Problèmes de jeux, de point fixe, de connectivité et de représentation sur des graphes, des ensembles ordonnés et des hypergraphes*. PhD thesis, Université de Paris VI, France, 1983.
- [7] M. Aigner and M. Fromme. A game of cops and robbers. *Discrete Applied Mathematics*, 8(1):1–12, 1984.
- [8] L.N. Bhuyan and D.P. Agrawal. Generalized hypercube and hyperbus structures for a computer network. *IEEE Transactions on Computers*, 100(4):323–333, 1984.
- [9] Jin Akiyama, Kiyoshi Ando, and David Avis. Eccentric graphs. *Discrete Mathematics*, 56(1):1–6, 1985.
- [10] M. Gromov. Hyperbolic groups. *Essays in Group Theory*, 8:75–263, 1987.
- [11] F. Harary, J. P. Hayes, and H.-J. Wu. A survey of the theory of hypercube graphs. *Computers & Mathematics with Applications*, 15(4):277–289, 1988.
- [12] S.B. Akers and B. Krishnamurthy. A group-theoretic model for symmetric interconnection networks. *IEEE Transactions on Computers*, 38(4):555–566, 1989.
- [13] J.-C. Bermond and C. Peyrat. De Bruijn and Kautz networks: a competitor for the hypercube? In *European Workshop on Hypercubes and Distributed Computers*, pages 279–293. North Holland, 1989.
- [14] F. Buckley. Self-centered graphs. *Annals of the New York Academy of Sciences*, 576(1):71–78, 1989.
- [15] J.-C. Bermond and P. Fraigniaud. Broadcasting and gossiping in de Bruijn networks. *SIAM Journal on Computing*, 23(1):212–225, 1994.
- [16] W. D. Neumann and M. Shapiro. Automatic structures, rational growth, and geometrically finite hyperbolic groups. *Inventiones mathematicae*, 120(1):259–287, 1995.
- [17] S. Latifi and P.K. Srimani. Transposition networks as a class of fault-tolerant robust networks. *IEEE Transactions on Computers*, 45(2):230–238, 1996.
- [18] S. Ohring and S. K. Das. Folded Petersen cube networks: New competitors for the hypercubes. *IEEE Transactions on Parallel and Distributed Systems*, 7(2):151–168, 1996.

- [19] I. Friš, I. Havel, and P. Liebl. The diameter of the cube-connected cycles. *Information processing letters*, 61(3):157–160, 1997.
- [20] F. Petrini and M. Vanneschi. k-ary n-trees: High performance networks for massively parallel architectures. In *International Parallel Processing Symposium (IPPS)*, pages 87–93. IEEE, 1997.
- [21] I. Benjamini. Expanders are not hyperbolic. *Israel Journal of Mathematics*, 108(1):33–36, 1998.
- [22] H. N. De Ridder and H. L. Bodlaender. Graph automorphisms with maximal projection distances. In *Fundamentals of Computation Theory*, pages 204–214. Springer, 1999.
- [23] J. Matoušek. On embedding trees into uniformly convex Banach spaces. *Israel Journal of Mathematics*, 114(1):221–237, 1999.
- [24] Jyh-Wen Mao and Chang-Biau Yang. Shortest path routing and fault-tolerant routing on de Bruijn networks. *Networks*, 35(3):207–215, 2000.
- [25] G. Brinkmann, J. H. Koolen, and V. Moulton. On the hyperbolicity of chordal graphs. *Annals of Combinatorics*, 5(1):61–69, 2001.
- [26] J. W. Cannon. Geometric group theory. *Handbook of geometric topology*, pages 261–305, 2002.
- [27] D. Coudert, A. Ferreira, and S. Pérennes. Isomorphisms of the De Bruijn digraph and free-space optical networks. *Networks*, 40(3):155 – 164, 2002.
- [28] J. H. Koolen and V. Moulton. Hyperbolic bridged graphs. *European Journal of Combinatorics*, 23(6):683–699, 2002.
- [29] Suohai Fan. Weakly symmetric graphs and their endomorphism monoids. *Southeast Asian Bulletin of Mathematics*, 27(3), 2003.
- [30] G. E. Jan, Y.-S. Hwang, M.-B. Lin, and D. Liang. Novel hierarchical interconnection networks for high-performance multicomputer systems. *Journal of information science and engineering*, 20:1213–1229, 2004.
- [31] R. Krauthgamer and J. R. Lee. Algorithms on negatively curved spaces. In *Symposium on Foundations of Computer Science (FOCS)*, pages 119–132. IEEE, 2006.
- [32] V. Chepoi and B. Estellon. Packing and covering δ -hyperbolic spaces by balls. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, volume 4627 of *Lecture Notes in Computer Science*, pages 59–73. Springer, 2007.
- [33] R. Kleinberg. Geographic routing using hyperbolic space. In *International Conference on Computer Communications (INFOCOM)*, pages 1902–1909. IEEE, 2007.
- [34] P. Potocnik, M. Sajna, and G. Verret. Mobility of vertex-transitive graphs. *Discrete Mathematics*, 307(3–5):579 – 591, 2007.

- [35] W. Xiao, W. Chen, M. He, W. Wei, and B. Parhami. Biswapped networks and their topological properties. In *ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD)*, volume 2, pages 193–198. IEEE, 2007.
- [36] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. *ACM SIGCOMM Computer Communication Review*, 38(4):63–74, 2008.
- [37] V. Chepoi, F. F. Dragan, B. Estellon, M. Habib, and Y. Vaxès. Diameters, centers, and approximating trees of delta-hyperbolic geodesic spaces and graphs. In *24th Symposium on Computational Geometry (SCG)*, pages 59–68. ACM, 2008.
- [38] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu. DCell: a scalable and fault-tolerant network structure for data centers. *ACM SIGCOMM Computer Communication Review*, 38(4):75–86, 2008.
- [39] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu. BCube: a high performance, server-centric network architecture for modular data centers. *ACM SIGCOMM Computer Communication Review*, 39(4):63–74, 2009.
- [40] D. Li, C. Guo, H. Wu, K. Tan, Y. Zhang, and S. Lu. Ficonn: Using backup port for server interconnection in data centers. In *International Conference on Computer Communications (INFOCOM)*, pages 2276–2285. IEEE, 2009.
- [41] Damien Noguès. δ -hyperbolicité et graphes. Master’s thesis, MPRI, Université Paris 7, 2009.
- [42] F. Papadopoulos, D. Krioukov, M. Boguna, and A. Vahdat. Greedy forwarding in scale-free networks embedded in hyperbolic metric spaces. *ACM SIGMETRICS Performance Evaluation Review*, 37(2):15–17, 2009.
- [43] H. Wu, G. Lu, D. Li, C. Guo, and Y. Zhang. MDCube: a high performance network structure for modular data center interconnection. In *International conference on Emerging networking experiments and technologies*, pages 25–36. ACM, 2009.
- [44] Y. Wu and C. Zhang. Chordality and hyperbolicity of a graph. Technical Report arXiv:0910.3544, ArXiv, 2009.
- [45] M. Boguna, F. Papadopoulos, and D. Krioukov. Sustaining the Internet with Hyperbolic Mapping. *Nature Communications*, 1(62), 2010.
- [46] M. Kliegl, J. Lee, J. Li, X. Zhang, C. Guo, and D. Rincon. Generalized DCell structure for load-balanced data center networks. In *INFOCOM IEEE Conference on Computer Communications Workshops*, pages 1–5. IEEE, 2010.
- [47] M.H. Mahafzah. Fault-tolerant routing in butterfly networks. *Journal of Applied Sciences*, 10(11):903–908, 2010.
- [48] C. Cassagnes, T. Tiendrebeogo, D. Bromberg, and D. Magoni. Overlay addressing and routing system based on hyperbolic geometry. In *IEEE Symposium on Computers and Communications (ISCC)*, pages 294–301. IEEE, 2011.

- [49] J. Chalopin, V. Chepoi, N. Nisse, and Y. Vaxès. Cop and robber games when the robber can hide and ride. *SIAM Journal on Discrete Mathematics*, 25(1):333–359, 2011.
- [50] L. Ferretti and M. Cortelezzi. Preferential attachment in growing spatial networks. *Physical Review E*, 84(1), 2011.
- [51] E. Jonckheere, M. Lou, F. Bonahon, and Y. Baryshnikov. Euclidean versus hyperbolic congestion in idealized versus experimental networks. *Internet Mathematics*, 7(1):1–27, 2011.
- [52] M. A. Soto Gómez. *Quelques propriétés topologiques des graphes et applications à internet et aux réseaux*. PhD thesis, Univ. Paris Diderot (Paris 7), 2011.
- [53] Y. Wu and C. Zhang. Hyperbolicity and chordality of a graph. *The Electronic Journal of Combinatorics*, 18(1), 2011.
- [54] I. Benjamini and O. Schramm. Finite transitive graph embeddings into a hyperbolic metric space must stretch or squeeze. In *Geometric aspects of functional analysis*, volume 2050 of *Lecture Notes in Mathematics*, pages 123–126. Springer, 2012.
- [55] N. Cohen, D. Coudert, and A. Lancin. Exact and approximate algorithms for computing the hyperbolicity of large-scale graphs. Research Report RR-8074, Inria, September 2012.
- [56] O. Narayan, I. Saniee, and G. H Tucci. Lack of spectral gap and hyperbolicity in asymptotic erdős-renyi sparse random graphs. In *International Symposium on Communications Control and Signal Processing (ISCCSP)*, pages 1–4. IEEE, 2012.
- [57] F. Papadopoulos, M. Kitsak, M. Serrano, M. Boguná, and D. Krioukov. Popularity versus similarity in growing networks. *Nature*, 489(7417):537–540, 2012.
- [58] J. M. Rodríguez and J. M. Sigarreta. Bounds on Gromov hyperbolicity constant in graphs. In *Proceedings Indian Acad. Sci. (Mathematical Sciences)*, volume 122, pages 53–65. Springer, 2012.
- [59] W. Xiao, H. Liang, and B. Parhami. A class of data-center network models offering symmetry, scalability, and reliability. *Parallel Processing Letters*, 22(04), 2012.
- [60] A. B. Adcock, B. D. Sullivan, and M. W. Mahoney. Tree-like structure in large social and information networks. In *13th International Conference on Data Mining*, pages 1–10. IEEE, 2013.
- [61] Wei Chen, Wenjie Fang, Guangda Hu, and Michael W. Mahoney. On the hyperbolicity of small-world and treelike random graphs. *Internet Mathematics*, 9(4):434–491, 2013.
- [62] W.S. Kennedy, O. Narayan, and I. Saniee. On the hyperbolicity of large-scale networks. Technical Report arXiv:1307.0031, ArXiv, 2013.
- [63] R. Albert, B. DasGupta, and N. Mobasher. Topological implications of negative curvature for biological and social networks. *Physical Review E*, 89(3):032811, 2014.
- [64] M. Camelo, D. Papadimitriou, L. Fabrega, and P. Vila. Geometric routing with word-metric spaces. *Communications Letters, IEEE*, 18(12):2125–2128, 2014.

- [65] M. Camelo, D. Papadimitriou, L. Fàbrega, and P. Vilà. Efficient routing in Data Center with underlying Cayley graph. In *Complex Networks V*, volume 549 of *Studies in Computational Intelligence*, pages 189–197. Springer, 2014.
- [66] M. Camelo, P. Vilà, L. Fàbrega, and D. Papadimitriou. Cayley-graph-based data centers and space requirements of a routing scheme using automata. In *International Conference on Distributed Computing Systems Workshops (ICDCS)*, pages 63–69. IEEE, 2014.
- [67] Nathann Cohen, David Coudert, Guillaume Ducoffe, and Aurélien Lancin. Applying clique-decomposition for computing Gromov hyperbolicity. Research Report RR-8535, Inria, June 2014.
- [68] X. Huang and Y. Peng. DCen: A dual-ports and cost-effective network architecture for modular datacenters. *Journal of Computational Information Systems*, 10(13):5697–5704, 2014.
- [69] D. Mitsche and P. Prałat. On the hyperbolicity of random graphs. *The Electronic Journal of Combinatorics*, 21(2):2–39, 2014.
- [70] K. Verbeek and S. Suri. Metric embedding, hyperbolic space, and social networks. In *Annual Symposium on Computational Geometry (SCG)*, pages 501–510. ACM, 2014.
- [71] H. Alrasheed and F. F. Dragan. Core-periphery models for graphs based on their δ -hyperbolicity: An example using biological networks. In *Complex Networks VI*, pages 65–77. Springer, 2015.
- [72] M. Borassi, A. Chessa, and G. Caldarelli. Hyperbolicity measures democracy in real-world networks. *Physical Review E*, 92, 2015.
- [73] M. Camelo, L. Fabrega, and P. Vilà. As yet untitled paper. in preparation, 2015.
- [74] N. Cohen, D. Coudert, and A. Lancin. On computing the Gromov hyperbolicity. *ACM Journal on Experimental Algorithmics*, 20(1):1–6, 2015.
- [75] S. Li and G. H. Tucci. Traffic congestion in expanders and (p, δ) -hyperbolic spaces. *Internet Mathematics*, 11(2):134–142, 2015.
- [76] A. Malyshev. Expanders are order diameter non-hyperbolic. Technical Report arXiv:1501.07904, ArXiv, 2015.
- [77] Muad Abu-Ata and F. F. Dragan. Metric tree-like structures in real-life networks: an empirical study. *Networks*, 67(1):49–69, 2016.
- [78] D. Coudert, G. Ducoffe, and N. Nisse. To approximate treewidth, use treelength! *SIAM Journal of Discrete Mathematics*, 2016. to appear.
- [79] N. Lichiardopol. Quasi-centers and radius related to some iterated line digraphs, proofs of several conjectures on de Bruijn and Kautz graphs. *Discrete Applied Mathematics*, 202:106 – 110, 2016.
- [80] J. A. Bondy and U. S. Murty. *Graph theory*. Berlin: Springer, 2008.

- [81] Jean De Rumeur. *Communications dans les réseaux de processeurs*. Masson, 1994.
- [82] R. Diestel. *Graph theory*, volume 173 of *Graduate texts in mathematics*. Springer, 1997.

Papers on tree decompositions

APPENDIX F

Clique-decomposition revisited

Clique-decomposition revisited*

David Coudert^{1,2} and Guillaume Ducoffe^{2,1}

¹Inria, France

²Univ. Nice Sophia Antipolis, CNRS, I3S, UMR 7271, 06900 Sophia Antipolis, France

Abstract

The decomposition of graphs by clique-minimal separators is a common algorithmic tool, first introduced by Tarjan. Since it allows to cut a graph into smaller pieces, it can be applied to preprocess the graphs in the computation of many optimization problems. However, the best known clique-decomposition algorithms have respective $\mathcal{O}(nm)$ -time and $\mathcal{O}(n^{2.69})$ -time complexity, that is prohibitive for large graphs. Here we prove that for every graph G , the decomposition can be computed in $\mathcal{O}(T(G) + \min\{n^{2.3729}, \omega^2 n\})$ -time with $T(G)$ and ω being respectively the time needed to compute a minimal triangulation of G and the clique-number of G . In particular, it implies that every graph can be clique-decomposed in $\tilde{\mathcal{O}}(n^{2.3729})$ -time. Based on prior work from Kratsch et al., we prove in addition that computing the clique-decomposition is at least as hard as triangle detection. Therefore, the existence of any $o(n^{2.3729})$ -time clique-decomposition algorithm would be a significant breakthrough in the field of algorithmic. Finally, our main result implies that planar graphs, bounded-treewidth graphs and bounded-degree graphs can be clique-decomposed in linear or quasi-linear time.

Keywords: clique-decomposition; minimal triangulation; clique-number; treewidth; planar graphs; bounded-degree graphs.

1 Introduction

Our purpose in this work is to study the complexity of separating a graph with all its minimal separators that are cliques. In the literature, such minimal separators are called *clique-minimal separators*, and the decomposition process is called *clique-decomposition*. We refer to [6] for a survey. The clique-decomposition has been introduced by Tarjan in [32], where it is studied for its algorithmic applications. Indeed, it is often the case that hard problems on graphs (theoretically or in practice) can be solved on each subgraph of the clique-decomposition separately. See for instance [4, 9, 14, 15, 18, 19, 26, 34]. In particular, there are NP-hard problems that can be solved on graphs when the subgraphs obtained with the clique-decomposition are “simple enough” w.r.t. the problem. This was first noted by Gavril in [22] for the so-called clique-separable graphs. Other classes of graphs with a “simple” clique-decomposition comprise the chordal graphs (that can be clique-decomposed into complete subgraphs), the EPT graphs [23] and the P_6 -free graphs [12, 13]. Note that general graphs may fail to contain a clique-minimal separator (we will call them *prime graphs* in the following), however in practice the biological networks, the graph of the autonomous systems of the Internet and some other complex networks do contain clique-minimal separators — as supported by some experimentations [1, 15].

With the exception of an $\mathcal{O}(n^{2.69})$ -time algorithm in [28], all the best known algorithms for computing clique-decomposition have an $\mathcal{O}(nm)$ -time complexity [2, 30, 32], that is cubic for dense graphs. Therefore,

*This work is partially supported by ANR project Stint under reference ANR-13-BS02-0007 and ANR program “Investments for the Future” under reference ANR-11-LABX-0031-01.

it becomes too prohibitive to run them on large graphs with thousands of vertices and sometimes billions of edges. Following a recent trend in algorithmic [11], we here investigate on the optimal time for computing the clique-decomposition.

Related work. To the best of our knowledge, the time complexity of clique-decomposition has received little attention in the literature. We are only aware of a recent article [7] introducing a generic framework to compute the clique-decomposition of graphs. This framework applies to all the best-known algorithms to compute the clique-decomposition. Indeed, all these algorithms follow the same three steps:

1. Compute a minimal triangulation of the graph;
2. Find the clique-minimal separators of the graphs (using the minimal triangulation);
3. Finally, recursively disconnect the graph with its clique-minimal separators.

We emphasize that the first step: computing a minimal triangulation, has been extensively studied (see [24] for a survey). So far, the best-known algorithm to compute a minimal triangulation of a graph has an $\tilde{\mathcal{O}}(n^{2.3729})$ -time complexity. Note that it is less than $\mathcal{O}(n^{2.69})$, that has been the best-known complexity for computing the clique-decomposition of a graph — until this note.

Furthermore, new clique-decomposition algorithms are proposed in [7] that are provably faster than the classical approach in some cases, that is, they run in $\mathcal{O}(nm_0)$ -time for some $m_0 < m$. In order to compare these algorithms with our work, let us note that the authors in [7] claim that bounded-treewidth graphs can be clique-decomposed in *quadratic-time*, whereas we will show that it can be done in quasi *linear-time*. Fast (quadratic-time) algorithms to compute the clique-decomposition can also be found in [3, 8] for some specific graph classes, but the latter algorithms deeply rely upon the structural properties of these graphs.

Closest to our work are two papers from Kratsch and Spinrad [27, 28]. In [28], they describe what has been, until this note, the best-known algorithm to compute the clique-decomposition. The latter algorithm has running time $\mathcal{O}(n^{2.69})$, that follows from an algorithm to compute a minimal triangulation of the graph within the same time bounds. We will generalize their result in our work, proving that the clique-decomposition can be computed in $\mathcal{O}(n^{2.3729})$ -time if *any* minimal triangulation of the graph is given¹. Furthermore, lower-bounds on the complexity of computing the clique-decomposition can be deduced from some results in [27]. In particular, they show that finding a clique-minimal separator in a graph is at least as hard as finding a simplicial vertex, *even if a minimal elimination ordering is given as part of the input*. The latter implies that computing a minimal triangulation is not the only complexity bottleneck of clique-decomposition algorithms.

Our contributions. On the negative side, we first prove a lower-bound on the complexity of computing the clique-decomposition. More precisely, we will build upon a result in [27] in order to prove that clique-decomposition is at least as hard as triangle detection (Theorem 4).

We next focus on the two last steps of clique-decomposition algorithms, that is, we ignore the first step of computing a minimal triangulation. Our main result is that the clique-minimal separators of a graph G can be computed in $\mathcal{O}(T(G) + \min\{n^{2.3729}, \omega^2 n\})$ -time, with $T(G)$ and ω being respectively the time needed to compute a minimal triangulation of G and the *clique-number* of G (let us remind that the clique-number of G is the size of a largest clique in G). The latter result follows from two simple algorithms that respectively run in $\mathcal{O}(T(G) + n^{2.3729})$ -time (Proposition 5) and in $\mathcal{O}(T(G) + \omega^2 n)$ -time (Proposition 6). Furthermore, whereas the first algorithm (in $\mathcal{O}(T(G) + n^{2.3729})$ -time) relies upon fast matrix multiplication, the second one is purely combinatorial and can be easily implemented.

We finally notice that any graph G can be clique-decomposed within the same time bound $\mathcal{O}(T(G) + \min\{n^{2.3729}, \omega^2 n\})$ (Theorem 7). Since a minimal triangulation can be computed in $T(G) = \tilde{\mathcal{O}}(n^{2.3729})$ -time for any graph G , our main result implies that any graph can be clique-decomposed in $\tilde{\mathcal{O}}(n^{2.3729})$ -time. Furthermore, faster and practical algorithms can be obtained in some cases — whenever the graphs have bounded clique-number and a minimal triangulation can be computed efficiently. We will show it is the case

¹It seems to us that the techniques in [28] could also be applied to any minimal triangulation. Nonetheless, we will propose a method that is, to our opinion, slightly simpler than theirs.

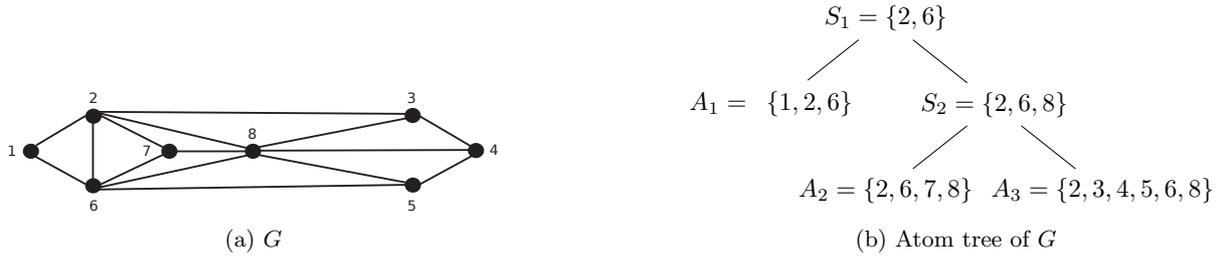


Figure 1: A connected graph G (Figure 1a), an atom tree of the graph (Figure 1b).

for interesting graph classes such as planar graphs, bounded-treewidth graphs and bounded-degree graphs (see Section 5.1 for details).

Altogether, this is hint that our $\tilde{O}(n^{2.3729})$ -time clique-decomposition algorithm is optimal up to polylogarithmic factors — due to the well-know equivalence between triangle detection and matrix multiplication [33].

Definitions and useful notations are given in Section 2. Last, we will conclude this paper with an open conjecture in Section 6.

2 Definitions and preliminaries

We will use standard graph terminology from [10]. Graphs in this study are finite, simple (hence without loops nor multiple edges) connected and unweighted, unless stated otherwise. Given a graph $G = (V, E)$ and a set $S \subseteq V$, we will denote by $G[S]$ the subgraph of G that is induced by S . The open neighbourhood of S , denoted by $N(S)$, is the set of all vertices in $G[V \setminus S]$ that are adjacent to at least one vertex in S . The closed neighbourhood of S is denoted by $N[S] = N(S) \cup S$.

Clique-minimal separators. A set $S \subseteq V$ is a *separator* in G if there are at least two connected components in $G[V \setminus S]$. In particular, a *full component* in $G[V \setminus S]$ is any connected component C in $G[V \setminus S]$ satisfying that $N(C) = S$ (note that a full component might fail to exist). The set S is called a *minimal separator* in G if it is a separator and there are at least two full components in $G[V \setminus S]$. In particular, S is a *clique-minimal separator* if it is a minimal separator and $G[S]$ is a complete subgraph.

Clique-decomposition and atom tree. A graph is *prime* if it does not contain any clique-minimal separator. Examples of prime graphs are the complete graph K_n and the cycle graph C_n . The *clique-decomposition* of a graph G is the family of all inclusionwise maximal subsets A_i such that $G[A_i]$ is prime, and it is unique [30]. The subsets A_i are called the *atoms* of G .

Usually, we represent the clique-decomposition with a binary rooted tree, that is called an *atom tree* and is recursively defined as follows (see Figure 1 for an illustration).

- If G is a prime graph then it has a unique atom tree, that is a single node labeled with V .
- Else, an atom tree of G is any binary rooted tree such that: its root is labeled with a clique-minimal separator S in G , the left child of the root is a leaf-node that is labelled with $A = S \cup C$ where C is a full component of $G[V \setminus S]$ and $G[A]$ is prime, furthermore the subtree that is rooted at the right child of the root is an atom tree of $G[V \setminus C]$.

Informally, an atom tree can be seen as the trace of some execution of a clique-decomposition algorithm (e.g., a decomposition ordering). Note that the atom tree of a graph may not be unique. Furthermore, any atom tree has linear-size (defined as the sum of the label cardinalities) $\mathcal{O}(n + m)$ [5].

Lemma 1 ([30]). *Let $G = (V, E)$ and T be an atom tree of G . Each leaf-node of T is labeled with an atom of G , and each atom of G appears exactly once as a leaf-node label in T .*

Since any atom tree has linear-size, we have by Lemma 1 that $\sum_i |A_i| = \mathcal{O}(n + m)$, where the sets A_i denote the atoms of G . In contrast with the above result, we observe that there may be $\Omega(\omega^2 n)$ edges in the subgraphs that are induced by the atoms of G , with ω being the clique-number of G , that is, the size of a largest complete subgraph in G (e.g., see Figure 2).

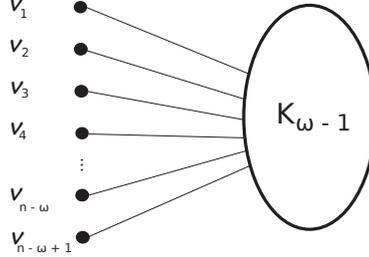


Figure 2: A split graph G with clique-number ω . The atoms of G are the sets $N[v_i]$ for $1 \leq i \leq n - \omega + 1$. Hence, there are $\omega(\omega - 1)(n - \omega + 1)/2$ edges in the subgraphs $G[N[v_i]]$ that are induced by the atoms of G .

Minimal triangulation. A *triangulation* of $G = (V, E)$ is any supergraph $H = (V, E \cup F)$ such that H does not contain any induced cycle of length at least four. In particular, H is a *minimal triangulation* of G if for any strict subset $F' \subset F$, the supergraph $H' = (V, E \cup F')$ is not a triangulation of G .

There exist strong relationships between minimal triangulations and clique-minimal separators. Namely, we will use the following lemma.

Lemma 2 ([6]). *For any minimal triangulation H of a graph G , the clique-minimal separators in G are exactly the minimal separators in H that induce complete subgraphs of G .*

3 Time complexity lower-bound

Let us start proving the hardness of clique-decomposition by reducing this problem from triangle detection. In the following, recall that a simplicial vertex is one whose closed neighbourhood induces a complete subgraph. We will need the following lemma.

Lemma 3 ([27]). *The problem of counting the number of simplicial vertices in a graph with $3n + 2$ vertices is at least as hard as determining whether a graph on n vertices has a triangle.*

Theorem 4. *The problem of computing the clique-decomposition of a graph with $3n + 2$ vertices is at least as hard as determining whether a graph on n vertices has a triangle.*

Proof. Let $G = (V, E)$ be any graph with $3n + 2$ vertices. In order to prove the theorem, by Lemma 3 it is sufficient to prove that counting the number of simplicial vertices in G can be done in $\mathcal{O}(n + m)$ -time if the clique-decomposition of G is given (encoded as an atom tree).

We claim that for every simplicial vertex $v \in V$, its closed neighbourhood $N[v]$ is an atom, and in particular it is the unique atom containing v . Indeed, since $G[N[v]]$ is complete, we have that $G[N[v]]$ is prime, and so, the subset $N[v]$ must be contained in *any* atom A containing v . Furthermore, if it were the case that there exists $u \in A \setminus N[v]$ then the clique $N(v)$ would be a uv -separator, thus contradicting the fact that $G[A]$ is prime. As a result, we have that $A = N[v]$, that proves the claim.

Recall that using an atom tree of G , every atom A_i of G can be written $A_i = C_i \cup S_i$ with S_i being a clique-minimal separator and $S_i \subseteq N(C_i)$. In particular, let $M_i \subseteq C_i$ contain every vertex in the atom that is not contained in any other atom A_j , with $j \neq i$. Note that all the subsets M_i can be computed by visiting the atoms sequentially, which takes $\mathcal{O}(\sum_i |A_i|) = \mathcal{O}(n + m)$ -time. Furthermore, we have by the above claim that in order to count the number of simplicial vertices in G , it is sufficient to sum together the cardinalities

$|M_i|$ of the subsets M_i such that the atom A_i is a clique. Here is a way to achieve the goal in linear-time. Since the subsets C_i are pairwise disjoint, let us reorder the vertices in G so that in any adjacency list, it first appears the neighbours in C_1 , then those in C_2 , and so on. In such case, the atom A_i is a clique if and only if each vertex in C_i has $|A_i| - 1$ neighbours in A_i , that is, $|A_i| - 1$ neighbours that are not contained in any C_j , with $j < i$. The latter can be verified by visiting the subsets C_i sequentially, while removing the vertices in C_i from all adjacency lists at the i^{th} step. Since the adjacency lists have been reordered, it can be done in $\mathcal{O}(m + \sum_i |C_i|) = \mathcal{O}(n + m)$ -time. So, overall, finding the atoms A_i that are cliques can be done in $\mathcal{O}(n + m)$ -time, which implies that counting the number of simplicial vertices in G can be done within the same time complexity. \square

4 Computing the clique-minimal separators

This section is devoted to fast computation of the clique-minimal separators in a graph. We will introduce two methods which both make use of Lemma 2.

Proposition 5. *Let $G = (V, E)$. Suppose that a minimal triangulation of G can be computed in time $T(G)$. Then, the clique-minimal separators of G can be computed in $\mathcal{O}(T(G) + n^{2.3729})$ -time.*

Proof. Let $H = (V, E \cup F)$ be a minimal triangulation of G , with $f = |F|$ fill edges. By the hypothesis it can be computed in time $T(G)$. Let $\Xi = (S_1, S_2, \dots, S_l)$ be the minimal separators of H , with $l \leq n$. By [21], the family Ξ can be computed in $\mathcal{O}(n + m + f) = \mathcal{O}(n^2)$ -time. Furthermore, recall that by Lemma 2 the clique-minimal separators of G are exactly the separators in Ξ that are cliques of G . In order to compute them, let $V = (v_1, v_2, \dots, v_n)$ be totally ordered. Let \mathcal{A}_G be the adjacency matrix of G , and let \mathcal{B}_H be the *clique matrix* of H (of dimensions $n \times l$) defined as follows. For every $1 \leq i \leq n$ and for every $1 \leq j \leq l$, we have $b_{ij} = 1$ if $v_i \in S_j$ and $b_{ij} = 0$ otherwise. Then, $\mathcal{C} = \mathcal{A}_G \mathcal{B}_H$ is a matrix of dimensions $n \times l$. It can be computed in $\mathcal{O}(n^{2.3729})$ -time by using fast matrix multiplication since $l \leq n$ [29]. Furthermore, for every $1 \leq i \leq n$ and for every $1 \leq j \leq l$, we have $c_{ij} = |N_G(v_i) \cap S_j|$. Therefore, $S_j \in \Xi$ is a clique-minimal separator of G if and only if we have $c_{ij} = |S_j| - 1$ for every $v_i \in S_j$. As a result, the clique-minimal separators of G are obtained from the matrix \mathcal{C} in time $\mathcal{O}(\sum_{j=1}^l |S_j|)$, that is $\mathcal{O}(n + m + f) = \mathcal{O}(n^2)$. \square

Proposition 6. *Let $G = (V, E)$. Suppose that a minimal triangulation of G can be computed in time $T(G)$. Then, the clique-minimal separators of G can be computed in $\mathcal{O}(T(G) + \omega^2 n)$ -time.*

Proof. Let $H = (V, E \cup F)$ be a minimal triangulation of G , with $f = |F|$ fill edges. By the hypothesis it can be computed in time $T(G)$. Let us compute the set Ξ of all minimal separators of H . By [21], the family Ξ can be computed in $\mathcal{O}(n + m + f) = \mathcal{O}(T(G))$ -time.

Let $\mathcal{S} = \Xi$. Our aim is to remove separators of H from \mathcal{S} until it only contains the clique-minimal separators of G . In order to achieve the result, let $V = (v_1, v_2, \dots, v_n)$ be totally ordered. We consider the vertices sequentially. For every $1 \leq i \leq n$, let $\mathcal{S}_i \subseteq \mathcal{S}$ contain every $S \in \mathcal{S}$ such that $v_i \in S$. Furthermore, let $S_{<i} := S \cap \{v_1, \dots, v_{i-1}\}$ for every $S \in \mathcal{S}_i$. If $S_{<i} \not\subseteq N_G(v_i)$ then S is not a clique and it is discarded from \mathcal{S} . Therefore, once the algorithm has terminated, subsets in \mathcal{S} are exactly the minimal separators of H that are cliques of G . By Lemma 2, these are exactly the clique-minimal separators of G . Hence the above algorithm is correct.

Let us focus on the time complexity. Assume for ease of computation that we maintain an “incidence graph” $\mathcal{I}_{\mathcal{S}}$: with vertex set $V \cup \mathcal{S}$ and an edge between every vertex $v_i \in V$ and every separator $S \in \mathcal{S}_i$. Note that $\mathcal{I}_{\mathcal{S}}$ can be constructed at the initialization step (when $\Xi = \mathcal{S}$) in $\mathcal{O}(|V| + \sum_{S \in \Xi} |S|) = \mathcal{O}(n + m + f)$ -time, that is $\mathcal{O}(T(G))$. Furthermore, for every $1 \leq i \leq n$ the separators in \mathcal{S}_i are exactly the neighbours of vertex v_i in $\mathcal{I}_{\mathcal{S}}$, hence it takes $\mathcal{O}(|\mathcal{S}_i|)$ -time to access to each of the separators in \mathcal{S}_i . Discarding a separator $S \in \mathcal{S}_i$ from \mathcal{S} is equivalent to deleting the vertex corresponding to S in $\mathcal{I}_{\mathcal{S}}$, which can be done in $\mathcal{O}(|S|)$ -time. Overall, these two types of operations (accessing and discarding) take $\mathcal{O}(\sum_{i=1}^n |\mathcal{S}_i| + \sum_{S \in \Xi} |S|)$ -time, that is $\mathcal{O}(\sum_{S \in \Xi} |S|) = \mathcal{O}(T(G))$ -time.

Finally, deciding whether $S_{<i} \not\subseteq N_G(v_i)$ for every $S \in \mathcal{S}_i$ takes time $\mathcal{O}(|N_G(v_i)| + \sum_{S \in \mathcal{S}_i} |S_{<i}|)$. Furthermore, since the vertices are considered sequentially, we have that $S_{<i}$ is a clique for every $S \in \mathcal{S}_i$ (or else, S would have been discarded from \mathcal{S} at some step $j < i$ of the algorithm). This implies that $\sum_{i|S \in \mathcal{S}_i} |S_{<i}| \leq \sum_{j=1}^{\omega} j = \omega(\omega + 1)/2 = \mathcal{O}(\omega^2)$ for every $S \in \Xi$. Hence, since H is triangulated, and so, $|\Xi| \leq n$, we have:

$$\sum_{i=1}^n \sum_{S \in \mathcal{S}_i} |S_{<i}| = \sum_{S \in \Xi} \sum_{i|S \in \mathcal{S}_i} |S_{<i}| = \mathcal{O}(\omega^2 n).$$

□

5 Faster computation of clique-decomposition

In Section 4, we proved that if a minimal triangulation of a graph G can be computed in time $T(G)$, then the clique-minimal separators of G can be computed in $\mathcal{O}(T(G) + \min\{n^{2.3729}, \omega^2 n\})$ -time. We now prove that an atom tree of G can be computed within the same time bounds.

Theorem 7. *Let $G = (V, E)$. Suppose that a minimal triangulation of G can be computed in time $T(G)$. Then, the clique-decomposition of G can be computed in $\mathcal{O}(T(G) + \min\{n^{2.3729}, \omega^2 n\})$ -time.*

Proof. Let $H = (V, E \cup F)$ be a minimal triangulation of G , with $f = |F|$ fill edges. By the hypothesis it can be computed in time $T(G)$. Furthermore, the clique-decomposition of G can be computed in $\mathcal{O}(n + m + f) = \mathcal{O}(T(G))$ -time if H and the clique-minimal separators of G are given [7]. By Propositions 5 and 6, the clique-minimal separators of G can be computed in $\mathcal{O}(T(G) + \min\{n^{2.3729}, \omega^2 n\})$ -time. So, overall it takes $\mathcal{O}(T(G) + \min\{n^{2.3729}, \omega^2 n\})$ -time to compute the clique-decomposition of G . □

On the combinatorial side, our approach for computing the clique-decomposition (Theorem 7) is at least as good as the state-of-the-art $\mathcal{O}(nm)$ -time algorithm. Indeed, for any graph G , a minimal triangulation of G can be computed in time $T(G) = \mathcal{O}(nm)$ [31]. Furthermore if G has clique-number ω then it has number of edges $m \geq \omega(\omega - 1)/2 = \Omega(\omega^2)$.

Corollary 8. *The clique-decomposition of a graph G can be computed in $\tilde{\mathcal{O}}(n^{2.3729})$ -time.*

Proof. Since a minimal triangulation of a graph G can be computed in $\tilde{\mathcal{O}}(n^{2.3729})$ -time [25], the result follows from Theorem 7 by replacing $T(G)$ with $\tilde{\mathcal{O}}(n^{2.3729})$. □

5.1 Applications

By Theorem 7, the clique-decomposition of a graph G can be computed in quasi linear-time if i) G has bounded clique-number and ii) a minimal triangulation of G can be computed efficiently. Below, we list a few graph classes for which it is the case.

- A graph G has *tree-width* at most k if there exists a triangulation of G with clique-number at most k . Note that the clique-number ω of G is a lower-bound on its tree-width. Furthermore, if G has tree-width k then a minimal triangulation of G can be computed in $\mathcal{O}(k^7 \cdot n \log n)$ -time [20]. Therefore, by Theorem 7 the clique-decomposition of bounded tree-width graphs can be computed in $\mathcal{O}(n \log n)$ -time.
- A graph G is *planar* if it can be drawn in the Euclidean plane so that edges may only intersect at their endpoints. By Kuratowski Theorem, G is planar if and only if G is $\{K_{3,3}, K_5\}$ -minor-free. So, a planar graph G has bounded clique-number $\omega \leq 4$. Furthermore, if G is planar then a minimal triangulation of G can be computed in $\mathcal{O}(n)$ -time [16]. As a result, by Theorem 7 the clique-decomposition of planar graphs can be computed in linear-time.

- Finally, let us consider bounded-degree graphs. Indeed, for every graph G , $\omega \leq \Delta + 1$ with ω and Δ being respectively the clique-number and the maximum degree of G . Therefore, bounded-degree graphs have bounded clique-number. Furthermore, if G has maximum degree Δ then a minimal triangulation of G can be computed in $\mathcal{O}(n \cdot (\Delta^3 + \alpha(n)))$ -time where $\alpha(n)$ here denotes the inverse of Ackerman's function [17]. Hence by Theorem 7 the clique-decomposition of bounded-degree graphs can be computed in $\mathcal{O}(n \cdot \alpha(n))$ -time.

6 Conclusion

By Corollary 8, the time complexity of computing the clique-decomposition of an n -vertex graph G is $\tilde{\mathcal{O}}(n^{2.3759})$. It is unlikely that the problem can be solved in $o(n^{2.3759})$ -time by Theorem 4 (recall that the two problems of triangle detection and matrix multiplication are equivalent [33]).

Finally, we proved in Theorem 7 that for every graph G with bounded clique-number ω , the clique-decomposition of G can be computed in $\mathcal{O}(T(G) + \omega^2 n)$ -time where $T(G)$ here denotes the time needed to compute a minimal triangulation of G . We conjecture that in fact, it can be computed in $\mathcal{O}(\omega^2 n)$ -time.

References

- [1] M. Abu-Ata and F. Dragan. Metric tree-like structures in real-world networks: an empirical study. *Networks*, 67(1):49–68, 2016.
- [2] A. Berry and J.-P. Bordat. Decomposition by clique minimal separators. Technical Report 97213, 1999.
- [3] A. Berry, A. Brandstädt, V. Giakoumakis, and M. F. Efficiently decomposing, recognizing and triangulating hole-free graphs without diamonds. *Discrete Applied Mathematics*, 184:50 – 61, 2015.
- [4] A. Berry, R. Pogorelnik, and A. Sigayret. Vertical decomposition of a lattice using clique separators. In *Proceedings of The Eighth International Conference on Concept Lattices and Their Applications, Nancy, France, October 17-20, 2011*, pages 15–29, 2011.
- [5] A. Berry, R. Pogorelnik, and G. Simonet. Efficient clique decomposition of a graph into its atom graph. Technical Report RR-10-07, Mar. 2010.
- [6] A. Berry, R. Pogorelnik, and G. Simonet. An introduction to clique minimal separator decomposition. *Algorithms*, 3(2):197, 2010.
- [7] A. Berry, R. Pogorelnik, and G. Simonet. Organizing the atoms of the clique separator decomposition into an atom tree. *Discrete Applied Mathematics*, 177:1 – 13, 2014.
- [8] A. Berry and A. Wagler. Triangulation and clique separator decomposition of claw-free graphs. In M. C. Golumbic, M. Stern, A. Levy, and G. Morgenstern, editors, *Graph-Theoretic Concepts in Computer Science: 38th International Workshop, WG 2012, Jerusalem, Israel, June 26-28, 2012, Revised Selected Papers*, pages 7–21. Springer Berlin Heidelberg, 2012.
- [9] H. Bodlaender and A. Koster. Safe separators for treewidth. *Discrete Mathematics*, 306(3):337 – 350, 2006.
- [10] J. Bondy and U. Murty. *Graph theory*. Berlin: Springer, 2008.
- [11] M. Borassi, P. Crescenzi, and M. Habib. Into the square - on the complexity of quadratic-time solvable problems. *arXiv preprint arXiv:1407.4972*, 2014.
- [12] A. Brandstädt and C. Hoàng. On clique separators, nearly chordal graphs, and the maximum weight stable set problem. In *Integer Programming and Combinatorial Optimization*, pages 265–275. Springer, 2005.
- [13] A. Brandstädt and R. Mosca. Weighted efficient domination for P_6 -free graphs in polynomial time. *arXiv preprint arXiv:1508.07733*, 2015.

- [14] M. Changat and J. Mathew. On triangle path convexity in graphs. *Discrete Mathematics*, 206(1):91–95, 1999.
- [15] N. Cohen, D. Coudert, G. Ducoffe, and A. Lancin. Applying clique-decomposition for computing Gromov hyperbolicity. Research Report RR-8535, Inria, June 2014.
- [16] E. Dahlhaus. Minimal elimination of planar graphs. In *Algorithm Theory - SWAT '98, 6th Scandinavian Workshop on Algorithm Theory, Stockholm, Sweden, July, 8-10, 1998, Proceedings*, pages 210–221, 1998.
- [17] E. Dahlhaus. Minimal elimination ordering for graphs of bounded degree. *Discrete Applied Mathematics*, 116(1-2):127–143, 2002.
- [18] M. Didi Biha, B. Kaba, M.-J. Meurs, and E. SanJuan. Graph decomposition approaches for terminology graphs. In *MICAI 2007: Advances in Artificial Intelligence*, volume 4827 of *Lecture Notes in Computer Science*, pages 883–893. Springer, 2007.
- [19] Y. Dourisboure and C. Gavaille. Tree-decompositions with bags of small diameter. *Discrete Mathematics*, 307(16):2008–2029, 2007.
- [20] F. Fomin, D. Lokshtanov, M. Pilipczuk, S. Saurabh, and W. Wrochna. Fully polynomial-time parameterized computations for graphs and matrices of low treewidth. Technical Report abs/1511.01379, arXiv, 2015.
- [21] F. Gavril. Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph. *SIAM Journal on Computing*, 1(2):180–187, 1972.
- [22] F. Gavril. Algorithms on clique separable graphs. *Discrete Mathematics*, 19(2):159–165, 1977.
- [23] M. Golumbic and R. Jamison. The edge intersection graphs of paths in a tree. *Journal of Combinatorial Theory, Series B*, 38(1):8–22, 1985.
- [24] P. Heggernes. Minimal triangulations of graphs: A survey. *Discrete Mathematics*, 306(3):297–317, 2006.
- [25] P. Heggernes, J. Telle, and Y. Villanger. Computing minimal triangulations in time $O(n^\alpha \log n) = o(n^{2.376})$. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 907–916. Society for Industrial and Applied Mathematics, 2005.
- [26] B. Kaba, N. Pinet, G. Lelandais, A. Sigayret, and A. Berry. Clustering gene expression data using graph separators. *In silico biology*, 7(4-5):433–452, 2007.
- [27] D. Kratsch and J. Spinrad. Between $O(nm)$ and $O(n^\alpha)$. *SIAM Journal on Computing*, 36(2):310–325, 2006.
- [28] D. Kratsch and J. Spinrad. Minimal fill in $O(n^{2.69})$ time. *Discrete mathematics*, 306(3):366–371, 2006.
- [29] F. Le Gall. Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation, ISSAC '14*, pages 296–303, New York, NY, USA, 2014. ACM.
- [30] H.-G. Leimer. Optimal decomposition by clique separators. *Discrete Mathematics*, 113(1):99–123, 1993.
- [31] D. Rose, R. Tarjan, and G. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM Journal on computing*, 5(2):266–283, 1976.
- [32] R. E. Tarjan. Decomposition by clique separators. *Discrete Mathematics*, 55(2):221 – 232, 1985.
- [33] V. Vassilevska Williams and R. Williams. Subcubic equivalences between path, matrix and triangle problems. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 645–654. IEEE, 2010.
- [34] H. Yaghi and H. Krim. Probabilistic graph matching by canonical decomposition. In *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*, pages 2368–2371. IEEE, 2008.

On the complexity of computing
tree decompositions with metric
constraints on the bags



On computing tree and path decompositions with metric constraints on the bags*

Guillaume Ducoffe^{† ‡}, Sylvain Legay[§], Nicolas Nisse^{‡ †}

Project-Teams COATI

Research Report n° 8842 — January 2016 — 63 pages

Abstract: We here investigate on the complexity of computing the *tree-length* and the *tree-breadth* of any graph G , that are respectively the best possible upper-bounds on the diameter and the radius of the bags in a tree decomposition of G . *Path-length* and *path-breadth* are similarly defined and studied for path decompositions. So far, it was already known that tree-length is NP-hard to compute. We here prove it is also the case for tree-breadth, path-length and path-breadth. Furthermore, we provide a more detailed analysis on the complexity of computing the tree-breadth. In particular, we show that graphs with tree-breadth one are in some sense the hardest instances for the problem of computing the tree-breadth. We give new properties of graphs with tree-breadth one. Then we use these properties in order to recognize in polynomial-time all graphs with tree-breadth one that are planar or bipartite graphs. On the way, we relate tree-breadth with the notion of *k-good* tree decompositions (for $k = 1$), that have been introduced in former work for routing. As a byproduct of the above relation, we prove that deciding on the existence of a *k-good* tree decomposition is NP-complete (even if $k = 1$). All this answers open questions from the literature.

Key-words: Tree-length; Tree-breadth; Path-length; Path-breadth; *k-good* tree decompositions;

* This work has been partially supported by ANR project Stint under reference ANR-13-BS02-0007, ANR program “Investments for the Future” under reference ANR-11-LABX-0031-01.

[†] Univ. Nice Sophia Antipolis, CNRS, I3S, UMR 7271, 06900 Sophia Antipolis, France

[‡] Inria, France

[§] LRI, Univ Paris Sud, Université Paris-Saclay, 91405 Orsay, France

RESEARCH CENTRE
SOPHIA ANTIPOLIS – MÉDITERRANÉE

2004 route des Lucioles - BP 93
06902 Sophia Antipolis Cedex

Sur la complexité du calcul de décompositions arborescentes ou linéaires avec des contraintes sur les distances dans les sacs

Résumé : Nous étudions la complexité du calcul de la *tree-length* et de la *tree-breadth* pour tout graphe G , qui sont respectivement les meilleures bornes supérieures possibles sur le diamètre et le rayon des sacs dans une décomposition arborescente de G . La *path-length* et la *path-breadth* sont définies (et étudiées) de manière identique pour les décompositions linéaires. Il était connu jusqu'à présent que le calcul de la *tree-length* est NP-difficile. Nous prouvons dans ce rapport qu'il en va de même pour les paramètres *tree-breadth*, *path-length* et *path-breadth*. Par ailleurs, nous analysons plus en détails la complexité du calcul de la *tree-breadth*. En particulier, nous montrons que les graphes de *tree-breadth* 1 sont, en un sens que nous préciserons par la suite, les instances les plus difficiles pour le problème du calcul de la *tree-breadth*. Nous démontrons de nouvelles propriétés des graphes de *tree-breadth* 1. Puis nous utilisons ces propriétés afin de reconnaître en temps polynomial si un graphe planaire ou biparti a sa *tree-breadth* égale à 1. En chemin dans la preuve des résultats ci-dessus, nous établissons une relation entre la *tree-breadth* d'un graphe et l'existence d'une décomposition arborescente "*k-good*" (avec $k = 1$), un problème qui avait été introduit dans la littérature sur le routage compact. Comme conséquence directe de cette relation, nous prouvons que décider de l'existence d'une décomposition arborescente "*k-good*" est un problème NP-complet (même si $k = 1$). Nous répondons ainsi à des questions ouvertes de la littérature.

Mots-clés : Tree-length; Tree-breadth; Path-length; Path-breadth; Décomposition *k-good*;

1 Introduction

Context. It is a fundamental problem in metric graph theory [5] to embed a graph into a simpler metric space while minimizing the (multiplicative) distortion of the distances in the graph. In particular, minimum distortion embeddings of a graph into a tree or a path have practical applications in computer vision [45], computational chemistry and biology [35] as well as in network design and distributed protocols [33]. The two problems to embed a graph into a tree or a path with minimum distortion are NP-hard [2, 6, 40]. However, there exists a nice setting in order to approximate these two problems. More precisely, a series of graph parameters has been introduced in recent work in order to measure how much the *distance distribution* of a graph is close to a tree metric or a path metric [24, 26, 27]. We refer to [25, 27] for details about the relationships between these parameters and the two above-mentioned embedding problems. Here we study the complexity of computing these parameters, thereby solving open problems in the literature.

The parameters that are considered in this note can be defined using the terminology of Robertson and Seymour tree decompositions [43]. Informally, a tree decomposition is a dividing of a graph G into “bags”: that are overlapping subgraphs that can be pieced together in a tree-like manner (formal definitions will be given in the technical sections of the paper). The shortest-path metric of G is “tree-like” when each bag of the tree decomposition has bounded diameter and bounded radius, where the distance taken between two vertices in a same bag is their distance in G . The *tree-length* [24] and the *tree-breadth* [26] of G are respectively the best possible upper-bounds on the diameter and the radius of the bags in a tree decomposition of G . *Path-length* [46] and *path-breadth* [27] are defined in the same fashion as tree-length and tree-breadth for path decompositions. In this paper, we focus on the complexity of computing the four parameters tree-length, tree-breadth, path-length and path-breadth.

Recent studies suggest that some classes of real-life networks – including biological networks and social networks – have bounded tree-length and tree-breadth [1]. This metric tree-likeness can be exploited in algorithms. For instance, bounded tree-length graphs admit compact distance labeling scheme [23] as well as a PTAS for the well-known Traveling Salesman problem [38]. Furthermore, the diameter and the radius of bounded tree-length graphs can be approximated up to an additive constant in linear-time [17]. In contrast to the above result, we emphasize that under classical complexity assumptions the diameter of general graphs *cannot* be approximated up to an additive constant in subquadratic-time, that is prohibitive for very large graphs [15].

Note that a large amount of the literature about tree decompositions rather seeks to minimize the *size* of the bags than their diameter. The *tree-width* [43] of a graph G is the best possible upper-bound on the size of the bags in a tree decomposition of G . However, tree-length and the other parameters that are considered in this paper can differ arbitrarily from tree-width; we refer to [20] for a global picture on the relations between tree-length and tree-width. Furthermore, one aim of this paper is to complete the comparison between tree-width and path-width on one side, and tree-length, tree-breadth, path-length and path-breadth on the other side, from the complexity point of view. Let us remind that computing the tree-width (resp. the path-width) is NP-hard [3, 36], however for every fixed $k \geq 1$ there is a linear-time algorithm to decide whether a graph has tree-width at most k (resp., path-width at most k) [8, 12].

Related work. The complexity of computing tree-length, tree-breadth, path-length and path-breadth has been left open in several works [24, 26, 27]. So far, it has been solved only for tree-length, that is NP-hard to compute.

Tree-length and tree-breadth. It is NP-complete to decide whether a graph has tree-length at most k for every constant $k \geq 2$ [39]. However, the reduction used for tree-length goes through

weighted graphs and then goes back to unweighted graphs using rather elegant gadgets. It does not seem to us these gadgets can be easily generalized in order to apply to the other parameters that are considered in this note. On a more positive side, there exists a 3-approximation algorithm for tree-length [24]. In this aspect, it looks natural to investigate on the complexity of computing the tree-breadth, since any polynomial-time algorithm would imply an improved 2-approximation algorithm for tree-length.

Path-length and path-breadth. There exist constant-factor approximation algorithms for path-length and path-breadth [27]. Recently, the minimum eccentricity shortest-path problem – that is close to the computation of path-length and path-breadth – has been proved NP-hard [28]. Let us point out that for every fixed k , it can be decided in polynomial-time whether a graph admits a shortest-path with eccentricity at most k [28]. Our results will show the situation is different for path-length and path-breadth than for the minimum eccentricity shortest-path.

Our contributions. On the negative side, we prove that tree-breadth, path-length and path-breadth are NP-hard to compute. More precisely:

- recognizing graphs with tree-breadth one is NP-complete;
- recognizing graphs with path-length two is NP-complete;
- recognizing graphs with path-breadth one is NP-complete.

It is remarkable the last two results (for path-length and path-breadth) are obtained using *the same reduction*. Our reductions have distant similarities with the reduction that was used for tree-length. However, they do not need any detour through weighted graphs.

We next focus our work on tree-breadth (although part of the results may extend to the three other parameters that are considered in this note). We give a more in-depth analysis on the complexity of computing this parameter. In particular, we prove it is equally hard to compute tree-breadth as to recognize graphs with tree-breadth one. Therefore, graphs with tree-breadth one are in some sense the hardest instances for the problem of computing the tree-breadth. The latter partly answers an open question from [26], where it was asked for a characterization of graphs with tree-breadth one. We also prove a few properties of graphs with tree-breadth one. In particular, graphs with tree-breadth one are exactly those graphs admitting a *1-good tree decomposition*, that is a tree decomposition whose each bag has a spanning star. The more general notion of k -good tree decompositions was introduced in [37] to obtain new compact routing schemes. Note that as a byproduct of the above relation between 1-good tree decompositions and graphs with tree-breadth one, we obtain that deciding on the existence of a k -good tree decomposition is an NP-complete problem even when $k = 1$.

Finally, on the algorithmic side, we show how to recognize in polynomial time all graphs of tree-breadth one that are planar or bipartite. In particular, our recognition algorithm for planar graphs of tree-breadth one relies upon deep structural properties of these graphs.

Definitions and useful notations are given in Section 2. All our results of NP-completeness are listed and proved in Section 3. Sections 4 and 5 are devoted to the computation of tree-breadth. In particular, in Section 5 we present and we prove correctness of an algorithm to recognize planar graphs with tree-breadth one. Finally, we discuss about some open questions in the conclusion (Section 6).

2 Definitions and preliminary results

We refer to [22] for a survey on graph theory. Graphs in this study are finite, simple, connected and unweighted. Let $G = (V, E)$ be a graph. For any $X \subseteq V$, let $G[X]$ denote the subgraph of G induced by X . For any subgraph H of G , let $N_H(v)$ denote the set of neighbors of $v \in V$ in H , and let $N_H[v] = N_H(v) \cup \{v\}$. The distance $dist_H(u, v)$ between two vertices $u, v \in V$ in H is the minimum length (number of edges) of a path between u and v in a subgraph H of G . In what follows, we will omit the subscript when no ambiguity occurs. A set $S \subseteq V$ is a dominating set of G if any vertex of $V \setminus S$ has a neighbor in S . The *dominating number* $\gamma(G)$ of a graph G is the minimum size of a dominating set of G .

Tree decompositions and path decompositions of a graph. A *tree decomposition* (T, \mathcal{X}) of G is a pair consisting of a tree T and of a family $\mathcal{X} = (X_t)_{t \in V(T)}$ of subsets of V indexed by the nodes of T and satisfying:

- $\bigcup_{t \in V(T)} X_t = V$;
- for any edge $e = \{u, v\} \in E$, there exists $t \in V(T)$ such that $u, v \in X_t$;
- for any $v \in V$, the set of nodes $t \in V(T)$ such that $v \in X_t$ induces a subtree, denoted by T_v , of T .

The sets X_t are called *the bags* of the decomposition. If no bag is contained into another one, then the tree decomposition is called *reduced*. Starting from any tree decomposition, a reduced tree decomposition can be obtained in polynomial-time by contracting any two adjacent bags with one contained into the other until it is no more possible to do that.

In the following we will make use of the *Helly property* in our proofs:

Lemma 1 [*4, Helly property*] *Let T be a tree and let T_1, T_2, \dots, T_k be a finite family of pairwise intersecting subtrees. Then, $\bigcap_{i=1}^k T_i \neq \emptyset$, or equivalently there is a node contained in all the k subtrees.*

Finally, let (T, \mathcal{X}) be a tree decomposition, it is called a *path decomposition* if T induces a path.

Metric tree-likeness and path-likeness. All graph invariants that we consider in the paper can be defined in terms of tree decompositions and path decompositions. Let (T, \mathcal{X}) be any tree decomposition of a graph G . For any $t \in V(T)$,

- the *diameter* of bag X_t equals $\max_{v, w \in X_t} dist_G(v, w)$;
- the *radius* $\rho(t)$ of a bag X_t equals $\min_{v \in V} \max_{w \in X_t} dist_G(v, w)$.

The *length* of (T, \mathcal{X}) is the maximum diameter of its bags, while the *breadth* of (T, \mathcal{X}) is the maximum radius of its bags. The *tree-length* and the *tree-breadth* of G , respectively denoted by $tl(G)$ and $tb(G)$, are the minimum length and breadth of its tree decomposition, respectively.

Let k be a positive integer, the tree decomposition (T, \mathcal{X}) is called *k-good* when each bag contains a dominating induced path of length at most $k - 1$. It is proved in [37] every graph G has a *k-good* tree decomposition for $k = ch(G) - 1$, with $ch(G)$ denoting the size of a longest induced cycle of G . Finally, path-length, path-breadth and *k-good* path decompositions are

similarly defined and studied for the path decompositions as tree-length, tree-breadth and k -good tree decompositions are defined and studied for the tree decompositions. The path-length and path-breadth of G are respectively denoted by $pl(G)$ and $pb(G)$.

It has been observed in [26, 27] that the four parameters tree-length, tree-breadth and path-length, path-breadth are contraction-closed invariants. We will use the latter property in our proofs.

Lemma 2 ([26, 27]) *For every $G = (V, E)$ and for any edge $e \in E$:*

$$tl(G/e) \leq tl(G), \quad tb(G/e) \leq tb(G) \quad \text{and} \quad pl(G/e) \leq pl(G), \quad pb(G/e) \leq pb(G).$$

Furthermore, it can be observed that for any graph G , $tb(G) \leq tl(G) \leq 2 \cdot tb(G)$ and similarly $pb(G) \leq pl(G) \leq 2 \cdot pb(G)$. Moreover, if a graph G admits a k -good tree decomposition, then $tb(G) \leq \lfloor k/2 \rfloor + 1$ and $tl(G) \leq k + 1$. Before we end this section, let us prove the stronger equivalence, $tb(G) = 1$ if and only if G admits a 1-good tree decomposition. This result will be of importance in the following. Since a tree decomposition is 1-good if and only if each bag contains a spanning star, we will name the 1-good tree decompositions *star-decompositions* in the following.

Definition 1 *Let $G = (V, E)$ be a connected graph, a star-decomposition is a tree decomposition (T, \mathcal{X}) of G whose each bag induces a subgraph of dominating number one, i.e., for any $t \in V(T)$, $\gamma(G[X_t]) = 1$.*

Clearly, if a graph admits a star-decomposition, then it has tree-breadth at most one. Let us prove that the converse also holds.

Lemma 3 *For any graph G with $tb(G) \leq 1$, every reduced tree decomposition of G of breadth one is a star-decomposition. In particular:*

- *any tree decomposition of G of breadth one can be transformed into a star-decomposition in polynomial-time;*
- *similarly, any path decomposition of G of breadth one can be transformed into a 1-good path decomposition in polynomial-time.*

Proof. Let (T, \mathcal{X}) be any reduced tree decomposition of G of breadth one. We will prove it is a star-decomposition. To prove it, let $X_t \in \mathcal{X}$ be arbitrary and let $v \in V$ be such that $\max_{w \in X_t} \text{dist}_G(v, w) = 1$, which exists because X_t has radius one. We now show that $v \in X_t$. Indeed, since the subtree T_v and the subtrees $T_w, w \in X_t$, pairwise intersect, then it comes by the Helly Property (Lemma 1) that $T_v \cap (\bigcap_{w \in X_t} T_w) \neq \emptyset$ i.e., there is some bag containing $\{v\} \cup X_t$. As a result, we have that $v \in X_t$ because (T, \mathcal{X}) is a reduced tree decomposition, hence $\gamma(G[X_t]) = 1$. The latter implies that (T, \mathcal{X}) is a star-decomposition because X_t is arbitrary.

Now let (T, \mathcal{X}) be any tree decomposition of G of breadth one. It can be transformed in polynomial-time into a reduced tree decomposition (T', \mathcal{X}') so that $\mathcal{X}' \subseteq \mathcal{X}$. Furthermore, (T', \mathcal{X}') has breadth one because it is the case for (T, \mathcal{X}) , therefore (T', \mathcal{X}') is a star-decomposition. In particular, if (T, \mathcal{X}) is a path decomposition then so is (T', \mathcal{X}') . ■

Corollary 1 *For any graph G , $tb(G) \leq 1$ if and only if G admits a star-decomposition.*

3 Intractability results

3.1 Path-length and path-breadth

This section is devoted to the complexity of all path-like invariants that we consider in this paper.

Theorem 1 *Deciding whether a graph has path-length at most k is NP-complete even if $k = 2$.*

In contrast to Theorem 1, graphs with path-length one are exactly the interval graphs [27], *i.e.*, they can be recognized in linear-time.

Theorem 2 *Deciding whether a graph has path-breadth at most k is NP-complete even if $k = 1$.*

From the complexity result of Theorem 2, we will also prove the hardness of deciding on the existence of k -good path decompositions.

Theorem 3 *Deciding whether a graph admits a k -good path decomposition is NP-complete even if $k = 1$.*

Proof. The problem is in NP. By Lemma 3, a graph G admits a 1-good path decomposition if and only if $pb(G) \leq 1$, therefore it is NP-hard to decide whether a graph admits a 1-good path decomposition by Theorem 2. ■

All of the NP-hardness proofs in this section will rely upon the same reduction from the Betweenness problem, defined below. The Betweenness problem, sometimes called the Total Ordering problem, is NP-complete [41]. In [31], it was used to show that the Interval Sandwich problem is NP-complete. What we here prove is that the Interval Sandwich problem remains NP-complete even if the second graph is a *power* of the first one, where the k^{th} power G^k of any graph G is obtained from G by adding an edge between every two distinct vertices that are at distance at most k in G for every integer $k \geq 1$. Indeed, a graph G has path-length at most k if and only if there is an Interval Sandwich between G and G^k (we refer to [39] for the proof of a similar equivalence between tree-length and the Chordal Sandwich problem).

Problem 1 (Betweenness)

Input: a set \mathcal{S} of n elements, a set \mathcal{T} of m ordered triples of elements in \mathcal{S} .

Question: Is there a total ordering of \mathcal{S} such that for every triple $t = (s_i, s_j, s_k) \in \mathcal{T}$, either $s_i < s_j < s_k$ or $s_k < s_j < s_i$?

Now, given an instance $(\mathcal{S}, \mathcal{T})$ of the Betweenness problem, we will construct from \mathcal{S} and \mathcal{T} a graph $G_{\mathcal{S}, \mathcal{T}}$ as defined below. We will then prove that $pl(G_{\mathcal{S}, \mathcal{T}}) \leq 2$ (resp. $pb(G_{\mathcal{S}, \mathcal{T}}) \leq 1$) if and only if $(\mathcal{S}, \mathcal{T})$ is a yes-instance of the Betweenness problem.

Definition 2 *Let \mathcal{S} be a set of n elements, let \mathcal{T} be a set of m ordered triples of elements in \mathcal{S} . The graph $G_{\mathcal{S}, \mathcal{T}}$ is constructed as follows:*

- For every element $s_i \in \mathcal{S}$, $1 \leq i \leq n$, there are two adjacent vertices u_i, v_i in $G_{\mathcal{S}, \mathcal{T}}$. The vertices u_i are pairwise adjacent *i.e.*, the set $U = \{u_i \mid 1 \leq i \leq n\}$ is a clique.
- For every triple $t = (s_i, s_j, s_k) \in \mathcal{T}$, let us add in $G_{\mathcal{S}, \mathcal{T}}$ the $v_i v_j$ -path (v_i, a_t, b_t, v_j) of length 3, and the $v_j v_k$ -path (v_j, c_t, d_t, v_k) of length 3.

- Finally, for every triple $t = (s_i, s_j, s_k) \in \mathcal{T}$ let us make adjacent a_t, b_t with every u_l such that $l \neq k$, similarly let us make adjacent c_t, d_t with every u_l such that $l \neq i$.

It can be noticed from Definition 2 that for any $1 \leq i \leq n$, the vertex u_i is adjacent to any vertex but those v_j such that $j \neq i$, those a_t, b_t such that s_i is the last element of triple t and those c_t, d_t such that s_i is the first element of triple t . We refer to Figure 1 for an illustration (see also Figure 2). Observe that $G_{\mathcal{S}, \mathcal{T}}$ has diameter 3 because the clique U dominates $G_{\mathcal{S}, \mathcal{T}}$, therefore $pl(G_{\mathcal{S}, \mathcal{T}}) \leq 3$ and we will show that it is hard to distinguish graphs with path-length two from graphs with path-length three. Similarly, the clique U dominates $G_{\mathcal{S}, \mathcal{T}}$ hence $pb(G_{\mathcal{S}, \mathcal{T}}) \leq 2$, thus we will show that it is hard to distinguish graphs with path-breadth one from graphs with path-breadth two.

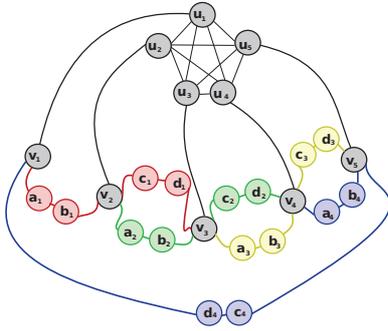


Figure 1: The graph $G_{\mathcal{S}, \mathcal{T}}$ for $\mathcal{S} = [1, 5]$ and $\mathcal{T} = \{(i, i+1, i+2) \mid 1 \leq i \leq 4\}$. Each colour corresponds to a given triple of \mathcal{T} . For ease of reading, the adjacency relations between the vertices u_i and the colored vertices a_t, b_t, c_t, d_t are not drawn.

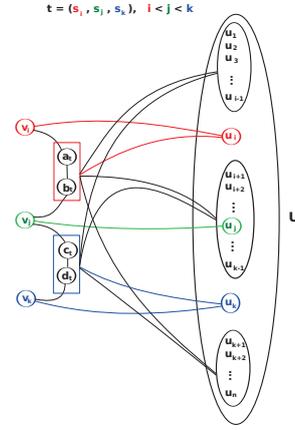


Figure 2: Adjacency relations in $G_{\mathcal{S}, \mathcal{T}}$ for one given triple $t = (s_i, s_j, s_k)$.

Lemma 4 *Let \mathcal{S} be a set of n elements, let \mathcal{T} be a set of m ordered triples of elements in \mathcal{S} . If $(\mathcal{S}, \mathcal{T})$ is a yes-instance of the Betweenness problem then $pb(G_{\mathcal{S}, \mathcal{T}}) \leq 1$ and $pl(G_{\mathcal{S}, \mathcal{T}}) \leq 2$, where $G_{\mathcal{S}, \mathcal{T}}$ is the graph that is defined in Definition 2.*

Proof. Since $pl(G_{\mathcal{S}, \mathcal{T}}) \leq 2 \cdot pb(G_{\mathcal{S}, \mathcal{T}})$ then we only need to prove that $pb(G_{\mathcal{S}, \mathcal{T}}) \leq 1$. For convenience, let us reorder the elements of \mathcal{S} so that for every triple $(s_i, s_j, s_k) \in \mathcal{T}$ either $i < j < k$ or $k < j < i$. It is possible to do that because by the hypothesis $(\mathcal{S}, \mathcal{T})$ is a yes-instance of the Betweenness problem. If furthermore $k < j < i$, let us also replace (s_i, s_j, s_k) with the inverse triple (s_k, s_j, s_i) . This way, we have a total ordering of \mathcal{S} such that $s_i < s_j < s_k$ for every triple $(s_i, s_j, s_k) \in \mathcal{T}$. Then, let us construct a path decomposition (P, \mathcal{X}) with n bags, denoted X_1, X_2, \dots, X_n , as follows. For every $1 \leq i \leq n$, $U \subseteq X_i$ and $v_i \in X_i$. For every $t = (s_i, s_j, s_k) \in \mathcal{T}$, we add both a_t, b_t into the bags X_l with $i \leq l \leq j$, similarly we add both c_t, d_t into the bags X_l with $j \leq l \leq k$. By construction, the clique U is contained in any bag of P and for every triple $t = (s_i, s_j, s_k) \in \mathcal{T}$ we have $a_t, b_t, v_i \in X_i$ and $a_t, b_t, c_t, d_t, v_j \in X_j$ and $c_t, d_t, v_k \in X_k$, therefore (P, \mathcal{X}) is indeed a path decomposition of $G_{\mathcal{S}, \mathcal{T}}$.

We claim that for every i , $X_i \subseteq N[u_i]$, that will prove the lemma. Indeed if it were not the case for some i then by Definition 2 there should exist $t \in \mathcal{T}, j, k$ such that: either $t = (s_i, s_j, s_k) \in \mathcal{T}$ and $c_t, d_t \in X_i$; or $t = (s_k, s_j, s_i) \in \mathcal{T}$ and $a_t, b_t \in X_i$. But then by construction either a_t, b_t

are only contained in the bags X_l for $k \leq l \leq j$, or c_t, d_t are only contained in the bags X_l for $j \leq l \leq k$, thus contradicting the fact that either $a_t, b_t \in X_i$ or $c_t, d_t \in X_i$. ■

Lemma 5 *Let \mathcal{S} be a set of n elements, let \mathcal{T} be a set of m ordered triples of elements in \mathcal{S} . If $pb(G_{\mathcal{S}, \mathcal{T}}) \leq 1$ or $pl(G_{\mathcal{S}, \mathcal{T}}) \leq 2$ then $(\mathcal{S}, \mathcal{T})$ is a yes-instance of the Betweenness problem, where $G_{\mathcal{S}, \mathcal{T}}$ is the graph that is defined in Definition 2.*

Proof. Since $pl(G_{\mathcal{S}, \mathcal{T}}) \leq 2 \cdot pb(G_{\mathcal{S}, \mathcal{T}})$ then we only need to consider the case when $pl(G_{\mathcal{S}, \mathcal{T}}) \leq 2$. Let (P, \mathcal{X}) be a path decomposition of length two, that exists by the hypothesis. Since the vertices v_i are pairwise at distance 3 then the subpaths P_{v_i} that are induced by the bags containing vertex v_i are pairwise disjoint. Therefore, starting from an arbitrary endpoint of P and considering each vertex v_i in the order that it appears in the path decomposition, this defines a total ordering over \mathcal{S} . Let us reorder the set \mathcal{S} so that vertex v_i is the i^{th} vertex to appear in the path-decomposition. We claim that for every triple $t = (s_i, s_j, s_k) \in \mathcal{T}$, either $i < j < k$ or $k < j < i$, that will prove the lemma.

By way of contradiction, let $t = (s_i, s_j, s_k) \in \mathcal{T}$ such that either $j < \min\{i, k\}$ or $j > \max\{i, k\}$. By symmetry, we only need to consider the case when $j < i < k$. In such case by construction the path between P_{v_j} and P_{v_k} in P contains P_{v_i} . Let $B \in P_{v_i}$, by the properties of a tree decomposition it is a $v_j v_k$ -separator, so it must contain one of c_t, d_t . However, vertex $v_i \in B$ is at distance 3 from both vertices c_t, d_t , thus contradicting the fact that (P, \mathcal{X}) has length 2. ■

We are now able to prove Theorems 1 and 2.

Proof of Theorem 1. To prove that a graph G satisfies $pl(G) \leq k$, it suffices to give as a certificate a tree decomposition of G with length at most k . Indeed, the all-pairs-shortest-paths in G can be computed in polynomial-time. Therefore, the problem of deciding whether a graph has path-length at most k is in NP. Given an instance $(\mathcal{S}, \mathcal{T})$ of the Betweenness problem, let $G_{\mathcal{S}, \mathcal{T}}$ be as defined in Definition 2. We claim that $pl(G_{\mathcal{S}, \mathcal{T}}) \leq 2$ if and only if the pair $(\mathcal{S}, \mathcal{T})$ is a yes-instance of the Betweenness problem. This will prove the NP-hardness because our reduction is polynomial and the Betweenness problem is NP-complete. To prove the claim in one direction, if $(\mathcal{S}, \mathcal{T})$ is a yes-instance then by Lemma 4 $pl(G_{\mathcal{S}, \mathcal{T}}) \leq 2$. Conversely, if $pl(G_{\mathcal{S}, \mathcal{T}}) \leq 2$ then $(\mathcal{S}, \mathcal{T})$ is a yes-instance by Lemma 5, that proves the claim in the other direction. ■

Proof of Theorem 2. To prove that a graph G satisfies $pb(G) \leq k$, it suffices to give as a certificate a tree decomposition of G with breadth at most k . Indeed, the all-pairs-shortest-paths in G can be computed in polynomial-time. Therefore, the problem of deciding whether a graph has path-breadth at most k is in NP. Given an instance $(\mathcal{S}, \mathcal{T})$ of the Betweenness problem, let $G_{\mathcal{S}, \mathcal{T}}$ be as defined in Definition 2. We claim that $pb(G_{\mathcal{S}, \mathcal{T}}) \leq 1$ if and only if the pair $(\mathcal{S}, \mathcal{T})$ is a yes-instance of the Betweenness problem. This will prove the NP-hardness because our reduction is polynomial and the Betweenness problem is NP-complete. To prove the claim in one direction, if $(\mathcal{S}, \mathcal{T})$ is a yes-instance then by Lemma 4 $pb(G_{\mathcal{S}, \mathcal{T}}) \leq 1$. Conversely, if $pb(G_{\mathcal{S}, \mathcal{T}}) \leq 1$ then $(\mathcal{S}, \mathcal{T})$ is a yes-instance by Lemma 5, that proves the claim in the other direction. ■

To conclude this section, we strengthen the above hardness results with two inapproximability results. Indeed, it has to be noticed that for any graph parameter $param$, an α -approximation algorithm for $param$ with $\alpha < 1 + \frac{1}{k}$ is enough to separate the graphs G such that $param(G) \leq k$ from those such that $param(G) \geq k + 1$. Therefore, the two following corollaries follow from our polynomial-time reduction.

Corollary 2 *For every $\varepsilon > 0$, the path-length of a graph cannot be approximated within a factor $\frac{3}{2} - \varepsilon$ unless $P=NP$.*

Proof. Let $G_{S,\mathcal{T}}$ be the graph of the reduction in Theorem 1. By Definition 2, it has diameter at most 3 and so $pl(G_{S,\mathcal{T}}) \leq 3$. Since it is NP-hard to decide whether $pl(G_{S,\mathcal{T}}) \leq 2$, therefore it does not exist a $(\frac{3}{2} - \varepsilon)$ -approximation algorithm for path-length unless $P=NP$. ■

Corollary 3 *For every $\varepsilon > 0$, the path-breadth of a graph cannot be approximated within a factor $2 - \varepsilon$ unless $P=NP$.*

Proof. Let $G_{S,\mathcal{T}}$ be the graph of the reduction in Theorem 2. By Definition 2, the set U is a dominating clique and so $pb(G_{S,\mathcal{T}}) \leq 2$. Since it is NP-hard to decide whether $pb(G_{S,\mathcal{T}}) \leq 1$, therefore it does not exist a $(2 - \varepsilon)$ -approximation algorithm for path-breadth unless $P=NP$. ■

So far, there exists a 2-approximation algorithm for path-length and a 3-approximation algorithm for path-breadth [27]. Therefore, we let open whether there exist $\frac{3}{2}$ -approximation algorithms for path-length and 2-approximation algorithms for path-breadth.

3.2 Tree-breadth

We prove next that computing the tree-breadth is NP-hard.

Theorem 4 *Deciding whether a graph has tree-breadth at most k is NP-complete even if $k = 1$.*

Theorem 5 *Deciding whether a graph admits a k -good tree decomposition is NP-complete even if $k = 1$.*

Proof. The problem is in NP. By Corollary 1, a graph G admits a star-decomposition if and only if $tb(G) \leq 1$, therefore it is NP-hard to decide whether a graph admits a 1-good path decomposition by Theorem 4. ■

In order to prove Theorem 4, we will reduce from the Chordal Sandwich problem (defined below). In [39], the author also proposed a reduction from the Chordal Sandwich problem in order to prove that computing tree-length is NP-hard. However, we will need different gadgets than in [39], and we will need different arguments to prove correctness of the reduction.

Problem 2 (Chordal Sandwich)

Input: graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ such that $E_1 \subseteq E_2$.

Question: Is there a chordal graph $H = (V, E)$ such that $E_1 \subseteq E \subseteq E_2$?

The Chordal Sandwich problem is NP-complete even when the $2n = |V|$ vertices induce a perfect matching in \bar{G}_2 (the complementary of G_2) [11, 32]. Perhaps surprisingly, the later constricton on the structure of \bar{G}_2 is a key element in our reduction. Indeed, we will need the following technical lemma.

Lemma 6 *Let $G_1 = (V, E_1)$, $G_2 = (V, E_2)$ such that $E_1 \subseteq E_2$ and \bar{G}_2 (the complementary of G_2) is a perfect matching. Suppose that $\langle G_1, G_2 \rangle$ is a yes-instance of the Chordal Sandwich problem.*

Then, there exists a reduced tree decomposition (T, \mathcal{X}) of G_1 such that for every forbidden edge $\{u, v\} \notin E_2$: $T_u \cap T_v = \emptyset$, $T_u \cup T_v = T$, furthermore there are two adjacent bags $B_u \in T_u$ and $B_v \in T_v$ such that $B_u \setminus u = B_v \setminus v$.

Proof. Let $H = (V, E)$ be any chordal graph such that $E_1 \subseteq E \subseteq E_2$ (that exists because $\langle G_1, G_2 \rangle$ is a yes-instance of the Chordal Sandwich problem by the hypothesis) and the number $|E|$ of edges is maximized. We will prove that any clique-tree (T, \mathcal{X}) of H satisfies the above properties (given in the statement of the lemma). To prove it, let $\{u, v\} \notin E_2$ be arbitrary. Observe that $T_u \cap T_v = \emptyset$ (else, $\{u, v\} \in E$, that would contradict that $E \subseteq E_2$).

Furthermore, let $B_u \in T_u$ minimize the distance in T to the subtree T_v , let B be the unique bag that is adjacent to B_u on a shortest-path between B_u and T_v in T . Note that $B \notin T_u$ by the minimality of $\text{dist}_T(B_u, T_v)$, however B may belong to T_v . Removing the edge $\{B_u, B\}$ in T yields two subtrees T_1, T_2 with $T_u \subseteq T_1$ and $T_v \subseteq T_2$. In addition, we have that for every $x \in V \setminus u$ such that $T_x \cap T_1 \neq \emptyset$, $\{u, x\} \in E_2$ since $x \neq v$ and \bar{G}_2 is a perfect matching by the hypothesis. Similarly, we have that for every $y \in V \setminus v$ such that $T_y \cap T_2 \neq \emptyset$, $\{v, y\} \in E_2$. Therefore, by maximality of the number $|E|$ of edges, it follows that $T_1 = T_u$ and $T_2 = T_v$, and so, $T_u \cup T_v = T$. In particular, $B = B_v \in T_v$.

Finally, let us prove that $B_u \setminus u = B_v \setminus v$. Indeed, assume for the sake of contradiction that $B_u \setminus u \neq B_v \setminus v$. In particular, $(B_u \setminus B_v) \setminus u \neq \emptyset$ or $(B_v \setminus B_u) \setminus v \neq \emptyset$. Suppose w.l.o.g. that $(B_u \setminus B_v) \setminus u \neq \emptyset$. Let $H' = (V, E')$ be obtained from H by adding an edge between vertex v and every vertex of $(B_u \setminus B_v) \setminus u$. By construction $|E'| > |E|$. Furthermore, H' is chordal since a clique-tree of H' can be obtained from (T, \mathcal{X}) by adding a new bag $(B_u \setminus u) \cup \{v\}$ in-between B_u and B_v . However, for every $x \in (B_u \setminus B_v) \setminus u$ we have that $\{x, v\} \in E_2$ since $x \neq u$ and \bar{G}_2 is a perfect matching by the hypothesis. As a result, $E' \subseteq E_2$, thus contradicting the maximality of the number $|E|$ of edges in H . ■

Proof of Theorem 4. The problem is in NP. To prove the NP-hardness, let $\langle G_1, G_2 \rangle$ be any input of the Chordal Sandwich problem such that \bar{G}_2 is a perfect matching. The graph G' is constructed from G_1 as follows. First we add a clique V' of $2n = |V|$ vertices in G' . Vertices $v \in V$ are in one-to-one correspondance with vertices $v' \in V'$. Then, for every forbidden edge $\{u, v\} \notin E_2$, vertices u, v are respectively made adjacent to all vertices in $V' \setminus v'$ and $V' \setminus u'$. Finally, we add a distinct copy of the gadget F_{uv} in Figure 3, and we make adjacent s_{uv} and t_{uv} to the two vertices u', v' (see also Figure 4 for an illustration). We will prove $tb(G') = 1$ if and only if $\langle G_1, G_2 \rangle$ is a yes-instance of the Chordal Sandwich problem. This will prove the NP-hardness because our reduction is polynomial and the Chordal Sandwich problem is NP-complete even when the $2n = |V|$ vertices induce a perfect matching in \bar{G}_2 (the complementary of G_2) [11, 32].

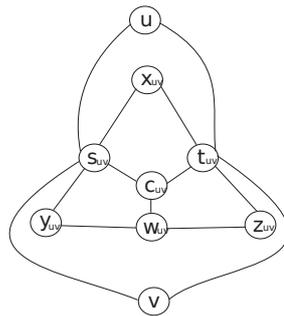


Figure 3: The gadget F_{uv} .

In one direction, assume $tb(G') = 1$, let (T, \mathcal{X}) be a star-decomposition of G' . Let $H = (V, \{\{u, v\} \mid T_u \cap T_v \neq \emptyset\})$, that is a chordal graph such that $E_1 \subseteq E(H)$. To prove that $\langle G_1, G_2 \rangle$ is a yes-instance of the Chordal Sandwich problem, it suffices to prove that $T_u \cap T_v = \emptyset$ for every forbidden edge $\{u, v\} \notin E_2$. More precisely, we will prove that $T_{s_{uv}} \cap T_{t_{uv}} \neq \emptyset$, for we claim that

the latter implies $T_u \cap T_v = \emptyset$. Indeed, assume $T_{s_{uv}} \cap T_{t_{uv}} \neq \emptyset$ and $T_u \cap T_v \neq \emptyset$. Since s_{uv} and t_{uv} are both adjacent to u and v , therefore the four subtrees $T_u, T_v, T_{s_{uv}}, T_{t_{uv}}$ pairwise intersect. By the Helly property (Lemma 1) $T_u \cap T_v \cap T_{s_{uv}} \cap T_{t_{uv}} \neq \emptyset$, hence there is a bag containing u, v, s_{uv}, t_{uv} but then it contradicts the fact that (T, \mathcal{X}) is a star-decomposition because no vertex dominates the four vertices. Therefore, $T_{s_{uv}} \cap T_{t_{uv}} \neq \emptyset$ implies $T_u \cap T_v = \emptyset$. Let us prove that $T_{s_{uv}} \cap T_{t_{uv}} \neq \emptyset$. By contradiction, assume $T_{s_{uv}} \cap T_{t_{uv}} = \emptyset$. Every bag B onto the path between $T_{s_{uv}}$ and $T_{t_{uv}}$ must contain c_{uv}, x_{uv} , furthermore $N[c_{uv}] \cap N[x_{uv}] = \{s_{uv}, t_{uv}\}$. Since, (T, \mathcal{X}) is a star-decomposition, the latter implies either $s_{uv} \in B$ and $B \subseteq N[s_{uv}]$ or $t_{uv} \in B$ and $B \subseteq N[t_{uv}]$. Consequently, there exist two adjacent bags $B_s \in T_{s_{uv}}, B_t \in T_{t_{uv}}$ such that $B_s \subseteq N[s_{uv}]$ and $B_t \subseteq N[t_{uv}]$. Furthermore, $B_s \cap B_t$ is an $s_{uv}t_{uv}$ -separator by the properties of a tree decomposition. In particular, $B_s \cap B_t$ must intersect the path (y_{uv}, w_{uv}, z_{uv}) because $y_{uv} \in N(s_{uv})$ and $z_{uv} \in N(t_{uv})$. However, $B_s \subseteq N[s_{uv}], B_t \subseteq N[t_{uv}]$ but $N[s_{uv}] \cap N[t_{uv}] \cap \{y_{uv}, w_{uv}, z_{uv}\} = \emptyset$, hence $B_s \cap B_t \cap \{y_{uv}, w_{uv}, z_{uv}\} = \emptyset$, that is a contradiction. As a result, $T_{s_{uv}} \cap T_{t_{uv}} \neq \emptyset$ and so, $T_u \cap T_v = \emptyset$.

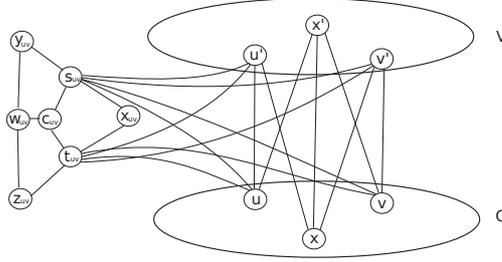


Figure 4: The graph G' (simplified view).

Conversely, assume that $\langle G_1, G_2 \rangle$ is a yes-instance of the Chordal Sandwich problem. Since \bar{G}_2 is a perfect matching by the hypothesis, by Lemma 6 there exists a reduced tree decomposition (T, \mathcal{X}) of G_1 such that for every forbidden edge $\{u, v\} \notin E_2$: $T_u \cap T_v = \emptyset, T_u \cup T_v = T$ and there are two adjacent bags $B_u \in T_u, B_v \in T_v$ so that $B_u \setminus u = B_v \setminus v$. Let us modify (T, \mathcal{X}) in order to obtain a star-decomposition of G' .

In order to achieve the result, we first claim that for every edge $\{t, t'\} \in E(T)$, the bags $X_t, X_{t'}$ differ in exactly one vertex, that is, $|X_t \setminus X_{t'}| = 1$ and similarly $|X_{t'} \setminus X_t| = 1$. Indeed, $X_t \setminus X_{t'} \neq \emptyset$ because (T, \mathcal{X}) is reduced, so, let $u_{tt'} \in X_t \setminus X_{t'}$. Let $v_{tt'} \in V$ be the unique vertex satisfying $\{u_{tt'}, v_{tt'}\} \notin E_2$, that is well-defined because \bar{G}_2 is a perfect matching by the hypothesis. Note that $v_{tt'} \in X_{t'}$ because $u_{tt'} \notin X_{t'}$ and $T_{u_{tt'}} \cup T_{v_{tt'}} = T$. Furthermore, $v_{tt'} \notin X_t$ because $u_{tt'} \in X_t$ and $T_{u_{tt'}} \cap T_{v_{tt'}} = \emptyset$. By construction of (T, \mathcal{X}) , there are two adjacent bags $B_{u_{tt'}} \in T_{u_{tt'}}, B_{v_{tt'}} \in T_{v_{tt'}}$ such that $B_{u_{tt'}} \setminus u_{tt'} = B_{v_{tt'}} \setminus v_{tt'}$. Since $u_{tt'} \in X_t \setminus X_{t'}$ and $v_{tt'} \in X_{t'} \setminus X_t$, therefore, $X_t = B_{u_{tt'}}$ and $X_{t'} = B_{v_{tt'}}$, and so, $X_t \setminus X_{t'} = \{u_{tt'}\}$ and $X_{t'} \setminus X_t = \{v_{tt'}\}$. In the following, we will keep the above notations $u_{tt'}, v_{tt'}$ for every edge $\{t, t'\} \in E(T)$ (in particular, $u_{tt'} = v_{t't}$ and $v_{tt'} = u_{t't}$).

Let us construct the star-decomposition (T', \mathcal{X}') of G' as follows.

- For every node $t \in V(T)$, let $S_t = X_t \cup V' \cup (\bigcup_{t' \in N_T(t)} \{s_{u_{tt'}v_{tt'}}, t_{u_{tt'}v_{tt'}}\})$ (in particular, $|S_t| = 2n + |X_t| + 2 \cdot \deg_T(t)$). We will first construct a path decomposition of $G'[S_t]$ whose bags are the sets $Y_{tt'} = X_t \cup V' \cup \{s_{u_{tt'}v_{tt'}}, t_{u_{tt'}v_{tt'}}\}$ for every edge $\{t, t'\} \in E(T)$ (note that the bags can be linearly ordered in an arbitrary way in the path decomposition). Furthermore, for every edge $\{t, t'\} \in E(T)$, $Y_{tt'} \subseteq N[u'_{tt'}]$, where $u'_{tt'} \in V'$ is the corresponding vertex to $u_{tt'} \in V$ in the clique V' (see Figure 5 for an illustration). Therefore, the above constructed path decomposition is a 1-good path decomposition.

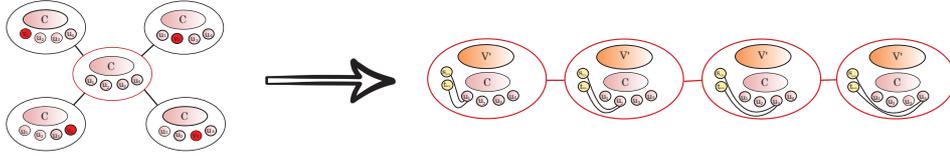


Figure 5: The 1-good path decomposition (right) obtained from the central bag with degree four (left).

- Then, we will connect the 1-good path decompositions together. More precisely, let us add an edge between the two bags $Y_{tt'}$ and $Y_{t't}$ for every edge $\{t, t'\} \in E(T)$ (see Figure 6 for an illustration).

In so doing, we claim that one obtains a star-decomposition of $G'[\bigcup_{t \in V(T)} S_t]$. Indeed, it is a tree decomposition since:

- the clique V' is contained in all bags;
- for every $\{t, t'\} \in E(T)$ the two vertices $s_{u_{tt'}v_{tt'}}, t_{u_{tt'}v_{tt'}}$ are only contained in the two adjacent bags $Y_{tt'}$ and $Y_{t't}$, furthermore $u_{tt'}, u'_{tt'}, v'_{tt'} \in Y_{tt'}$ and $v_{tt'}, u'_{t't}, v'_{t't} \in Y_{t't}$;
- last, each vertex $v \in V$ is contained in $\{Y_{tt'} \mid v \in X_t \text{ and } t' \in N_T(t)\}$ which induces a subtree since (T, \mathcal{X}) is a tree decomposition of G_1 .

Since in addition every bag $Y_{tt'}$, with $\{t, t'\} \in E(T)$, is dominated by $u'_{tt'} \in V'$, this proves the claim that one obtains a star-decomposition.

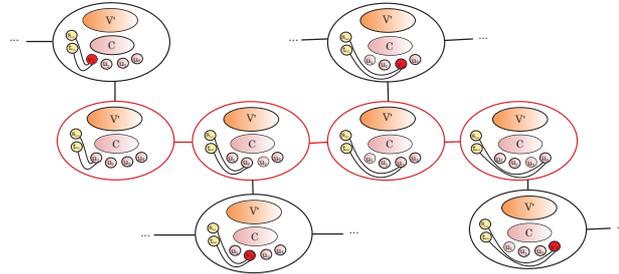


Figure 6: Connection of the 1-good path decomposition in Figure 5 to the neighbouring 1-good path decompositions.

- In order to complete the construction, let us observe that for every forbidden edge $\{u, v\} \notin E_2$, there is a star-decomposition of $F_{uv} \setminus \{u, v\}$ with three leaf-bags $\{x_{uv}, s_{uv}, t_{uv}\}$, $\{y_{uv}, s_{uv}, w_{uv}\}$ and $\{z_{uv}, s_{uv}, w_{uv}\}$ and one internal bag of degree three $B_{uv} = \{c_{uv}, s_{uv}, t_{uv}, w_{uv}\}$. For every $\{t, t'\} \in E(T)$, we simply connect the above star-decomposition of $F_{u_{tt'}v_{t't}} \setminus \{u_{tt'}, v_{t't}\}$ by making the internal bag $B_{u_{tt'}v_{t't}}$ adjacent to one of $Y_{tt'}$ or $Y_{t't}$ (see Figure 7 for an illustration).

By construction, the resulting tree decomposition (T', \mathcal{X}') of G' is a star-decomposition, hence $tb(G') = 1$. ■

Recall that we can use our reduction from Definition 2 in order to prove that computing path-length and path-breadth is NP-hard. By contrast, our reduction from Theorem 4 cannot

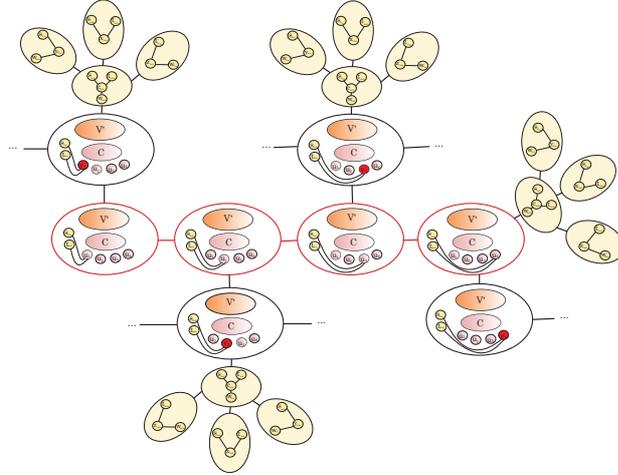


Figure 7: The respective star-decompositions of the gadgets $F_{u_i v_i}$ are connected to the other bags.

be used to prove that tree-length is NP-hard to compute (in fact, the graph G' resulting from the reduction has tree-length two).

Finally, as in the previous Section 3.1, let us strengthen Theorem 4 with an inapproximability result.

Corollary 4 *For every $\varepsilon > 0$, the tree-breadth of a graph cannot be approximated within a factor $2 - \varepsilon$ unless $P=NP$.*

4 General properties of graphs with tree-breadth one

In Section 3.2, we prove that computing the tree-breadth is NP-hard. In particular, the recognition of graphs with tree-breadth one is NP-complete. In light of this result, we focus on graphs with tree-breadth one (in order to obtain a better understanding of what makes the problem difficult)

Problem 3 (1-tree-breadth)

Input: a connected graph $G = (V, E)$

Question: $tb(G) \leq 1$?

In Lemma 7, we show that the problem of recognizing graphs with tree-breadth at most one is equivalent to the problem of computing tree-breadth. This further motivates our study of these graphs. Then, we will prove necessary conditions for a graph to be of tree-breadth one.

- One is that all graphs with a star-decomposition have a *domination elimination ordering* (see Section 4.1). We will outline a few implications of this property.
- Second, we will prove in Lemma 9 that if a graph G admits a star-decomposition then so do all the *blocks* of G , where the blocks here denote a particular case of induced subgraphs of G (e.g., see Definition 4).

Finally, we will obtain from the latter result a polynomial-time algorithm to decide whether a bipartite graph has tree-breadth at most one (e.g., see Section 4.3).

Definition 3 Let G be a graph with n vertices, denoted by v_1, v_2, \dots, v_n , and let r be a positive integer. The graph G'_r is obtained from G by adding a clique with n vertices, denoted by $U = \{u_1, u_2, \dots, u_n\}$, so that for every $1 \leq i \leq n$, vertex u_i is adjacent to $B_G(v_i, r) = \{x \in V(G) \mid \text{dist}_G(v_i, x) \leq r\}$.

Lemma 7 For every graph G , for every positive integer r , let G'_r be as defined in Definition 3, $\text{tb}(G) \leq r$ if and only if $\text{tb}(G'_r) \leq 1$.

Proof. If $\text{tb}(G) \leq r$ then we claim that starting from any tree decomposition (T, \mathcal{X}) of G with breadth at most r , one obtains a star-decomposition of G'_r by adding the clique U in every bag X_t , $t \in V(T)$. Indeed, in such case for every bag X_t , $t \in V(T)$, by the hypothesis there is $v_i \in V(G)$ such that $\max_{x \in X_t} \text{dist}_G(v_i, x) \leq r$, whence $X_t \cup U \subseteq N_{G'_r}[u_i]$. Conversely, if $\text{tb}(G'_r) \leq 1$ then we claim that starting from any tree decomposition (T', \mathcal{X}') of G'_r with breadth at most one, one obtains a tree decomposition of G with breadth at most r by removing every vertex of the clique U from every bag X'_t , $t \in V(T')$. Indeed, in such case for every bag X'_t , $t \in V(T')$, by the hypothesis there is $y \in V(G'_r)$ such that $X'_t \subseteq N_{G'_r}[y]$. Furthermore, $y \in \{u_i, v_i\}$ for some $1 \leq i \leq n$, and so, since $N_{G'_r}[v_i] \subseteq N_{G'_r}[u_i]$ by construction, therefore $X'_t \setminus U \subseteq N_{G'_r}(u_i) \setminus U = \{x \in V(G) \mid \text{dist}_G(v_i, x) \leq r\}$. ■

4.1 Existence of specific elimination orderings

Independently from the remaining of the section, let us prove some interesting properties of graphs with tree-breadth one in terms of *elimination orderings*. More precisely, a *domination elimination ordering* [21] of a graph G is a total ordering of its vertex-set, denoted by v_1, v_2, \dots, v_n , so that for every $1 \leq i < n$, there is $j > i$ satisfying that $N_G(v_i) \cap \{v_{i+1}, v_{i+2}, \dots, v_n\} \subseteq N_G[v_j]$. The existence of domination elimination orderings in some graph classes and their algorithmic applications has been studied in [16]. Let us prove that graphs with tree-breadth one all admit a domination elimination ordering.

Lemma 8 Let G be such that $\text{tb}(G) \leq 1$, G admits a domination elimination ordering.

Proof. Assume G has at least two vertices (or else, the lemma is trivial). To prove the lemma, it suffices to prove the existence of $u, v \in V(G)$ distinct such that $N(v) \subseteq N[u]$ and $\text{tb}(G \setminus v) \leq 1$ (then, the lemma follows by induction on the order of the graph).

If G admits a universal vertex u , then one can pick $v \in V(G) \setminus u$ arbitrary, $N(v) \subseteq N[u]$ because u is universal in G , furthermore $\text{tb}(G \setminus v) \leq 1$ because $G \setminus v$ admits a universal vertex u .

Else, G does not admit any universal vertex, let (T, \mathcal{X}) be a reduced tree decomposition of G of breadth one, that is a star-decomposition by Lemma 3. Let X_t , $t \in V(T)$ be a leaf. Since the tree decomposition is reduced, there must be $v \in X_t$ satisfying $T_v = \{X_t\}$. Now there are two cases.

- Suppose there is $u \in X_t \setminus v$ such that $X_t \subseteq N[u]$. Then, $N(v) \subseteq X_t \subseteq N[u]$, and $\text{tb}(G \setminus v) \leq 1$ because $G \setminus v$ can be obtained from G by contracting the edge $\{u, v\}$ and tree-breadth is contraction-closed by Lemma 2.
- Else, $X_t \subseteq N[v]$, and for every $x \in X_t \setminus v$, $X_t \not\subseteq N[x]$. Let $t' \in V(T)$ be the unique node adjacent to node t in T , that exists because G does not admit any universal vertex

and so, T has at least two bags. Let us assume that for every $x \in X_t \setminus v$, $x \in X_t \cap X_{t'}$ (for otherwise, $N(x) \subseteq N[v]$ and $tb(G \setminus x) \leq 1$ because $G \setminus x$ can be obtained from G by contracting the edge $\{v, x\}$ to v and tree-breadth is contraction-closed by Lemma 2). In particular, let $u \in X_{t'}$ satisfy $X_{t'} \subseteq N[u]$. Then, $N(v) = X_t \cap X_{t'} \subseteq N[u]$, furthermore $tb(G \setminus v) \leq 1$ because $(T \setminus t, \mathcal{X} \setminus X_t)$ is a star-decomposition of $G \setminus v$. ■

Note that for a graph to have tree-breadth one, it must satisfy the necessary condition of Lemma 8 and this can be checked in polynomial-time. However, the existence of some domination elimination ordering is not a sufficient condition for the graph to have tree-breadth one. Indeed, every grid has a domination elimination ordering but the *tree-length* of the $n \times m$ grid is at least $\min\{n, m\} - 1$ [24] (recall that $tl(G) \leq 2tb(G)$ for any graph G).

The existence of a domination elimination ordering has some interesting consequences about the graph structure. Let us recall one such a consequence about the *cop-number* of the graph.

Corollary 5 *For any graph G with $tb(G) \leq 1$, G has cop-number ≤ 2 and the upper-bound is sharp.*

Proof. By Lemma 8, G admits a domination elimination ordering. Therefore, by [19, Theorem 4] G has cop-number ≤ 2 . One can prove the sharpness of the upper-bound by setting $G := C_4$, the cycle with four vertices. ■

4.2 Properties of particular decompositions

In the following, it will be useful not only to constrain the properties of the star-decomposition whose existence we are interested in, but also to further constrain the properties of the graph G that we take as input. Let us first remind basic terminology about graph separators.

Definition 4 *Let $G = (V, E)$ be connected, a separator of G is any subset $S \subseteq V$ such that $G \setminus S$ has at least two connected components.*

In particular, a full component for S is any connected component C of $G \setminus S$ satisfying $N(C) = S$. A block is any induced subgraph $G[C \cup S]$ with S being a separator and C being a full component for S .

Finally, a minimal separator is a separator with at least two full components.

Our objective is to prove that if a graph G has tree-breadth one then so do all its blocks. In fact, we will prove a slightly more general result:

Lemma 9 *Let $G = (V, E)$ be a graph, $S \subseteq V$ be a separator, and $W \subseteq V \setminus S$ be the union of some connected components of $G \setminus S$. If $tb(G) = 1$ and W contains a full component for S , then $tb(G[W \cup S]) = 1$. More precisely if (T, \mathcal{X}) is a tree decomposition of G of breadth one, then $(T, \{X_t \cap (W \cup S) \mid X_t \in \mathcal{X}\})$ is a tree decomposition of $G[W \cup S]$ of breadth one.*

Proof. Let (T, \mathcal{X}) be a tree decomposition of breadth one of G . Let us remove all vertices in $V \setminus (W \cup S)$ from bags in (T, \mathcal{X}) , which yields a tree decomposition (T', \mathcal{X}') of the induced subgraph $G[W \cup S]$. To prove the lemma, we are left to prove that (T', \mathcal{X}') has breadth one. Let X_t be a bag of (T', \mathcal{X}') . By construction, X_t is fully contained into some bag of (T, \mathcal{X}) , so it has radius one in G . Let $v \in V$ be such that $X_t \subseteq N_G[v]$. If $v \in W \cup S$, then we are done. Else, since for all $x \notin S \cup W$, $N(x) \cap (S \cup W) \subseteq S$ (because S is a separator by the hypothesis), we must have that $X_t \subseteq S$. Let $A \subseteq W$ be a full component for S , that exists by the hypothesis, and let T_A be the subtree that is induced by the bags intersecting the component. Since we have that the

subtree T_A and the subtrees $T_x, x \in X_t$ pairwise intersect — because for all $x \in X_t, x \in S$ and so, x has a neighbour in A —, then by the Helly property (Lemma 1) $T_A \cap (\bigcap_{x \in X_t} T_x) \neq \emptyset$ *i.e.*, there exists a bag in (T, \mathcal{X}) containing X_t and intersecting A . Moreover, any vertex dominating this bag must be either in S or in A , so in particular there exists $u \in A \cup S$ dominating X_t , which proves the lemma. ■

Lemma 9 implies that, under simple assumptions, a graph of tree-breadth one can be disconnected using any (minimal) separator, and the components must still induce subgraphs with tree-breadth one. The converse does not hold in general, yet there are interesting cases when it does.

Lemma 10 *Let $G = (V, E)$ be a graph, $S \subseteq V$ be a clique-minimal-separator and A be a full component for S . Then, $tb(G) = 1$ if and only if both $tb(G[A \cup S]) = 1$ and $tb(G[V \setminus A]) = 1$.*

Proof. By the hypothesis $V \setminus (A \cup S)$ contains a full component because S is a minimal separator. Therefore, if G has tree-breadth one, then so do $G[A \cup S]$ and $G[V \setminus A]$ by Lemma 9. Conversely, suppose that we have both $tb(G[A \cup S]) = 1$ and $tb(G[V \setminus A]) = 1$. Let (T^1, \mathcal{X}^1) be a tree decomposition of $G[A \cup S]$ with breadth one, let (T^2, \mathcal{X}^2) be a tree decomposition of $G[V \setminus A]$ with breadth one. Then for every $i \in \{1, 2\}$ we have that since S is a clique the subtrees $T_s^i, s \in S$, pairwise intersect, so by the Helly Property (Lemma 1) $\bigcap_{s \in S} T_s^i \neq \emptyset$ *i.e.*, S is fully contained into some bag of (T^1, \mathcal{X}^1) and it is fully contained into some bag of (T^2, \mathcal{X}^2) . Moreover, $(A \cup S) \cap (V \setminus A) = S$, therefore a tree decomposition of G with breadth one can be obtained by adding an edge between some bag of (T^1, \mathcal{X}^1) containing S and some bag of (T^2, \mathcal{X}^2) containing S . ■

Recall that computing the clique-minimal-decomposition of a graph G can be done in $\mathcal{O}(nm)$ -time, where m denotes the number of edges [7]. By doing so, one replaces a graph G with the maximal subgraphs of G that have no clique-separator, *a.k.a. atoms*. Therefore, we will assume in the remaining of the proofs that there is no clique-separator in the graphs that we will study, we will call them *prime graphs*.

4.3 Application to bipartite graphs

In this section, we describe an $\mathcal{O}(nm)$ -time algorithm so as to decide whether a prime bipartite graph has tree-breadth one. This combined with Lemma 10 proves that it can be decided in polynomial-time whether a bipartite graph has tree-breadth one.

We will first describe a more general problem and how to solve it in polynomial-time.

Tree decompositions with constrained set of bags. Our algorithm for bipartite graphs makes use of the correspondance between tree decompositions and triangulations of a graph. Indeed, recall that any reduced tree decomposition (T, \mathcal{X}) of a graph G is a clique-tree for some chordal supergraph H of G whose maximal cliques are the bags of \mathcal{X} . Conversely, for any chordal supergraph H of G , every clique-tree of H is a tree decomposition of G whose bags are the maximal cliques of H [30]. Therefore as shown below, the following subproblem can be solved in polynomial-time:

Problem 4

Input: a graph G , a family \mathcal{X} of subsets of $V(G)$.

Question: Does there exist a tree T such that (T, \mathcal{X}) is a tree decomposition of G ?

Let us assume w.l.o.g. that no subset of \mathcal{X} is properly contained into another one. To solve Problem 4.3, it suffices to make every subset $X \in \mathcal{X}$ a clique in G , then to verify whether the resulting supergraph H of G is a chordal graph whose maximal cliques are exactly the sets in \mathcal{X} . Since chordal graphs can be recognized in linear-time, and so can be enumerated their maximal cliques [29], therefore Problem 4.3 can be solved in polynomial-time.

The algorithm for bipartite graphs. Now, given a bipartite graph G , we aim to exhibit a family \mathcal{X} so that $tb(G) = 1$ if and only if there is a star-decomposition of G whose bags are \mathcal{X} . By doing so, we will reduce the recognition of bipartite graph with tree-breadth at most one to the more general Problem 4.3.

Lemma 11 *Let $G = (V_0 \cup V_1, E)$ be a prime bipartite graph with tree-breadth one. There is (T, \mathcal{X}) a star-decomposition of G such that either $\mathcal{X} = \{N[v_0] \mid v_0 \in V_0\}$, or $\mathcal{X} = \{N[v_1] \mid v_1 \in V_1\}$.*

Proof. Let (T, \mathcal{X}) be a star-decomposition of G , that exists by Lemma 3, minimizing the number $|\mathcal{X}|$ of bags. Suppose there is some $v_0 \in V_0$, there is $t \in V(T)$ such that $X_t \subseteq N_G[v_0]$ (the case when there is some $v_1 \in V_1$, there is $t \in V(T)$ such that $X_t \subseteq N_G[v_1]$ is symmetrical to this one). We claim that for every $t' \in V(T)$, there exists $v'_0 \in V_0$ satisfying $X_{t'} \subseteq N_G[v'_0]$. By contradiction, let $v_0 \in V_0, v_1 \in V_1$, let $t, t' \in V(T)$ be such that $X_t \subseteq N_G[v_0], X_{t'} \subseteq N_G[v_1]$. By connectivity of the tree T we may assume w.l.o.g. that $\{t, t'\} \in E(T)$. Moreover, $N_G(v_0) \cap N_G(v_1) = \emptyset$ because G is bipartite. Therefore, $X_t \cap X_{t'} \subseteq \{v_0, v_1\}$, and in particular if $X_t \cap X_{t'} = \{v_0, v_1\}$ then v_0, v_1 are adjacent in G . However, by the properties of a tree decomposition this implies that $X_t \cap X_{t'}$ is a clique-separator (either an edge or a single vertex), thus contradicting the fact that G is prime.

Now, let $v_0 \in V_0$ be arbitrary. We claim that there is a unique bag $X_t, t \in V(T)$, containing v_0 . Indeed, any such bag X_t must satisfy $X_t \subseteq N_G[v_0]$, whence the subtree T_{v_0} can be contracted into a single bag $\bigcup_{t \in T_{v_0}} X_t$ without violating the property for the tree decomposition to be a star-decomposition. As a result, the unicity of the bag X_t follows from the minimality of $|\mathcal{X}|$. Finally, since X_t is unique and $X_t \subseteq N_G[v_0]$, therefore $X_t = N_G[v_0]$ and so, $\mathcal{X} = \{N[v_0] \mid v_0 \in V_0\}$. ■

We can easily deduce from Lemma 11 the following algorithm for deciding whether a prime bipartite graph G has tree-breadth one. Let (V_0, V_1) be the (unique) bipartition of the vertex-set of G into two stable sets. Let $\mathcal{X}_0 = \{N[v_0] \mid v_0 \in V_0\}$, let $\mathcal{X}_1 = \{N[v_1] \mid v_1 \in V_1\}$. By Lemma 11, $tb(G) = 1$ if and only if one of $(G, \mathcal{X}_0), (G, \mathcal{X}_1)$ is a yes-instance of Problem 4.3.

5 Algorithm for planar graphs

We are now ready to present our main result. In this section, we describe a quadratic-time algorithm for deciding whether a prime planar graph has tree-breadth one. Overall, we claim that it gives us a quadratic-time algorithm for deciding whether a general planar graph has tree-breadth one. Indeed, the clique-decomposition of a planar graph takes $\mathcal{O}(n^2)$ -time to be computed, furthermore the disjoint union of the atoms has $\mathcal{O}(n + m)$ vertices [7], that is $\mathcal{O}(n)$ for planar graphs.

Roughly, we will construct a star-decomposition of the graph by increments. The main principle of the recursive algorithm is to find a particular vertex, called *leaf-vertex*. Informally, it extracts a new bag of the star-decomposition from some ball around the leaf-vertex. Then, depending on the case, either the leaf-vertex vertex is removed or some edge is added or contracted. In both cases, the resulting graph remains prime and planar and has tree-breadth one if and only if the initial one has tree-breadth one.

We prove that each inductive step takes a linear time. Moreover, we prove that there are at most a linear number of recursive iterations (Lemma 27).

There are three kinds of leaf-vertices (e.g., see Figure 8).

Definition 5 Let $G = (V, E)$ be a graph. A vertex v is a leaf-vertex if one of the following conditions hold.

Type 1. $N(v)$ induces an $a_v b_v$ -path for some $a_v, b_v \in V \setminus \{v\}$, denoted by Π_v , of length at least 3 and there exists $d_v \in V \setminus \{v\}$ such that $N(v) \subseteq N(d_v)$, i.e., d_v dominates Π_v .

Type 2. $N(v)$ induces a path, denoted by $\Pi_v = (a_v, b_v, c_v)$, of length 2.

Type 3. $N(v)$ consists of two non adjacent vertices a_v and c_v , and there exists $b_v \in (N(a_v) \cap N(c_v)) \setminus \{v\}$.

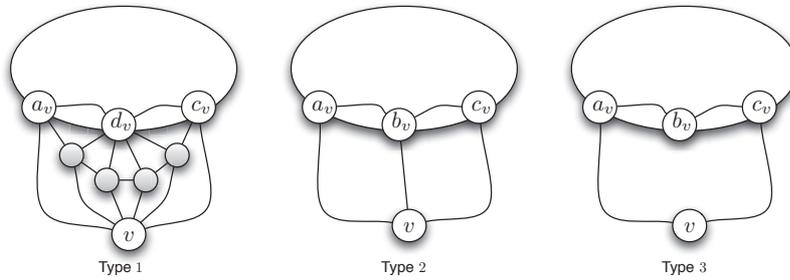


Figure 8: The three kinds of leaf-vertices.

We are now ready to describe the algorithm.

5.1 Algorithm Leaf-BottomUp

Let $G = (V, E)$ be prime planar graph. Assume G has at least 7 vertices (else, it is easy to conclude).

Step 1 The first step is to find a leaf-vertex in G . In Section 5.4.1, we describe how to decide whether G has a leaf-vertex in linear-time.

- if G has no leaf-vertex, then, by Theorem 7, no minimal separator of G induces a path of length 2. Therefore, by Lemma 20, $tb(G) = 1$ only if G has a star-decomposition with at most 2 bags. In that case, Algorithm Leaf-BottomUp checks whether it exists a star-decomposition with at most 2 bags, which can be done in quadratic time (see Lemma 26). If it exists, then $tb(G) = 1$. Otherwise, $tb(G) > 1$.
- otherwise, let v be a leaf-vertex of G and go to Step 2 if v is of Type 1 and go to Step 3 otherwise.

Step 2 Case v is of Type 1. Let Π_v and d_v be defined as in Definition 5. If $V = N[v] \cup \{d_v\}$ then trivially $tb(G) = 1$. Else by Theorem 8, G' is prime and planar, where G' is the graph obtained from $G \setminus v$ by contracting the internal nodes of Π_v to a single edge, and $tb(G') = 1$

if and only if $tb(G') = 1$. In that case, Algorithm **Leaf-BottomUp** is recursively applied on G' .

Step 3 Case v is of Type 2 or 3. Let a_v, b_v, c_v be defined as in Definition 5.

In that case, Algorithm **Leaf-BottomUp** checks whether $G \setminus v$ is prime. By Theorem 6, for any clique minimal separator S of $G \setminus v$ (if any), there exists $u_v \in V \setminus \{a_v, b_v, c_v, v\}$ such that $S = \{b_v, u_v\}$. Therefore, this can be checked in linear time (by checking with a Depth-First-Search whether there is a cut-vertex of $G \setminus \{a_v, b_v, c_v, v\}$ in the neighbors of b_v). If $G \setminus v$ is prime then go to Step 3.1, else go to Step 3.2.

Step 3.1 Case v is of Type 2 or 3 and $G \setminus v$ is prime. There are 4 cases that can be determined in linear-time.

(a) **Case $|N(a_v) \cap N(c_v)| \geq 3$ in $G \setminus v$, or there exists a minimal separator $S \subseteq (N(a_v) \cap N(c_v)) \cup \{a_v, c_v\}$ in $G \setminus v$ and $\{a_v, c_v\} \subseteq S$.**

By Theorem 9, $tb(G) = 1$ if and only if $tb(G \setminus v) = 1$. Since, moreover, $G \setminus v$ is planar and prime, then Algorithm **Leaf-BottomUp** is recursively applied on $G \setminus v$.

(b) **Case: $|N(a_v) \cap N(c_v)| < 3$ in $G \setminus v$ and there is no minimal separator $S \subseteq (N(a_v) \cap N(c_v)) \cup \{a_v, c_v\}$ in $G \setminus v$ such that $\{a_v, c_v\} \subseteq S$.**

i **Subcase: $|N(a_v) \cap N(c_v)| = 1$ in $G \setminus v$.** In that subcase, $N(a_v) \cap N(c_v) = \{v, b_v\}$ and, by Theorem 10, $tb(G) = 1$ if and only if $G = C_4$, a cycle with four vertices. Note that here it implies that $tb(G) > 1$ because G has at least 7 vertices.

ii **Subcase: $|N(a_v) \cap N(c_v)| = 2$ in $G \setminus v$.** In that subcase, let $N(a_v) \cap N(c_v) = \{v, b_v, u_v\}$. By Theorem 11, since G has more than 5 vertices, the graph G' obtained from G by adding edges $\{v, u_v\}$ and $\{b_v, v\}$ is planar and prime, and moreover $tb(G) = 1$ if and only if $tb(G') = 1$. In that case, Algorithm **Leaf-BottomUp** is recursively applied on G' .

Step 3.2 Case v is of Type 2 or 3 and $G \setminus v$ has a clique separator. As mentioned in Step 3, in that case, there exists $u_v \in V \setminus \{a_v, b_v, c_v, v\}$ such that $S = \{b_v, u_v\}$ is a minimal clique separator of $G \setminus v$. Moreover, by Theorem 6, $G \setminus \{a_v, b_v, c_v, v\}$ is connected.

By Theorem 12, $tb(G) = 1$ if and only if $tb(G') = 1$ where G' is obtained from G by adding the edge $\{v, b_v\}$ (if it were not already there). Moreover, G' is prime and planar. Hence, we may assume that $\{v, b_v\} \in E$ (if not Algorithm **Leaf-BottomUp** adds it).

Furthermore by Theorem 6, since G has more than 5 vertices, $u_v \notin N(a_v) \cap N(c_v)$. In the latter case, let us assume w.l.o.g. that $u_v \notin N(a_v)$, that is either $u_v \notin N(a_v) \cup N(c_v)$ or $u_v \in N(c_v) \setminus N(a_v)$. There are several cases to be considered.

(a) **Case $u_v \notin N(a_v) \cup N(c_v)$,
or $(N(u_v) \cap N(a_v)) \cup \{v, c_v\}$ does not separate u_v and a_v in G .**

By Theorem 13, G/va_v is prime and planar, and $tb(G) = 1$ if and only if $tb(G/va_v) = 1$. In that case, Algorithm **Leaf-BottomUp** is recursively applied on G/va_v , the graph obtained from G by contracting the edge $\{v, a_v\}$.

(b) **Case $u_v \in N(c_v) \setminus N(a_v)$,
and $(N(u_v) \cap N(a_v)) \cup \{v, c_v\}$ separates u_v and a_v in G .**

In that case, recall that G has at least 7 vertices. Again, Algorithm **Leaf-BottomUp** distinguishes several subcases.

- i **Subcase** $N(b_v) = \{v, a_v, c_v, u_v\}$. In that subcase, by Theorem 14, we can find in linear-time a vertex $x \in (N(a_v) \cap N(u_v)) \setminus \{b_v\}$ such that G' is planar, where G' is obtained from G by adding the edge $\{b_v, x\}$. Moreover, by Theorem 15, G'/b_vx (obtained by contracting $\{b_v, x\}$) is prime and $tb(G) = 1$ if and only if $tb(G'/b_vx) = 1$. In that case, Algorithm **Leaf-BottomUp** is recursively applied on G'/b_vx .
- ii **Subcase** $\{v, a_v, c_v, u_v\} \subset N(b_v)$ and $N(b_v) \cap N(a_v) \cap N(u_v) \neq \emptyset$. In that subcase, $|N(b_v) \cap N(a_v) \cap N(u_v)| = 1$ by Lemma 23 and let x be this common neighbor. By Theorem 15, G/b_vx (obtained by contracting $\{b_v, x\}$) is prime and $tb(G) = 1$ if and only if $tb(G/b_vx) = 1$. In that case, Algorithm **Leaf-BottomUp** is recursively applied on G/b_vx .
- iii **Subcase** $\{v, a_v, c_v, u_v\} \subset N(b)$ and $N(b_v) \cap N(a_v) \cap N(u_v) = \emptyset$. In that subcase, by Theorem 16, there must be a unique $x \in (N(a_v) \cap N(u_v)) \setminus \{b_v\}$ such that $N(b_v) \cap N(x)$ is a b_vx -separator of G and $|N(b_v) \cap N(x)| \geq 3$ (or else, $tb(G) > 1$).
 - **Suppose there is a leaf-vertex** $\ell \in N(b_v) \cap N(x)$. By Lemma 24, ℓ is of Type 1 or $G \setminus \ell$ is prime. In that case, go to Step 2 if ℓ is of Type 1, and go to Step 3.1 if ℓ has Type 2 or 3 (in both cases, ℓ takes the role of v). Note that we never go back to Step 3.2 in such case, so the algorithm cannot loop.
 - Otherwise, by Theorem 17, there exist $y, z \in N(b_v) \cap N(x)$ two non-adjacent vertices, such that G' is prime and planar, and $tb(G) = 1$ if and only if $tb(G') = 1$, where G' is obtained from G by adding the edge $\{x, y\}$. In that case, Algorithm **Leaf-BottomUp** is recursively applied on G' .

5.2 Properties of prime planar graphs with tree-breadth one

5.2.1 General lemmas

We will first investigate on general properties of prime planar graphs. In particular, the following properties do not depend on the existence of a star-decomposition, therefore we do not use tree decompositions in our proofs. However, note that we refer to Definition 5 in Theorem 6. For clarity, we will separate the properties that hold for every *biconnected* planar graph from those that only hold for prime planar graphs.

Properties of biconnected planar graphs. In order to obtain these properties, we will mostly rely on the notion of *intermediate graphs*, defined below.

Definition 6 [13, Definition 6] *Let $G = (V, E)$ be a planar graph. We fix a plane embedding of G . Let F be the set of faces of this embedding. The intermediate graph $G_I = (V \cup F, E_I)$ has vertex-set $V \cup F$, furthermore $E \subseteq E_I$ and we add an edge in G_I between an original vertex $v \in V$ and a face-vertex $f \in F$ whenever the corresponding vertex and face are incident in G (see Figure 9).*

Note that an intermediate graph is planar. Furthermore, since a plane embedding can be constructed in linear-time [34], therefore so can be an intermediate graph. This is important for

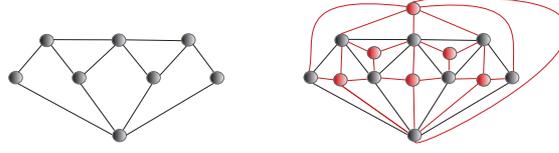


Figure 9: A plane embedding of some planar graph (left) and the corresponding intermediate graph (right). Face-vertices are coloured in red.

the quadratic-time complexity of Algorithm `Leaf-BottomUp`. To prove the correctness of Algorithm `Leaf-BottomUp` in the following, we will rely upon the following property of intermediate graphs.

Lemma 12 [13, Proposition 9] *Let S be a minimal separator of some biconnected planar graph $G = (V, E)$ and let C be a full component of $G \setminus S$. We fix a plane embedding of G . Then S corresponds to a cycle $v_S(C)$ of G_I , of length $2|S|$ and with $V \cap v_S(C) = S$, and such that $G_I \setminus v_S(C)$ has at least two connected components. Moreover, the original vertices of one of these components are exactly the vertices of C .*

In the following, we will rely upon two properties which both follow from Lemma 12. The first one is the following structural property of minimal separators of planar graphs.

Corollary 6 *Let S be a minimal separator of a biconnected planar graph $G = (V, E)$. Then, S either induces a cycle or a forest of paths.*

Proof. Let us fix a plane embedding of G , let G_I be the corresponding intermediate graph. Then, let C_S be a smallest cycle of G_I such that $V \cap C_S = S$, that exists by Lemma 12. To prove the corollary, it suffices to prove that C_S is an induced cycle of G_I . By contradiction, assume the existence of a chord xy of C_S . Note that $x \in S$ or $y \in S$ because face-vertices are pairwise non-adjacent in G_I . Therefore assume w.l.o.g. that $x \in S$. Let us divide C_S in two cycles C_1, C_2 such that $C_1 \cap C_2 = \{x, y\}$. By the minimality of C_S , S intersects both $C_1 \setminus C_2$ and $C_2 \setminus C_1$. Therefore, let $z_1, z_2 \in S$ such that $z_1 \in C_1 \setminus C_2$ and $z_2 \in C_2 \setminus C_1$. Finally, let A, B be two full components of $G \setminus S$. Observe that $(A \cup B) \cap (C_1 \cup C_2) = \emptyset$ because $V \cap C_S = S$. Let us contract C_1, C_2 in order to obtain the two triangles (z_1, x, y) and (z_2, x, y) . In such case, there is a $K_{3,3}$ -minor of G_I with $\{A, B, y\}$ and $\{x, z_1, z_2\}$ being the respective sides of the bipartition, thus contradicting the fact that G_I is planar. Therefore, C_S is an induced cycle of G_I and so, S induces a subgraph of a cycle in G , that is either a cycle or a forest of path. ■

On the algorithmic side, one can also deduce from Lemma 12 the following corollary.

Corollary 7 *Let G be a biconnected planar graph, let S be a minimal separator of G . There is a planar supergraph G_S of G with same vertex-set so that S either induces an edge (if $|S| = 2$) or a cycle of G_S , and it can be constructed in linear-time.*

Proof. Let us fix a plane embedding of G , let G_I be the corresponding intermediate graph. For every face-vertex f of G_I , let us consider $S_f = S \cap N_{G_I}(f)$. We first claim that $|S_f| \leq 2$. Indeed, let A, B be two full components of $G \setminus S$, let us contract them to any two vertices $a \in A, b \in B$. Then, there is a $K_{3,|S_f|}$ -minor of G_I with $\{a, b, f\}$ and S_f being the respective parts of the bipartition. Since G_I is planar by construction, therefore, $|S_f| \leq 2$.

Now, the graph G_S is constructed from G as follows (we refer to Figure 10 for an illustration of the proof). For every face-vertex f of G_I , if $S_f = (x, y)$ then we add the edge $\{x, y\}$ in G_S .

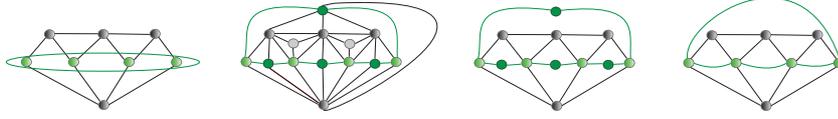


Figure 10: Addition of edges in a planar graph G so as to make a minimal separator of G induce a cycle.

Note that G_S is a minor of G_I and so, it is a planar graph. Moreover, by Lemma 12 there is a cycle of G_I whose original vertices are exactly S and so, S induces a connected subgraph of G_S . In particular if $|S| = 2$, then it must be an edge. Else, $|S| > 2$ and the connected subgraph $G_S[S]$ contains a cycle by construction. Since S is a minimal separator of G_S by construction and $G_S[S]$ is not acyclic, it follows from Corollary 6 that S induces a cycle of G_S . ■

We will often make use of the routine of Corollary 7 in order to prove the quadratic-time complexity of Algorithm `Leaf-BottomUp`.

Properties of prime planar graphs. Unlike the above Corollaries 6 and 7 (which hold for every biconnected planar graph), the following results only hold for prime planar graphs. We will make use of the following structural properties of prime planar graphs in order to prove the correctness of Algorithm `Leaf-BottomUp`.

Lemma 13 *Let $G = (V, E)$ be a prime graph that is $K_{3,3}$ -minor-free. Let $v \in V$, for every minimal separator $S \subseteq N_G(v)$ of the subgraph $G \setminus v$, S consists of two non-adjacent vertices.*

Proof. Let $S \subseteq N_G(v)$ be a minimal separator of $G \setminus v$. There must exist two full components A and B of S in $G \setminus (S \cup \{v\})$. Let us remove all nodes of the components of $G \setminus (S \cup \{v\})$ but the ones in A or B . Then, let us contract A (resp., B) in a single vertex a (resp., b). We get a $K_{3,|S|}$ as a minor of G where $\{a, b, v\}$ is one part of the bipartition, and so $|S| \leq 2$. Finally, since $S \cup \{v\}$ is also a separator of G , then $|S| \geq 2$ because otherwise $S \cup \{v\}$ would be an edge-separator. Therefore, $|S| = 2$ and it is a stable set because otherwise there would be a clique-separator of size 3 in G . ■

Lemma 14 *Let G be a prime planar graph, let the path $\Pi = (a, b, c)$ be a separator of G , and let C be a component of $G \setminus \Pi$. Then, there is at most one common neighbour of a, b in C .*

Proof. First note that Π is induced or else it would be a clique-separator of G . Furthermore, $a, c \in N(C')$ for every component C' of $G \setminus (\Pi \cup C)$ or else $N(C')$ would be a clique-separator of G (either a vertex-separator or an edge-separator). In particular, it is always possible to make vertices a, c adjacent by contracting an arbitrary component of $G \setminus (\Pi \cup C)$.

By contradiction, let $u, u' \in N(a) \cap N(b) \cap C$ be distinct. We claim that there exists a uc -path Q in $C \cup \{c\}$ that does not contain u' , because else the triangle a, b, u' would separate u from Π , that contradicts the fact that G is prime. By symmetry, there also exists a $u'c$ -path Q' in $C \cup \{c\}$ that does not contain u . There are two cases.

- Q and Q' are internally vertex-disjoint paths (see Figure 11 for an illustration). Let us contract $Q \setminus c, Q' \setminus c$ to the vertices u, u' , let us contract an arbitrary component of $G \setminus (\Pi \cup C)$ in order to make vertices a, c adjacent, then let us contract a path from Q to Q' in C (that exists, because C is connected by the hypothesis) in order to make vertices u, u' adjacent. Then one obtains from a, b, c, u, u' a K_5 -minor, which contradicts the fact that G is planar.

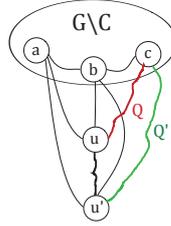


Figure 11: Case where the paths Q and Q' are internally vertex-disjoint paths.

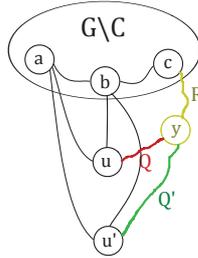


Figure 12: Case where the paths Q and Q' intersect.

- Q and Q' intersect (see Figure 12 for an illustration). Let $y \in (Q \cap Q') \setminus c$ be such that the uy -subpath of Q does not intersect Q' . Let R be the yc -subpath of Q' . We may assume w.l.o.g. that $R \subseteq Q \cap Q'$ for the remaining of the proof, whence $Q \cap Q' = R$. Let us contract $Q \setminus R, Q' \setminus R, R \setminus c$ in order to make vertices u, u', c adjacent to vertex y , then let us contract an arbitrary component of $G \setminus (P \cup C)$ in order to make vertices a, c adjacent. One obtains from a, b, c, u, u', y a $K_{3,3}$ -minor with $\{a, b, y\}$ being one side of the bipartition, that contradicts the fact that G is planar. ■

Lemma 15 *Let G be a prime planar graph, let the path $\Pi = (a, b, c)$ be a separator of G , and let C be a component of $G \setminus \Pi$. Suppose there is some vertex $v \in C$ that is a common neighbour of a, b, c . Then, either C is reduced to v , or (a, v, c) is a separator of G . Furthermore, in the latter case, the path (a, v, c) separates vertex b from $C \setminus v$.*

Proof. Let us assume that $C \setminus v \neq \emptyset$. Let D be a connected component of $G[C \setminus v]$. Note that $v \in N(D)$ because C is a connected component of $G \setminus \Pi$ by the hypothesis. To prove the lemma, it suffices to prove that $b \notin N(D)$. By contradiction, suppose that $b \in N(D)$ (see Figure 13 for an illustration). Since v, b, a and v, b, c are pairwise adjacent and G has no clique-separator by the hypothesis, then necessarily $N(D) = \{a, b, c, v\}$.

Let us contract the component D to a single vertex x . Then, let C' be any component of $G \setminus (\Pi \cup C)$. We have that $a, c \in N(C')$ or else $N(C')$ would be a clique-separator of G (either a vertex-separator or an edge-separator). So, let us contract the component C' onto vertex a in order to make a and c adjacent. One obtains from a, b, c, v, x a K_5 -minor, which contradicts the fact that G is planar. ■

We recall that the gist of Algorithm `Leaf-BottomUp` is (informally) to try to remove a leaf-vertex v from G then to apply recursively the algorithm on $G \setminus v$. Because the algorithm is strongly dependent on the fact that G is prime, it is important to characterize the cases when

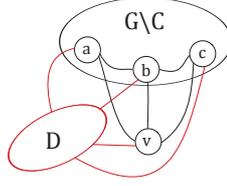


Figure 13: Existence of a component $D \subseteq C \setminus v$ that is adjacent to b .

$G \setminus v$ is also prime. Indeed, new clique-decompositions are needed when $G \setminus v$ is not prime, which may provoke a combinatorial explosion of the number of subgraphs to be considered. Therefore, before we conclude this section, let us characterize whenever there may be clique-separators in $G \setminus v$ with v being a leaf-vertex. This will first require the following lemma.

Lemma 16 *Let $G = (V, E)$ be a graph and let the path $\Pi = (a, b, c)$ be a separator of G . Let C be the union of some components of $G \setminus \Pi$ and let S be a separator of $G[C \cup \Pi]$. Then, S is a separator in G or S separates a and c in $G[C \cup \Pi]$.*

Moreover, in the latter case, $G[C \cup \Pi] \setminus S$ has exactly two components C_a and C_c containing a and c respectively.

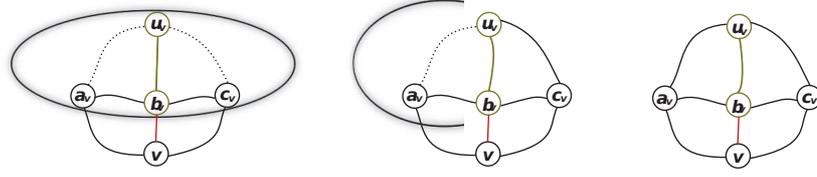
Proof. There are two cases.

- Suppose there exists a component D of $G[C \cup \Pi] \setminus S$ such that $N_G(D) \subseteq C \cup \Pi$. Since $D \cap S = \emptyset$, $S \neq V \setminus D$ and $N_G(D) \subseteq S$ therefore S is a separator of G with D being a component of $G \setminus S$.
- Else, every component D of $G[C \cup \Pi] \setminus S$ has a neighbour in $V \setminus (C \cup \Pi)$. The latter implies that $D \cap \Pi \neq \emptyset$ for every component D of $G[C \cup \Pi] \setminus S$ because C is a union of components of $G \setminus \Pi$ by the hypothesis. In particular, since there exist at least two components of $G[C \cup \Pi] \setminus S$ then there must be one containing an endpoint of Π . W.l.o.g. assume there is a component C_a of $G[C \cup \Pi] \setminus S$ such that $a \in C_a$. Let C_c be any component of $G[C \cup \Pi] \setminus (S \cup C_a)$. We have that $C_c \cap N[C_a] = \emptyset$ because S separates C_a and C_c in $G[C \cup \Pi]$. Therefore, $a, b \notin C_c$ and so, $c \in C_c$ because $C_c \cap \Pi \neq \emptyset$. This finally proves that $G[C \cup \Pi] \setminus S$ has exactly two components C_a and C_c containing a and c respectively. ■

Theorem 6 *Let $G = (V, E)$ be a prime planar graph, let v be a leaf-vertex of Type either 2 or 3 and let $\Pi_v = (a_v, b_v, c_v)$ be as defined in Definition 5. Suppose that there exists a minimal separator S in $G \setminus v$ that is a clique. Then, $S = \{u_v, b_v\}$, $u_v \notin \Pi_v$ and the following hold:*

- $V \setminus (\Pi_v \cup \{v\})$ is a full component of $G \setminus \Pi_v$.
- If $u_v \in N(a_v)$ (resp. $u_v \in N(c_v)$), then a_v (resp. c_v) is simplicial in $G \setminus v$ with neighbours $\{u_v, b_v\}$;
- Furthermore $u_v \notin N(a_v) \cap N(c_v)$ unless $V = \Pi_v \cup \{u_v, v\}$;

Proof. Note that the subgraph $G \setminus v$ is planar and S is a minimal separator of $G \setminus v$ by the hypothesis, therefore by Corollary 6 either S induces a cycle or a forest of path. Since in addition S is a clique by the hypothesis, it follows that S either induces a singleton, an edge

Figure 14: Existence of a clique-separator in $G \setminus v$.

or a triangle. Since S is a clique and G is prime, S is not a separator of G , so by Lemma 16 with $C = V \setminus (\Pi_v \cup \{v\})$, S is an $a_v c_v$ -separator of $G \setminus v$. This both implies that $b_v \in S$ and $S' := S \cup \{v\}$ is a minimal $a_v c_v$ -separator of G . In particular, S being a strict subset of some minimal separator of G it cannot induce a cycle (by Corollary 6), hence it must induce either a singleton or an edge. Furthermore, still by Lemma 16 with $C = V \setminus (\Pi_v \cup \{v\})$ there exist exactly two components C_a, C_c of $G \setminus (S \cup \{v\})$, with $a_v \in C_a, c_v \in C_c$. As a result, $S \setminus b_v \neq \emptyset$, or else $\{a_v, b_v\}, \{b_v, c_v\}$ would be edge-separators of G , thus contradicting the hypothesis. Let $S = \{u_v, b_v\}, u_v \notin \Pi_v \cup \{v\}$. If $u_v \in N(a_v)$, then $C_a \setminus a_v = \emptyset$ (and so, a_v is simplicial in $G \setminus v$), for otherwise (a_v, u_v, b_v) would be a clique-separator of G . Similarly, if $u_v \in N(c_v)$ then $C_c \setminus c_v = \emptyset$ (and so, c_v is simplicial in $G \setminus v$). In particular if $u_v \in N(a_v) \cap N(c_v)$ then $\Pi_v \cup \{u_v, v\} = V$. Last, as there exists an $a_v c_v$ -path in every component C' of $G \setminus (\Pi_v \cup \{v\})$ because G has no clique-separator by the hypothesis, therefore $u_v \in C'$. This implies that $V \setminus (\Pi_v \cup \{v\})$ is a full component of $G \setminus \Pi_v$. ■

5.2.2 Constrained star-decompositions

In the following, it will be useful to impose additional structure on the star-decompositions. In order to do that, we will prove properties on some pairs of vertices in the graph. Namely, we will prove that when $x, y \in V$ satisfy a few technical conditions, then it can be assumed that $T_x \cup T_y$ is a subtree of the star-decomposition (T, \mathcal{X}) .

Lemma 17 *Let G be a connected graph with $tb(G) = 1$, let $x, y \in V(G)$ be non-adjacent (and $x \neq y$).*

Suppose the pair (x, y) satisfies that for every xy -separator S of G , if there is $z \notin \{x, y\}$ that dominates S then $z \in N_G(x) \cap N_G(y)$.

Then, there is a star-decomposition (T, \mathcal{X}) of G with $B_x, B_y \in \mathcal{X}$, $x \in B_x, y \in B_y$ and either $B_x = B_y$ or B_x, B_y are adjacent in T . Moreover, in the latter case, $B_x \subseteq N[x], B_y \subseteq N[y]$.

Proof. Consider a star-decomposition (T, \mathcal{X}) of G , that exists by Lemma 3. If x and y are not in a same bag, let B_x and B_y be the bags containing respectively x and y and as close as possible in T . By the properties of a tree decomposition, $N(x) \cap N(y) \subseteq B_x \cap B_y$. Hence, for any bag B between B_x and B_y in T , $N(x) \cap N(y) \subseteq B$.

- **Case 1:** If B_x and B_y are not adjacent in T , let B be any bag in the path between B_x and B_y in T . By the properties of a tree decomposition, B is an xy -separator. Moreover, let $z \in B$ dominate the bag, by the hypothesis $z \in N(x) \cap N(y)$ because $x, y \notin B$. As a result, adding x and y in each bag B between B_x and B_y achieves a star-decomposition of G that has a bag containing both x, y .
- **Case 2:** Now, let us assume that B_x and B_y are adjacent in T . Note that, if $B_x \subseteq N[z]$ for some $z \in N(x) \cap N(y)$ (resp., if $B_y \subseteq N[z]$ for some $z \in N(x) \cap N(y)$) the result holds.

Indeed, adding y in B_x (resp., x in B_y) achieves a star-decomposition of G that has a bag containing both x, y .

So, let us consider the case when none of the two bags B_x, B_y is dominated by a vertex of $N(x) \cap N(y)$. Then, $B_x \setminus x$ and $B_y \setminus y$ are xy -separators by the properties of a tree decomposition. Let $z_x \in B_x, z_y \in B_y$ satisfy $B_x \subseteq N[z_x]$ and $B_y \subseteq N[z_y]$. By the hypothesis, $z_x \in \{x\} \cup (N(x) \cap N(y))$ and $z_y \in \{y\} \cup (N(x) \cap N(y))$. Thus it follows that $z_x = x$ and $z_y = y$ (or else, we are back to Case 1). Note that $B_x \cap B_y = N(x) \cap N(y)$ in such a case. ■

We will mostly use the following two weaker versions of Lemma 17 in our proofs.

Corollary 8 *Let G be a connected graph with $tb(G) = 1$, let $x, y \in V(G)$ be non-adjacent (and $x \neq y$).*

Suppose there exists a minimal separator $S \subseteq (N(x) \cap N(y)) \cup \{x, y\}$ in G and $\{x, y\} \subseteq S$.

Then, there is a star-decomposition (T, \mathcal{X}) of G with $B_x, B_y \in \mathcal{X}$, $x \in B_x, y \in B_y$ and either $B_x = B_y$ or B_x, B_y are adjacent in T . Moreover, in the latter case, $B_x \subseteq N[x], B_y \subseteq N[y]$.

Proof. We claim that for every xy -separator S' of G , if there is $z \notin \{x, y\}$ such that $S' \subseteq N[z]$ then $z \in N(x) \cap N(y)$. Observe that if the claim holds, then the corollary follows from Lemma 17. To prove the claim, let $S \subseteq (N(x) \cap N(y)) \cup \{x, y\}$ be a separator of G and $\{x, y\} \subseteq S$, that exists by the hypothesis. Note that for any full component C of $G \setminus S$, the xy -separator S' must contain some vertex in C . Since there are at least two full components of $G \setminus S$, then $z \in S \setminus \{x, y\} \subseteq N(x) \cap N(y)$, that finally proves the claim. ■

So far, the two above results in this section (Lemma 17 and Corollary 8) apply to *general* graphs with tree-breadth one. However, we will need the fact that the graph is planar for the following corollary.

Corollary 9 *Let G be a connected graph with $tb(G) = 1$, let $x, y \in V(G)$ be non-adjacent (and $x \neq y$).*

Suppose G is $K_{3,3}$ -minor-free and $|N_G(x) \cap N_G(y)| \geq 3$.

Then, there is a star-decomposition (T, \mathcal{X}) of G with $B_x, B_y \in \mathcal{X}$, $x \in B_x, y \in B_y$ and either $B_x = B_y$ or B_x, B_y are adjacent in T . Moreover, in the latter case, $B_x \subseteq N[x], B_y \subseteq N[y]$.

Proof. We claim that for every xy -separator S of G , if there is $z \notin \{x, y\}$ such that $S \subseteq N[z]$ then $z \in N(x) \cap N(y)$. Observe that if the claim holds, then the corollary follows from Lemma 17. To prove the claim, first recall that $|N(x) \cap N(y)| \geq 3$. Since vertex z dominates S and S is an xy -separator, therefore, z dominates $N(x) \cap N(y)$ because $N(x) \cap N(y) \subseteq S$. In such case, $z \in N(x) \cap N(y)$, or else, G admits a $K_{3, |N(x) \cap N(y)|}$ -minor with $\{x, y, z\}$ and $N(x) \cap N(y)$ being the respective of the bipartition, which contradicts the hypothesis. ■

Before we conclude this section, let us emphasize a useful consequence of Corollary 8 regarding minimal 2-separators.

Lemma 18 *Let $G = (V, E)$ with $tb(G) = 1$, let $x, y \in V$ be non-adjacent such that $S = \{x, y\}$ is a minimal separator of G ($x \neq y$). For every full component C of $G \setminus S$, we have that $N(x) \cap N(y) \cap C \neq \emptyset$.*

Proof. Let (T, \mathcal{X}) be a star-decomposition of G , that exists by Lemma 3, minimizing the distance in T between the subtrees T_x and T_y (respectively induced by the bags containing x and y). There are two cases.

- First, suppose that $T_x \cap T_y \neq \emptyset$. For any full component C of $G \setminus S$, let T_C be the subtree that is induced by all bags intersecting C . Because C is a full component, there must be an edge between x and a vertex of C , and this edge is in a bag of $T_x \cap T_C$. Similarly, there must be an edge between y and a vertex of C , and this edge is in a bag of $T_y \cap T_C$. As a result, the subtrees T_x, T_y, T_C are pairwise intersecting, and so by the Helly property (Lemma 1) $T_x \cap T_y \cap T_C \neq \emptyset$ i.e., there exists a bag X_t which contains S and it intersects C . Let $z \in X_t$ dominate the bag. Note that $z \in C \cup S$ because it has to dominate some vertices in C and so, it cannot be in $V \setminus (C \cup S)$. Furthermore, recall that x, y are non-adjacent by the hypothesis. Therefore, $z \in C \cap N(x) \cap N(y)$, and the result holds for any full component C of $G \setminus S$.
- Else, since $S = \{x, y\}$ is a minimal separator and we assume (T, \mathcal{X}) to minimize the distance in T between T_x and T_y , by Corollary 8 there are two adjacent bags B_x, B_y such that $x \in B_x \setminus B_y$ dominates $B_x, y \in B_y \setminus B_x$ dominates B_y . Since $B_x \cap B_y$ is an xy -separator by the properties of a tree-decomposition, then $B_x \cap B_y \cap C \neq \emptyset$ for every full component C of $G \setminus S$, that is $N(x) \cap N(y) \cap C \neq \emptyset$.

■

5.2.3 Bounded Treewidth

Independently from Algorithm `Leaf-BottomUp`, let us introduce in this section another property of (not necessarily prime) planar graphs with tree-breadth one. More precisely, we prove these graphs have bounded treewidth. To prove this property, we will use the same terminology as for the previous subsections.

Lemma 19 *Let G be planar with $tb(G) \leq 1$. Then, $tw(G) \leq 4$ and the upper-bound is sharp.*

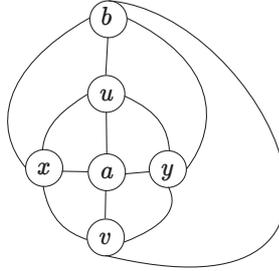


Figure 15: A planar graph G with $tb(G) = 1$ and $tw(G) = 4$.

Proof. The treewidth of G is the maximum treewidth of its atoms [10], so, let us assume G to be a prime planar graph. Let (T, \mathcal{X}) be any star-decomposition of G , the graph $H = (V, \{\{u, v\} \mid T_u \cap T_v \neq \emptyset\})$ is chordal. Furthermore, if H' is a chordal graph with same vertex-set V and such that $E(G) \subseteq E(H') \subseteq E(H)$ then any clique-tree of H' is still a star-decomposition of G . Therefore, we will assume w.l.o.g. that H is a minimal triangulation of G and (T, \mathcal{X}) is a clique-tree of H (in particular, (T, \mathcal{X}) is reduced). Additional properties of (T, \mathcal{X}) will be deduced from the latter assumption about H by using the results from [14]. Let us now prove the lemma by induction on $|V(G)|$ (the base-case of the graph with a single vertex is trivial).

- If $|\mathcal{X}| = 1$, then G has some universal vertex u . Furthermore, since G is planar therefore, $G \setminus u$ is outerplanar [44]. Consequently, $tw(G \setminus u) \leq 2$ [9], so, $tw(G) \leq 3$.

- Suppose $|\mathcal{X}| = 2$. Let $\mathcal{X} = \{B, B'\}$. Since (T, \mathcal{X}) is assumed to be a clique-tree of some minimal triangulation H of G , therefore, $B \cap B'$ is a minimal separator [14]. Let us remind that by Corollary 7 there is a planar supergraph G' of G with same vertex-set so that $B \cap B'$ induces either an edge or a cycle of G' . Furthermore (T, \mathcal{X}) is also a star-decomposition of G' , so, $tb(G') \leq 1$. In addition, $tw(G) \leq tw(G')$. Recall that we can further assume G' to be prime (or else, we apply the induction hypothesis on the atoms of G'), hence $B \cap B'$ induces a cycle of G' of length at least four. Let $B \Delta B' = (B \setminus B') \cup (B' \setminus B)$. Since H is assumed to be a minimal triangulation and B, B' are leaves of a clique-tree of H , therefore, $B \setminus B'$ is a (nonempty) dominating clique of the subgraph $G[B]$, and similarly $B' \setminus B$ is a (nonempty) dominating clique of the subgraph $G[B']$ [14]. Thus, every vertex $u \in B \Delta B'$ satisfies $B \cap B' \in N(u)$. Since $|B \cap B'| \geq 4$ because $B \cap B'$ induces a cycle of G' of length at least four, therefore, $|B \Delta B'| \leq 2$ or else there would be a $K_{3,3}$ -minor of G' with any three vertices of $B \cap B'$ being one part of the bipartition. As a result, since $B \cap B'$ induces a cycle, $tw(G) \leq tw(G') \leq 2 + |B \Delta B'| = 4$.
- Finally, suppose $|\mathcal{X}| \geq 3$. Let $t \in V(T)$ be an internal node, by the properties of a tree decomposition the bag X_t is a separator of G . Let $b \in X_t$ satisfy $X_t \subseteq N_G[b]$. Since G is prime and so, biconnected, therefore $X_t \setminus b$ is a separator of $G \setminus b$. In such case, let us remind by Lemma 13 that there exist $a, c \in X_t \setminus b$ non-adjacent such that $\{a, c\}$ is a minimal separator of $G \setminus b$. In particular, the path $\Pi = (a, b, c)$ is a separator of G . Let C_1, C_2, \dots, C_l be the components of $G \setminus \Pi$. For every $1 \leq i \leq l$, let G_i be obtained from $G[C_i \cup \Pi]$ by making the two endpoints a, c of Π adjacent. Note that G_i can be obtained from G by edge-contractions (because G is prime and so, $a, c \in N(C_j)$ for every $1 \leq j \leq l$), therefore, $tb(G_i) \leq 1$ because tree-breadth is stable under edge-contractions (Lemma 2). In addition, $tw(G) \leq \max_i tw(G_i)$ because Π induces a triangle in every graph G_i by construction. As a result, for every $1 \leq i \leq l$, since $|V(G_i)| < |V(G)|$ by construction, therefore $tw(G_i) \leq 4$ by the induction hypothesis, whence $tw(G) \leq 4$.

Let G be constructed from the cycle (u, v, x, y) of length four by adding two vertices a, b such that $N_G(a) = N_G(b) = \{u, x, v, y\}$ (see Figure 15 for an illustration). Since there exists a star-decomposition of G with two bags (respectively dominated by a, b), $tb(G) \leq 1$. Moreover, G is 4-regular by construction, therefore $tw(G) \geq 4$ [10]. This proves the sharpness of the upper-bound. ■

Note that since it is well-known that many difficult problems can be solved on bounded-treewidth graphs in linear-time, therefore, it may be the case that the recognition of planar graphs with tree-breadth at most one can be simplified by using Lemma 19. However, we were unable to find a way to use it in our proofs (actually, the star-decomposition that can be computed using our algorithm may have unbounded width — because of leaf-vertices of Type 1).

5.3 Correctness of Algorithm Leaf-BottomUp

5.3.1 Existence of a P_3 -separator

As a first step to prove correctness of Algorithm Leaf-BottomUp, let us prove correctness of Step 1. That is, we will prove that for every planar graph G with $tb(G) = 1$, G contains a leaf-vertex or G admits a star-decomposition with at most two bags.

To prove this step, we will prove additional properties of the minimal separators of prime planar graphs with tree-breadth one. In the following, let $\mathcal{P}_3(G)$ be the set of (not necessarily minimal) separators of G that induce paths of length 2 (we will call them P_3 -separators since

they have three vertices). We will distinguish the case when $\mathcal{P}_3(G) \neq \emptyset$ from the case when $\mathcal{P}_3(G) = \emptyset$.

Theorem 7 *Let G be a prime planar graph with $tb(G) = 1$. If $\mathcal{P}_3(G) \neq \emptyset$, then G has a leaf-vertex.*

Proof. Let $\Pi = (a, b, c) \in \mathcal{P}_3(G)$ minimize the size of a smallest component of $G \setminus \Pi$. We recall that $\{a, c\} \notin E(G)$ because G is assumed to be prime by the hypothesis (the latter fact will be used in the following). Let C be any component of $G \setminus \Pi$ of minimum size. Our aim is to prove the existence of some leaf-vertex $v \in C$ (the latter dominating the component C), that will prove Theorem 7.

Claim 1 *There do not exist $\Pi' \subseteq \Pi \cup C$, $C' \subset C$ such that $\Pi' \in \mathcal{P}_3(G)$ and C' is a component of $G \setminus \Pi'$.*

Proof. The claim follows from the minimality of C . ◇

We will often use Claim 1 in the remaining of the proof.

Let (T, \mathcal{X}) be a star-decomposition of G , that exists by Lemma 3. In particular, let T_a, T_c be the subtrees that are respectively induced by the bags containing a or c . Assume w.l.o.g. that (T, \mathcal{X}) minimizes the distance in T between the subtrees T_a and T_c . We will distinguish the case $T_a \cap T_c \neq \emptyset$ from the case $T_a \cap T_c = \emptyset$.

Case $T_a \cap T_c \neq \emptyset$. In such case, the subtrees T_a, T_b, T_c are pairwise intersecting and so, by the Helly property (Lemma 1) $T_a \cap T_b \cap T_c \neq \emptyset$. Let us remove all vertices in $V \setminus (\Pi \cup C)$ from bags in (T, \mathcal{X}) . Let us call (T, \mathcal{X}^C) the resulting tree decomposition of $G[\Pi \cup C]$.

Claim 2 *(T, \mathcal{X}^C) has breadth one.*

Proof. There are two cases to be considered.

- If b has some neighbour in C , then C must be a full component of $G \setminus \Pi$, or else one of $\{a, b\}, \{b, c\}$ should be a clique-separator thus contradicting the fact that G is prime by the hypothesis. In such case, the claim follows from Lemma 9.
- Else, b has no neighbour in C , and let D be the connected component of b in $G \setminus (a, c)$. Let H be obtained from G by contracting D to b . By Lemma 2, $tb(H) = 1$. Let (T, \mathcal{X}^H) be the tree decomposition of breadth one of H where for every $t \in V(T)$, $X_t^H = X_t$ if $X_t \cap D = \emptyset$, $X_t^H = (X_t \setminus D) \cup \{b\}$ else. Moreover, since b has no neighbour in C , $D \cap N_G[C] = \emptyset$ and so, $H[C \cup \Pi] = G[C \cup \Pi]$ by construction. Finally, since $\{b\}$ is a full component of $H \setminus (a, c)$, therefore, by Lemma 9 applied to H , the tree decomposition (T, \mathcal{X}^C) is indeed a tree decomposition of breadth one of $G[C \cup \Pi]$. ◇

Let (T', \mathcal{X}') be any reduced tree decomposition obtained from (T, \mathcal{X}^C) . We point out that $T'_a \cap T'_b \cap T'_c \neq \emptyset$ by construction (because $T_a \cap T_b \cap T_c \neq \emptyset$). Furthermore, since by Claim 2 (T, \mathcal{X}^C) has breadth one, therefore (T', \mathcal{X}') is a star-decomposition of $G[C \cup \Pi]$ by Lemma 3.

We will prove that C contains a leaf-vertex by contradiction. Informally, we will show, using the properties of the star-decomposition (T', \mathcal{X}') , that if it is not the case that C contains a leaf-vertex, then $\mathcal{P}_3(G[C \cup \Pi]) \cap \mathcal{P}_3(G) \neq \emptyset$ and the latter contradicts Claim 1.

In order to prove this, first note that a has at least one neighbour in C because G is prime by the hypothesis (indeed, (b, c) cannot be an edge-separator of G). We now distinguish between several subcases.

- **Case 1.** There is $u \in C$ such that $u \in N(a) \cap N(b) \cap N(c)$ (e.g., see Figure 16). By Lemma 15, either C is reduced to u or there exist $\Pi' = (a, u, c) \in \mathcal{P}_3(G)$, $C' \subseteq C \setminus u$ and C' is a component of $G \setminus \Pi'$. The latter case contradicts Claim 1, therefore, C is reduced to u and so u is a leaf-vertex of Type 2.

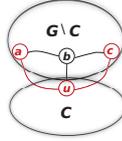


Figure 16: Case 1

Thus, from now on let us assume that no such vertex u exists.

- **Case 2.** By contradiction, assume $N(a) \cap C \subseteq N(b) \cap C$. By Lemma 14, $|N(a) \cap N(b) \cap C| \leq 1$, so, $|N(a) \cap C| = 1$. Let $u \in N(a) \cap N(b) \cap C$ be the unique neighbour of vertex a in C (see Figure 17). Since in such case we can assume that $u \notin N(c)$ (for otherwise, we are back to Case 1), and vertex c has some neighbour in C because G is prime (and so, (a, b) cannot be an edge-separator of G), therefore, C is not reduced to vertex u . Then, $\Pi' = (u, b, c) \in \mathcal{P}_3(G)$ because it separates a from $C \setminus u$, and so there is at least one component of $G \setminus \Pi'$ that is strictly contained into C by construction. This contradicts Claim 1, so, Case 2 cannot occur.

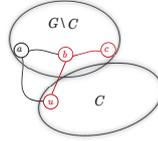


Figure 17: Case 2

- **Case 3.** There is $u \in C$ satisfying $u \in N(a) \setminus N(b)$. By the properties of a tree decomposition, there is some bag $B' \in T'_a \cap T'_u$. Let $v \in B'$ dominate the bag B' . By construction, $v \neq b$ because $u \in B' \setminus N(b)$, similarly $v \neq c$ because $a \in B' \setminus N(c)$. We will also prove later that $v \neq a$. Moreover, $\Pi \setminus N[v] \neq \emptyset$ (or else, we are back to Case 1), hence $\Pi \setminus B' \neq \emptyset$. So let B be the bag adjacent to B' onto the unique path in T' from B' to $T'_a \cap T'_b \cap T'_c$ (we remind that the latter subtree is nonempty by construction). By the properties of the tree decomposition (T', \mathcal{X}') , $B \cap B'$ is a separator of $G[C \cup \Pi]$. Furthermore, $a \in B \cap B'$. More generally $\Pi \cap B' \subseteq B \cap B'$ by construction, therefore $B \cap B'$ is also a separator of G by Lemma 16. Let $w \in B$ dominate this bag. Observe that $w \neq c$ because $a \in B \cap B'$.

We will prove that $v \in C$ and v is a leaf-vertex. In order to prove these two results, we will need to prove that $C \cup \Pi$ is fully contained into the two adjacent bags B, B' (Claim 7). The latter will require intermediate claims.

Claim 3 $c \in B \cap B'$.

Proof. Assume for the sake of contradiction that $c \notin B \cap B'$ (see Figure 18). Then, $c \notin B'$ because $\Pi \cap B' \subseteq B \cap B'$ by construction. We will prove that the latter contradicts Claim 1.

Indeed, first observe that $G \setminus w$ is connected because G is prime and so, biconnected, by the hypothesis. In addition $(B \cap B') \setminus w$ is a (not necessarily minimal) separator of $G \setminus w$ because it separates $B' \setminus B$ from c . Let $S \subseteq (B \cap B') \setminus w$ be a minimal separator of $G \setminus w$. By Lemma 13, there exist $x, y \in (B \cap B') \setminus w$ non-adjacent such that $S = \{x, y\}$, and so, $\Pi' = (x, w, y) \in \mathcal{P}_3(G)$. Note that $\Pi' \neq \Pi$, because we assume that $c \notin B \cap B'$ and so $c \notin \{x, y\}$. Moreover, since (T', \mathcal{X}') is a star-decomposition of $G[C \cup \Pi]$ by construction we have that $\Pi' \subseteq \Pi \cup C$, therefore $x \in C$ or $y \in C$, because $c \notin \{x, y\}$ and a, b are adjacent whereas x, y are non-adjacent. W.l.o.g. let $x \in C$.

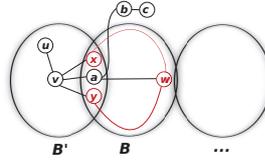


Figure 18: Case $c \notin B \cap B'$

Subclaim 3.1 Π' is not an ac -separator.

Proof. We refer to Figure 19 for an illustration of the proof. Let C' be any component of $G \setminus (\Pi \cup C)$. Observe that C' is fully contained into some component D of $G \setminus \Pi'$, because $\Pi' \subseteq \Pi \cup C$. In addition, $a, c \in N(C')$ because G is prime by the hypothesis (and so, neither a nor b nor c nor (a, b) nor (b, c) can be a separator of G). In particular, since we assume $c \notin B \cap B'$ and so, $c \notin \Pi'$, therefore, $c \in D$. As a result, either $a \in \Pi'$ or $a, c \in D$, that finally proves the subclaim. \circ

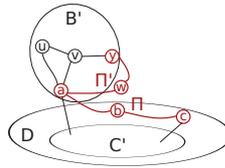


Figure 19: Π' is not an ac -separator.

Let D be the component of $G \setminus \Pi'$ such that $c \in D$, that exists because we assume $c \notin B \cap B'$ and so, $c \notin \Pi'$. Since b, c are adjacent and Π' is not an ac -separator by Claim 3.1, therefore, $\Pi \subseteq \Pi' \cup D$.

Moreover, let us show that Claim 3.1 implies the existence of some $D' \subset C$ being a component of $G \setminus \Pi'$, thus contradicting Claim 1. Indeed, let D' be any component of $G \setminus (\Pi' \cup D)$. Since G is prime by the hypothesis, x has some neighbour in D' and so, $D' \cap C \neq \emptyset$ because $x \in C$ and $\Pi \cap D' = \emptyset$ by construction. But then, $D' \subseteq C \setminus x$, for the existence of some $z \in D' \setminus C$ would imply that $D' \cap \Pi \neq \emptyset$.

To sum up, we conclude that it must be the case that $c \in B \cap B'$. \diamond

We will use Claim 3 to prove that $v \in C$, as follows:

Claim 4 $v \in C$. Furthermore, the two vertices b, v are non-adjacent.

Proof. Recall that $v \in C \cup \Pi$ because (T', \mathcal{X}') is a star-decomposition of $G[C \cup \Pi]$ by construction. So we will only need to prove that $v \notin \Pi$. First, since $a \in B \cap B'$ by construction and $c \in B \cap B'$ by Claim 3, therefore, $a, c \in B' \subseteq N[v]$. The latter implies that $v \notin \{a, c\}$ because $a, c \in N[v]$ whereas a, c are non-adjacent. Furthermore, this implies $b \notin N[v]$ because we assume that $\Pi \not\subseteq N[v]$ (for otherwise, we are back to Case 1). As a result, $v \notin \Pi$, whence $v \in C$. \diamond

Then, we will need the following technical claim in order to prove that $w = b$ (Claim 6).

Claim 5 $G[C \cup \Pi]$ is prime.

Proof. Suppose by contradiction there exists a clique-separator S of $G[C \cup \Pi]$. Then, S could not be a separator of G because G is prime by the hypothesis. By Lemma 16, the latter implies that S is an ac -separator of $G[C \cup \Pi]$. Therefore, the two vertices $b, v \in N(a) \cap N(c)$ must be in S , and so, since b, v are non-adjacent by Claim 4, the latter contradicts the fact that S is a clique. \diamond

Claim 6 $w = b$.

Proof. Assume for the sake of contradiction $w \neq b$ (see Figure 20). We will prove that it contradicts Claim 1.

Indeed, the graph $G[C \cup \Pi] \setminus w$ is connected because $G[C \cup \Pi]$ is prime by Claim 5 and so, biconnected. In addition, $(B \cap B') \setminus w$ is a (not necessarily minimal) separator of $G[C \cup \Pi] \setminus w$ because it separates b from $B' \setminus B$ (recall that b, v are non-adjacent by Claim 3, and so, $b \notin B' \subseteq N[v]$). Let $S \subseteq (B \cap B') \setminus w$ be a minimal separator of $G[C \cup \Pi] \setminus w$. By Lemma 13, there exist $x, y \in (B \cap B') \setminus w$ non-adjacent such that $S = \{x, y\}$, and so, $\Pi' = (x, w, y) \in \mathcal{P}_3(G[C \cup \Pi])$. Furthermore, $b \notin \Pi' \subseteq (B \cap B') \cup \{w\} \subseteq N[v] \cup \{w\}$ and so, Π' cannot be an ac -separator of $G[C \cup \Pi]$, whence by Lemma 16 Π' is a separator of G , and so, $\Pi' \in \mathcal{P}_3(G)$.

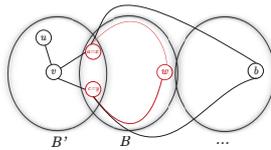


Figure 20: Case $c \in B \cap B'$ and $w \neq b$.

Let $D \subseteq C \cup \Pi$ be the component of $G[C \cup \Pi] \setminus \Pi'$ containing vertex b . Note that $\Pi \subseteq D \cup \Pi'$ because Π' is not an ac -separator of $G[C \cup \Pi]$. Let D' be any component of $G[C \cup \Pi] \setminus (\Pi' \cup D)$, that exists because $\Pi' \in \mathcal{P}_3(G[C \cup \Pi])$. Since $N(D') \subseteq C \cup \Pi$ by construction, $D' \cap \Pi = \emptyset$ by construction and D' is a component of $G[C \cup \Pi] \setminus \Pi'$, therefore, D' is also a component of $G \setminus \Pi'$. The latter contradicts Claim 1 because $D' \subset C$. \diamond

Let $S = \{v, b\} \cup (B \cap B')$. We are now able to prove that $S = C \cup \Pi$ (Claim 7). That is, $C \cup \Pi$ is fully contained in the two adjacent bags B, B' (respectively dominated by b, v).

Claim 7 $S = C \cup \Pi$.

Proof. Assume by contradiction $S \neq C \cup \Pi$, let D be a component of $G[C \cup \Pi] \setminus S$ (see Figure 21). Note that $D \subset C$ because $\Pi \subset S$ by construction. Furthermore, $v, b \notin B \cap B'$ because $w = b$ by Claim 6 and $b \notin N(v)$ by Claim 3, so, $B \cap B'$ is a (minimal) vb -separator of $G[C \cup \Pi]$. The latter implies $v \notin N(D)$ or $b \notin N(D)$ because D induces a connected subgraph, $D \cap B \cap B' = \emptyset$ by construction, and $B \cap B'$ is a bv -separator of $G[C \cup \Pi]$. As a result, there exists $z \in \{v, b\}$ such that $N(D) \setminus z \subseteq B \cap B'$.

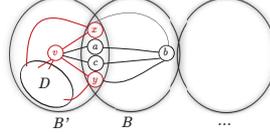


Figure 21: Case $z = v$.

Moreover let $\{z, z'\} = \{v, b\}$. $G[C \cup \Pi] \setminus z$ is connected because $G[C \cup \Pi]$ is prime by Claim 5, and so, biconnected. In addition, $N(D) \setminus z$ is a minimal separator of $G[C \cup \Pi] \setminus z$ because it separates D from z' and $N(D) \setminus z \subseteq B \cap B' \subseteq N(z')$ by construction. By Lemma 13, one obtains the existence of two non-adjacent vertices $x, y \in B \cap B'$ such that $N(D) \setminus z = \{x, y\}$, whence $N(D) \subseteq \{x, y, z\}$. Then, by construction $\Pi' = (x, z, y) \in \mathcal{P}_3(G)$ with $D \subset C$ being a component of $G \setminus \Pi'$, that contradicts Claim 1. \diamond

By Claim 7, $C \cup \Pi = S$ (see Figure 22). Note that it implies that $C \subseteq N[v]$ because $C \setminus v = (B \cap B') \setminus (a, c)$. In order to conclude that v is a leaf-vertex, we will finally prove in Claim 8 that either $B \cap B' = \{a, c\}$ or $B \cap B'$ induces a path.

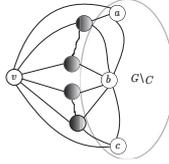


Figure 22: Case $c \in B \cap B'$, $w = b$ and $C \cup \Pi = \{v, b\} \cup (B \cap B')$.

Claim 8 *If $B \cap B' \neq \{a, c\}$, then $G[B \cap B']$ is a path.*

Proof. Recall that $b, v \notin B \cap B'$ because $w = b$ by Claim 6 and $b \notin N(v)$ by Claim 3. Hence by the properties of a tree decomposition, $B \cap B'$ is a bv -separator of $G[C \cup \Pi]$. Since $v \in C$ by Claim 4, $a \in B \cap B'$ by construction and $c \in B \cap B'$ by Claim 3, therefore $B \cap B'$ is also a vb -separator of G . In particular, $B \cap B'$ is a minimal bv -separator of G because $B \cap B' \subseteq N(v) \cap N(w) = N(v) \cap N(b)$ (indeed, recall that $w = b$ by Claim 6). By Corollary 6, $B \cap B'$ either induces a cycle or it induces a forest of paths.

Subclaim 8.1 *$B \cap B'$ does not induce a cycle.*

Proof. By contradiction, let $B \cap B'$ induce a cycle. Recall that $B \cap B'$ contains the pair of non-adjacent vertices a, c (because $a \in B \cap B'$ by construction and $c \in B \cap B'$ by Claim 3). Therefore, one can contract $B \cap B'$ until one obtains an induced quadrangle (a, x, c, y) . Let us contract an arbitrary component of $G \setminus (\Pi \cup C)$ so as to obtain a vertex z . Note that $a, c \in N(z)$ because G is prime by the hypothesis (indeed, neither a nor b nor c nor (a, b)

nor (a, c) can be a separator of G). Then, let us contract the edge $\{a, z\}$ to a . By doing so, one obtains a $K_{3,3}$ -minor with $\{a, b, v\}$ being one part of the bipartition and $\{x, y, c\}$ being the other part. This contradicts the fact that G is planar by the hypothesis, therefore $B \cap B'$ does not induce a cycle. \circ

It follows from Claim 8.1 that $B \cap B'$ induces a forest of paths. Suppose for the sake of contradiction that $B \cap B'$ induces a forest of at least two paths. Let $x \notin \{a, c\}$ be the endpoint of some path in the forest, that exists because we assume that $B \neq \{a, c\}$. Observe that $|N(x)| \geq 2$ because $b, v \in N(x)$, and $|N(x)| = |N(x) \cap (C \cup \Pi)| \leq 3$ because x is the endpoint of some path of $B \cap B'$ and $x \in C$. Furthermore, $N(x) \setminus (b, v) \subseteq B \cap B' \subseteq N(b) \cap N(v)$, and so, if $|N(x)| = 3$ then $N(x)$ induces a path. Let $\Pi' = N(x)$ if $|N(x)| = 3$, else $\Pi' = (b, a, v)$. By construction, $\Pi' \subseteq \Pi \cup C$ is a separator of G with $\{x\} \subset C$ being a component of $G \setminus \Pi'$, thus contradicting Claim 1. Consequently, $B \cap B'$ induces a path. \diamond

By Claim 8, either $B \cap B' = \{a, c\}$ or $B \cap B'$ induces a path. Furthermore, $B \cap B' = N(v)$ because $v \in C$ (Claim 4) and $C \cup \Pi = \{v, b\} \cup (B \cap B')$ (Claim 7). In particular, if $B \cap B' = \{a, c\}$ then v is a leaf-vertex of Type 3. Else, $B \cap B'$ induces a path and the latter implies that $|B \cap B'| \geq 4$ or else the path $B \cap B'$ would be a separator of G with $\{v\}$ being a component of $G \setminus (B \cap B')$, thus contradicting Claim 1. As a result, since we also have that $B \cap B' \subseteq N(b)$ and b, v are non-adjacent by Claim 4, therefore, v is a leaf-vertex of Type 1.

Case $T_a \cap T_c = \emptyset$. Since Π is a separator of G and G is prime by the hypothesis, one of Π or $\Pi \setminus b$ must be a minimal separator of G . Therefore, since (T, \mathcal{X}) is assumed to minimize the distance in T between T_a and T_c , by Corollary 8 there exist two bags B_a, B_c that are adjacent in T and such that $a \in B_a \setminus B_c$ and $c \in B_c \setminus B_a$. Furthermore, a dominates B_a while c dominates B_c . Note that $B_a \cap B_c = N(a) \cap N(c)$, so, $b \in B_a \cap B_c$. In particular, by the properties of a tree decomposition this implies that $S = N(a) \cap N(c)$ is a minimal ac -separator of G .

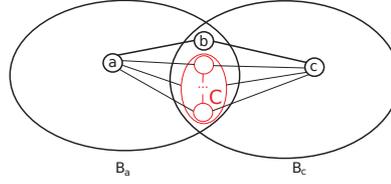
We will prove that C is reduced to a vertex (Claim 10), the latter being a leaf-vertex.

Claim 9 $C \subseteq S$.

Proof. Assume for the sake of contradiction that $C \not\subseteq B \cap B'$. By the properties of a tree decomposition it comes that some vertex $y \in C$ is separated from a or c by the set $S = B \cap B' = N(a) \cap N(c)$. Say w.l.o.g. that S is an yc -separator. Let $C' \subset C$ be the connected component containing y in $G \setminus (S \cup \{a\})$. Since we have that $G \setminus a$ is connected because G is prime by the hypothesis (and so, biconnected), that $c \notin C'$ and $N(C') \setminus a \subseteq S \cap (C \cup \Pi) \subseteq N(c) \cap (C \cup \Pi)$, then it comes that $N(C') \setminus a$ is a minimal yc -separator of $G \setminus a$. So, by Lemma 13 there exist $x', y' \in S$ such that $N(C') \setminus a = \{x', y'\}$. Therefore, $\Pi' = (x', a, y') \in \mathcal{P}_3(G)$ and $C' \subset C$ is a component of $G \setminus \Pi'$, that contradicts Claim 1. \diamond

By Claim 9, $C \subseteq S$ (see Figure 23 for an illustration). Since S is an ac -separator and for any component C' of $G \setminus (\Pi \cup C)$, $a, c \in N(C')$ because G is prime, therefore $S \cap C' \neq \emptyset$. One thus obtains the following chain of strict subset containment relations $C \subset C \cup \{b\} \subset S$. Furthermore, by Corollary 6, S either induces a cycle or a forest of paths, so, C being a strict connected subset of S , it must induce a path. In particular, $C \cup \{b\}$ also being a strict subset of S , either it induces a path or it is the union of the path induced by C with the isolated vertex b .

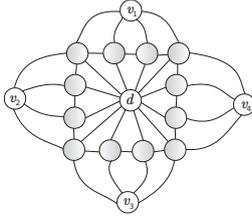
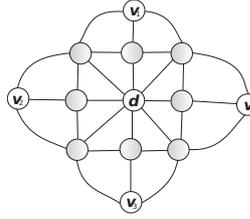
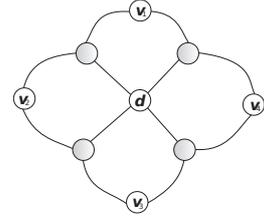
Claim 10 $|C| = 1$.

Figure 23: Case $T_a \cap T_c = \emptyset$.

Proof. Assume for the sake of contradiction that $|C| \geq 2$. Since C induces a path, let us pick an endpoint $v \in C$ that is not adjacent to vertex b (recall that $C \cup \{b\}$ being a strict subset of S , it does not induce a cycle). In such a case, $N(v)$ induces a path $\Pi' \in \mathcal{P}_3(G)$, with $a, c \in \Pi'$ and $\{v\} \subset C$ is a component of $G \setminus \Pi'$, thus contradicting Claim 1. \diamond

By Claim 10, C is reduced to a vertex v , that is either a leaf-vertex of Type 2 (if $v \in N(b)$) or of Type 3 (if $v \notin N(b)$). \blacksquare

Note that in some cases, there may only exist leaf-vertices of only one Type (*i.e.*, see respectively Figure 24, 25 and 26 for Types 1, 2 and 3). Therefore, there is none of the three Types of leaf-vertices that can be avoided in our algorithm.

Figure 24: A planar graph G with $tb(G) = 1$ and all of its four leaf-vertices v_1, v_2, v_3, v_4 of Type 1.Figure 25: A planar graph G with $tb(G) = 1$ and all of its four leaf-vertices v_1, v_2, v_3, v_4 of Type 2.Figure 26: A planar graph G with $tb(G) = 1$ and all of its four leaf-vertices v_1, v_2, v_3, v_4 of Type 3.

Examples of planar graphs G with $tb(G) = 1$ and $\mathcal{P}_3(G) = \emptyset$ include C_4 , the cycle with four vertices. To prove correctness of Step 1, it now suffices to prove that all these graphs (with $\mathcal{P}_3(G) = \emptyset$) admit a star-decomposition with at most two bags.

Lemma 20 *For any prime planar graph G , if $tb(G) = 1$ and $\mathcal{P}_3(G) = \emptyset$, then G admits a star-decomposition with at most 2 bags.*

Proof. By contradiction, let (T, \mathcal{X}) be a star-decomposition of G with at least three bags. Let $t \in V(T)$ be an internal node, by the properties of a tree decomposition the bag X_t is a separator of G . Let $u \in X_t$ satisfy $X_t \subseteq N_G[u]$. Since G is biconnected, therefore $X_t \setminus u$ is a separator of $G \setminus u$. By Lemma 13, there exist $x, y \in X_t \setminus u$ non-adjacent such that $\{x, y\}$ is a minimal separator of $G \setminus u$. In such case, $(x, u, y) \in \mathcal{P}_3(G)$, which contradicts the fact that $\mathcal{P}_3(G) = \emptyset$. \blacksquare

5.3.2 Case of leaf-vertex v of Type 1

Lemma 21 *Let G be a prime planar graph and v be a leaf-vertex of Type 1. Let Π_v be the path induced by $N(v)$ and let a_v, c_v be the ends of Π_v . Suppose $V(G) \neq N[v] \cup \{d_v\}$.*

Then $\Pi' = (a_v, d_v, c_v) \in \mathcal{P}_3(G)$ and $N[v] \setminus \{a_v, c_v\}$ is a component of $G \setminus \Pi'$.

Proof. Let C be a component of $G \setminus (N[v] \cup \{d_v\})$, that exists by the hypothesis. By construction, $v \notin N[C]$, so, $N(C) \subseteq N(v) \cup \{d_v\}$ separates v from C . Furthermore, since G is prime by the hypothesis, there exist $x, y \in N(C)$ non-adjacent. Note that $d_v \notin \{x, y\}$ because $N(v) \subseteq N(d_v)$ by the hypothesis, hence $x, y \in N(v)$.

We claim that $\{x, y\} = \{a_v, c_v\}$. By contradiction, suppose $x \notin \{a_v, c_v\}$. Let us write $\Pi_v = (P, x, Q, y, R)$ with P, Q non-empty subpaths of Π_v and R a (possibly empty) subpath of Π_v . In such a case, the connected subsets $S_1 := \{v\} \cup P$, $S_2 := \{d_v\}$, $S_3 := \{x\}$, $S_4 := Q$ and $S_5 := \{y\} \cup C$ induce a K_5 -minor of G , that contradicts the hypothesis that G is planar. Therefore, the claim is proved, that is, $\{x, y\} = \{a_v, c_v\}$.

To prove the lemma, it now suffices to prove that $N(C) \cap N(v) = \{a_v, c_v\}$ for in such a case the result will hold for any component C' of $G \setminus (N[v] \cup \{d_v\})$. By contradiction, let $x' \in (N(C) \cap N(v)) \setminus \{a_v, c_v\}$. Since $|N(v)| \geq 4$ because v is a leaf-vertex of Type 1 by the hypothesis, therefore, x' and a_v are non-adjacent or x' and c_v are non-adjacent. Let $y' \in \{a_v, c_v\}$ be non-adjacent to x' . Since $x', y' \in N(C) \cap N(v)$ are non-adjacent, therefore, by the same proof as for the above claim $\{x', y'\} = \{a_v, c_v\}$, that would contradict the assumption that $x' \notin \{a_v, c_v\}$. As a result, $N(C) \subseteq \{a_v, d_v, c_v\}$ and so, since the result holds for any component C' of $G \setminus (N[v] \cup \{d_v\})$, $\Pi' = (a_v, d_v, c_v) \in \mathcal{P}_3(G)$ with $N[v] \setminus \{a_v, c_v\}$ being a full component of $G \setminus \Pi'$. ■

Theorem 8 *Let G be a prime planar graph and v be a leaf-vertex of Type 1. Let Π_v be the path induced by $N(v)$ and let a_v, c_v be the ends of Π_v . Suppose $V(G) \neq N[v] \cup \{d_v\}$.*

Then, the graph G' , obtained from $G \setminus v$ by contracting the internal vertices of Π_v to a single edge, is prime and planar, and $tb(G) = 1$ if and only if $tb(G') = 1$.

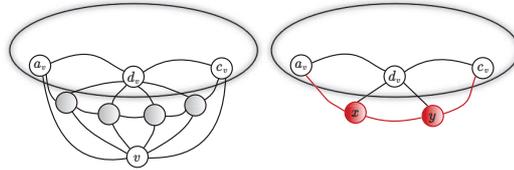


Figure 27: Contraction of the internal vertices of Π_v to a single edge and removal of v .

Proof. For the remaining of the proof, let $\Pi'_v = (a_v, x, y, c_v)$ be the path resulting from the contraction of the internal vertices of Π_v to the edge $\{x, y\}$ in G' . By Lemma 21 $(a_v, d_v, c_v) \in \mathcal{P}_3(G)$ with $(N[v] \setminus \{a_v, c_v\})$ being a full component of $G \setminus (a_v, d_v, c_v)$. Consequently, $N_{G'}(x) = \{a_v, d_v, y\}$ and $N_{G'}(y) = \{c_v, d_v, x\}$.

The graph G' is a minor of G , that is a planar graph by the hypothesis, so, G' is also planar. In order to prove that G' is prime, by contradiction, let S be a minimal clique-separator of G' . There are two cases to be considered.

- Suppose $x \in S$ or $y \in S$. In such case, $S \subseteq (a_v, x, d_v)$, or $S \subseteq (x, d_v, y)$, or $S \subseteq (y, d_v, c_v)$. By Lemma 21 $(a_v, d_v, c_v) \in \mathcal{P}_3(G)$ with $(N[v] \setminus \{a_v, c_v\})$ being a full component of $G \setminus (a_v, d_v, c_v)$, and so, for every component C of $G' \setminus (\Pi'_v \cup \{d_v\}) = G \setminus (N[v] \cup \{d_v\})$ $a_v, c_v \in N(C)$ because G is prime by the hypothesis. In such case, since $a_v \notin S$ or $c_v \notin S$, therefore, $G' \setminus S$ is connected, that contradicts the assumption that S is a clique-separator of G' .
- Else, $x, y \notin S$. Since $a_v \notin S$ or $c_v \notin S$ because a_v and c_v are non-adjacent in G' , therefore, S must be a separator of $G' \setminus (x, y)$ or else $G' \setminus S$ would be connected because Π'_v induces a path of G' (thus contradicting the assumption that S is a separator of G'). In such a

case, since by Lemma 21 $(a_v, d_v, c_v) \in \mathcal{P}_3(G)$ with $(N[v] \setminus (a_v, c_v))$ being a full component of $G \setminus (a_v, d_v, c_v)$, since S is a separator of $G' \setminus (x, y) = G \setminus (N[v] \setminus (a_v, c_v))$ and since S is not a separator of G because G is prime by the hypothesis, therefore, by Lemma 16 there are exactly two components C_a, C_c in $G' \setminus (S \cup \{x, y\})$ with $a_v \in C_a$ and $c_v \in C_c$. However, Π'_v is an $a_v c_v$ -path of $G' \setminus S$, thus contradicting the assumption that S is a separator of G' .

As a result, G' is a prime planar graph.

Finally, let us prove $tb(G) = 1$ if and only if $tb(G') = 1$.

- If $tb(G) = 1$ then $tb(G \setminus v) = 1$ because $N(v) \subseteq N(d_v)$ by the hypothesis, and so, $tb(G') = 1$ because G' is obtained from $G \setminus v$ by edge-contractions and tree-breadth is contraction-closed (Lemma 2).
- Conversely, let us prove that $tb(G) = 1$ if $tb(G') = 1$. To prove this, let (T', \mathcal{X}') be a reduced star-decomposition of G' , that exists by Lemma 3, minimizing the distance in T' between the two subtrees T'_{a_v} and T'_{c_v} . In order to prove $tb(G) = 1$, it suffices to show how to construct a star-decomposition of G from (T', \mathcal{X}') .

We will prove as an intermediate claim that $T'_{a_v} \cap T'_{c_v} \neq \emptyset$. By contradiction, suppose $T'_{a_v} \cap T'_{c_v} = \emptyset$. Since by Lemma 21 $(a_v, d_v, c_v) \in \mathcal{P}_3(G)$ with $(N[v] \setminus (a_v, c_v))$ being a full component of $G \setminus (a_v, d_v, c_v)$, therefore, $(a_v, d_v, c_v) \in \mathcal{P}_3(G')$ with $\{x, y\}$ being a full component of $G' \setminus (a_v, d_v, c_v)$. Since we proved that G' is prime, it follows that one of (a_v, c_v) or (a_v, d_v, c_v) is a minimal separator of G' . In such a situation, since (T', \mathcal{X}') is assumed to minimize the distance in T' between T'_{a_v} and T'_{c_v} , therefore, by Corollary 8 there are two adjacent bags B'_{a_v}, B'_{c_v} such that $a_v \in B'_{a_v} \setminus B'_{c_v}$ and $c_v \in B'_{c_v} \setminus B'_{a_v}$ respectively dominate B'_{a_v} and B'_{c_v} in G' . However by the properties of a tree decomposition this implies that $B'_{a_v} \cap B'_{c_v} = N(a_v) \cap N(c_v)$ is an $a_v c_v$ -separator of G' , thus contradicting the existence of the $a_v c_v$ -path Π'_v . Therefore, the claim is proved and $T'_{a_v} \cap T'_{c_v} \neq \emptyset$.

Recall that $T'_{a_v} \cap T'_{d_v} \neq \emptyset$ and similarly $T'_{c_v} \cap T'_{d_v} \neq \emptyset$ by the properties of a tree decomposition. Hence, the subtrees $T'_{a_v}, T'_{c_v}, T'_{d_v}$ are pairwise intersecting, and so, by the Helly property (Lemma 1), $T'_{a_v} \cap T'_{d_v} \cap T'_{c_v} \neq \emptyset$. Let us now proceed as follows so as to obtain a star-decomposition of G . Let us remove x, y from all bags in \mathcal{X}' , that keeps the property for (T', \mathcal{X}') to be a star-decomposition because x and y are dominated by d_v in G' . Then, let us add two new bags $B_1 = N[v]$, $B_2 = N(v) \cup \{d_v\}$, and finally let us make B_1, B_2 pairwise adjacent and let us make B_2 adjacent to some bag of $T'_{a_v} \cap T'_{d_v} \cap T'_{c_v}$. By construction, the resulting tree decomposition is indeed a star-decomposition of G , whence $tb(G) = 1$.

■

5.3.3 Proof of Step 3.1 (a)

In the following three subsections (5.3.3, 5.3.4 and 5.3.5) we will prove correctness of the algorithm for the case of a leaf-vertex v of Type 2 or 3 and $G \setminus v$ is prime (Step 3.1). Our proofs in these subsections will mostly rely on Lemma 17.

Let us first show how we can use Lemma 17 in order to prove correctness of Step 3.1 (a). Note that since we are in the case when $G \setminus v$ is prime, we needn't prove it in the following Theorem 9.

Theorem 9 *Let $G = (V, E)$ be a prime planar graph, let v be a leaf-vertex of Type 2 or 3, and let $\Pi_v = (a_v, b_v, c_v)$ be as in Definition 5.*

Suppose that $|N(a_v) \cap N(c_v)| \geq 3$ in $G \setminus v$, or there exists a minimal separator $S \subseteq (N(a_v) \cap N(c_v)) \cup \{a_v, c_v\}$ in $G \setminus v$ and $\{a_v, c_v\} \subseteq S$.

Then, $tb(G) = 1$ if and only if $tb(G \setminus v) = 1$.

Proof. First we prove that $tb(G) = 1$ implies that $tb(G \setminus v) = 1$, that is the easy part of the result. Let (T, \mathcal{X}) be a tree decomposition of G of breadth one, let (T, \mathcal{X}') be such that for every node $t \in V(T)$, $X'_t = X_t \setminus v$. Observe that (T, \mathcal{X}') is a tree decomposition of $G \setminus v$. Furthermore, we claim that it has breadth one, indeed, for every $t \in V(T)$ such that $X_t \subseteq N_G[v]$, $X'_t \subseteq N_G[b_v]$ because $N_G(v) \subseteq N_G[b_v]$. As a result, $tb(G \setminus v) = 1$.

Conversely, we prove that $tb(G \setminus v) = 1$ implies that $tb(G) = 1$. Let (T', \mathcal{X}') be a star-decomposition of $G \setminus v$ minimizing the distance in T' between the subtrees T'_{a_v} and T'_{c_v} . There are two cases. If $T'_{a_v} \cap T'_{c_v} \neq \emptyset$, then the subtrees $T'_{a_v}, T'_{b_v}, T'_{c_v}$ are pairwise intersecting, hence by the Helly property (Lemma 1) $T'_{a_v} \cap T'_{b_v} \cap T'_{c_v} \neq \emptyset$, and so it suffices to make adjacent to any bag of $T'_{a_v} \cap T'_{b_v} \cap T'_{c_v}$ the new bag $N_G[v] \subseteq \{a_v, b_v, c_v, v\}$ so as to obtain a star-decomposition of G . Else $T'_{a_v} \cap T'_{c_v} = \emptyset$ and so, by Corollary 9 if $|N(a_v) \cap N(c_v)| \geq 3$ in $G \setminus v$ or by Corollary 8 else, there are two adjacent bags B'_{a_v}, B'_{c_v} such that $a_v \in B'_{a_v} \setminus B'_{c_v}, b_v \in B'_{a_v} \cap B'_{c_v} \subseteq N(a_v) \cap N(c_v)$ and $c_v \in B'_{c_v} \setminus B'_{a_v}$. Furthermore, a_v dominates B'_{a_v} while c_v dominates B'_{c_v} . One obtains a star-decomposition of G simply by adding vertex v into bags B'_{a_v} and B'_{c_v} . ■

5.3.4 Proof of Step 3.1 (b) i

The proof of this step is more involved than the proof of previous Step 3.1 (a). We will need the following intermediate lemma.

Lemma 22 *Let $G = (V, E)$ be a prime graph with $tb(G) = 1$, let v be a leaf-vertex of Type 2 or 3 and let $\Pi_v = (a_v, b_v, c_v)$ be as in Definition 5. Suppose that $N(a_v) \cap N(c_v) = \{v, b_v\}$ and $V \neq \Pi_v \cup \{v\}$. Then, $N_G[b_v] \setminus (a_v, c_v, v)$ is an $a_v c_v$ -separator of $G \setminus v$.*

Proof. Let (T, \mathcal{X}) be a star-decomposition of G , that exists by Lemma 3, minimizing the distance in T between the subtrees T_{a_v} and T_{c_v} . We claim that $T_{a_v} \cap T_{c_v} \neq \emptyset$, i.e., a_v, c_v are in a same bag of the decomposition. By contradiction, let $T_{a_v} \cap T_{c_v} = \emptyset$. Since G is prime and Π_v is a separator of G , therefore, one of Π_v or $\Pi_v \setminus b_v$ is a minimal separator of G . Since (T, \mathcal{X}) minimizes the distance in T between T_{a_v} and T_{c_v} , therefore, by Corollary 8 there exist two adjacent bags B_{a_v}, B_{c_v} such that $a_v \in B_{a_v} \setminus B_{c_v}$ and $c_v \in B_{c_v} \setminus B_{a_v}$. Furthermore, vertices a_v and c_v respectively dominate the bags B_{a_v} and B_{c_v} . This implies $B_{a_v} \cap B_{c_v} = N_G(a_v) \cap N_G(c_v)$ and so, $N_G(a_v) \cap N_G(c_v)$ is a minimal $a_v c_v$ -separator of G by the properties of the tree decomposition. However, let C be any component of $G \setminus (\Pi_v \cup \{v\})$, that exists because $V \neq \Pi_v \cup \{v\}$ by the hypothesis. Since G is prime, therefore, $a_v, c_v \in N(C)$ (or else, one of the cliques a_v or b_v or c_v or (a_v, b_v) or (b_v, c_v) would be a clique-separator of G , thus contradicting the assumption that G is prime). Then, the $a_v c_v$ -separator $N_G(a_v) \cap N_G(c_v)$ must contain some vertex of C , which contradicts the fact that $N_G(a_v) \cap N_G(c_v) = \{v, b_v\}$ by the hypothesis. As a result, we proved that $T_{a_v} \cap T_{c_v} \neq \emptyset$.

Let H be the chordal supergraph of G such that (T, \mathcal{X}) is a clique-tree of H . Equivalently, every two vertices $x, y \in V$ are adjacent in H if and only if they are in a same bag of \mathcal{X} . In particular, a_v, c_v are adjacent in H . Let $S := N_H(a_v) \cap N_H(c_v)$. We claim that S is an $a_v c_v$ -separator of G . By contradiction, if it is not an $a_v c_v$ -separator of G , then there exists an $a_v c_v$ -path $P_{a_v c_v}$ of G which does not intersect S . Furthermore, $P_{a_v c_v}$ is a path of H because H is a supergraph of G , and it has length at least two because a_v, c_v are non-adjacent in G . So, let $Q_{a_v c_v}$ be taken of minimum length amongst all $a_v c_v$ -paths of length at least two in H that do not intersect S (the existence of such a path follows from the existence of $P_{a_v c_v}$). Observe that

$Q_{a_v c_v}$ may be not a path in G . By minimality of $Q_{a_v c_v}$, the vertices of $Q_{a_v c_v}$ induce a cycle of H because a_v, c_v are adjacent in H . Therefore, the vertices of $Q_{a_v c_v}$ induce a triangle because H is chordal. However, this contradicts the fact that $Q_{a_v c_v}$ does not intersect $S = N_H(a_v) \cap N_H(c_v)$, so, the claim is proved.

Finally, let us prove that $S \setminus v \subseteq N_G[b_v] \setminus (a_v, c_v, v)$, that will conclude the proof that $N_G[b_v] \setminus (a_v, c_v, v)$ is an $a_v c_v$ -separator of $G \setminus v$. For every vertex $x \in S \setminus v$, $x \in N_H(a_v) \cap N_H(c_v)$, therefore, $T_{a_v} \cap T_x \neq \emptyset$ and $T_{c_v} \cap T_x \neq \emptyset$ by construction of H . Since the subtrees T_{a_v}, T_{c_v}, T_x are pairwise intersecting, by the Helly property (Lemma 1) $T_{a_v} \cap T_{c_v} \cap T_x \neq \emptyset$, or equivalently there is some bag $B \in T_{a_v} \cap T_{c_v} \cap T_x$. Let $z \in B$ dominate the bag. Clearly, $x \in N_G[z]$. Furthermore, $z \in N_G(a_v) \cap N_G(c_v)$ because a_v, c_v are non-adjacent in G . As a result, either $z = b_v$ or $z = v$. Since $x \neq v$ and $N_G(v) \subseteq N_G[b_v]$, we have that $x \in N_G[b_v]$ in both cases. ■

Theorem 10 *Let $G = (V, E)$ be a prime planar graph with $tb(G) = 1$, v be a leaf-vertex of Type 2 or 3, and let $\Pi_v = (a_v, b_v, c_v)$ be as in Definition 5. Suppose $N(a_v) \cap N(c_v) = \{b_v, v\}$, and $G \setminus v$ is prime, and there is no minimal separator $S \subseteq (N(a_v) \cap N(c_v)) \cup \{a_v, c_v\}$ in $G \setminus v$ such that $\{a_v, c_v\} \subseteq S$. Then, $G = C_4$, a cycle with four vertices.*

Proof. By contradiction, assume $G \neq C_4$. Since G is prime by the hypothesis, G has at least five vertices (the single other graph with four vertices and a leaf-vertex of Type 2 or 3 is the diamond, which is not prime). Equivalently, $V \neq \Pi_v \cup \{v\}$. By Lemma 22, this implies that $N[b_v] \setminus (a_v, c_v, v)$ is an $a_v c_v$ -separator of $G \setminus v$. Since $G \setminus v$ is prime by the hypothesis, and so, biconnected, therefore, $G \setminus (b_v, v)$ is connected, and so, $N(b_v) \setminus (a_v, c_v, v) \neq \emptyset$ is an $a_v c_v$ -separator of $G \setminus (b_v, v)$. In particular, $a_v, b_v, c_v \in N(V \setminus (\Pi_v \cup \{v\}))$.

Moreover, we claim that $V \setminus (\Pi_v \cup \{v\})$ induces a connected subgraph (note that the latter implies that $V \setminus (\Pi_v \cup \{v\})$ is a full component of $G \setminus \Pi_v$). By contradiction, let C_1, C_2 be distinct components of $V \setminus (\Pi_v \cup \{v\})$. Since G is prime, $a_v, c_v \in N_G(C_1) \cap N_G(C_2)$ (or else, one of the cliques a_v or b_v or c_v or (a_v, b_v) or (b_v, c_v) would be a clique-separator of G , thus contradicting the assumption that G is prime). Therefore, $b_v \in N_G(C_1) \cap N_G(C_2)$ because $N[b_v] \setminus (a_v, c_v, v)$ is an $a_v c_v$ -separator of $G \setminus v$. It follows that Π_v is a minimal separator of $G \setminus v$, that contradicts the hypothesis that there is no minimal separator $S \subseteq (N(a_v) \cap N(c_v)) \cup \{a_v, c_v\}$ in $G \setminus v$ and $\{a_v, c_v\} \subseteq S$. Consequently, $V \setminus (\Pi_v \cup \{v\})$ induces a connected subgraph.

Let $S' \subseteq N(b_v) \setminus (a_v, c_v, v)$ be a minimal $a_v c_v$ -separator of $G \setminus (b_v, v)$. By Lemma 13, there exist $x, y \in N(b_v) \setminus (a_v, c_v, v)$ non-adjacent such that $S' = \{x, y\}$. Finally, let $\Pi' = (x, b_v, y)$ and let A, C be the respective components of a_v, c_v in $G \setminus (\Pi' \cup \{v\})$. Note that $x, y \in N(A) \cap N(C)$ because $G \setminus v$ is prime by the hypothesis (indeed, neither x nor b_v nor y nor (b_v, x) nor (b_v, y) can be a separator of $G \setminus v$). Let P be an xy -path of $V \setminus (\Pi_v \cup \{v\})$, that exists because $V \setminus (\Pi_v \cup \{v\})$ is connected. Also, let $A' \subseteq A$ and $C' \subseteq C$ be the respective components of a_v, c_v in $G \setminus (P \cup \Pi' \cup \{v\})$. Note that the subpath $P \setminus (x, y)$ lies onto a unique component of $G \setminus (\Pi' \cup \{v\})$ because it does not intersect $\Pi_v \cup \{v\}$ by construction, so, $A' = A$ or $C' = C$. By symmetry, assume that $C' = C$. There are two cases to consider.

- Assume $A' = A$ (see Figure 28 for an illustration). Let us contract the internal vertices of P so as to make vertices x, y adjacent. Then, let us contract the components A, C to the two vertices a_v, c_v , respectively. Finally, let us contract v to either a_v or c_v . By construction, the five vertices a_v, b_v, c_v, x, y now induce a K_5 , that contradicts the fact that G is planar by the hypothesis.
- Else, $A' \neq A$. Equivalently, $P \subseteq A \cup \{x, y\}$ (see Figure 29 for an illustration). Since A is connected, $N(A') \cap (P \setminus (x, y)) \neq \emptyset$. Let $z \in N(A') \cap P$. Let us contract the internal vertices of P to vertex z . Then, let us contract the components A' and $C' = C$ to the two vertices

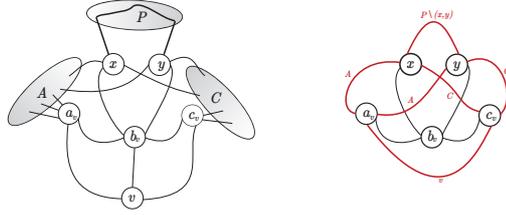


Figure 28: Case $A' = A$ (left). A K_5 -minor is drawn (right), with edges resulting from contractions labeled in red.

a_v, c_v , respectively. Finally let us contract v to either a_v or c_v . By construction, there is a $K_{3,3}$ -minor whose sides of the bipartition are $\{a_v, x, y\}$ and $\{b_v, c_v, z\}$, respectively, that contradicts the fact that G is planar by the hypothesis.

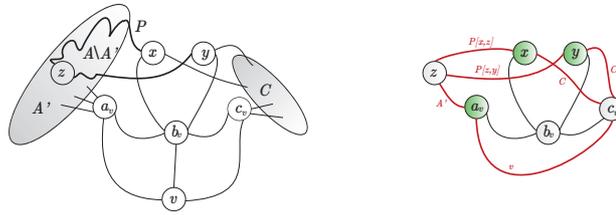


Figure 29: Case $A' \neq A$ (left). A $K_{3,3}$ -minor is drawn (right), with each side of the bipartition being coloured differently. Edges resulting from contractions are labeled in red.

Since both cases contradict the hypothesis that G is planar, therefore, $G = C_4$. ■

5.3.5 Proof of Step 3.1 (b) ii

Theorem 11 *Let $G = (V, E)$ be a prime planar graph, let v be a leaf-vertex of Type 2 or 3, and let $\Pi_v = (a_v, b_v, c_v)$ be as in Definition 5.*

Suppose that all of the following statements hold:

- $N(a_v) \cap N(c_v) = \{v, b_v, u_v\}$ with $u_v \notin \{v\} \cup \Pi_v$;
- $V \neq \{a_v, b_v, c_v, u_v, v\}$;
- *there is no minimal separator $S \subseteq (N(a_v) \cap N(c_v)) \cup \{a_v, c_v\}$ in $G \setminus v$ and $\{a_v, c_v\} \subseteq S$.*

Let G' be the graph obtained from G by adding edges $\{v, u_v\}$ and $\{b_v, v\}$, then $tb(G) = 1$ if and only if $tb(G') = 1$. Moreover, G' is planar and prime.

Proof. We will first prove that $G \setminus v$ is prime. By contradiction, let S' be a minimal clique-separator of $G \setminus v$. By Theorem 6, there is $w_v \neq v$ such that $S' = \{b_v, w_v\}$, and by Lemma 16, S' must be an $a_v c_v$ -separator of $G \setminus v$. Then, it follows that $w_v = u_v \in N(a_v) \cap N(c_v)$, whence $V = \{a_v, b_v, c_v, u_v, v\}$ by Theorem 6, that contradicts the hypothesis. Therefore, $G \setminus v$ is prime.

Let us prove that $tb(G) = 1$ implies that $tb(G') = 1$. Let (T, \mathcal{X}) be a star-decomposition of G , which exists by Lemma 3, minimizing the distance in T between the subtrees T_{a_v} and T_{c_v} . Since $N_G(v) \subseteq N_G[b_v]$ then removing v from all bags leaves a tree decomposition of $G \setminus v$ of breadth

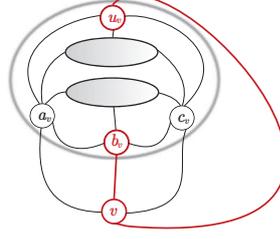


Figure 30: Case of a leaf-vertex v of Type 3 with $G \setminus v$ is prime and $N(a_v) \cap N(c_v) = \{u_v, b_v, v\}$. Red edges are those added by Algorithm `Leaf-BottomUp`.

one. Up to reducing the tree decomposition, let (T', \mathcal{X}') be any reduced tree decomposition of $G \setminus v$ that is obtained from (T, \mathcal{X}) by first removing v from the bags. Note that (T', \mathcal{X}') is a star-decomposition of $G \setminus v$ by Lemma 3. Now, there are two cases.

- Suppose $T'_{a_v} \cap T'_{c_v} \neq \emptyset$. We will need to prove in this case that the two subtrees $T'_{a_v} \cap T'_{b_v} \cap T'_{c_v}$ and $T'_{a_v} \cap T'_{u_v} \cap T'_{c_v}$ are nonempty and disjoint.

Claim 11 b_v, u_v are non-adjacent in G .

Proof. By contradiction, if it were the case that b_v, u_v are adjacent, then by Lemma 15, either u_v is an isolated vertex of $G \setminus (\Pi_v \cup \{v\})$ — in which case, $\Pi_v \in \mathcal{P}_3(G \setminus v)$ because we assume $V \neq \{a_v, b_v, c_v, u_v, v\}$ by the hypothesis —, or $(a_v, u_v, b_v) \in \mathcal{P}_3(G \setminus v)$. Since $G \setminus v$ is prime, it follows that one of Π_v or $\Pi_v \setminus b_v$ must be a minimal separator of $G \setminus v$, similarly one of (a_v, u_v, c_v) or (a_v, c_v) must be a minimal separator of $G \setminus v$. Therefore, both cases contradict the hypothesis that there is no minimal separator $S \subseteq (N(a_v) \cap N(c_v)) \cup \{a_v, c_v\}$ in $G \setminus v$ and $\{a_v, c_v\} \subseteq S$, which proves that b_v, u_v are non-adjacent. \diamond

Recall that we are in the case when $T'_{a_v} \cap T'_{c_v} \neq \emptyset$. The subtrees $T'_{a_v}, T'_{c_v}, T'_{u_v}$ are pairwise intersecting, similarly the subtrees $T'_{a_v}, T'_{c_v}, T'_{b_v}$ are pairwise intersecting. Therefore, by the Helly property (Lemma 1) $T'_{a_v} \cap T'_{b_v} \cap T'_{c_v} \neq \emptyset$ and $T'_{a_v} \cap T'_{u_v} \cap T'_{c_v} \neq \emptyset$. Furthermore, $T'_{a_v} \cap T'_{b_v} \cap T'_{c_v} \cap T'_{u_v} = \emptyset$, because since b_v, u_v are non-adjacent by Claim 11 no vertex dominates all of $\{a_v, b_v, c_v, u_v\}$ in G , and so, $T'_{a_v} \cap T'_{b_v} \cap T'_{c_v} \cap T'_{u_v} = \emptyset$.

Claim 12 The subtrees $T'_{a_v} \cap T'_{b_v} \cap T'_{c_v}$ and $T'_{a_v} \cap T'_{u_v} \cap T'_{c_v}$ are adjacent in T' .

Proof. By contradiction, let B be an internal bag onto the path between both subtrees in T' , let $z \in B$ dominate the bag. Note that $a_v, c_v \in B$ by the properties of the tree decomposition, $z \notin \{a_v, c_v\}$ because a_v, c_v are non-adjacent, and so, $z \in N(a_v) \cap N(c_v) \setminus v = \{u_v, b_v\}$. This contradicts the fact that $B \notin T'_{a_v} \cap T'_{b_v} \cap T'_{c_v}$ and $B \notin T'_{a_v} \cap T'_{u_v} \cap T'_{c_v}$, therefore, the subtrees $T'_{a_v} \cap T'_{b_v} \cap T'_{c_v}$ and $T'_{a_v} \cap T'_{u_v} \cap T'_{c_v}$ are adjacent in T' . \diamond

Finally, let $B \in T'_{a_v} \cap T'_{b_v} \cap T'_{c_v}, B' \in T'_{a_v} \cap T'_{u_v} \cap T'_{c_v}$ be adjacent, that exist by Claim 12. Observe that b_v dominates B , u_v dominates B' . To obtain a star-decomposition of G' from (T', \mathcal{X}') , it now suffices to add vertex v in B and B' , whence $tb(G') = 1$.

- Else, $T'_{a_v} \cap T'_{c_v} = \emptyset$. This implies $T_{a_v} \cap T_{c_v} = \emptyset$. Since the tree decomposition (T, \mathcal{X}) minimizes the distance in T between T_{a_v} and T_{c_v} , G is planar and $|N(a_v) \cap N(c_v)| \geq 3$, therefore by Corollary 9, the subtrees T_{a_v} and T_{c_v} are adjacent in T , whence the subtrees T'_{a_v}, T'_{c_v} are also adjacent in T' . In particular, by Corollary 9 there exist two adjacent bags

$B'_{a_v}, B'_{c_v} \in \mathcal{X}'$ such that $a_v \in B'_{a_v} \setminus B'_{c_v}, B'_{a_v} \cap B'_{c_v} = N_G(a_v) \cap N_G(c_v) \setminus v = \{u_v, b_v\}, c_v \in B'_{c_v} \setminus B'_{a_v}$. Furthermore, a_v dominates B'_{a_v} while c_v dominates B'_{c_v} . Therefore, in order to obtain a star-decomposition of G' from (T', \mathcal{X}') , it now suffices to add vertex v in B'_{a_v} and B'_{c_v} – that yields exactly (T, \mathcal{X}) –, whence $tb(G') = 1$.

Before we can prove the equivalence, *i.e.*, $tb(G) = 1$ if and only if $tb(G') = 1$, we need to prove first that G' is prime and planar.

Claim 13 G' is prime.

Proof. Let S' be a clique-separator of G' . Note that $v \in S'$ by construction of G' . Therefore, $S' \setminus v$ is a clique-separator of $G \setminus v$, that contradicts the fact that $G \setminus v$ is prime. Consequently, G' is prime. \diamond

Claim 14 G' is planar.

Proof. Let us fix a plane embedding of G . By Jordan Theorem, the cycle induced by (a_v, b_v, c_v, u_v) separates the plane into two regions. Let G_1, G_2 be respectively the subgraphs of G that are induced by all the vertices in each region.

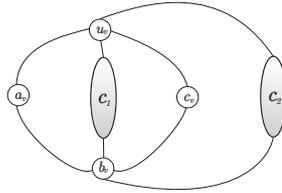


Figure 31: Proof that the graph G' of Theorem 11 is planar.

We claim that either $V \setminus (a_v, b_v, c_v, u_v, v) \subseteq V(G_1)$, or $V \setminus (a_v, b_v, c_v, u_v, v) \subseteq V(G_2)$. Note that it will prove that G' is planar, because then drawing vertex v onto the region that does not contain the set $V \setminus (a_v, b_v, c_v, u_v, v)$ yields a planar embedding of G' . By contradiction, let $C_1 \subseteq V(G_1), C_2 \subseteq V(G_2)$ be connected components of $V \setminus (a_v, b_v, c_v, u_v, v)$. Let $\Pi'_v = (a_v, u_v, c_v)$. If one of Π_v or Π'_v belongs to $\mathcal{P}_3(G \setminus v)$, then, there exists a minimal separator $S \subseteq (N(a_v) \cap N(c_v)) \cup \{a_v, c_v\}$ in $G \setminus v$ and since $G \setminus v$ is prime, $\{a_v, c_v\} \subseteq S$. This would contradict the hypothesis, so, $\Pi_v, \Pi'_v \notin \mathcal{P}_3(G \setminus v)$. As a result, since $(a_v, b_v, c_v, u_v) = \Pi_v \cup \Pi'_v$ separates C_1 from C_2 , therefore, $u_v, b_v \in N(C_1) \cap N(C_2)$ (or else, $\Pi_v \in \mathcal{P}_3(G \setminus v)$ or $\Pi'_v \in \mathcal{P}_3(G \setminus v)$). Let us remove all other components of $V \setminus (a_v, b_v, c_v, u_v, v)$ but C_1 and C_2 , and let us remove all edges between $\{a_v, c_v\}$ and $C_1 \cup C_2$ if any (see Figure 31). Finally, let us contract C_1, C_2 to the two vertices x_1, x_2 . The cycle induced by (u_v, x_1, b_v, x_2) separates the plane into two regions with a_v, c_v being into different regions by construction. Vertex v must belong to one of the regions, but then it is a contradiction because $v \in N(a_v) \cap N(c_v)$ by the hypothesis. \diamond

To conclude the proof, let us prove that conversely, $tb(G') = 1$ implies that $tb(G) = 1$. Let (T', \mathcal{X}') be a star-decomposition of G' minimizing the distance in T' between the subtrees T'_{a_v} and T'_{c_v} . As an intermediate step, we claim that if removing vertex v from all bags of \mathcal{X}' leaves a tree decomposition of $G \setminus v$ of breadth one, then it implies that $tb(G) = 1$. To prove the claim, there are two cases to be considered.

- If $T'_{a_v} \cap T'_{c_v} \neq \emptyset$, then the subtrees $T'_{a_v}, T'_{b_v}, T'_{c_v}$ are pairwise intersecting, hence by the Helly property (Lemma 1) $T'_{a_v} \cap T'_{b_v} \cap T'_{c_v} \neq \emptyset$. Equivalently there is bag containing Π_v , and so it suffices to remove v from all bags and then to make any bag containing Π_v adjacent to the new bag $N_G[v]$ in order to obtain a tree decomposition of G of breadth one.
- Else, $T'_{a_v} \cap T'_{c_v} = \emptyset$. Since (T', \mathcal{X}') minimizes the distance in T' between the subtrees T'_{a_v} and T'_{c_v} , G' is planar by Claim 14 and a_v, c_v have three common neighbours in G' , therefore, by Corollary 9 there must exist two adjacent bags B'_{a_v}, B'_{c_v} such that $a_v \in B'_{a_v} \setminus B'_{c_v}, B'_{a_v} \cap B'_{c_v} = N(a_v) \cap N(c_v)$ and $c_v \in B'_{c_v} \setminus B'_{a_v}$. Furthermore, vertex a_v dominates the bag B'_{a_v} , while vertex c_v dominates the bag B'_{c_v} . As a result, removing vertex v from all bags but B'_{a_v}, B'_{c_v} leads to a tree decomposition of G of breadth one.

Consequently, we are left to modify the tree decomposition (T', \mathcal{X}') so as to ensure that none of the bags is only dominated by vertex v in G' , for if it is the case then removing v from all bags does leave a tree decomposition of $G \setminus v$ of breadth one. We will call the latter property the *removal property*. Observe that if it is the case that (T', \mathcal{X}') does not satisfy the removal property, then there must be a bag B fully containing $N_{G'}(v)$ because any strict subset of $N_{G'}(v)$ is dominated by some vertex of $G \setminus v$. In particular, $B = N_{G'}[v]$ because only vertex v dominates $N_{G'}(v)$ in G' , and so we can further assume that $T'_v = \{B\}$ without violating the property for (T', \mathcal{X}') to be a tree decomposition of G' of breadth one. Therefore in the following, assume that (T', \mathcal{X}') is a reduced star-decomposition of G' and $T'_v = \{B\}$, that is always possible to achieve by Lemma 3 and above remarks.

Since $V \neq \{a_v, b_v, c_v, u_v, v\} = N_{G'}[v]$ by the hypothesis, therefore, $\mathcal{X}' \setminus B \neq \emptyset$. Let B' be adjacent to B in T' . Note that $B \cap B' \neq \{a_v, b_v, c_v, u_v\}$ because no other vertex than v dominates the subset $\{a_v, b_v, c_v, u_v\}$ in G' . By the properties of a tree decomposition, $B \cap B'$ is a separator of G' . Consequently, $B \cap B'$ is not a clique because G' is prime by Claim 13. Furthermore, since $B \cap B' \neq \{a_v, b_v, c_v, u_v\}$ it holds that $B \setminus (B' \cup \{v\}) \neq \emptyset$, consequently $B \cap B'$ is also a separator of $G \setminus v$. Since $G \setminus v$ is prime, $B \cap B'$ cannot be any of $(a_v, c_v), \Pi_v$ or Π'_v because by the hypothesis there is no minimal separator $S \subseteq (N(a_v) \cap N(c_v)) \cup \{a_v, c_v\}$ in $G \setminus v$ and $\{a_v, c_v\} \subseteq S$. It follows that $B \cap B' \subseteq \{a_v, b_v, u_v\}$ or $B \cap B' \subseteq \{b_v, c_v, u_v\}$. Let us substitute the bag B with the two adjacent bags $B_1 = \{a_v, u_v, b_v, v\}, B_2 = \{b_v, c_v, u_v, v\}$, then we make adjacent all bags B'' that were formerly adjacent to B to some bag amongst B_1, B_2 containing $B \cap B''$. Note that $B_1 \subseteq N[a_v]$ and that $B_2 \subseteq N[c_v]$. Therefore, the resulting tree decomposition is a tree decomposition of G of breadth one such that v dominates no bag. ■

5.3.6 Case of leaf-vertex v of Type 2 or 3 and $G \setminus v$ not prime

The remaining subsections will be devoted to the proof of correctness of Step 3.2. In particular, this subsection is devoted to the proof that when $G \setminus v$ is not prime one can only consider the case when the leaf-vertex v is of Type 2, *i.e.*, v and b_v are adjacent in G . Note that when v is of Type 3, then in general one cannot add an edge between v and b_v without violating the property for the graph G to be planar, as shown in Figure 32. We will now prove that whenever we are in the conditions of Step 3.2, it is always possible to do so while preserving the planarity of the graph G and the property to be of tree-breadth one.

Theorem 12 *Let G be a prime planar graph. Let v be a leaf-vertex of Type 3 such that $G \setminus v$ is not prime. Finally, let $\Pi_v = (a_v, b_v, c_v)$ be as in Definition 5. Let G' be obtained from G by adding the edge $\{v, b_v\}$.*

Then, G' is prime and planar, and $tb(G) = 1$ if and only if $tb(G') = 1$.

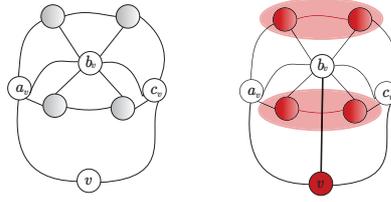


Figure 32: A planar graph G with $tb(G) = 1$ (left), and a leaf-vertex v of Type 3 so that adding an edge between v and b_v violates the property for the graph to be planar (right). In the latter case, one side of the bipartition of the $K_{3,3}$ -minor is coloured red.

Proof. First, we prove that G' is prime and planar.

- In order to prove that G' is prime, by contradiction let S be a clique-separator of G' . Since G' is a supergraph of G , therefore S is a separator of G but it does not induce a clique in G . Hence, S contains the edge $\{v, b_v\}$, and so either $S \subseteq \{a_v, b_v, v\}$ or $S \subseteq \{b_v, c_v, v\}$. Let $C = V \setminus (\Pi_v \cup \{v\})$, by Theorem 6, C is a full component of $G \setminus \Pi_v$ because $G \setminus v$ is not prime. In particular, C is connected and $a_v, c_v \in N(C)$, that contradicts the fact that $G' \setminus S$ is unconnected.
- Then in order to prove that G' is planar, let us fix a plane embedding of G . The cycle induced by (a_v, b_v, c_v, v) separates the plane into two regions. To prove that G' is planar, we claim that it suffices to prove that all vertices in $C = V \setminus (\Pi_v \cup \{v\})$ are in the same region, for then drawing the edge $\{b_v, v\}$ in the other region leads to a plane embedding of G' . By contradiction, let $x, y \in C$ be in different regions. By [13, Proposition 8], the cycle (a_v, b_v, c_v, v) is an xy -separator of G , that contradicts the fact that C is connected.

Let us now prove that $tb(G) = 1$ implies that $tb(G') = 1$. Let (T, \mathcal{X}) be a star-decomposition of G , that exists by Lemma 3, minimizing the distance in T between the subtrees T_{a_v} and T_{c_v} . Let us remove vertex v from all bags, that leads to a tree decomposition (T, \mathcal{X}_{-v}) of $G \setminus v$ of breadth one because $N_G(v) \subseteq N_G(b_v)$. Then, let (T', \mathcal{X}') be any reduced tree decomposition that is obtained from (T, \mathcal{X}_{-v}) , that is a star-decomposition of $G \setminus v$ by Lemma 3. Now, there are two cases. If $T'_{a_v} \cap T'_{c_v} \neq \emptyset$, then the subtrees $T'_{a_v}, T'_{b_v}, T'_{c_v}$ are pairwise intersecting and so, by the Helly property (Lemma 1) $T'_{a_v} \cap T'_{b_v} \cap T'_{c_v} \neq \emptyset$. Hence one obtains a star-decomposition of G' simply by making some bag of $T'_{a_v} \cap T'_{b_v} \cap T'_{c_v}$ adjacent to the new bag $N_{G'}[v] = \{a_v, b_v, c_v, v\}$. Else, $T'_{a_v} \cap T'_{c_v} = \emptyset$, so, $T_{a_v} \cap T_{c_v} = \emptyset$. Since $\Pi_v \in \mathcal{P}_3(G)$ and G is prime by the hypothesis, therefore, one of Π_v or $\Pi_v \setminus b_v$ must be a minimal separator of G . As a result, since (T, \mathcal{X}) is assumed to minimize the distance in T between the subtrees T_{a_v} and T_{c_v} , by Corollary 8 there exist two adjacent bags $B_{a_v}, B_{c_v} \in \mathcal{X}$ so that $a_v \in B_{a_v} \setminus B_{c_v}$ and $c_v \in B_{c_v} \setminus B_{a_v}$ respectively dominate the bags B_{a_v} and B_{c_v} . In such case, $B_{a_v} \cap B_{c_v} = N_G(a_v) \cap N_G(c_v)$ and so, since $b_v, v \in B_{a_v} \cap B_{c_v}$, (T, \mathcal{X}) is also a star-decomposition of G' . So, in conclusion, $tb(G') = 1$ in both cases.

Conversely, let us prove that $tb(G') = 1$ implies that $tb(G) = 1$. Let (T', \mathcal{X}') be a star-decomposition of G' , that exists by Lemma 3, minimizing the distance in T' between the subtrees T'_{a_v} and T'_{c_v} . Let us remove vertex v from all bags, that leads to a tree decomposition (T', \mathcal{X}'_{-v}) of $G' \setminus v = G \setminus v$ of breadth one because $N_{G'}[v] \subseteq N_{G'}[b_v]$. Then, let (T, \mathcal{X}) be any reduced tree decomposition that is obtained from (T', \mathcal{X}'_{-v}) , that is a star-decomposition of $G \setminus v$ by Lemma 3. There are two cases. If $T_{a_v} \cap T_{c_v} \neq \emptyset$, then one obtains a star-decomposition of G simply by making some bag of $T_{a_v} \cap T_{c_v}$ adjacent to the new bag $N_G[v] = \{a_v, c_v, v\}$. Else, $T_{a_v} \cap T_{c_v} = \emptyset$,

so, $T'_{a_v} \cap T'_{c_v} = \emptyset$. Since $\Pi_v \in \mathcal{P}_3(G')$ and G' is also prime, therefore, one of Π_v or $\Pi_v \setminus b_v$ must be a minimal separator of G' . As a result, since (T', \mathcal{X}') is assumed to minimize the distance in T' between the subtrees T'_{a_v} and T'_{c_v} , by Corollary 8 there exist two adjacent bags $B'_{a_v}, B'_{c_v} \in \mathcal{X}'$ so that $a_v \in B'_{a_v} \setminus B'_{c_v}$ and $c_v \in B'_{c_v} \setminus B'_{a_v}$ respectively dominate the bags B'_{a_v} and B'_{c_v} . In such case, one obtains a star-decomposition of G by adding v in the two bags B'_{a_v}, B'_{c_v} . So, in conclusion, $tb(G) = 1$ in both cases. \blacksquare

5.3.7 Proof of Step 3.2 (a)

Theorem 13 *Let G be a prime planar graph, let v be a leaf-vertex of Type 2, $\Pi_v = (a_v, b_v, c_v)$ be as in Definition 5, and let $u_v \notin \Pi_v \cup \{v\}$ be such that (b_v, u_v) is an edge-separator of $G \setminus v$.*

Suppose a_v and u_v are non-adjacent, and either c_v and u_v are non-adjacent or the subset $N_G(a_v) \cap N_G(u_v)$ is not an $a_v u_v$ -separator in the subgraph $G \setminus (c_v, v)$.

Then, G/va_v (obtained by contracting $\{v, a_v\}$) is planar and prime and $tb(G) = 1$ if and only if $tb(G/va_v) = 1$.

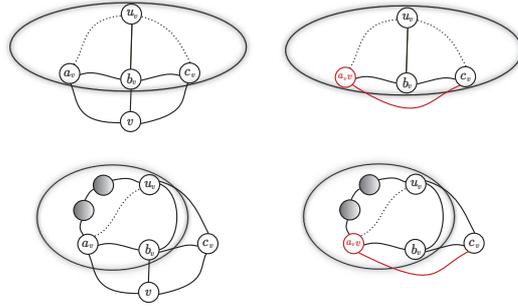


Figure 33: Cases when Theorem 13 applies and the edge $\{v, a_v\}$ can be contracted to a_v .

Proof. The graph G/va_v is a contraction of the planar graph G , therefore it is planar. Let us prove that G/va_v is prime. By contradiction, let S be a minimal clique-separator of G/va_v . Since G/va_v is a supergraph of $G \setminus v$, S is also a separator of $G \setminus v$. Furthermore, it is not an $a_v c_v$ -separator because a_v, c_v are adjacent in G/va_v , therefore, by Lemma 16 S is a separator of G . Since G is prime by the hypothesis, S does not induce a clique of G , whence $a_v, c_v \in S$. However, since (b_v, u_v) is not a separator of G because G is prime by the hypothesis, therefore by Lemma 16 (u_v, b_v) is an $a_v c_v$ -separator of $G \setminus v$. So, $N_G(a_v) \cap N_G(c_v) \subseteq \{v, b_v, u_v\}$, that implies $N_G(a_v) \cap N_G(c_v) = \{v, b_v\}$ because a_v and u_v are non-adjacent by the hypothesis. In such a case $S \subseteq \Pi_v$, but then $V \setminus (\Pi_v \cup \{v\})$ cannot be a full component of $G \setminus \Pi_v$, thus contradicting Theorem 6. As a result, the graph G/va_v is planar and prime.

If $tb(G) = 1$ then $tb(G/va_v) = 1$ because tree-breadth is contraction-closed by Lemma 2. Conversely, let us prove that $tb(G/va_v) = 1$ implies $tb(G) = 1$. To show this, let (T, \mathcal{X}) be a star-decomposition of G/va_v , that exists by Lemma 3, minimizing the number of bags $|\mathcal{X}|$ (in particular, (T, \mathcal{X}) is a reduced tree decomposition). Assume moreover (T, \mathcal{X}) to minimize the number of bags that are not contained into the closed neighbourhood of some vertex in G w.r.t. this property. Note that there is a bag of (T, \mathcal{X}) containing Π_v , because since it is a clique of G/va_v the subtrees $T_{a_v}, T_{b_v}, T_{c_v}$ are pairwise intersecting and so, by the Helly property (Lemma 1), $T_{a_v} \cap T_{b_v} \cap T_{c_v} \neq \emptyset$. So, we can add in (T, \mathcal{X}) a new bag $N_G[v]$, and by making

this bag adjacent to any bag of $T_{a_v} \cap T_{b_v} \cap T_{c_v}$, one obtains a tree decomposition of G (not necessarily a star-decomposition). Consequently, we claim that to prove that $tb(G) = 1$, it suffices to prove that (T, \mathcal{X}) is a star-decomposition of $G \setminus v$, for then the above construction leads to a star-decomposition of G .

By contradiction, suppose it is not the case that (T, \mathcal{X}) is a star-decomposition of $G \setminus v$. Since G/va_v and $G \setminus v$ only differ in the edge $\{a_v, c_v\}$, there must be a bag B of $T_{a_v} \cap T_{c_v}$ that is only dominated by some of a_v, c_v . We make the stronger claim that the bag B has a unique dominator, that is either a_v or c_v . Since B is only dominated by some of a_v, c_v , then in order to prove the claim by contradiction we only need to consider the case when $B \subseteq N_{G/va_v}[a_v] \cap N_{G/va_v}[c_v]$. Recall that $N_{G/va_v}[a_v] \cap N_{G/va_v}[c_v] = \{a_v, b_v, c_v\}$ by the above remarks (because (u_v, b_v) is an $a_v c_v$ -separator of $G \setminus v$), therefore either $B = \{a_v, b_v, c_v\}$ or $B = \{a_v, c_v\}$. In the first case ($B = \{a_v, b_v, c_v\}$) we have that $B \subseteq N[b_v]$, thus contradicting the fact that B is only dominated by some of a_v, c_v . However in the second case ($B = \{a_v, c_v\}$) the bag B is strictly contained in any bag of the nonempty subtree $T_{a_v} \cap T_{b_v} \cap T_{c_v}$, thus contradicting the fact that (T, \mathcal{X}) is a reduced tree decomposition by minimality of $|\mathcal{X}|$. Therefore, the claim is proved and so, the bag B has a unique dominator, that is either a_v or c_v . Note that if $B \subseteq N_{G/va_v}[c_v]$ then we may further assume that c_v, u_v are nonadjacent, or else by Theorem 6 $N_{G/va_v}[c_v] = \{a_v, b_v, c_v, u_v\} \subseteq N[b_v]$ and so, $B \subseteq N[b_v]$, that would contradict the claim that B is only dominated by some of a_v, c_v . In addition, since a_v and c_v play symmetrical roles in the case when u_v, c_v are nonadjacent, let us assume w.l.o.g. that vertex a_v is the sole dominator of the bag B .

In such a case, $N_{G/va_v}(a_v) \cap N_{G/va_v}(c_v) = \{b_v\}$ because (u_v, b_v) is an $a_v c_v$ -separator of $G \setminus v$, so, since $N(c_v) \setminus (\Pi_v \cup \{v\}) \neq \emptyset$ because G is prime by the hypothesis, the existence of a bag B' containing vertex c_v and adjacent to B follows. By the properties of a tree decomposition, $B \cap B'$ is a separator of G/va_v . Now, let C_a be the component of vertex a_v in $G \setminus (b_v, u_v, v)$. Observe that $c_v \notin C_a$ because (u_v, b_v) is an $a_v c_v$ -separator of $G \setminus v$. Since $B \cap B' \subseteq N_{G/va_v}[a_v] \subseteq C_a \cup \Pi_v$, therefore, $B \cap B' \cap C_a \neq \emptyset$ or else $B \cap B'$ would be a clique-separator in G/va_v (impossible since it is a prime graph). There are several cases to be considered depending on the dominators of bag B' .

- If a_v dominates B' then B, B' can be merged into one, thus contradicting the minimality of $|\mathcal{X}|$;
- Else, B' must be dominated by one of b_v or u_v because $B \cap B' \cap C_a \neq \emptyset$, $c_v \in (B \cap B') \setminus C_a$ and (b_v, u_v) separates c_v from C_a . In fact, we claim that it cannot be dominated by vertex u_v . By contradiction, suppose that it is the case. Since a_v and u_v are non-adjacent, therefore, $a_v \in B \setminus B'$ and $u_v \in B' \setminus B$. So, it follows by the properties of a tree decomposition that $B \cap B'$ is an $a_v u_v$ -separator of G/va_v . However, $B \cap B' \subseteq N(a_v) \cap N(u_v)$, that contradicts the hypothesis that $N_G(a_v) \cap N_G(u_v)$ is not an $a_v u_v$ -separator in the subgraph $G \setminus (c_v, v)$.

Therefore, $b_v \in B'$ dominate the bag. Observe that if it were the case that there are at least two bags that are both adjacent to B and dominated by b_v , then they could all be merged into one without violating the property for (T, \mathcal{X}) to be a star-decomposition. As a result, by minimality of $|\mathcal{X}|$, B' is the unique bag that is both adjacent to B and dominated by b_v , whence it is also the unique bag adjacent to B containing vertex c_v . Let us substitute the two bags B, B' with $B \setminus c_v, B' \cup \{a_v\}$. Since $N_{G/va_v}(a_v) \cap N_{G/va_v}(c_v) = \{b_v\}$, it is still a star-decomposition of G/va_v with equal number of bags $|\mathcal{X}|$. Furthermore, there is one less bag that is not contained in the closed neighbourhood of some vertex in G , thus contradicting the minimality of (T, \mathcal{X}) . ■

5.3.8 Proof of Step 3.2 (b) i and Step 3.2 (b) ii

In order to deal with all remaining cases, it will require us to further study the neighbourhood of vertex b_v in the graph. Observe that in the following Theorem 14 we needn't prove that the resulting graph G' is prime because it will be proved in Theorem 15.

Theorem 14 *Let G be a prime planar graph, let v be a leaf-vertex of Type 2, $\Pi_v = (a_v, b_v, c_v)$ be as in Definition 5, and let $u_v \notin \Pi_v \cup \{v\}$ be such that (b_v, u_v) is an edge-separator of $G \setminus v$.*

Suppose $u_v \in N(c_v) \setminus N(a_v)$, $N(a_v) \cap N(u_v)$ is an $a_v u_v$ -separator of $G \setminus (c_v, v)$, and $N(b_v) = \{a_v, c_v, u_v, v\}$.

Then, there exists $x \in (N(a_v) \cap N(u_v)) \setminus b_v$ such that the graph G' , obtained from G by adding the edge $\{b_v, x\}$, is planar and satisfies $tb(G') = 1$ if $tb(G) = 1$. Moreover, the vertex x can be found in linear-time.

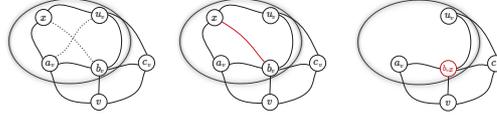


Figure 34: Cases when one of Theorem 14 or Theorem 15 applies and vertex b_v can be eventually contracted to another vertex.

Proof. First, we claim that (a_v, u_v) is a minimal 2-separator of G . Indeed, by the hypothesis c_v and u_v are adjacent, therefore, by Theorem 6 $N_G(c_v) = \{b_v, u_v, v\}$. In addition, $N(b_v) = \{a_v, c_v, u_v, v\}$ by the hypothesis. Last, since G is prime by the hypothesis, therefore, $N(a_v) \setminus (\Pi_v \cup \{v\}) \neq \emptyset$, and so, since a_v and u_v are non-adjacent by the hypothesis, $V(G) \setminus (a_v, b_v, c_v, u_v, v) \neq \emptyset$. As a result, (a_v, u_v) is a minimal 2-separator of G with $\{b_v, c_v, v\}$ being a full component of $G \setminus (a_v, u_v)$.

Since $N(a_v) \cap N(u_v)$ is an $a_v u_v$ -separator of $G \setminus (c_v, v)$ by the hypothesis, therefore, $N(a_v) \cap N(u_v) \setminus b_v \neq \emptyset$, for it has to contain a vertex from every component of $G \setminus (a_v, b_v, c_v, u_v, v)$. For now, let $x \in N(a_v) \cap N(u_v) \setminus b_v$ be arbitrary. Let us prove that $tb(G) = 1$ implies that $tb(G') = 1$ where G' is obtained by adding an edge between b_v and x (for now, G' may not be planar, depending on the choice for x). To prove this, let (T, \mathcal{X}) be a star-decomposition of G , that exists by Lemma 3, minimizing the distance in T between the subtrees T_{a_v} and T_{u_v} . We claim that $T_{a_v} \cap T_{u_v} \neq \emptyset$. By contradiction, if $T_{a_v} \cap T_{u_v} = \emptyset$, then by Corollary 8, there are two bags B_{a_v}, B_{u_v} that are adjacent in T and such that $a_v \in B_{a_v} \setminus B_{u_v}, u_v \in B_{u_v} \setminus B_{a_v}$ respectively dominate B_{a_v}, B_{u_v} . However, this implies by the properties of a tree decomposition that $B_{a_v} \cap B_{u_v} \subseteq N(a_v) \cap N(u_v)$ is an $a_v u_v$ -separator of G . Since the $a_v u_v$ -path (a_v, v, c_v, u_v) does not intersect $N(a_v) \cap N(u_v)$, that is clearly a contradiction, and so, $T_{a_v} \cap T_{u_v} \neq \emptyset$.

Furthermore, since there is a full component of $G \setminus (a_v, u_v)$ in the subgraph $G \setminus (b_v, c_v, v)$, therefore, by Lemma 9 the removal of vertices b_v, c_v, v from all bags in \mathcal{X} leads to a tree decomposition (T, \mathcal{X}^-) of breadth one of $G \setminus (b_v, c_v, v)$. Let (T', \mathcal{X}') be a reduced star-decomposition obtained from (T, \mathcal{X}^-) , that exists by Lemma 3. Since the subtrees T'_{a_v}, T'_x, T'_{u_v} are pairwise intersecting (because $x \in N(a_v) \cap N(u_v)$ and $T_{a_v} \cap T_{u_v} \neq \emptyset$), therefore by the Helly property (Lemma 1) $T'_{a_v} \cap T'_x \cap T'_{u_v} \neq \emptyset$. Let $B \in T'_{a_v} \cap T'_x \cap T'_{u_v}$. To obtain a star-decomposition of G' , it now suffices to make the bag B adjacent to the new bag $N_{G'}[b_v] = \{a_v, b_v, c_v, u_v, v, x\}$.

The above result holds for any choice of vertex $x \in (N_G(a_v) \cap N_G(u_v)) \setminus b_v$. Let us finally prove that one such a vertex x exists so that G' is planar. Indeed, since $N(a_v) \cap N(u_v)$ is an $a_v u_v$ -separator of $G \setminus (c_v, v)$ by the hypothesis, therefore, $S := (N(a_v) \cap N(u_v)) \cup \{v\}$ is

an $a_v u_v$ -separator of G , and in particular it is a minimal $a_v u_v$ -separator (because for every vertex $s \in S$, there is an $a_v u_v$ -path that intersects S only in s). By Corollary 7, it can be computed in linear-time a planar supergraph G_S of G so that S induces a cycle of G_S . Then, let $N_{G_S}(b_v) \cap S = \{x, v\}$, by construction the graph G' is planar for such a choice of vertex x . ■

In Theorem 14, we show conditions so that vertex b_v can be made adjacent to some other vertex of $N_G(a_v) \cap N_G(u_v)$. Lemma 23 completes the picture by proving that if it is the case that $N_G(a_v) \cap N_G(u_v) \cap N_G(b_v) \neq \emptyset$, then $|N_G(a_v) \cap N_G(u_v) \cap N_G(b_v)| = 1$ and vertex b_v has exactly five neighbours.

Lemma 23 *Let G be a prime planar graph, let v be a leaf-vertex of Type 2, $\Pi_v = (a_v, b_v, c_v)$ be as in Definition 5, and let $u_v \notin \Pi_v \cup \{v\}$ be such that (b_v, u_v) is an edge-separator of $G \setminus v$.*

Suppose $u_v \in N_G(c_v) \setminus N_G(a_v)$ and there exists $x \in N_G(a_v) \cap N_G(u_v) \cap N_G(b_v)$.

Then, $N_G(b_v) = \{a_v, c_v, u_v, v, x\}$.

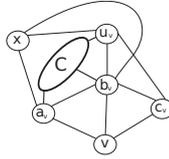


Figure 35: Case when $N_G(b_v) \neq \{a_v, c_v, u_v, v, x\}$.

Proof. By contradiction, let C be a component of $G \setminus (a_v, b_v, c_v, u_v, v, x)$ such that $b_v \in N(C)$ (see Figure 35 for an illustration). By Theorem 6 $N_G(c_v) = \{b_v, u_v, v\}$, therefore, $c_v, v \notin N(C)$. It follows that $N(C)$ is a separator of G . In particular, $N(C) \subseteq \{a_v, b_v, u_v, x\}$, so, $a_v, u_v \in N(C)$ or else $N(C)$ should be a clique-separator of the prime graph G . As a result, there is a $K_{3,3}$ -minor with $\{a_v, b_v, u_v\}$ and $\{C, x, \{c_v, v\}\}$ being the two sides of the bipartition. It contradicts the fact that G is planar by the hypothesis. ■

Theorem 15 *Let G be a prime planar graph, let v be a leaf-vertex of Type 2, $\Pi_v = (a_v, b_v, c_v)$ be as in Definition 5, and let $u_v \notin \Pi_v \cup \{v\}$ be such that (b_v, u_v) is an edge-separator of $G \setminus v$.*

Suppose $u_v \in N(c_v) \setminus N(a_v)$, $N(a_v) \cap N(u_v)$ is an $a_v u_v$ -separator of $G \setminus (c_v, v)$, and either $N(b_v) = \{a_v, c_v, u_v, v\}$ or $N(b_v) \cap N(a_v) \cap N(u_v) \neq \emptyset$.

Then, there is $x \in N(a_v) \cap N(u_v)$ such that one of the following must hold:

- $V(G) = \{a_v, b_v, c_v, u_v, v, x\}$, and G admits a star-decomposition with two bags $N_G[b_v], N_G[x]$;
- or $\Pi' = (a_v, x, u_v) \in \mathcal{P}_3(G)$, and let G' be obtained from G by adding the edge $\{b_v, x\}$ (if it is not already present) then contracting this edge. The graph G' is planar and prime, furthermore $tb(G) = 1$ if and only if $tb(G') = 1$.

Moreover, vertex x can be computed in linear-time.

Proof. There are two cases. If $N_G(b_v) = \{a_v, c_v, u_v, v\}$, then let x be set as in the statement of Theorem 14. Else, let x be the unique vertex of $N(b_v) \cap N(a_v) \cap N(u_v)$, that is well-defined by Lemma 23. Note that in both cases, vertex x can be computed in linear-time. In addition, $N(b_v) \subseteq \{a_v, c_v, u_v, v, x\}$ (the latter property following from Lemma 23 when b_v and x are adjacent, and being trivial else). Suppose for the proof that $V(G) \neq \{a_v, b_v, c_v, u_v, v, x\}$ (else, Theorem 15 is trivial). We claim that $\{b_v, c_v, v\}$ is a component of $G \setminus \Pi'$. Indeed, $N(b_v) \subseteq \Pi' \cup \{c_v, v\}$

by the hypothesis, and by Theorem 6 $N_G(c_v) = \{b_v, u_v, v\}$. Since $V(G) \neq \{a_v, b_v, c_v, u_v, v, x\}$, then it indeed follows that $\Pi' \in \mathcal{P}_3(G)$, with $\{b_v, c_v, v\}$ being a component of $G \setminus \Pi'$.

Let us prove that G' is prime and planar. By Theorem 14, adding an edge between b_v and x if it is not already present does not violate the property for the graph G to be planar. Therefore, G' is planar because it is obtained by an edge-contraction from some planar graph. To prove that G' is prime, by contradiction suppose the existence of a minimal clique-separator S' of G' .

Let us denote by x' the vertex resulting from the contraction of the edge $\{b_v, x\}$. Let $S := S'$ if $x' \notin S'$, $S := (S' \setminus x') \cup \{b_v, x\}$ else. By construction, S is a separator of G . In particular, S is not a clique because G is prime by the hypothesis. Therefore, $S \neq S'$, whence $x' \in S'$ or equivalently, $x, b_v \in S$. We now claim that $c_v \in S \cap S'$ or $v \in S \cap S'$ (possibly, $v, c_v \in S \cap S'$). There are two cases.

- Suppose that $S \setminus b_v$ is a separator of G . Then, $S \setminus b_v$ is not a clique because G is prime by the hypothesis. Since $S \setminus (b_v, x) = S' \setminus x'$ is a clique, there must be some vertex of $S \setminus (b_v, x) = S \cap S'$ that is adjacent to x' in G' but non-adjacent to x in G . Consequently, $v \in S \cap S'$ or $c_v \in S \cap S'$.
- Else, $S \setminus b_v$ is not a separator of G . Recall that by construction, S is a separator of G . In particular, there must be two neighbours of b_v in G that are separated by S in G . Since $N_G(b_v) \setminus x$ induces the path (a_v, v, c_v, u_v) , it follows that S must contain an internal node of the path, whence $c_v \in S \cap S'$ or $v \in S \cap S'$.

However, in such case S' must be contained in one of (a_v, x', v) , (v, x', c_v) or (c_v, x', u_v) , for it is a clique of G' . In particular, let $z \in \{a_v, u_v\} \setminus S'$. Since z has a neighbour in every component C' of $G \setminus \Pi'$, $\{z\} \cup C'$ is not disconnected by S' in G' . Furthermore, let us contract C' to z so as to make a_v and u_v adjacent, S' intersects the resulting cycle (a_v, u_v, c_v, v) either in an edge (different from $\{a_v, u_v\}$) or a single vertex because it is a clique of G' , therefore, $(a_v, u_v, c_v, v) \setminus S'$ is not disconnected by S' . Altogether, this contradicts the fact that S' is a separator of G' , and so, G' is prime.

Finally, let us prove that $tb(G') = 1$ if and only if $tb(G) = 1$. If $tb(G) = 1$, then let us assume b_v and x to be adjacent (if they are not, then Theorem 14 ensures we can add the edge without violating the property for the graph to be of tree-breadth one). Then, $tb(G') = 1$ because it is obtained by an edge-contraction from some graph with tree-breadth one and that tree-breadth is contraction-closed by Lemma 2.

Conversely, let us prove that $tb(G') = 1$ implies that $tb(G) = 1$. To prove this, let (T, \mathcal{X}) be a star-decomposition of G' , that exists by Lemma 3, minimizing the distance in T between the subtrees T_{a_v} and T_{u_v} . We claim that $T_{a_v} \cap T_{u_v} \neq \emptyset$. By contradiction, suppose $T_{a_v} \cap T_{u_v} = \emptyset$. Recall that $(a_v, x', u_v) \in \mathcal{P}_3(G')$ (because $\Pi' \in \mathcal{P}_3(G)$) and G' is prime, therefore one of (a_v, x', u_v) or (a_v, u_v) is a minimal separator of G' . Since we assume the distance in T between T_{a_v} and T_{u_v} to be minimized, by Corollary 8, there are two bags B_{a_v}, B_{u_v} that are adjacent in T so that $a_v \in B_{a_v} \setminus B_{u_v}, u_v \in B_{u_v} \setminus B_{a_v}$ respectively dominate B_{a_v}, B_{u_v} . However, by the properties of a tree decomposition $B_{a_v} \cap B_{u_v} \subseteq N(a_v) \cap N(u_v)$ is an $a_v u_v$ -separator of G' , that is impossible due to the existence of the path (a_v, v, c_v, u_v) in G' that does not intersect $N(a_v) \cap N(u_v)$. Therefore, $T_{a_v} \cap T_{u_v} \neq \emptyset$. Hence the subtrees $T_{a_v}, T_{x'}, T_{u_v}$ are pairwise intersecting and so, by the Helly Property (Lemma 1), $T_{a_v} \cap T_{x'} \cap T_{u_v} \neq \emptyset$. Furthermore, $N_{G'}[c_v] \cup N_{G'}[v] \subseteq N_{G'}[x']$ by construction. So, let us construct a tree decomposition of G of breadth one as follows. First, let us remove c_v and v from all bags in \mathcal{X} . Since $N_{G'}[c_v] \cup N_{G'}[v] \subseteq N_{G'}[x']$, one obtains a tree decomposition of $G' \setminus (c_v, v)$ of breadth one. Then let us replace x' with x in all bags. Note that in so doing, one obtains a tree decomposition of $G \setminus (b_v, c_v, v)$ of breadth one. Finally, let us

make adjacent the new bag $N_G[b_v]$ with any bag of $T_{a_v} \cap T_x \cap T_{u_v}$. The result is indeed a tree decomposition of G' because $N_G[b_v] \subseteq \{a_v, b_v, c_v, u_v, v, x\}$ and $N_G[c_v] \cup N_G[v] \subseteq N_G[b_v]$. ■

5.3.9 Proof of Step 3.2 (b) iii

Theorem 16 *Let G be a prime planar graph, let v be a leaf-vertex of Type 2, $\Pi_v = (a_v, b_v, c_v)$ be as in Definition 5, and let $u_v \notin \Pi_v \cup \{v\}$ be such that (b_v, u_v) is an edge-separator of $G \setminus v$.*

Suppose $u_v \in N(c_v) \setminus N(a_v)$, $N(a_v) \cap N(u_v)$ is an $a_v u_v$ -separator in the subgraph $G \setminus (c_v, v)$, $N(b_v) \neq \{a_v, c_v, u_v, v\}$ and $N(a_v) \cap N(b_v) \cap N(u_v) = \emptyset$.

Then it can be computed in linear-time (a unique) $x \in N(a_v) \cap N(u_v)$ such that if $tb(G) = 1$, $N(b_v) \cap N(x)$ is a $b_v x$ -separator and $|N(b_v) \cap N(x)| \geq 3$.

Proof. Let $W = (N(a_v) \cap N(u_v)) \cup \{a_v, c_v, u_v, v\}$. By the hypothesis, $N(b_v) \neq \{a_v, c_v, u_v, v\}$ and $N(a_v) \cap N(b_v) \cap N(u_v) = \emptyset$, therefore, it exists a component C_0 of $G \setminus W$ such that $b_v \in N(C_0)$. We claim that there is $x \in N(a_v) \cap N(u_v) \cap N(C_0)$ satisfying that $N(C_0) \subseteq (a_v, b_v, x)$ or $N(C_0) \subseteq (u_v, b_v, x)$. Indeed, first observe that $v, c_v \notin N(C_0)$ because by Theorem 6 $N_G(c_v) = \{b_v, u_v, v\}$. Furthermore, $a_v \notin N(C_0)$ or $u_v \notin N(C_0)$ because $N(a_v) \cap N(u_v)$ is an $a_v u_v$ -separator of $G \setminus (c_v, v)$ by the hypothesis. So, let $\{z, z'\} = \{a_v, u_v\}$ satisfy $z' \notin N(C_0)$. Since G is prime by the hypothesis, and so, biconnected, $G \setminus z$ is connected. Furthermore, $N(C_0) \setminus z \subseteq N(z')$ (by the definition of W), therefore, $N(C_0) \setminus z$ is a minimal separator of $G \setminus z$. By Lemma 13 there exist $s, t \in N(a_v) \cap N(u_v)$ non-adjacent such that $N(C_0) \setminus z = \{s, t\}$. Since $b_v \in N(C_0)$ by construction, therefore, let us set $\{s, t\} = \{b_v, x\}$, that finally proves the claim.

We claim in addition that x does not depend on the choice of the component C_0 . By contradiction, let C, C' be two components of $G \setminus W$ such that $b_v \in N(C) \cap N(C')$ and let $x, x' \in N(a_v) \cap N_G(u_v)$ be distinct and such that $x \in N(C)$, $x' \in N(C')$. Then, there exists a $K_{3,3}$ -minor with $\{a_v, b_v, u_v\}$ and $\{\{c_v, v\}, C \cup \{x\}, C' \cup \{x'\}\}$ being the sides of the bipartition, that contradicts the hypothesis that G is planar. Thus from now on, let $x \in N(a_v) \cap N(u_v) \setminus b_v$ be the unique vertex satisfying that for every component C of $G \setminus W$, if $b_v \in N(C)$ then $x \in N(C)$.

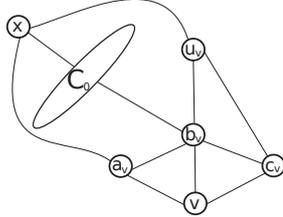


Figure 36: Component C_0 such that $b_v, x \in N(C_0)$.

Recall that C_0 is a fixed component of $G \setminus W$ such that $b_v, x \in N(C_0)$ (see Figure 36 for an illustration). Finally, assume for the remaining of the proof that $tb(G) = 1$ and let us prove that $N(b_v) \cap N(x)$ is a $b_v x$ -separator and $|N(b_v) \cap N(x)| \geq 3$. To prove it, we will only need to prove that $N(b_v) \cap N(x)$ is a $b_v x$ -separator of G . Indeed, in such a case $N(b_v) \cap N(x) \cap C_0 \neq \emptyset$, and so, $|N(b_v) \cap N(x)| \geq 3$ because $a_v, u_v \in N(b_v) \cap N(x)$ and $a_v, u_v \notin C_0$.

Let (T, \mathcal{X}) be star-decomposition of G , that exists by Lemma 3, minimizing the distance in T between the subtrees T_{b_v} and T_x . We claim that $T_{b_v} \cap T_x = \emptyset$. By contradiction, suppose $T_{b_v} \cap T_x \neq \emptyset$. Let us prove as an intermediate subclaim that $T_{a_v} \cap T_{u_v} \neq \emptyset$. By contradiction, let $T_{a_v} \cap T_{u_v} = \emptyset$. By the properties of a tree decomposition, every bag B onto the path in T between T_{a_v} and T_{u_v} must contain $N(a_v) \cap N(u_v)$ and at least one of v or c_v . If $c_v \in B$ then $B \subseteq N[u_v]$ since $N(c_v) = \{b_v, u_v, v\}$ and $x \in B$. Similarly if $v \in B$ then $B \subseteq N[a_v]$ since $N(v) = \{a_v, b_v, c_v\}$

and $x \in B$. Consequently, there are two adjacent bags B_{a_v}, B_{u_v} such that $a_v \in B_{a_v} \setminus B_{u_v}$ and $u_v \in B_{u_v} \setminus B_{a_v}$ respectively dominate B_{a_v} and B_{u_v} . However, by the properties of a tree decomposition, $B_{a_v} \cap B_{u_v} = N(a_v) \cap N(u_v)$ is an $a_v u_v$ -separator of G , thus contradicting the existence of the path (a_v, v, c_v, u_v) in G . Therefore, it follows that $T_{a_v} \cap T_{u_v} \neq \emptyset$, that proves the subclaim.

If $T_{a_v} \cap T_{u_v} \neq \emptyset$ and $T_{b_v} \cap T_x \neq \emptyset$ then the subtrees $T_{a_v}, T_{b_v}, T_{u_v}, T_x$ are pairwise intersecting and so, it implies $T_{a_v} \cap T_{u_v} \cap T_{b_v} \cap T_x \neq \emptyset$ by the Helly property (Lemma 1). However, let $B' \in T_{a_v} \cap T_{u_v} \cap T_{b_v} \cap T_x$, no vertex in G can dominate B' because $N(b_v) \cap N(a_v) \cap N(u_v) = \emptyset$ by the hypothesis, thus contradicting the fact that (T, \mathcal{X}) is a star-decomposition. As a result, we proved the claim that $T_{b_v} \cap T_x = \emptyset$.

Finally, since there exists $S \subseteq (N(b_v) \cap N(x)) \cup \{b_v, x\}$ a minimal separator of G such that $b_v, x \in S$ (namely, $S := N(C_0)$), and (T, \mathcal{X}) is assumed to minimize the distance in T between T_{b_v} and T_x , by Corollary 8 there exist two adjacent bags B_{b_v}, B_x such that $b_v \in B_{b_v} \setminus B_x$, $x \in B_x \setminus B_{b_v}$ respectively dominate B_{b_v} and B_x . By the properties of a tree decomposition, $B_{b_v} \cap B_x = N(b_v) \cap N(x)$ is indeed a $b_v x$ -separator of G . ■

Lemma 24 *Let G be a prime planar graph, let v be a leaf-vertex of Type 2, $\Pi_v = (a_v, b_v, c_v)$ be as in Definition 5, and let $u_v \notin \Pi_v \cup \{v\}$ be such that (b_v, u_v) is an edge-separator of $G \setminus v$.*

Suppose $u_v \in N(c_v) \setminus N(a_v)$, $N(a_v) \cap N(u_v)$ is an $a_v u_v$ -separator in the subgraph $G \setminus (c_v, v)$, $N(b_v) \neq \{a_v, c_v, u_v, v\}$ and $N(a_v) \cap N(b_v) \cap N(u_v) = \emptyset$. Assume furthermore that there is $x \in N(a_v) \cap N(u_v)$, and there exists a leaf-vertex $l \in N(b_v) \cap N(x)$.

Then, l is a leaf-vertex of Type 1, or l is a leaf-vertex of Type 2 or 3 and $G \setminus l$ is prime.

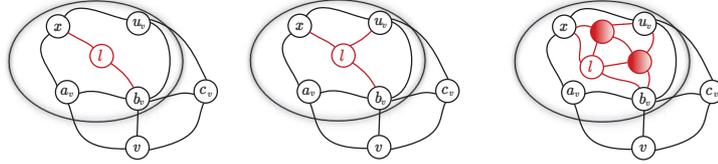


Figure 37: Existence of a leaf-vertex in $N(b_v) \cap N(x)$.

Proof. Suppose for the proof that l is not of Type 1 (else, Lemma 24 is trivial). Then, l is of Type 2 or 3, let Π_l be as in Definition 5. Note that $l \neq a_v$ because $v, b_v, x \in N(a_v)$ do not induce a path, similarly $l \neq u_v$ because $b_v, c_v, x \in N(u_v)$ do not induce a path. Furthermore by the hypothesis, b_v and x are the two endpoints of Π_l . Suppose by way of contradiction that there is a minimal clique-separator S of $G \setminus l$. Since G is prime by the hypothesis, by Lemma 16 S is a $b_v x$ -separator of $G \setminus l$. However, it implies that $a_v, u_v \in S$, that contradicts the fact that S is a clique. As a result, $G \setminus l$ is prime. ■

Equipped with Lemma 24, we can assume from now on that there is no leaf-vertex that is adjacent to both vertices b_v, x , or else it could be immediately processed by the algorithm.

Theorem 17 *Let G be a prime planar graph, let v be a leaf-vertex of Type 2, $\Pi_v = (a_v, b_v, c_v)$ be as in Definition 5, and let $u_v \notin \Pi_v \cup \{v\}$ be such that (b_v, u_v) is an edge-separator of $G \setminus v$.*

Suppose $u_v \in N(c_v) \setminus N(a_v)$, $N(a_v) \cap N(u_v)$ is an $a_v u_v$ -separator in the subgraph $G \setminus (c_v, v)$, $N(b_v) \neq \{a_v, c_v, u_v, v\}$ and $N(a_v) \cap N(b_v) \cap N(u_v) = \emptyset$. Assume furthermore that there is $x \in N(a_v) \cap N(u_v)$ such that $N(b_v) \cap N(x)$ is a $b_v x$ -separator, $|N(b_v) \cap N(x)| \geq 3$, and there is no leaf-vertex in $N(b_v) \cap N(x)$.

Then, there exist $y, z \in N(b_v) \cap N(x)$ non-adjacent such that the graph G' , obtained from G by making y, z adjacent, is planar and prime, and it holds $tb(G) = 1$ if and only if $tb(G') = 1$. Furthermore, the pair y, z can be computed in linear-time.

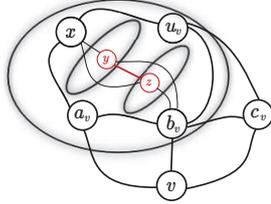


Figure 38: Illustration of Theorem 17.

Proof. Let us first show how to find the pair y, z . Let $W = \{a_v, c_v, v, u_v\} \cup (N(a_v) \cap N(u_v))$. Choose any component C_0 of $G \setminus W$ such that $b_v, x \in N(C_0)$ and $N(C_0) \subseteq (a_v, b_v, x)$ or $N(C_0) \subseteq (u_v, b_v, x)$ (the existence of such a component has been proved in Theorem 16). Note that $N(b_v) \cap N(x) \cap C_0 \neq \emptyset$ since $N(b_v) \cap N(x)$ is a $b_v x$ -separator of G by the hypothesis. Then, let $S := N(b_v) \cap N(x)$. By the hypothesis S is a minimal separator of G and $|S| \geq 3$, therefore, by Corollary 7 there is a planar supergraph G_S of G so that S induces a cycle of G_S . Furthermore, G_S can be computed in linear-time. Let P be an $a_v u_v$ -path of the cycle $G_S[S]$ that intersects C_0 . Since by the above claim $a_v \notin N_G(C_0)$ or $u_v \notin N_G(C_0)$, therefore, there is $y \in C_0 \cap V(P)$, there is z adjacent to vertex y in P so that either $z \in C_1$ for some component C_1 of $G \setminus (W \cup C_0)$ or $z \in \{a_v, u_v\} \setminus N_G(C_0)$. In particular, $z \notin N_G[C_0] = C_0 \cup N_G(C_0)$. Moreover, the graph G' , obtained from G by adding an edge between y and z , is planar by construction.

Claim 15 G' is prime.

Proof. By contradiction, let X be a minimal clique-separator of G' . Since G' is a supergraph of G , X is a separator of G . As a result, $y, z \in X$ because G is prime by the hypothesis. Let us prove as an intermediate step that $N(y) \cap N(z) = \{b_v, x\}$. There are two cases. If $z \in \{a_v, u_v\}$, then let $\{z, z'\} = \{a_v, u_v\}$. Since $N(C_0) \subseteq (z', b_v, x)$ (because $z \notin N_G(C_0)$) and z, z' are non-adjacent by the hypothesis, therefore, the claim immediately follows in this case. Else, $z \notin \{a_v, u_v\}$. Let C_1 be the component of $G \setminus (W \cup C_0)$ containing z . In such case, $b_v, x \in N(C_1)$ because $z \in S$ by construction. Therefore, $N(C_1) \subseteq (a_v, b_v, x)$ or $N(C_1) \subseteq (u_v, b_v, x)$ because the respective roles of components C_0, C_1 are symmetrical in this case. Suppose by way of contradiction $N(C_0) = N(C_1) = \{s, b_v, x\}$ for some $s \in \{a_v, u_v\}$ and let $\{s, t\} = \{a_v, u_v\}$. Then, there is a $K_{3,3}$ -minor of G with $\{b_v, x, s\}$ and $\{C_0, C_1, \{c_v, v, t\}\}$ being the sides of the bipartition, that contradicts the hypothesis that G is planar. As a result, $N(C_0) \cap N(C_1) = \{b_v, x\}$, that finally proves the claim.

Since X is assumed to be a clique of G' and $y, z \in X$, it follows $X \subseteq \{b_v, y, z\}$ or $X \subseteq \{x, y, z\}$. Consequently, $G[W \setminus X]$ is connected because $a_v, u_v \notin X$ is a dominating pair of W , $b_v, x \in N(a_v) \cap N(u_v)$ and $b_v \notin X$ or $x \notin X$. However, since $y, z \in X$ and $N_G(y) \subseteq W \cup C_0$, then in such case there must be a component A of $G \setminus X$ so that $A \subset C_0$. Since $z \notin N_G[C_0]$ by construction, $N_G(A) \subseteq X \setminus z$ is a clique-separator of G , thus contradicting the hypothesis that G is prime. As a result, G' is prime. \diamond

Now, let us prove $tb(G) = 1$ if and only if $tb(G') = 1$.

If $tb(G) = 1$, then let (T, \mathcal{X}) be star-decomposition of G , that exists by Lemma 3, minimizing the distance in T between T_{a_v} and T_{u_v} . Let us prove that (T, \mathcal{X}) is a star-decomposition of

G' , whence $tb(G') = 1$. To prove it, it is sufficient to prove $T_y \cap T_z \neq \emptyset$. We will prove as an intermediate claim that $T_{a_v} \cap T_{u_v} \neq \emptyset$. By contradiction, assume $T_{a_v} \cap T_{u_v} = \emptyset$. Observe that $\Pi' = (a_v, b_v, u_v) \in \mathcal{P}_3(G)$ with $\{c_v, v\}$ being a full component of $G \setminus \Pi'$. Therefore, since G is prime by the hypothesis, one of Π' or $\Pi' \setminus b_v$ is a minimal separator of G . Since (T, \mathcal{X}) is assumed to minimize the distance in T between T_{a_v} and T_{u_v} , therefore, by Corollary 8 there are two adjacent bags B_{a_v}, B_{u_v} such that $a_v \in B_{a_v} \setminus B_{u_v}$ and $u_v \in B_{u_v} \setminus B_{a_v}$ respectively dominate B_{a_v} and B_{u_v} . However, by the properties of a tree decomposition $B_{a_v} \cap B_{u_v} = N(a_v) \cap N(u_v)$ is an $a_v u_v$ -separator of G , that contradicts the existence of the path (a_v, v, c_v, u_v) in G . Consequently the claim is proved, hence $T_{a_v} \cap T_{u_v} \neq \emptyset$. The latter claim implies $T_{b_v} \cap T_x = \emptyset$, for if $T_{b_v} \cap T_x \neq \emptyset$ then the subtrees $T_{a_v}, T_{b_v}, T_{u_v}, T_x$ are pairwise intersecting, hence $T_{a_v} \cap T_{b_v} \cap T_x \cap T_{u_v} \neq \emptyset$ by the Helly property (Lemma 1), that would contradict the fact that (T, \mathcal{X}) is a star-decomposition because $N(a_v) \cap N(u_v) \cap N(b_v) = \emptyset$ by the hypothesis. Finally, since (x, y, b_v, z) induces a cycle of G and $T_{b_v} \cap T_x = \emptyset$, therefore, by the properties of a tree decomposition $T_y \cap T_z \neq \emptyset$, and so, (T, \mathcal{X}) is indeed a star-decomposition of G' .

Conversely, let us prove $tb(G') = 1$ implies $tb(G) = 1$. To prove it, let (T', \mathcal{X}') be a star-decomposition of G' , that exists by Lemma 3, minimizing the number $|\mathcal{X}'|$ of bags. Assume furthermore (T', \mathcal{X}') to minimize the number of bags $B \in \mathcal{X}'$ that are not contained into the closed neighbourhood of some vertex in G w.r.t. the minimality of $|\mathcal{X}'|$. In order to prove $tb(G) = 1$, it suffices to prove that (T', \mathcal{X}') is a star-decomposition of G . We will start proving intermediate claims.

Claim 16 $a_v, u_v \notin N_G(y)$.

Proof. By contradiction, assume the existence of $z' \in \{a_v, u_v\}$ so that z' and y are adjacent in G . In particular, $z' \neq z$ (since $z \notin N_G[C_0]$) and $N_G(C_0) = \{b_v, x, z'\}$ since either $N_G(C_0) \subseteq \{b_v, x, a_v\}$ or $N_G(C_0) \subseteq \{b_v, x, u_v\}$. Hence, the path (b_v, z', x) is a separator of G . Since $y \in N_G(b_v) \cap N_G(z') \cap N_G(x)$, by Lemma 15 either C_0 is reduced to y or $(b_v, y, x) \in \mathcal{P}_3(G)$ separates z' from $C_0 \setminus y$. The case $C_0 \setminus y = \emptyset$ implies that y is a leaf-vertex of Type 2, that contradicts the hypothesis that there is no leaf-vertex in $N_G(b_v) \cap N_G(x)$. Therefore, let $(b_v, y, x) \in \mathcal{P}_3(G)$ separates z' from $C_0 \setminus y$ in G , and let $C'_0 \subseteq C_0 \setminus y$ be a component of $G \setminus (b_v, y, x)$ (such a component C'_0 exists because $N_G(C_0) = \{z', b_v, x\}$). Since G is prime, $b_v, x \in N_G(C'_0)$ (indeed, neither b_v nor y nor x nor (b_v, y) nor (y, x) can be a separator of G). Therefore, $N_G(b_v) \cap N_G(x) \cap C'_0 \neq \emptyset$ because $N_G(b_v) \cap N_G(x)$ is a $b_v x$ -separator of G by the hypothesis. Furthermore, $y \in N_G(C'_0)$ because C_0 is connected. However in such case, there is a $K_{3,3}$ -minor of G' with $\{b_v, x, y\}$ and $\{a_v, C'_0, u_v\}$ being the sides of the bipartition, that contradicts the fact that G' is planar. \diamond

Claim 17 *There is no vertex dominating the cycle (a_v, b_v, u_v, x) in G' .*

Proof. By contradiction, if it were the case that such a vertex exists, then, since $N_G(b_v) \cap N_G(a_v) \cap N_G(u_v) = \emptyset$ by the hypothesis, the dominator should be y and furthermore $z \in \{a_v, u_v\}$. In particular, $y \in N_G(b_v) \cap N_G(x) \cap N_G(z')$ with $\{z, z'\} = \{a_v, u_v\}$, thus contradicting Claim 16. \diamond

Claim 18 $T'_{a_v} \cap T'_{u_v} \neq \emptyset$.

Proof. By contradiction, let $T'_{a_v} \cap T'_{u_v} = \emptyset$. By the properties of a tree decomposition, every bag B onto the path in T' between T'_{a_v} and T'_{u_v} (including the endpoints) must contain $N_{G'}(a_v) \cap N_{G'}(u_v)$ and at least one of v or c_v . Then, if $c_v \in B$ then $B \subseteq N_{G'}[u_v]$ and so, $B \in T'_{u_v}$, since $N_{G'}(c_v) = \{b_v, u_v, v\}$ and $x \in B$. Similarly if $v \in B$ then $B \subseteq N_{G'}[a_v]$ and so, $B \in T'_{a_v}$, since

$N_{G'}(v) = \{a_v, b_v, c_v\}$ and $x \in B$. Consequently, there are two adjacent bags B_{a_v}, B_{u_v} such that $a_v \in B_{a_v} \setminus B_{u_v}$ and $u_v \in B_{u_v} \setminus B_{a_v}$ respectively dominate B_{a_v} and B_{u_v} in G' . However, by the properties of a tree decomposition, $B_{a_v} \cap B_{u_v} = N_{G'}(a_v) \cap N_{G'}(u_v)$ is an $a_v u_v$ -separator of G' , thus contradicting the existence of the path (a_v, v, c_v, u_v) in G' . Therefore, it follows that $T'_{a_v} \cap T'_{u_v} \neq \emptyset$, that proves the claim. \diamond

Claim 19 $T'_{b_v} \cap T'_x = \emptyset$

Proof. Suppose for the sake of contradiction that $T'_{b_v} \cap T'_x \neq \emptyset$. By Claim 18, $T'_{a_v} \cap T'_{u_v} \neq \emptyset$, and so, the subtrees $T'_{a_v}, T'_{b_v}, T'_{u_v}, T'_x$ are pairwise intersecting. Hence by the Helly property (Lemma 1), $T'_{a_v} \cap T'_{b_v} \cap T'_{u_v} \cap T'_x \neq \emptyset$. However in such case, since (T', \mathcal{X}') is a star-decomposition of G' there must be a vertex dominating the cycle (a_v, b_v, u_v, x) in G' , thereby contradicting Claim 17. \diamond

As a result, $T'_{a_v} \cap T'_{u_v} \neq \emptyset$ by Claim 18 and $T'_{b_v} \cap T'_x = \emptyset$ by Claim 19.

Finally, suppose by way of contradiction (T', \mathcal{X}') is not a star-decomposition of G . In such case, since G and G' only differ in the edge $\{y, z\}$, there must exist $B \in T'_y \cap T'_z$ that is uniquely dominated by some of y, z in G' . More precisely, let us prove that only one of y, z can dominate B . By contradiction, suppose $B \subseteq N_{G'}[y] \cap N_{G'}[z] = \{b_v, x, y, z\}$ (indeed, $y \in C_0$ whereas $z \notin N_G[C_0]$). Since $T'_{b_v} \cap T'_x = \emptyset$ by Claim 19, $B \subseteq \{b_v, y, z\}$ or $B \subseteq \{x, y, z\}$. Therefore, B is a clique of G' . However, since $B \neq V(G')$, there is a bag B' adjacent to B and by the properties of a tree decomposition $B \cap B'$ is a clique-separator of G' , thus contradicting the fact that G' is prime by Claim 15. Consequently, either $B \subseteq N_{G'}[y]$ or $B \subseteq N_{G'}[z]$, and either $B \not\subseteq N_{G'}[y]$ or $B \not\subseteq N_{G'}[z]$. In the following, let $\{s, t\} = \{y, z\}$ satisfy $B \subseteq N_{G'}[s]$, that is well-defined. Let B' be any bag adjacent to B so that $t \in B'$ (such bag exists because $y, z \in N(b_v) \cap N(x)$, and $b_v \notin B$ or $x \notin B$ because $T'_{b_v} \cap T'_x = \emptyset$). There are three cases.

- Suppose no vertex of b_v, x, y, z dominates B' in G' (see Figure 39 for an illustration). Since $b_v \notin B$ or $x \notin B$ because $T'_{b_v} \cap T'_x = \emptyset$, therefore, $(B \cap B') \setminus (b_v, x, y, z) \neq \emptyset$, or else by the properties of a tree decomposition that would be a clique-separator of G' , thus contradicting the fact that G' is prime by Claim 15. Let $t' \in (B \cap B') \setminus (b_v, x, y, z)$. Note that t' and t are non-adjacent in G' because $t' \in N_{G'}[s]$ and $N_{G'}[y] \cap N_{G'}[z] = \{b_v, x, y, z\}$. Let $s' \in B'$ dominate this bag. Note that s' and s are non-adjacent in G' because we assume $s' \notin \{b_v, x, y, z\}$, $t \in N_{G'}(s')$ and $N_{G'}[y] \cap N_{G'}[z] = \{b_v, x, y, z\}$. In particular, $s' \neq t'$ and (s, t', s', t) induces a path in G . By construction, $y \in C_0$ and $z \notin N_G[C_0]$, hence there must be some of s', t' in $N_G(C_0)$. Since $N_G(C_0) \subseteq \{a_v, b_v, u_v, x\}$ and $s', t' \notin \{b_v, x, y, z\}$, therefore the pairs $\{s', t'\}$ and $\{a_v, u_v\}$ intersect. However, by Claim 16 $a_v, u_v \notin N_G(y)$, similarly $a_v, u_v \notin N_G(z)$, that contradicts the existence of the path (s, t', s', t) in G . Consequently, assume in the remaining cases that there is some vertex of b_v, x, y, z dominating B' in G' .

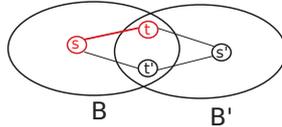


Figure 39: $B \subseteq N_{G'}[s]$, $B' \subseteq N_{G'}[s']$.

- Suppose B' is dominated by one of y, z in G' . We claim that B and B' are dominated by the same vertex of y, z , for if it were not the case $B \cap B' \subseteq N_{G'}[y] \cap N_{G'}[z] = \{b_v, x, y, z\}$,

and so, since by the properties of a tree decomposition $B \cap B'$ is a separator of G' , and $b_v \notin B$ or $x \notin B$ because $T'_{b_v} \cap T'_x = \emptyset$, $B \cap B'$ should be a clique-separator of G' , thus contradicting the fact that G' is prime by Claim 15. However, in such a case bags B, B' could be merged into one while preserving the property for the tree decomposition to be a star-decomposition of G' , that would contradict the minimality of $|\mathcal{X}'|$.

- Therefore, B' is dominated by some of b_v, x . We claim that there is a unique such bag B' that is adjacent to B . By contradiction, let $B'' \neq B'$ be adjacent to B and such that B'' is also dominated by some of b_v, x . In particular, if $B'' \cup B' \subseteq N[b_v]$ or $B'' \cup B' \subseteq N[x]$ then both bags B', B'' could be merged into one without violating the property for the tree decomposition to be a star-decomposition of G' , that would contradict the minimality of $|\mathcal{X}'|$. Else, w.l.o.g. $B' \subseteq N[b_v]$ and $B'' \subseteq N[x]$. Since $T'_{b_v} \cap T'_x = \emptyset$, $a_v, u_v \in N(b_v) \cap N(x)$ and B', B'' are adjacent to B , therefore, by the properties of the tree decomposition $a_v, u_v \in B$. However, $a_v, u_v \notin N_G[y]$ by Claim 16, a_v and u_v are non-adjacent, and either $z \in \{a_v, u_v\}$ or $a_v, u_v \notin N_G[z]$ (by the same proof as for Claim 16), thus contradicting the fact that either $B \subseteq N_{G'}[y]$ or $B \subseteq N_{G'}[z]$. Hence, the claim is proved and B' is assumed to be the unique bag adjacent to B such that $B \subseteq N[b_v]$ or $B \subseteq N[x]$. In particular, B' is the unique bag adjacent to B containing vertex t (recall that $\{s, t\} = \{y, z\}$ and $B \subseteq N_{G'}[s]$, $B \not\subseteq N_{G'}[t]$).

Let us substitute the bags B, B' with $B \setminus t, B' \cup \{s\}$. We claim that this operation keeps the property for (T', \mathcal{X}') to be a star-decomposition of G' . To prove the claim, first note that the operation only modifies bags B and B' , furthermore $B \setminus t \subseteq N[s]$ and $B' \cup \{s\} \subseteq N[b_v]$ or $B' \cup \{s\} \subseteq N[x]$. Consequently, to prove the claim, it suffices to prove that the operation keeps the property for (T', \mathcal{X}') to be a tree decomposition of G' (for in such a case, it is always a star-decomposition). Since $T'_t \setminus B$ is connected because B' is the only bag containing vertex t that is adjacent to the bag B , therefore, we are left to prove that there is no $w \in N_{G'}(t) \setminus s$ such that $T'_w \cap T'_t = \{B\}$. By contradiction, let $w \in N_{G'}(t) \setminus s$ satisfy $T'_w \cap T'_t = \{B\}$. Since $w \in B \subseteq N_{G'}[s]$, therefore $w \in N_{G'}(s) \cap N_{G'}(t) = N_G(y) \cap N_G(z) = \{b_v, x\}$. Moreover, $w \notin B'$ because $t \in B'$ and we assume that $T'_w \cap T'_t = \{B\}$. In such a case, since it is assumed that $B' \subseteq N[b_v]$ or $B' \subseteq N[x]$, and in addition $T'_x \cap T'_{b_v} = \emptyset$, let us write $\{w, w'\} = \{b_v, x\}$ such that $w \in B \setminus B'$, $w' \in B' \setminus B$ and $B' \subseteq N_{G'}[w']$. By the properties of a tree decomposition, $B \cap B'$ is a b_v, x -separator of G' , so, $a_v, u_v \in B \cap B'$. However, $a_v, u_v \notin N_G(y)$ by Claim 16 and similarly $a_v, u_v \notin N_G(z)$, that contradicts the fact that $B \subseteq N_{G'}[s]$ for some $s \in \{y, z\}$. This finally proves the claim that substituting the bags B, B' with $B \setminus t, B' \cup \{s\}$ keeps the property for (T', \mathcal{X}') to be a star-decomposition of G' .

However, the above operation does not increase the number of bags $|\mathcal{X}'|$, furthermore there is one less bag that is not contained in the closed neighbourhood of some vertex in G . This contradicts the minimality of (T', \mathcal{X}') w.r.t. these two properties.

As a result, we proved by contradiction that (T', \mathcal{X}') is a star-decomposition of G , hence $tb(G) = 1$. ■

5.4 Complexity of Algorithm Leaf-BottomUp

To complete this section, let us emphasize on some computational aspects of Algorithm Leaf-BottomUp, that will ensure the quadratic-time complexity of the algorithm. We here assume that the planar graph G is encoded with adjacency lists. Note that the adjacency lists can be updated in linear-time before each recursive call to the algorithm.

We will need as a routine to test whether two vertices are adjacent in *constant-time*. In order to achieve the goal, the following result (relying upon the bounded degeneracy of planar graphs) will be used:

Lemma 25 ([18]) *There exists a data structure such that each entry in the adjacency matrix of a planar graph can be looked up in constant time. The data structure uses linear storage, and can be constructed in linear time.*

5.4.1 Finding a leaf-vertex

At each call to the algorithm, it is first required to decide whether a leaf-vertex exists. If that is the case, then one such a vertex must be computed. Here is a way to achieve the goal in linear-time. Let us start computing the degree sequence of G , then let us order the vertices of the graph G by increasing degree.

Finding a leaf-vertex of Type 1. Let v be any vertex of degree at least four. We claim that a necessary condition for v to be a leaf-vertex of Type 1 is that all but at most two neighbours of v have degree four. Indeed, if v is a leaf-vertex of Type 1, then let Π_v, d_v be defined as in Definition 5. By Lemma 21, either $V(G) = N[v] \cup \{d_v\}$ or $\Pi' = (a_v, d_v, c_v) \in \mathcal{P}_3(G)$ and $N[v] \setminus (a_v, c_v)$ is a full component of $G \setminus \Pi'$. In both cases, all neighbours in $N(v) \setminus (a_v, c_v)$ have degree four.

- Therefore, let us count the number of neighbours of degree four in $N(v)$, that can be done in $\mathcal{O}(\deg(v))$ -time simply by traversing the adjacency list of vertex v (recall that the degree sequence of G has been computed).
- If there are all but at most two neighbours in $N(v)$ that have degree four, then we claim that one can construct the induced subgraph $G[N(v)]$ in $\mathcal{O}(\deg(v))$ -time. Indeed, for every neighbour $u \in N(v)$ that has degree four, let us test in constant-time for each of its four neighbours whether they are adjacent to vertex v — we only keep those for which it is the case in the adjacency list of u in $G[N(v)]$. Then, for every $u \in N(v)$ that does not have degree four (there are at most two such vertices), let us construct the adjacency list of u in $G[N(v)]$ simply by testing to which vertices in $N(v) \setminus u$ it is adjacent — the latter takes constant-time by neighbour.
- Once $G[N(v)]$ has been computed, it is easy to check whether it is a path in $\mathcal{O}(|N(v)|) = \mathcal{O}(\deg(v))$ -time.
- Finally, let $u \in N(v)$ have degree four. Let us pick in constant-time any neighbour $d_v \in N(u) \setminus N(v)$ (note that such a vertex is unique if $G[N(v)]$ induces a path). In order to decide whether v is a leaf-vertex of Type 1, it is now sufficient to test whether vertex d_v is adjacent to every vertex in $N(v)$ — that takes constant-time by neighbour.

Finding a leaf-vertex of Type 2. Recall that a vertex is a leaf-vertex of Type 2 if and only if it has degree three and its three neighbours induce a path. Given any vertex of degree three, three adjacency tests are enough in order to determine whether its three neighbours induce a path — and each adjacency tests takes constant-time. Therefore, it can be checked in constant-time whether a vertex is a leaf-vertex of Type 2.

Finding a leaf-vertex of Type 3. By Definition 5, a vertex v is a leaf-vertex of Type 3 if and only if it has degree two and its two neighbours are non-adjacent and they have at least two common neighbours (including v). Note that given a degree-two vertex, it can be checked whether its two neighbours are non-adjacent in constant-time. We now distinguish three cases.

1. First, suppose there is a vertex v such that $N(v) = \{x, y\}$ and neighbour x is a degree-two vertex. In such case, let $N(x) = \{v, z\}$, in order to decide whether v is a leaf-vertex of Type 3, it is sufficient to test in constant-time whether y, z are adjacent.
2. Second, suppose there are two degree-two vertices v, v' that share the same two non-adjacent neighbours (*i.e.*, $N(v) = N(v') = \{x, y\}$ and x, y are non-adjacent). In such case, both vertices v, v' are leaf-vertices of Type 3 (this case may happen if for instance, $G = K_{2,q}$ with $q \geq 2$). In order to check whether this case happens, it is sufficient to sort the pairs $N(v)$ with v being a degree-two vertex in linear-time (for instance, using a bucket-sort).
3. Else, let V' contain every degree-two vertex v with two non-adjacent neighbours of degree at least three (if one of the two neighbours of v has degree two, we fall in the first case) W.l.o.g., every vertex $v \in V'$ is uniquely determined by the pair $N(v)$ composed of its two neighbours (or else, we fall in the second case). In such case, let us contract every $v \in V'$ to one of its two neighbours. By doing so, we remove v and we make the two vertices in $N(v)$ adjacent. Note that all these edge-contractions are pairwise independent. Let us call G' the graph resulting from all edge-contractions, and let us call “virtual edges” any new edge resulting from an edge-contraction. Then, let us list all triangles in the resulting graph G' , it can be done in linear-time [42]. By construction, $v \in V'$ is a leaf-vertex of Type 3 if and only if the virtual edge resulting from its contraction belongs to a triangle in which it is the unique virtual edge.

Overall, finding a leaf-vertex in G takes $\mathcal{O}(\sum_{v \in V} \deg(v))$ -time, that is $\mathcal{O}(n)$ -time because G is planar.

5.4.2 Existence of a star-decomposition with two bags

Lemma 26 *Let G be a planar graph, it can be decided in quadratic-time whether G admits a star-decomposition with one or two bags.*

Proof. G admits a star-decomposition with one bag if and only if there is a universal vertex in G , hence it can be decided in linear-time. Assume for the remaining of the proof that G does not admit a star-decomposition with less than two bags. We will consider two necessary conditions for some fixed pair x, y to be the dominators of the only two bags in some star-decomposition of G . For each of the two conditions, we will prove that all pairs satisfying the condition can be computed in quadratic-time. Then, we will conclude the proof by showing that the two conditions are sufficient to ensure the existence of a star-decomposition of G with two bags.

1. Recall that if it exists a star-decomposition of G with two bags, then by the properties of a tree decomposition every vertex of G must be contained in at least one bag. Therefore, if x, y are the only two dominators of the bags in some star-decomposition of G , they must be a dominating pair of G . It can be decided in $\mathcal{O}(\deg(x) + \deg(y))$ -time whether a fixed pair x, y is a dominating pair. So, overall, it takes $\mathcal{O}(n^2)$ -time to compute all dominating pairs of G with n being the order of the graph, for the graph is planar and so, it is a sparse graph.

2. Furthermore, recall that if it exists a star-decomposition of G with two bags, then by the properties of a tree decomposition every edge of G must be contained in at least one bag. Therefore, if there is a star-decomposition of G with two bags that are respectively dominated by x and y , then it must be the case that there does not exist any edge $e = \{u, v\}$ so that $u \in N[x] \setminus N[y]$ and $v \in N[y] \setminus N[x]$ (else, such an edge could not be contained in any of the two bags). In order to decide whether the latter condition holds for some fixed pair x, y , it suffices to test whether every vertex of $N[x] \setminus N[y]$ is non-adjacent to all vertices in $N[y] \setminus N[x]$ — it takes constant-time per test and so, $\mathcal{O}(\deg(x) \cdot \deg(y))$ -time in total. As a result, computing all pairs x, y satisfying the condition requires $\mathcal{O}(\sum_{x,y} \deg(x) \cdot \deg(y)) = \mathcal{O}([\sum_x \deg(x)][\sum_y \deg(y)]) = \mathcal{O}(n^2)$ -time because the graph G is planar and so, it is a sparse graph.

Finally, let x, y satisfy the two above necessary conditions. We claim that (T, \mathcal{X}) with T being an edge and $\mathcal{X} = \{N[x], N[y]\}$ is a star-decomposition of G . Indeed, every vertex is contained into a bag because the pair x, y satisfies the first necessary condition. Furthermore, every edge has its both ends contained into a common bag because the pair x, y satisfies the second necessary condition. Last, all the bags containing a common vertex induce a subtree because there are only two bags. As a result, (T, \mathcal{X}) is a tree decomposition of G . Since each bag of \mathcal{X} is respectively dominated by x or y , therefore (T, \mathcal{X}) is indeed a star-decomposition of G , that proves the claim, hence the lemma. \blacksquare

Note that in any execution of Algorithm `Leaf-BottomUp`, it is verified at most once whether some planar graph admits a star-decomposition with one or two bags.

5.4.3 Upper-bound on the number of steps in the algorithm

Lemma 27 *Let G be a prime planar graph with n vertices and m edges. Then, there are at most $5n - m$ recursive calls to the Algorithm `Leaf-BottomUp`, that is $\mathcal{O}(n)$.*

Proof. First note that since G is planar by the hypothesis, $5n - m \geq 0$ and $5n - m = \mathcal{O}(n)$. Let G' with n' vertices and m' edges so that Algorithm `Leaf-BottomUp` is recursively applied on G' when G is the input. Since there is at most one such a graph G' (i.e., there is no more than one recursive call at each call of the algorithm), furthermore G' is prime and planar, therefore, in order to prove the lemma it suffices to prove that $5n' - m' < 5n - m$. To prove it, let us consider at which step of the algorithm the recursive call occurs.

- If it is at Step 2, then G' is obtained by removing a leaf-vertex of Type 1, denoted by v , and then contracting all the internal vertices in the path Π_v (induced by $N(v)$) to a single edge. Therefore, $n' = n - \deg(v) + 3$, $m' = m - 3\deg(v) + 8$ and so, $5n' - m' = 5n - m - (2\deg(v) - 7) < 5n - m$ because $\deg(v) \geq 4$.

Thus, from now on let us assume we fall in Step 3, i.e., a leaf-vertex of Type 2 or 3 is considered, denoted by v .

- If the recursion happens at Step 3.1 (a), then G' is obtained by removing v . Therefore, $n' = n - 1$ and either $m' = m - 3$ (if v is of Type 2) or $m' = m - 2$ (if v is of Type 3), hence $m' \geq m - 3$ and so, $5n' - m' \leq 5n - m - 2 < 5n - m$.
- If it is at Step 3.1 (b), then we fall in Step 3.1 (b) ii (no recursion occurs in Step 3.1 (b) i), thus G' is obtained by making v adjacent to the two vertices in $(N(a_v) \cap N(c_v)) \setminus v$ (including b_v in the case when v is of Type 3). Therefore, $n' = n$ and either $m' = m + 1$ (if v is of Type 2) or $m' = m + 2$ (if v is of Type 3), hence $m' \geq m + 1$ and so, $5n' - m' \leq 5n - m - 1 < 5n - m$.

- Else, the recursion happens at Step 3.2. Recall that in such case, there exists a vertex u_v such that (b_v, u_v) is a clique-separator of $G \setminus v$. Adding an edge between v and b_v if it does not exist, decreases $5n - m$ by 1, therefore from now on let us assume that v is a leaf-vertex of Type 2.
 - if it is at Step 3.2 (a), then G' is obtained by contracting the edge $\{v, a_v\}$. Therefore, $n' = n - 1$, $m' = m - 2$, hence $5n' - m' = 5n - m - 3 < 5n - m$.
 - If it is at Step 3.2 (b) i, then G' is obtained by adding an edge between b_v and some vertex $x \in (N(a_v) \cap N(u_v)) \setminus b_v$ then contracting this edge. Furthermore, $N(b_v) = \{a_v, c_v, u_v, v\}$ in such case and $c_v, v \notin N(x)$. Therefore, $n' = n - 1$, $m' = m - 2$ and so, $5n' - m' = 5n - m - 3 < 5n - m$.
 - If it is at Step 3.2 (b) ii, then G' is obtained by contracting the edge $\{b_v, x\}$ where $x \in N(a_v) \cap N(u_v) \cap N(b_v)$. Furthermore, $N(b_v) = \{a_v, c_v, u_v, v, x\}$ in such case and $c_v, v \notin N(x)$. Therefore, $n' = n - 1$, $m' = m - 3$ and so, $5n' - m' = 5n - m - 2 < 5n - m$.
 - Finally, in all other cases the recursive call happens at Step 3.2 (b) iii. Then, G' is obtained by adding an edge between two vertices $y, z \in N(b_v) \cap N(x)$ for some $x \in (N(a_v) \cap N(u_v)) \setminus b_v$. Therefore, $n' = n$, $m' = m + 1$ and so, $5n' - m' = 5n - m - 1 < 5n - m$.

■

6 Conclusion and Open questions

On the negative side, we proved the NP-hardness of computing five metric graph invariants (namely, tree-breadth, path-length, path-breadth, k -good tree and path decompositions) whose complexity has been left open in several works [24, 26, 27]. These results add up to the proof in [39] that it is NP-hard to compute the tree-length.

We leave as a future work further study on the border between tractable and intractable instances for the problem of computing the above metric graph invariants. Especially, what are the graph classes for which it can be decided in polynomial-time whether a graph admits a star-decomposition? In this paper, we partially answer to this question by proving that it is the case for bipartite graphs and planar graphs. Based on these two positive results, we conjecture that the problem is Fixed-Parameter Tractable when it is parameterized by the *clique-number* of the graph (note that there is a large clique in all the graphs obtained from our polynomial-time reductions). Intermediate challenges could be to determine whether the problem is Fixed-Parameter Tractable when it is parameterized by the genus, the tree-width or the Hardwiger number.

Finally, we notice that all our NP-hardness results imply that the above metric graph invariants cannot be approximated below some constant-factor. There remains a gap between our inapproximability results and the constant-ratio of the approximation algorithms in [24, 27]. Therefore, we leave as an interesting open question whether we can fill in this gap.

References

- [1] M. Abu-Ata and F. Dragan. Metric tree-like structures in real-world networks: an empirical study. *Networks*, pages n/a–n/a, 2015.

-
- [2] R. Agarwala, V. Bafna, M. Farach, M. Paterson, and M. Thorup. On the approximability of numerical taxonomy (fitting distances by tree metrics). *SIAM Journal on Computing*, 28(3):1073–1085, 1998.
- [3] S. Arnborg, D. Corneil, and A. Proskurowski. Complexity of finding embeddings in k -tree. *SIAM Journal on Algebraic Discrete Methods*, 8(2):277–284, 1987.
- [4] R. Balakrishnan and K. Ranganathan. *A textbook of graph theory*. Springer Science & Business Media, 2012.
- [5] H.-J. Bandelt and V. Chepoi. Metric graph theory and geometry: a survey. *Contemporary Mathematics*, 453:49–86, 2008.
- [6] M. Bdoiu, K. Dhamdhare, A. Gupta, Y. Rabinovich, H. Räcke, R. Ravi, and A. Sidiropoulos. Approximation algorithms for low-distortion embeddings into low-dimensional spaces. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 119–128. Society for Industrial and Applied Mathematics, 2005.
- [7] A. Berry, R. Pogorelcnik, and G. Simonet. An introduction to clique minimal separator decomposition. *Algorithms*, 3(2):197–215, 2010.
- [8] H. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25(6):1305–1317, 1996.
- [9] H. Bodlaender. A partial k -arboretum of graphs with bounded treewidth. *Theoretical computer science*, 209(1):1–45, 1998.
- [10] H. Bodlaender. Discovering treewidth. In *SOFSEM 2005: Theory and Practice of Computer Science, 31st Conference on Current Trends in Theory and Practice of Computer Science, Liptovský Ján, Slovakia, January 22-28, 2005, Proceedings*, pages 1–16, 2005.
- [11] H. Bodlaender, M. Fellows, and T. Warnow. Two strikes against perfect phylogeny. In *Automata, Languages and Programming, 19th International Colloquium, ICALP92, Vienna, Austria, July 13-17, 1992, Proceedings*, pages 273–283, 1992.
- [12] H. Bodlaender and T. Kloks. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *Journal of Algorithms*, 21(2):358–402, 1996.
- [13] V. Bouchitté, F. Mazoit, and I. Todinca. Chordal embeddings of planar graphs. *Discrete Mathematics*, 273(1):85–102, 2003.
- [14] V. Bouchitté and I. Todinca. Listing all potential maximal cliques of a graph. *Theoretical Computer Science*, 276(1):17–32, 2002.
- [15] S. Chechik, D. Larkin, L. Roditty, G. Schoenebeck, R. Tarjan, and V. Vassilevska Williams. Better approximation algorithms for the graph diameter. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1041–1052. SIAM, 2014.
- [16] V. Chepoi. On distance-preserving and domination elimination orderings. *SIAM Journal on Discrete Mathematics*, 11(3):414–436, 1998.
- [17] V. Chepoi, F. Dragan, B. Estellon, M. Habib, and Y. Vaxès. Diameters, centers, and approximating trees of delta-hyperbolic geodesic spaces and graphs. In *Proceedings of the Twenty-fourth Annual Symposium on Computational Geometry, SCG '08*, pages 59–68, New York, NY, USA, 2008. ACM.

-
- [18] M. Chrobak and D. Eppstein. Planar orientations with low out-degree and compaction of adjacency matrices. *Theor. Comput. Sci.*, 86(2):243–266, 1991.
- [19] N. Clarke and R. Nowakowski. Tandem-win graphs. *Discrete Mathematics*, 299(1):56–64, 2005.
- [20] D. Coudert, G. Ducoffe, and N. Nisse. Diameter of Minimal Separators in Graphs. Research Report RR-8639, Inria Sophia Antipolis ; I3S ; INRIA, Nov. 2014.
- [21] E. Dahlhaus, P. Hammer, F. Maffray, and S. Olariu. On domination elimination orderings and domination graphs. In *Graph-Theoretic Concepts in Computer Science, 20th International Workshop, WG '94, Herrsching, Germany, June 16-18, 1994, Proceedings*, pages 81–92, 1994.
- [22] R. Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.
- [23] Y. Dourisboure, F. Dragan, C. Gavoille, and Y. Chenyu. Spanners for bounded tree-length graphs. *Theor. Comput. Sci.*, 383(1):34–44, 2007.
- [24] Y. Dourisboure and C. Gavoille. Tree-decompositions with bags of small diameter. *Discrete Mathematics*, 307(16):2008–2029, 2007.
- [25] F. Dragan. Tree-like structures in graphs: A metric point of view. In *Graph-Theoretic Concepts in Computer Science - 39th International Workshop, WG 2013, Lübeck, Germany, June 19-21, 2013, Revised Papers*, pages 1–4, 2013.
- [26] F. Dragan and E. Köhler. An approximation algorithm for the tree t-spanner problem on unweighted graphs via generalized chordal graphs. *Algorithmica*, 69(4):884–905, 2014.
- [27] F. Dragan, E. Köhler, and A. Leitert. Line-distortion, bandwidth and path-length of a graph. In *Algorithm Theory–SWAT 2014*, pages 158–169. Springer, 2014.
- [28] F. Dragan and A. Leitert. On the minimum eccentricity shortest path problem. In *Algorithms and Data Structures*, pages 276–288. Springer, 2015.
- [29] F. Gavril. Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph. *SIAM Journal on Computing*, 1(2):180–187, 1972.
- [30] F. Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *Journal of Combinatorial Theory, Series B*, 16(1):47–56, 1974.
- [31] M. Golumbic, H. Kaplan, and R. Shamir. On the complexity of dna physical mapping. *Advances in Applied Mathematics*, 15(3):251–261, 1994.
- [32] M. Golumbic, H. Kaplan, and R. Shamir. Graph sandwich problems. *Journal of Algorithms*, 19(3):449–473, 1995.
- [33] M. Herlihy, F. Kuhn, S. Tirthapura, and R. Wattenhofer. Dynamic analysis of the arrow distributed protocol. *Theory of Computing Systems*, 39(6):875–901, 2006.
- [34] J. Hopcroft and R. Tarjan. Efficient planarity testing. *Journal of the ACM (JACM)*, 21(4):549–568, 1974.

-
- [35] P. Indyk. Algorithmic applications of low-distortion geometric embeddings. In *Proceedings of the 42Nd IEEE Symposium on Foundations of Computer Science, FOCS '01*, pages 10–, Washington, DC, USA, 2001. IEEE Computer Society.
- [36] T. Kashiwabara and T. Fujisawa. Np-completeness of the problem of finding a minimum-clique-number interval graph containing a given graph as a subgraph. In *Proc. Symposium of Circuits and Systems*, 1979.
- [37] A. Kosowski, B. Li, N. Nisse, and K. Suchan. k-Chordal Graphs: from Cops and Robber to Compact Routing via Treewidth. *Algorithmica*, 72(3):758–777, 2015.
- [38] R. Krauthgamer and J. Lee. Algorithms on negatively curved spaces. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 119–132. IEEE, 2006.
- [39] D. Lokshantov. On the complexity of computing treelength. *Discrete Applied Mathematics*, 158(7):820–827, 2010.
- [40] B. Monien. The bandwidth minimization problem for caterpillars with hair length 3 is np-complete. *SIAM Journal on Algebraic Discrete Methods*, 7(4):505–512, 1986.
- [41] J. Opatrny. Total ordering problem. *SIAM Journal on Computing*, 8(1):111–114, 1979.
- [42] C. Papadimitriou and M. Yannakakis. The clique problem for planar graphs. *Inf. Process. Lett.*, 13(4/5):131–133, 1981.
- [43] N. Robertson and P. Seymour. Graph minors. ii. algorithmic aspects of tree-width. *Journal of algorithms*, 7(3):309–322, 1986.
- [44] M. Sysło. Characterizations of outerplanar graphs. *Discrete Mathematics*, 26(1):47–53, 1979.
- [45] J. Tenenbaum, V. De Silva, and J. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.
- [46] K. Umezawa and K. Yamazaki. Tree-length equals branch-length. *Discrete Mathematics*, 309(13):4656–4660, 2009.



**RESEARCH CENTRE
SOPHIA ANTIPOLIS – MÉDITERRANÉE**

2004 route des Lucioles - BP 93
06902 Sophia Antipolis Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399

To approximate treewidth, use
treelength!

TO APPROXIMATE TREewidth, USE TREELENGTH!*

DAVID COUDERT^{†‡} AND GUILLAUME DUOFFE^{‡†} AND NICOLAS NISSE^{†‡}

Abstract. Tree-likeness parameters have proven their utility in the design of efficient algorithms on graphs. In this paper, we relate the *structural* tree-likeness of graphs with their *metric* tree-likeness. To this end, we establish new upper-bounds on the diameter of *minimal separators* in graphs. We prove that in any graph G , the diameter of any minimal separator S in G is at most $\lfloor \ell(G)/2 \rfloor \cdot (|S| - 1)$, with $\ell(G)$ the length of a longest isometric cycle in G . Our result relies on algebraic methods and on the cycle basis of graphs. We improve our bound for the graphs admitting a *distance preserving elimination ordering*, for which we prove that any minimal separator S has diameter at most $2 \cdot (|S| - 1)$.

We use our results to prove that the *treelength* $tl(G)$ of any graph G is at most $\lfloor \ell(G)/2 \rfloor$ times its *treewidth* $tw(G)$. In addition, we prove that, for any graph G that excludes an *apex graph* H as a minor, $tw(G) \leq c_H \cdot tl(G)$ for some constant c_H only depending on H . We refine this constant when G has bounded genus. Altogether, we obtain a simple $\mathcal{O}(\ell(G))$ -approximation algorithm for computing the treewidth of n -node apex-minor-free graphs in $\mathcal{O}(n^2)$ -time.

Key words. Graph; Treewidth; Treelength; Cycle basis; Genus.

AMS subject classifications. 05C85, 68Q17, 68Q25, 68R10

1. Introduction. It turns out that for a vast range of graph problems, the boundary between tractable and intractable cases depends on the *tree-like properties* of the graphs. This motivates us to study two tree-likeness invariants, that are called the *treewidth* [31] and the *treelength* [20]. Informally, the *width* of a tree-decomposition is the maximum size of its bags and its *length* is the maximum “diameter” of its bags. The treewidth and treelength of a graph are respectively the minimum width and length of its tree-decompositions (formal definitions can be found in Section 2). Note that bounded treelength graphs generalize the chordal graphs, split graphs, etc. which are well studied graph classes that have unbounded tree-width.

The treewidth aims to measure how close is the *structure* of a graph from the structure of a tree, whereas the treelength aims to measure the minimum distortion of the *distances* in a graph when it is embedded into a tree [1]. Since both invariants provide distinct, yet complementary, pieces of information on the closeness of a graph to a tree, we wish to relate treewidth and treelength through other graph properties so as to obtain a unifying view of tree-likeness in graphs.

Let us further motivate the need to compare treewidth with treelength, before presenting our main results.

One of the motivations is that we want to take the algorithmic advantages from both sides. Indeed, on the one hand there are many NP-hard problems that can be solved in polynomial-time on bounded-treewidth graphs [13]; on the other hand there exist compact routing schemes [7], approximation algorithms for packing, covering, and augmentation problems [10] up to an additive constant, as well as a PTAS for the well-known Traveling Salesman Problem on bounded-treelength graphs [26]. Therefore, finding relations between both invariants might lead to extend the use of some of the above-mentioned algorithms to a larger class of graphs. In particular,

*This work has been partially supported by ANR project Stint under reference ANR-13-BS02-0007, ANR program “Investments for the Future” under reference ANR-11-LABX-0031-01, and the Inria associated team AIDyNet.

[†]Inria, France

[‡]Univ. Nice Sophia Antipolis, CNRS, I3S, UMR 7271, 06900 Sophia Antipolis, France.

this might be beneficial to bounded-treelength graphs that are more common than bounded-treewidth graphs amongst the complex networks, but for which there seem to be less algorithmic applications. For instance, the graph of Autonomous Systems has large treewidth, while it is *hyperbolic* [1, 14], and so, it has bounded treelength [9].

Another advantage of relating treewidth and treelength is that such relations can improve the best-known complexities for their computation on certain graph classes. Indeed, despite the fact that treewidth and treelength are both NP-hard to compute [3, 28], treelength seems much easier to approximate than treewidth. Namely, there are 3-approximation algorithms for treelength that rely on a few breadth-first-searches [20], while the best known approximation algorithms for treewidth only achieve a ratio $\mathcal{O}(\sqrt{\log tw(G)})$ for general graphs —and a constant-ratio for minor-free graphs— through the use of semi-definite programming [22]. We thus highlight that by relating treelength with treewidth, one can obtain practical algorithms for approximating the treewidth on certain graph classes.

1.1. State of the art. To put our contributions in context, let it be said that treewidth and treelength are uncomparable in general. This fact prevents from expecting simple relations between both invariants.

On the one hand, it comes from the fact that the cycle C_n , with $n \geq 4$ vertices, satisfies that $tw(C_n) = 2$ while $tl(C_n) = \lceil n/3 \rceil$. This suggests that having a large treelength relies on the existence of long cycles in the graph. The authors in [20] supported this intuition, proving that the treelength of a graph G is bounded from above by half of the maximum length of a chordless cycle in G . Yet it is a strong constraint, as seen with the case of the wheel W_n which contains an induced C_n while it has treelength ≤ 2 . Therefore, it is natural to constrain ourselves to the subcase of *isometric* cycles in graphs. We remind that a subgraph H of G is isometric if for any two vertices of H , the distance between them is the same in H as in G . Note that it is known how to compute a longest isometric cycle in a graph in polynomial-time [27]. Unfortunately, there are graphs such as grids with bounded-length isometric cycles and arbitrarily large treelength. As we will show below, our results imply that in such a case, we always have that $tl(G) = \mathcal{O}(tw(G))$.

On the other hand, the complete graph with n vertices has treewidth $n - 1$ and treelength 1. Another interesting example is the graph H obtained by adding a universal vertex to a square-grid with n^2 vertices, for which it holds $tw(H) = n + 1$ and $tl(H) = 2$. Note that such graphs have a large *genus*, *i.e.*, they are in a sense arbitrarily far from planar graphs. In contrast, it holds that $tw(G) < 12 \cdot tl(G)$ for planar graphs [17]. Consequently, it is quite natural to ask whether a treewidth arbitrarily larger than the treelength requires a large genus. We will prove it is the case, *i.e.*, $tw(G) = \mathcal{O}(tl(G))$ for bounded-genus graphs.

Finally, and independently from this work, Belmonte et al. [5] proved that $tw(G) = \mathcal{O}(\Delta^{tl(G)})$ for any graph G with *maximum degree* Δ . They built upon this relation in order to design a fixed-parameter-tractable algorithm to compute the metric dimension on bounded treelength graphs. We will use different techniques in order to upper-bound the treewidth with linear dependency on the treelength.

1.2. Our contributions. We introduce a very generic method to upper-bound the *diameter*¹ of *minimal separators* in graphs, the latter denoting inclusion wise minimal subsets whose removal disconnects some fixed pair of vertices. Let us emphasize

¹The *diameter* of a set S of vertices of a graph G is the maximum distance in G between two vertices in S .

that the minimal separators are at the cornerstone of various graph decompositions, such as the so-called k -connected decompositions [24]. Therefore, our method may find applications beyond the scope of tree-decompositions.

In a few more details, we prove that minimal separators in a graph G induce connected subsets in some of its powers G^j , where j only depends on the length of cycles in some arbitrary *cycle basis* of G (see Section 2 for a formal definition). We deduce from our method that, for any graph G with longest isometric cycle of size $\ell(G)$, and for any minimal separator S in G , the diameter of S is at most $\lfloor \ell(G)/2 \rfloor \cdot (|S| - 1)$ and the upper-bound is sharp as shown by any cycle.

Then, we prove that for any graph G which is not a tree, $tl(G) \leq \lfloor \ell(G)/2 \rfloor \cdot (tw(G) - 1)$. This upper-bound on treelength follows from our upper-bound on the diameter of minimal separators, and it is tight up to a small constant-factor². We refine our bound in several particular graph classes (the formal definition of these classes are postponed to the technical sections of the paper).

- For any graph G in the class of null-homotopic graphs (including the class of dismantlable graphs), we prove that $tl(G) \leq tw(G)$. This is tight, as one can easily see on trees.
- In the class of graphs G that admit a distance preserving elimination ordering, we prove that $tl(G) < 2 \cdot tw(G)$.

We emphasize that the latter class contains the cobipartite graphs. Though the treelength of cobipartite graphs is trivially bounded by 3, computing their treewidth is an NP-hard problem. As a consequence, the treewidth of graphs that admit a distance preserving elimination ordering is also NP-hard to compute. Our results combined with the 3-approximation algorithm for treelength [20] provide a polynomial-time algorithm for computing a new non-trivial lower-bound for treewidth.

Finally, we consider lower-bounds for treelength. We prove that, for any graph excluding an *apex graph* H as a minor, there is a constant c_H such that $tw(G) \leq c_H \cdot tl(G)$. The constant c_H only depends on H . Our proofs in this part make use of the bidimensionality theory [15]. In the particular case of graphs with bounded genus $g > 0$, we use results from this theory so as to prove the more precise relation $tw(G) \leq 72\sqrt{2}(g+1)^{\frac{3}{2}} \cdot tl(G) + \mathcal{O}(g^2)$.

So, to sum up, we obtain that *any* approximation algorithm for treelength can be turned into an approximation algorithm for treewidth up to multiplying by an $\mathcal{O}(\ell(G) \cdot (g+1)^{\frac{3}{2}})$ the approximation ratio.

2. Preliminaries. In this section, we recall some useful definitions and known results that will be used in the sequel. All graphs considered in this paper are simple (*i.e.*, without loops or multiple edges), connected and finite. Given a graph $G = (V, E)$, the number $|V|$ of vertices will be denoted by n and the number of edges $|E|$ by m . For any vertex $v \in V$, let $N_G(v) = \{u \in V \mid \{u, v\} \in E\}$ be the set of neighbors of v in G . Let $N_G[v]$ denote $N_G(v) \cup \{v\}$.

Minimal separators. A set $S \subseteq V$ is a *minimal separator* if there exist $a, b \in V \setminus S$ such that any path from a to b intersects S and, for any proper subset $S' \subset S$, there is a path from a to b which does not intersect S' . We name any such a set S an *a-b*

²Recently and independently of this work (see research report in [12]), Diestel and Muller [19] proved that $tl(G) \leq \ell(G) \cdot (tw(G) - 1)$ (see also [2] for a slightly looser bound). Note that our upper-bound for treelength is sharper than theirs. Furthermore, unlike our results which apply to any minimal separator in a graph, theirs rely on minimal separators in a specific tree-decomposition called an atomic tree-decomposition.

minimal separator. A connected component $C \subseteq V \setminus S$ of $G[V \setminus S]$ is *full* with respect to S if every node in S has a neighbour in C . Any a - b minimal separator has at least two full components: the one containing a and the one containing b . Conversely, any separator having at least two full components is a minimal separator. A graph is said *well-connected* if each of its minimal separators induces a connected subgraph [21].

Cycle space. The set $\mathcal{C}(G)$ of Eulerian subgraphs of G is called the *cycle space* of G . It is well-known that every Eulerian subgraph can be obtained from the symmetric difference (on the edges) of cycles in G . In fact, the set $\mathcal{C}(G)$ with the symmetric difference is a vector space of dimension $m - n + 1$ if G is connected [18, Theorem 1.9.6]. We will call the symmetric difference of two subgraphs H_1, H_2 , denoted $H_1 \oplus H_2$, the *sum* of H_1 with H_2 . A cycle basis is an inclusion wise minimal set of cycles generating the whole cycle space. In particular, a graph is said *null-homotopic* if it has a cycle basis with only triangles.

THEOREM 2.1. [21] *Any connected null-homotopic graph is well-connected.*

In this paper, we will extend the class of null-homotopic graphs as follows.

DEFINITION 2.2. *Let $l \geq 3$. We define \mathcal{G}_l as the class of graphs whose cycle space can be generated by cycles of length at most l .*

Note that \mathcal{G}_3 is exactly the class of null-homotopic graphs. Moreover, the isometric cycles in a graph can generate its cycle space [25] (see also Exercice 1.32 of the textbook [18]), so the class \mathcal{G}_l contains all graphs with no isometric cycle longer than l . Therefore by varying the parameter l , classes \mathcal{G}_l include all graphs and they form an inclusion wise increasing hierarchy. By [11], the smallest integer $l \geq 3$ such that a graph belongs to \mathcal{G}_l can be computed in polynomial-time.

Diameter and Graph powers. For any $X \subseteq V$, let $\text{diam}_G(X)$ denote the maximum distance in G between any pair of vertices in X , *a.k.a.* the *diameter* of X . Last, for any $j \geq 1$, the graph G^j is obtained from G by adding an edge between any two distinct nodes that are at distance at most j in G .

2.1. Tree-decompositions.

Minimal fill-in. A graph is *chordal* if all its induced cycles have length at most 3. For any graph $G = (V, E)$, we define a *fill-in* of G as any chordal supergraph $H = (V, E \cup F)$ of G ³. A fill-in $H = (V, E \cup F)$ is *minimal* if, for any $f \in F$, the graph $H' = (V, E \cup F \setminus \{f\})$ is not chordal.

Let H be a fill-in of a graph G , and let V_C be the set of maximal cliques of H . A *clique-tree* of H is a tree $T_C = (V_C, F)$ such that for each vertex $x \in V$, the set of maximal cliques containing x induces a subtree of T_C . We define a (reduced) *tree-decomposition* of G as any clique-tree of an arbitrary fill-in of G . Equivalently, a tree-decomposition of G consists of a pair (T, \mathcal{X}) where T is a tree and $\mathcal{X} = (X_t)_{t \in V(T)}$ is a family of subsets of V , called *bags*, indexed by the nodes of T and that satisfies the following three properties.

1. $\bigcup_{t \in V(T)} X_t = V$;
2. for any $\{u, v\} \in E$, there is $t \in V(T)$ with $u, v \in X_t$;
3. for any $u \in V$, the set of bags containing u induces a subtree of T .

Tree-likeness invariants. Given a graph G , the *length* of a tree-decomposition (T, \mathcal{X}) equals the maximum diameter in G of its bags. The *treelength* of G , denoted by $tl(G)$, is the minimum length over all tree-decompositions of G . Equivalently, the treelength of G is the smallest integer j such that G^j contains a fill-in of G [28]. The *width* of (T, \mathcal{X}) equals the maximum size of its bags minus one. The *treewidth*

³Here we use the term fill-in to avoid confusion with planar triangulations.

of G , denoted by $tw(G)$, is the minimum width over all tree-decompositions of G . Equivalently, the treewidth of G is the minimum over all minimal fill-ins H of G of $\omega(H) - 1$, where $\omega(H)$ is the clique-number of H [6]. It can be checked that both invariants are *contraction-closed* i.e., the contraction of an edge in the graph cannot increase its treewidth nor its treelength. We will often use the fact that treewidth and treelength are contraction-closed invariants in the following.

Parallel minimal separators. Finally, let S_1, S_2 be two minimal separators in a graph G . The separator S_1 *crosses* S_2 if there are two components of $G \setminus S_2$ that S_1 intersects. If S_1 does not cross S_2 , then S_1 is said to be *parallel* to S_2 .

THEOREM 2.3. [30] *H is a minimal fill-in of the graph G if and only if H is obtained by completing⁴ all sets of a maximal set of pairwise parallel minimal separators in G .*

3. Diameter of Minimal Separators in Graphs. In this section, we show the diameter of any minimal separator S in a graph G is $\mathcal{O}(\ell(G) \cdot |S|)$, where $\ell(G)$ is the length of a longest isometric cycle in G (Theorem 3.4). We then strengthen our results in particular graph classes that are defined by the existence of some *elimination ordering* of their vertices.

Before going into the details of the proof, let us describe the main intuition behind it and the difficulties we had to face on. Let us consider a minimal separator S . If it is connected, then the result easily follows. Hence, we may assume S consists of several connected components. The idea is to find a set of paths, each of length at most $\lfloor \ell(G)/2 \rfloor$, such that any of these paths connects two components and the subgraph induced by S and these paths is connected. If we do so, the result easily follows. Hence, the main difficulty is to find such paths. For this purpose, let us consider a minimum-length cycle crossing two components of S (such a cycle surely exists because there are at least two full components in $G \setminus S$). If this cycle is isometric, then the distance between the two components cannot exceed $\lfloor \ell(G)/2 \rfloor$. Otherwise, it means that there is a shortcut between two nodes of the cycle. However, this shortcut could intersect S more than once which does not help our purpose. The key point is that, using the shortcut, the initial cycle can be viewed as the combination (symmetric difference) of two cycles. This kind of local view can be generalized to a global one using our main tool, namely the cycle basis. Indeed, the initial cycle is actually the symmetric difference of a set of isometric cycles [25, 18]. Using this set, we can then prove our theorem.

3.1. Case of general graphs. We start proving some properties of graphs in the class \mathcal{G}_l . This will lead us to the main result in this section (Theorem 3.4).

Let us first prove that the class \mathcal{G}_l is stable under the following two operations.

LEMMA 3.1. *Let $l \geq 3$, the class \mathcal{G}_l is stable under edge-contraction.*

Proof. Let $G \in \mathcal{G}_l$ with n vertices and m edges. W.l.o.g., G is connected. The dimension $\dim(\mathcal{C}(G))$ of the cycle space $\mathcal{C}(G)$ is $s = m - n + 1$ ([18, Theorem 1.9.6]). Let $e \in E(G)$ such that e lies on $k \geq 0$ triangles in G . By contracting e , we loose one vertex and $k + 1$ edges, the edge e and for each triangle which contains e we have to remove one of the resulting multi-edges. Hence, $\dim(\mathcal{C}(G/e)) = \dim(\mathcal{C}(G)) - k$. Let $\{C_1, \dots, C_s\}$ be a basis of $\mathcal{C}(G)$ such that each C_i has length at most l . Let $\{C'_1, \dots, C'_t\}$ be the set of cycles in G/e which are obtained by contracting e on each C_i and by removing triangles that contain e from the list. Then $t \geq \dim(\mathcal{C}(G/e)) = s - k$ (since at most k triangles have been removed) and each C'_i has length at most l . We

⁴Completing a set of vertices is to make the set a clique.

show that C'_1, \dots, C'_t are linearly independent in $\mathcal{C}(G/e)$, which proves that they form a basis of $\mathcal{C}(G/e)$. For purpose of contradiction, let us assume that $C'_{i_1} \oplus \dots \oplus C'_{i_r} = 0$ for $1 \leq i_1 < \dots < i_r \leq s$, $r > 0$. Then $C_{i_1} \oplus \dots \oplus C_{i_r}$ is either 0 or e . Therefore, the sum equals e since the C_{i_j} 's are linearly independent in $\mathcal{C}(G)$. This is a contradiction as $(V(G), \{e\})$ is not Eulerian.

Hence, since all cycles in the basis $\{C'_1, \dots, C'_t\}$ have length at most ℓ , it implies that $G/e \in \mathcal{G}_l$. \square

LEMMA 3.2. *Let G_1 and G_2 be two graphs such that $V(G_1) \cap V(G_2) = \{x, y\}$ and $E(G_1) \cap E(G_2) = \emptyset$, and let $G = G_1 \cup G_2 = (V(G_1) \cup V(G_2), E(G_1) \cup E(G_2))$. If $G_1, G_2 \in \mathcal{G}_l$ and $d_{G_1}(x, y) + d_{G_2}(x, y) \leq l$, then $G \in \mathcal{G}_l$.*

Proof. Let C be a cycle in G . We will prove that it is a sum of cycles of length at most l in G . If it is a cycle in G_1 (resp. in G_2), then we are done as it is the sum of cycles of length at most l by Definition 2.2. Else, it must contain the pair x, y and it can be decomposed into: a xy -path in G_1 , and a xy -path in G_2 . Let C_l be obtained from the union of a shortest xy -path in G_1 with a shortest xy -path in G_2 . Note that C_l has length $d_{G_1}(x, y) + d_{G_2}(x, y) \leq l$ by the hypothesis. Furthermore, $H = C \oplus C_l$ is an Eulerian subgraph of G . Let H_1, H_2 be the respective subgraphs of H that are induced by the edges in G_1, G_2 (possibly empty). Note that $E(H_1) \cap E(H_2) = \emptyset$ by construction. We claim that both graphs H_1, H_2 are Eulerian subgraphs. Indeed, on the one hand the subsets $V(H_1) \setminus \{x, y\}, V(H_2) \setminus \{x, y\}$ are disjoint by the hypothesis and so, any vertex $\neq x, y$ in one of these graphs, say in H_1 , has the same (even) degree in H_1 as in H . On the other hand, by construction each node amongst x, y is incident exactly to one edge in $E(C) \cap E(G_1)$ (resp. in $E(C) \cap E(G_2)$) and to one edge in $E(C_l) \cap E(G_1)$ (resp. in $E(C_l) \cap E(G_2)$). As a result, nodes x, y have degree either null or equal to 2 in H_1 , and similarly they have degree either null or equal to 2 in H_2 , which is even in both cases. Consequently, both H_1, H_2 are sums of cycles of length at most l by the hypothesis because they are respective Eulerian subgraphs of $G_1, G_2 \in \mathcal{G}_l$. Hence $H = H_1 \oplus H_2$ is also a sum of cycles of length at most l in G . This concludes the proof because $C = H \oplus C_l$. \square

Then, we prove that for any graph $G \in \mathcal{G}_l$, every minimal separator in G must contain a pair of vertices that are at small distance to each other.

LEMMA 3.3. *Let $l \geq 3$, let $G \in \mathcal{G}_l$ and let S be a minimal separator in G . Either S is a cut-vertex, or there are two distinct nodes $x, y \in S$ such that $d_G(x, y) \leq \lfloor l/2 \rfloor$.*

Proof. Suppose that S does not consist of a single cut-vertex. If the subgraph induced by S contains at least one edge $\{x, y\}$, then we are done as in such case $d_G(x, y) = 1 \leq \lfloor l/2 \rfloor$. So, we assume that S is a stable set. Let A, B be two distinct full components of $G \setminus S$ and let $s, t \in S$ be two distinct vertices. By connectivity, there is an st -path P whose internal vertices are contained in A , and in the same way there is a st -path Q whose internal vertices are contained in B . Let C be a cycle composed of P and Q . Because $G \in \mathcal{G}_l$, there is some set \mathcal{C} of cycles of length at most l whose sum equals C . We claim that there is a cycle $C' \in \mathcal{C}$ which intersects both A and another component of $G \setminus S$. Otherwise, because S is a stable set, the sum of all cycles that intersect A must generate P . This is not possible, since it is not Eulerian. As S separates the components, there are $x, y \in S \cap V(C')$ and so, since the length of C' is at most l , we deduce that $d_G(x, y) \leq \lfloor l/2 \rfloor$. \square

Finally, we can prove Theorem 3.4 and Corollary 3.5 below. Intuitively, we consider a pair of nodes $x, y \in S$, where S is a minimal separator in some graph $G \in \mathcal{G}_l$. If x, y are connected in the induced subgraph $G[S]$, then it is clear that their distance in G is at most $|S| - 1$. Else, we prove that there is a xy -path that intersects

the connected components C_1, C_2, \dots, C_k of $G[S]$ consecutively, and that satisfies: $x \in C_1, y \in C_k$, and $\forall 1 \leq i < k$, there exists a cycle of length at most l which intersects both C_i and C_{i+1} . Every two consecutive components C_i, C_{i+1} of $G[S]$ are thus at distance at most $\lfloor l/2 \rfloor$ in G , hence their union $C_i \cup C_{i+1}$ induces a connected subgraph of the power $G^{\lfloor l/2 \rfloor}$.

THEOREM 3.4. *Let $l \geq 3$. For any graph $G \in \mathcal{G}_l$, every minimal separator in G induces a connected subgraph in the power $G^{\lfloor l/2 \rfloor}$.*

Proof. By contradiction, let $G \in \mathcal{G}_l$, and let S be a minimal separator in G that does not satisfy the property. We first make adjacent every two vertices in S that are at distance at most $\lfloor l/2 \rfloor$ in G . We claim that the resulting graph is still in \mathcal{G}_l . Indeed, let $x, y \in S$ be non-adjacent and at distance at most $\lfloor l/2 \rfloor$ in G , let $G_1 = G$ and let G_2 be the complete graph on the two vertices x, y (i.e., G_2 is isomorphic to K_2). Since we have that $G_1 \in \mathcal{G}_l$ by the hypothesis, that $G_2 \in \mathcal{G}_3 \subseteq \mathcal{G}_l$ and that $d_{G_1}(x, y) + d_{G_2}(x, y) \leq \lfloor l/2 \rfloor + 1 \leq l$, then we deduce from Lemma 3.2 that $G_1 \cup G_2 \in \mathcal{G}_l$. The same argument can be applied iteratively because adding an edge in G cannot increase the distances between nodes in S . So, the claim is proved. Finally, we contract each connected component of the subgraph induced by S in a single node, thus contracting S to obtain a stable set S' , and the resulting graph G' still belongs to \mathcal{G}_l by Lemma 3.1. Furthermore, the stable set S' is a minimal separator in G' by construction. Since S does not satisfy the property of the theorem, we have that all nodes in S' are pairwise at distance at least $\lfloor l/2 \rfloor + 1$, but then it contradicts Lemma 3.3. \square

Theorem 3.4 is tight, as it can be shown with any cycle C_l .

COROLLARY 3.5. *Let G be a graph that is not a tree. Any minimal separator S in G has diameter at most $\lfloor \ell(G)/2 \rfloor \cdot (|S| - 1)$, where $\ell(G)$ denotes the length of a longest isometric cycle in G .*

Proof. It follows from Theorem 3.4 combined with the fact that isometric cycles generate the cycle space [25, 18]. \square

3.2. Graphs with distance-preserving elimination ordering. We strengthen the result of Corollary 3.5 in the case of graphs with a distance-preserving elimination ordering. Formally, we say that G admits a *distance-preserving elimination ordering* if there exists a total order of V , denoted by v_1, v_2, \dots, v_n , such that, for any $1 \leq i \leq n$, the subgraph $G_i = G \setminus \{v_1, \dots, v_i\}$ is isometric. Graphs with a distance-preserving elimination ordering arise from applications in graph searching (e.g., dismantlable graphs [29]) and geometry [8]. Note that they contain the class of cobipartite graphs, for which computing the treewidth is NP-hard. Our main result in this section is that for every graph G with a distance-preserving elimination ordering, it holds that $G \in \mathcal{G}_4$.

PROPOSITION 3.6. *A graph G that admits a distance-preserving elimination ordering has its cycle space generated by all its triangles and quadrangles.*

Proof. We claim that it is enough to prove that the induced cycles of G can be generated by all its triangles and quadrangles. Indeed, the induced cycles of G generate its cycle space [18]. Let v_1, v_2, \dots, v_n be a distance-preserving elimination ordering of G . By contradiction, amongst all induced cycles that do not satisfy the property let C maximize the smallest index j such that $v_j \in C$. Note that C is a cycle of $G_{j-1} = G[\{v_j, \dots, v_n\}]$ by the hypothesis. Moreover, all cycles contained in G_j are the sum of triangles and quadrangles of G because of the maximality of index j . Let $x, y \in V(C)$ be the two neighbours of v_j in cycle C . By the hypothesis, x, y are not adjacent because C is induced. So, because $x, y, v_j \in G_{j-1}$ which has a distance-

preserving elimination ordering, there is $v_i, i > j$ such that x, y are adjacent to v_i . Moreover, $v_i \notin C$ because otherwise C would be the quadrangle (v_j, x, v_i, y, v_j) , thus contradicting the fact that it does not satisfy the property. As a result, $C = Q \oplus C'$, with Q the quadrangle (v_j, x, v_i, y, v_j) and C' is the cycle of G_j obtained from C by replacing the path x, v_j, y with x, v_i, y . Furthermore, cycle C' is a sum of induced cycles of G_j that are themselves a sum of triangles and quadrangles by maximality of j . Hence so is cycle C , which contradicts the fact that it does not satisfy the property. \square

COROLLARY 3.7. *Let G be a graph that admits a distance-preserving elimination ordering. Every minimal separator S in G has diameter at most $2 \cdot (|S| - 1)$.*

Given that the cycle of length four C_4 admits a distance-preserving elimination ordering, one can see that Corollary 3.7 is sharp.

Dismantlable graphs are an interesting subclass of graphs with a distance-preserving elimination ordering. Formally, a graph G is dismantlable if, for any $1 \leq i < n$, there exists $j > i$, such that $N_G[u_i] \setminus \{u_1, \dots, u_{i-1}\} \subseteq N_G[u_j]$. It is immediate that if a graph is dismantlable, then it admits a distance-preserving elimination ordering. We obtain the following improvement over Proposition 3.6 for the subclass of dismantlable graphs.

LEMMA 3.8. *A dismantlable graph is null-homotopic and so, well-connected.*

Proof. Let G be a dismantlable graph. We prove that cycles of G can be generated by its triangles, which proves that G is null-homotopic. The fact that G is well-connected follows from the fact that dismantlable graphs are connected and from Theorem 2.1.

It is enough to prove that all induced cycles can be generated by triangles. Let (u_1, u_2, \dots, u_n) be a dismantling ordering of G . By contradiction, amongst all induced cycles that do not satisfy the property, let C maximize the smallest index j such that $u_j \in C$. Let $x, y \in V(C)$ be the two neighbours of u_j in cycle C , and let u_i , with $i > j$, be a dominator of u_j in G_{j-1} . We have that $u_i \notin C$ because C is induced and it has length at least 4 by the hypothesis. As a result, $C = T_1 \oplus T_2 \oplus C'$, with T_1 the triangle induced by nodes u_i, x, u_j ; with T_2 the triangle induced by nodes u_i, y, u_j ; and with C' a cycle of G_j obtained from C by replacing the path x, u_j, y with x, u_i, y . Furthermore, cycle C' is a sum of induced cycles of G_j that are themselves a sum of triangles of G by maximality of j . Hence, so is cycle C , which contradicts the fact that it does not satisfy the property. \square

We note that it was already noticed in [21] that dismantlable graphs are null-homotopic. However the proof was left to the reader. We give it in the paper for self-containment.

COROLLARY 3.9. *Let G be a dismantlable graph. Every minimal separator S in G has diameter at most $|S| - 1$.*

Last, we point out that by a result from [4], every graph is an isometric subgraph of some dismantlable graph. Therefore, there are graphs with arbitrarily long isometric cycles that admit a distance-preserving elimination ordering.

4. Relating treewidth with treelength.

4.1. Upper-bounds for treelength. Using the results recalled in Section 2.1, we are now able to upper-bound the treelength of a graph by a linear function depending on the size of its minimal separators. We then show that the treelength of a

graph is upper-bounded by a function that is linear in its treewidth.

LEMMA 4.1. *Let G be a graph and \mathcal{S} be a maximal set of pairwise parallel minimal separators in G . If there is a constant $c_{\mathcal{S}}$ such that $\text{diam}_G(S) \leq c_{\mathcal{S}}(|S| - 1)$ for all $S \in \mathcal{S}$, then $tl(G) \leq \max\{1\} \cup \{c_{\mathcal{S}} \cdot (|S| - 1) \mid S \in \mathcal{S}\}$.*

Proof. Let H be the supergraph of G obtained by completing all sets of \mathcal{S} . By Theorem 2.3, H is a minimal fill-in of G . Moreover, any clique-tree T_C of H corresponds to a reduced tree-decomposition of G where each clique of H induces a bag. Let Ω be any maximal clique in H , i.e., Ω is any bag of the tree-decomposition T_C . Let $x, y \in \Omega$. By definition of H , either $\{x, y\} \in E(G)$ or there is a minimal separator $S \in \mathcal{S}$ that contains both x and y . In the latter case, $d(x, y) \leq \text{diam}_G(S) \leq c_{\mathcal{S}} \cdot (|S| - 1)$. \square

THEOREM 4.2. *If every minimal separator in a graph G induces a connected subgraph in its power G^j , then $tl(G) \leq \max\{1, j \cdot (tw(G) - 1)\}$.*

Proof. Let H be a minimal fill-in of G with maximum clique-size $tw(G) + 1$. By Theorem 2.3, there is a maximal set \mathcal{S} of pairwise parallel minimal separators of G such that H results from the completion of all elements in \mathcal{S} . Note that any $S \in \mathcal{S}$ induces a minimal separator in H that is a clique —*a.k.a.* a clique-minimal separator in H — and therefore S is strictly contained in a maximal clique in H . Hence, $\max_{S \in \mathcal{S}} |S| \leq tw(G)$. By Lemma 4.1, $tl(G) \leq \max\{1\} \cup \{j \cdot (|S| - 1) \mid S \in \mathcal{S}\} \leq \max\{1, j \cdot (tw(G) - 1)\}$. \square

COROLLARY 4.3. *Let G be a connected graph which is not a tree, then $tl(G) \leq c_G \cdot (tw(G) - 1)$, where:*

- $c_G = 2$ if G admits a distance-preserving elimination ordering;
- $c_G = \lfloor \ell(G)/2 \rfloor$, with $\ell(G)$ the length of a longest isometric cycle in G .

Proof. First item follows from Proposition 3.6 combined with Theorem 3.4 and Theorem 4.2. Second item follows from Theorem 3.4 combined with Theorem 4.2. \square

We emphasize that it is NP-hard to compute the treelength of a graph [28], but there exist 3-approximation algorithms to compute it in polynomial-time [20]. Moreover, a longest isometric cycle in a graph can also be computed in polynomial-time [27]. Hence, the previous result gives a new way to compute lower-bounds for treewidth.

4.2. Lower-bound in case of bounded-genus graphs. In this section, we prove that the treewidth of a graph is upper-bounded by a function of its treelength and of its genus. Our result is mainly based on the result from [16] stating that any graph with large treewidth and genus contains a large “grid-like” graph as a contraction. We use their terminology.

Let us remind that a planar triangulation of a planar graph G is a planar supergraph of G whose faces are bounded by triangles. A *partially triangulated $(r \times r)$ -grid* is any graph that contains an $(r \times r)$ -grid as a subgraph and is a subgraph of some planar triangulation of the same $(r \times r)$ -grid. A (r, k) -*gridoid* G is a partially triangulated $(r \times r)$ -grid in which k extra edges have been added⁵.

THEOREM 4.4. [16] *Let G be a graph with genus g and $tw(G) > 4k(g + 1)$ with $k \geq 12g$, then G contains a $(k - 12g, g)$ -gridoid as a contraction.*

We prove that such a gridoid has large treelength and so, since the treelength is contraction-closed, such a graph has large treelength too.

LEMMA 4.5. *Let G be a partially triangulated $(r \times r)$ -grid, then $tl(G) \geq \lfloor r/3 \rfloor - 1$.*

⁵Note that the notion of (r, k) -gridoid is more general in [16].

Proof. The result holds if $r \leq 3$ because in such a case $tl(G) \geq 1 \geq \lfloor r/3 \rfloor - 1$. Else, let G' be the $(r \times r)$ -grid from which G is obtained by planar triangulation. Let V' be the set of vertices that are at distance at least $\lfloor \frac{r-1}{3} \rfloor$ from the external face of G' . The vertices of V' induce a partially triangulated $(r' \times r')$ -grid F in G , $r = 2 \lfloor \frac{r-1}{3} \rfloor + r'$, such that the external face has not been triangulated. Moreover, F is isometric in G . Hence, $tl(G) \geq tl(F)$. We show that $tl(F) \geq \lfloor r/3 \rfloor - 1$.

Our proof adapts from the lower-bound techniques in [20, Sec. 2.3]. Let (T, \mathcal{X}) be any tree-decomposition of F . Consider the two subsets of vertices A, B that contain the first and the last row of F respectively. Since A induces a connected subgraph of F , by the properties of tree-decompositions the bags in \mathcal{X} that intersect A form a subtree T_A of T . Similarly, the bags in \mathcal{X} that intersect B form a subtree T_B of T . Furthermore, either $T_A \cap T_B \neq \emptyset$ (in which case, the diameter of every bag in $T_A \cap T_B$ is at least $r' - 1$), or by [20, Lemma 5] there exists a bag which intersects all paths between A and B in F . In the latter case, such bag must intersect the first and last column of F , and so, it has diameter at least $r' - 1$. Therefore, (T, \mathcal{X}) has length at least $r' - 1$ in both cases, that proves that $tl(F) \geq r' - 1 \geq \lfloor r/3 \rfloor - 1$. \square

LEMMA 4.6. *Let G be a (r, k) -gridoid, then $tl(G) > r/(18\sqrt{2k+1}) - 2$.*

Proof. The result holds if $r \leq 36\sqrt{2k+1}$ because in such case $tl(G) \geq 1 > r/(18\sqrt{2k+1}) - 2$. Hence, let us assume that $r > 36\sqrt{2k+1}$.

Let M be a set of at most k edges whose removal in G yields a partially triangulated $(r \times r)$ -grid. Let $S = V(M)$ be the set of end-vertices of the edges of M . Note that $|S| \leq 2k$. Also, let G' be the $(r \times r)$ -grid whose $G \setminus M$ is a partial planar triangulation. Let finally $4 \leq x \leq r$ be an integer. There are $(r-x+1)^2$ distinct $(x \times x)$ -grids as subgraphs in G' , that give us as many distinct partially triangulated $(x \times x)$ -grids as subgraphs in G . Furthermore, each node in S belongs to at most x^2 such subgraphs. Therefore assuming $(r-x+1)^2 - 2k \cdot x^2 \geq 1$, there is one of these partially triangulated $(x \times x)$ -grids, say H , that does not contain any node incident to one of the k extra edges. Consider the partially triangulated $(x' \times x')$ -grid R which is in the center of H , with $x = 2 \cdot \lfloor \frac{x-1}{3} \rfloor + x'$. That is, R is a subgraph of H and any node of R is at distance at least $\lfloor \frac{x-1}{3} \rfloor$ from a node of $G \setminus H$ (it is possible because H does not contain an extremity of an extra edge). Therefore, R is isometric in G and $tl(R) \leq tl(G)$. By Lemma 4.5,

$$tl(R) \geq \lfloor x'/3 \rfloor - 1 \geq x/9 - 1.$$

It remains to maximize x satisfying the inequality $(r-x+1)^2 - 2k \cdot x^2 \geq 1$ so that we maximize the above lower-bound for $tl(R)$. The polynomial

$$\begin{aligned} (r-X+1)^2 - 2k \cdot X^2 - 1 &= r^2 + X^2 + 1 - 2r \cdot X + 2r - 2X - 2k \cdot X^2 - 1 \\ &= -[(2k-1) \cdot X^2 + 2(r+1) \cdot X - r(r+2)] \end{aligned}$$

has for reduced discriminant $(r+1)^2 + r(r+2)(2k-1) = 2k \cdot r(r+2) + 1$, hence its roots are equal to

$$\left\{ -\frac{\sqrt{2k \cdot r(r+2) + 1} + r + 1}{2k - 1}, \frac{\sqrt{2k \cdot r(r+2) + 1} - r - 1}{2k - 1} \right\}.$$

Since this polynomial is nonnegative only between its roots, the value maximizing x

is:

$$\begin{aligned}
x_0 &= \left\lfloor \frac{\sqrt{2k \cdot r(r+2) + 1} - r - 1}{2k - 1} \right\rfloor \geq \frac{\sqrt{2k \cdot r(r+2) + 1} - r - 1}{2k - 1} - 1 + \frac{1}{2k - 1} \\
&= \frac{r(r+2)}{\sqrt{2k \cdot r(r+2) + 1} + r + 1} - 1 + \frac{1}{2k - 1} > \frac{r(r+2)}{2\sqrt{2k \cdot r(r+2) + 1}} - 1 \\
&> \frac{1}{2} \sqrt{\frac{r(r+2)}{2k+1}} - 1 > \frac{r}{2\sqrt{2k+1}} - 1.
\end{aligned}$$

Hence, $tl(G) \geq tl(R) \geq x_0/9 - 1 \geq r/(18\sqrt{2k+1}) - 2$. \square

THEOREM 4.7. *Let G be a graph with genus g and $tw(G) > 4k(g+1)$ with $k \geq 12g$. Then*

$$tw(G) \leq 72\sqrt{2}(g+1)^{\frac{3}{2}} \cdot tl(G) + \mathcal{O}(g^2).$$

Proof. By Theorem 4.4, G contains a $(k - 12g, g)$ -gridoid R as a contraction. By Lemma 4.6,

$$tl(R) > \frac{k - 12g}{18\sqrt{2g+1}} - 2.$$

Thus, by setting $k = (tw(G) - 1)/(4(g+1))$, we obtain that:

$$tl(R) > \frac{tw(G) - 48g(g+1) - 1}{72(g+1)\sqrt{2g+1}} - 2 > \frac{tw(G)}{72\sqrt{2}(g+1)^{\frac{3}{2}}} - \frac{\sqrt{2}}{3} \cdot \sqrt{g+1} - 3.$$

The result then follows from the fact that treelength is contraction-closed. \square

Extensions.. Theorem 4.7 can be extended to the broader class of apex-minor-free graphs. An *apex graph* is a graph such that the removal of one vertex creates a planar graph. Similar techniques from the bidimensionality theory allow us to deal with graphs that exclude a fixed apex graph as a minor. Namely, we will make use of the graph Γ_k as it is defined in [23]. The graph Γ_k is obtained from a $(k \times k)$ -grid by triangulating its internal faces such that all internal vertices become of degree 6, all non-corner external vertices are of degree 4, and then one corner of degree two is joined by edges with all vertices of the external face.

THEOREM 4.8. [23] *For every apex graph H , there is a constant $c_H > 0$ such that every connected H -minor-free graph of treewidth at least $c_H \cdot k$ contains Γ_k as a contraction.*

THEOREM 4.9. *Let H be any apex graph and G be a connected H -minor-free graph of treewidth at least $c_H \cdot k$, where c_H is the constant of Theorem 4.8. Then $tl(G) \geq tw(G)/(3 \cdot c_H) - 1$.*

Proof. By Theorem 4.8, G contains Γ_k as a contraction. Moreover, Γ_k is a partially triangulated grid. The result follows from Lemma 4.5 and the fact that treelength is contraction-closed. \square

5. Conclusion. We can deduce from Corollary 3.5 and Theorem 4.7 that for every n -node graph of genus g , the 3-approximation algorithms for treelength in [20] compute in $\mathcal{O}(g \cdot n^2)$ -time an integer t^* satisfying:

$$\frac{tw(G)}{72\sqrt{2}(g+1)^{\frac{3}{2}}} - \frac{\sqrt{2}}{3} \cdot \sqrt{g+1} - 3 \leq t^* \leq 3 \lceil \ell(G)/2 \rceil \cdot tw(G).$$

Observe that in case an upper-bound on the treewidth is given, we can also deduce from our relations a lower-bound on the graph genus.

The main drawback with our above approximation algorithm for treewidth is that it may output a tree-decomposition with unbounded width (the length is upper-bounded by t^*). We let open whether our method can be modified so that it outputs a tree-decomposition of width $\mathcal{O}(\ell(G) \cdot (g + 1)^{3/2} \cdot t^*)$.

Acknowledgments. We wish to thank the referees for their careful reading of the first version of this manuscript, and their useful comments. Their remarks and suggestions have improved the presentation of this paper significantly.

REFERENCES

- [1] MUAD ABU-ATA AND FEODOR F. DRAGAN, *Metric tree-like structures in real-world networks: an empirical study*, Networks, 67 (2016), pp. 49–68.
- [2] AARON B. ADCOCK, BLAIR D. SULLIVAN, AND MICHAEL W. MAHONEY, *Tree decompositions and social graphs*, arXiv preprint arXiv:1411.1546, (2014).
- [3] STEFAN ARNBORG, DEREK G. CORNEIL, AND ANDRZEJ PROSKUROWSKI, *Complexity of finding embeddings in a k -tree*, SIAM J. Algebraic Discrete Methods, 8 (1987), pp. 277–284.
- [4] HANS-JÜRGEN BANDELT AND VICTOR CHEPOI, *Metric graph theory and geometry: a survey*, Contemporary Mathematics, 453 (2008), pp. 49–86.
- [5] RÉMY BELMONTE, FEDOR V. FOMIN, PETR A. GOLOVACH, AND M. S. RAMANUJAN, *Metric dimension of bounded width graphs*, in Mathematical Foundations of Computer Science 2015: 40th International Symposium, MFCS 2015, Milan, Italy, August 24–28, 2015, Proceedings, Part II, 2015, pp. 115–126.
- [6] HANS L. BODLAENDER, *A partial k -arboretum of graphs with bounded treewidth*, Theor. Comput. Sci., 209 (1998), pp. 1–45.
- [7] MARIÁN BOGUNA, FRAGKISKOS PAPAIOPOULOS, AND DMITRI KRIOUKOV, *Sustaining the Internet with hyperbolic mapping*, Nature Communications, 1 (2010), pp. 1–18.
- [8] JÉRÉMIE CHALOPIN, VICTOR CHEPOI, HIROSHI HIRAI, AND DAMIAN OSAJDA, *Weakly modular graphs and nonpositive curvature*, arXiv preprint arXiv:1409.3892, (2014).
- [9] VICTOR CHEPOI, FEODOR DRAGAN, BERTRAND ESTELLON, MICHEL HABIB, AND YANN VAXÈS, *Diameters, centers, and approximating trees of δ -hyperbolic geodesic spaces and graphs*, in Proceedings of the twenty-fourth annual symposium on Computational geometry, ACM, 2008, pp. 59–68.
- [10] VICTOR CHEPOI AND BERTRAND ESTELLON, *Packing and covering δ -hyperbolic spaces by balls*, in Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, vol. 4627 of Lecture Notes in Computer Science, Springer, 2007, pp. 59–73.
- [11] DAVID M CHICKERING, DAN GEIGER, AND DAVID HECKERMAN, *On finding a cycle basis with a shortest maximal cycle*, Information Processing Letters, 54 (1995), pp. 55–58.
- [12] DAVID COUDERT, GUILLAUME DUCOFFE, AND NICOLAS NISSE, *Diameter of Minimal Separators in Graphs*, Research Report RR-8639, Inria Sophia Antipolis ; I3S, nov 2014. <https://hal.inria.fr/hal-01088423>.
- [13] BRUNO COURCELLE, *The monadic second-order logic of graphs. I. recognizable sets of finite graphs*, Information and Computation, 85 (1990), pp. 12 – 75.
- [14] FABIEN DE MONTGOLFIER, MAURICIO SOTO, AND LAURENT VIENNOT, *Treewidth and hyperbolicity of the internet*, in 10th IEEE International Symposium on Network Computing and Applications (NCA), Boston, 2011, IEEE, pp. 25–32.
- [15] ERIK D. DEMAINE AND MOHAMMAD TAGHI HAJIAGHAYI, *The bidimensionality theory and its algorithmic applications*, Comput. J., 51 (2008), pp. 292–302.
- [16] ERIK D. DEMAINE, MOHAMMAD TAGHI HAJIAGHAYI, AND DIMITRIOS M. THILIKOS, *The bidimensional theory of bounded-genus graphs*, SIAM J. Discrete Math., 20 (2006), pp. 357–371.
- [17] YOUSOU DIENG AND CYRIL GAVOILLE, *On the tree-width of planar graphs*, Electronic Notes in Discrete Mathematics, 34 (2009), pp. 593–596.
- [18] REINHARD DIESTEL, *Graph theory*, Heidelberg, Graduate Texts in Mathematics, 173 (2010), p. 451 pp. 4th edition.
- [19] REINHARD DIESTEL AND MALTE MÜLLER, *Connected tree-width*, nov 2014. <http://arxiv.org/abs/1211.7353>.

- [20] YON DOURISBOURE AND CYRIL GAVOILLE, *Tree-decompositions with bags of small diameter*, Discrete Mathematics, 307 (2007), pp. 2008–2029.
- [21] PIERRE DUCHET, MICHEL LAS VERGNAS, AND HENRY MEYNIEL, *Connected cutsets of a graph and triangle bases of the cycle space*, Discrete Mathematics, 62 (1986), pp. 145–154.
- [22] URIEL FEIGE, MOHAMMADTAGHI HAJIAGHAYI, AND JAMES R. LEE, *Improved approximation algorithms for minimum weight vertex separators*, SIAM J. Comput., 38 (2008), pp. 629–657.
- [23] FEDOR V. FOMIN, PETR A. GOLOVACH, AND DIMITRIOS M. THILIKOS, *Contraction obstructions for treewidth*, J. Comb. Theory, Ser. B, 101 (2011), pp. 302–314.
- [24] WALTER HOHBERG, *The decomposition of graphs into k -connected components*, Discrete mathematics, 109 (1992), pp. 133–145.
- [25] JOHN D. HORTON, *A polynomial-time algorithm to find the shortest cycle basis of a graph*, SIAM Journal on Computing, 16 (1987), pp. 358–366.
- [26] ROBERT KRAUTHGAMER AND JAMES R. LEE, *Algorithms on negatively curved spaces*, in Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on, IEEE, 2006, pp. 119–132.
- [27] DANIEL LOKSHTANOV, *Finding the longest isometric cycle in a graph*, Discrete Applied Mathematics, 157 (2009), pp. 2670–2674.
- [28] ———, *On the complexity of computing treelength*, Discrete Applied Mathematics, 158 (2010), pp. 820–827.
- [29] RICHARD NOWAKOWSKI AND PETER WINKLER, *Vertex-to-vertex pursuit in a graph*, Discrete Mathematics, 43 (1983), pp. 235–239.
- [30] ANDREA PARRA AND P. SCHEFFLER, *Characterizations and algorithmic applications of chordal graph embeddings*, Discrete Applied Mathematics, 79 (1997), pp. 171–188.
- [31] NEIL ROBERTSON AND PAUL D. SEYMOUR, *Graph minors. II. algorithmic aspects of tree-width*, Journal of Algorithms, 7 (1986), pp. 309–322.

Papers on coloring games

APPENDIX I

The Complexity of Hedonic
coalitions under bounded
cooperation

The Complexity of Hedonic Coalitions under Bounded Cooperation *

G. Ducoffe^{1,3}, D. Mazauric^{2,3}, and A. Chaintreau³

¹Université Côte d'Azur, Inria, CNRS, I3S, France

²Inria Sophia Antipolis - Méditerranée, F-06902 Sophia Antipolis, France

³Columbia University in the City of New York, U.S.A.

Abstract

We consider additively separable symmetric hedonic games, that are a specific type of hedonic coalition formation in which players value joining a given group solely based on its members. In graph-theoretic terms, the game is defined by an edge-weighted complete graph, and a configuration of the game is a vertex-partition of this graph. The utility of a given vertex v w.r.t. the configuration is the sum of the weights of the edges that are incident to v and another vertex in the same group as v . Then, given an integer $k \geq 1$, a better-response dynamics for the hedonic game previously mentioned consists in iterating the following operation, until it is no more possible to do so. We move at most k vertices to a same group of the partition, provided they all increase their utility in the process.

We study the complexity of deciding whether a given such game admits an equilibrium that is robust to coalition of at most k players, where k is a fixed constant. In some cases where an equilibrium is known to exist, referred to as *colouring games* in the literature, we also study the time of convergence of the better-response dynamics toward an equilibrium.

Our main result proves how even a minimal amount of cooperation between players makes finding the final outcome of such games prohibitively difficult. This applies even in the simpler case of colouring games, for which it was previously conjectured that the better-response dynamics always converge in polynomial time, while our tight analysis proves the opposite. Precisely, our main result is that, already for $k = 4$, the better-response dynamics for colouring games converges in $\Omega(n^{\Theta(\ln(n))})$ steps. Furthermore, we show that minimum addition to the simplest game reinforces this view, making the mere existence of a stable outcome prohibitively hard to decide. That is, when all edge weights are in a fixed subset \mathcal{W} , either an equilibrium always exists or it is NP-complete to decide on its existence.

While our results may be interpreted as a criticism of solution concepts involving bounded cooperation in general, we also present conditions (based on the edge weights) under which the complexity of this form of cooperation remains tractable.

Coloring games; Hedonic Games; Equilibrium computation; Better-response dynamics; Graph theory; NP-hardness; Social networks.

1 Introduction

We consider process by which players of a game choose to belong to a group solely based on the members that it contains, as a way to model multiple situations pertaining to creation of

*This work has been partially supported by ANR program “Investments for the Future” under reference ANR-11-LABX-0031-01.

coalitions. Below we introduce a model that generalizes previous games and dynamics related to information sharing [22], graph colouring [11, 26], creation of clubs and societies [4, 8] referred to as Hedonic [13]. Roughly in *Hedonic Games*, the following natural assumption is made: a player’s appreciation of a coalition structure only depends on the members of the coalition she belongs to, and not on how players in other coalitions are arranged [13]. Our goal, like multiple works before [3, 4, 8, 16, 24, 25, 31], is to determine the merit of a solution concept for revealing stable and meaningful structures as final outcome of this game. However, unlike previous studies, we carefully control the level of cooperation among players to see how it impacts the complexity of computing an equilibrium and the convergence of multiple better response dynamics.

We consider the effect of two important features, only found in [22]: *Incompatibility* between some pairs of players, in the form of arbitrarily high penalty, is here to ensure that the dynamics of the game always avoid cases where any such two players set are in the same coalition. *Bounded cooperation of size k* stipulates that a new set of coalitions can be formed whenever a subset containing at most k players can move simultaneously so that each player in the subset benefits from this move. Computing stable outcomes of this game for any value of k allows one to gradually compare configurations that are individually stable ($k = 1$) — and hence attainable as Nash Equilibrium of a non-cooperative dynamics — with those that satisfy a stronger notion of stability ($k > 1$). This eventually culminates (when k is equal to the number of players) in finding exactly the configurations that remain stable even against any coalition of players. Such configurations are sometimes referred to as strong Nash Equilibrium, coalition proof equilibrium, or core stable coalition, as they relate to those in the core of an associated cooperative game.

On the one hand, it seems intuitive that, when it comes to understand the formation of clubs or groups, a purely individually rational solution concept (corresponding to $k = 1$) is rather limiting. On the other hand, allowing cooperation between players is expected to increase dynamics’ complexity. Prohibitive complexity does not only come as computational cost for prediction, it also casts a doubt that the solution concept represents well the outcome of a process of decisions made by players. We hence aim at determining the conditions under which this complexity remains tractable (*i.e.*, a final outcome is reached by the dynamics in a number of steps that grows no more than a polynomial function of the number of players, where ideally this polynomial function grows not too large). It appears that a limited form of cooperation (*i.e.*, $k < \infty$) helps to reduce complexity as long as k is chosen small enough. As an example, the authors in [22] show that for the simplest version of the game above (that we call uniform), simple dynamics converge in polynomial time when $k = 1, 2$ and 3. They also conjecture that a similar property extends when $k > 3$. In this article, we disprove the above conjecture and shows that $k = 3$ is the maximum level of cooperation ensuring polynomial convergence for simple dynamics. Furthermore if $k \leq 2$ our novel analysis does not only provide the first non-trivial lower bounds but it determines the exact worst case convergence time in a closed form.

1.1 Related work

The role of coalitions – and under which conditions on player’s incentive those can be sustained – has been a premier theme of Game Theory since its beginning. Ever since, it has played a key role in determining the theoretical and practical merits of solution concepts [1, 6, 7, 23, 28]. Most of the earliest works center on situations in which utility can be transferred among players. That leads to the possibility of bargaining and hence reach specific equilibria (Nash, Core, Strong Nash) depending on the axioms applying to players’ expected behavior in similar situations. With a slight simplification, one can summarize the role of cooperation among players in this context as a major factor increasing complexity: indeed, determining that a Nash equilibrium exists is often

trivial, while being assured that an equilibrium exists that is stable under cooperation (*i.e.*, the core is non-empty) remains prohibitively difficult. In other words, expecting solution concepts with cooperations to be normative implies that players or analyst in general have access to infinite or at least very large computational powers to behave accordingly. The only general case with utility transfers where cooperation does not contradict stability and leads to predictable outcome is when the game is convex and the only stable coalition is one that contains all players.

Here, we wish to determine the impact of cooperation among players in a different context where utility cannot be transferred between players. The systematic study of those games has been more recent, motivated primarily by the need to determine how individual choices governed by constraints set on player's interaction lead to the formation of different types of cartels, alliances, clubs, interest groups. The studies of such groups and how users are connected through them in a network paved the way to a novel economic analysis where a player's position in a structure governs its expected behaviors and eventual gains. Most of the work concentrates on studying the outcomes of so called *Hedonic Games*. In the general case, a stable outcome may not exist, even for a weak definition of stability that only allows certain individual strategies [8]. This holds even when preferences among coalitions are anonymous (*i.e.*, the utility of a player only depends on the size of her group). This also holds if the preferences are additively separable (*i.e.*, they derive from a simple utility function that sums up the effects of pairwise interaction between a player and members of her coalition), as long as they are not required to be symmetric. Perhaps unsurprisingly, for stronger stability notion when cooperation among players – and hence possibly more deviations from a given status quo – are allowed to take place, those negative results are only reinforced. Only when additional properties constraining preferences are assumed (see *e.g.*, [5, 10, 20]) can it be shown that such solutions (core stable or strong Nash equilibrium) exist, and/or are unique [27].

The results we mention above judge the merit of solution concepts merely by existence, and occasionally by unicity of a stable outcome. More recently, merit of solution concepts were discussed using an algorithmic approach [12], by analyzing how the cost of deciding on the existence of a stable outcome, or computing one when it exists, grows with the number of players n . Indeed, all the negative results aforementioned were strengthened in recent works: for hedonic games in general, even anonymous ones, deciding if a coalition system that satisfies a stability condition exists is NP-hard [4]. Those results were shown to hold when preferences are additively separable (in the asymmetric case) [24]; and more recently even when the parameters defining the preferences among coalitions are bounded by a polynomial function of n [30]. It is noteworthy that those negative results are each different, but one can be found for *any* stability condition, whether or not cooperation between players is allowed¹. Similar NP-hardness results apply in general to solution concepts defined by efficiency instead of stability such as maximizing sum of utility, or satisfying Pareto optimality under weak conditions (as shown in [2]).

For hedonic games, the effect of players' cooperation is chiefly of interest in the case where preferences are additively separable and symmetric. The interesting situation is when players' preferences among coalitions are defined using parameters which are constant or bounded by a polynomial function of the number of players². Under those natural conditions, depending on which cooperation is assumed the properties of solution concepts are entirely at odds. First, if one assumes that only individual deviation is allowed, outcomes are Nash or individually stable equilibrium of this game (see definition below). Those always exist and they can be always attained

¹Note that here and in the rest of the paper we ignore the case where in order to move a player is required to get approval from members of his current coalition [29]. This leads, in the separable case and many others, to have almost all coalition structures being stable, and hence provide little insight into the game.

²Otherwise, it is already known that finding stable outcome is hard even without cooperation, as computing Nash or Individually stable equilibria requires solving a PLS complete problem [16]

using some better response dynamics of the game within a pseudopolynomial number of steps [8] — following a potential function argument. In contrast, when unrestricted cooperation is allowed between players, sustainable outcomes are only those that are so called core stable; such an outcome may not exist in general. In fact, deciding if one exists in general is NP-hard, as previously shown independently in [3] for a specific set of weights, a result that we prove in this article in a more general case. This chasm motivates us to answer more precisely which forms of cooperation renders stability so much complex. Are pairwise cooperation involving two players, or triadic cooperation involving three, sufficient to cause a sudden increase in complexity? We gradually increase the size of possible subsets of players acting in concert to create a new deviation, from a subset of size 1 denoting absence of coordination to size n denoting unrestricted cooperation.

The above question was recently formulated in [22] when preferences are additively separable, symmetric, and formulated to either denote a complete incompatibility between two players, or a benefit to share the same coalition that is uniform among compatible players. In this case, solution concepts involving simultaneous cooperation of at most 2 or 3 players are shown to be tractable for those particular hedonic games. This result is remarkable: it offers the first example of a solution concept involving some cooperation and that has not been shown to be computationally prohibitive. Unfortunately, this paper brings no more light to extend any such results to other hedonic games. Indeed the tractability of any form of cooperation involving more than 3 players is left open. Here we explain and show why this is the case. Furthermore, although it has been proposed a partial analysis beyond the case $k = 3$, where it is proved that the outcome can be NP-Hard to find in general, this negative result relies on a unique form of deviations (called *gossip*) introduced by the authors and that has no equivalent in the theory of hedonic games.

1.2 Contribution of this paper

The main contribution of this paper is to completely solve the complexity analysis of hedonic games for any level of cooperation, in the general case where preferences are additively separable and symmetric. This includes in particular a specific study of the effect of cooperation when the values taken by weights defining players' preferences are fixed in advance.

- We first consider the binary incompatible/compatible case (uniform), where all compatible players benefit from each other uniformly as introduced in [22] (the latter corresponds to the above-mentioned uniform case). We prove that in such case better response dynamics always converge to an equilibrium, for any possible level of cooperation. Hence we focus on the time of convergence for the dynamics. We report on our results in Table 1. Previous work focuses on upper-bounds on the number of steps. We provide not only the first non-trivial lower-bound but even derive the exact worst case of convergence time when only individual and pairwise deviations are allowed. This requires an entire new analysis of this problem connecting its dynamic to sand piles, which is of independent interest.
- By leveraging more properties of our novel analysis, we are able to prove that the complexity suddenly increases when groups of more than 3 players are allowed to coordinate to act simultaneously. This result solves an open conjecture from [22]. It is also remarkable as it proves that cooperation affects computational complexity in multiple stages as more and more players are allowed to coordinate. To the best of our knowledge, no such progressive phase transitions were ever identified.

All our results on the uniform case are listed in Table 1.

- We then consider general hedonic games, and for the first time analyze them when cooperation among players is gradually increased. We start with a case where preferences follow incompatible/indifferent/compatible interactions among players. We first show that solution concepts for pairwise cooperation in this case can be computed efficiently. On the other hand, this result is tight in multiple ways: any amount of further cooperation or slightly more complex preferences is sufficient to impose a prohibitive cost to decide if a stable outcome exists in the general case. We present new sufficient conditions on preferences under which an outcome is found in polynomial time, but those remain stringent.

Those results present a new level of details to assess solution concepts for hedonic games. Under different assumptions on cooperation, it allows one to see which coalition systems are likely to form among players or, on the contrary, are unlikely to exist or be found using reasonable time and resources.

2 Hedonic Games and Stability

Let us first define colouring games in a formal way.

Network We model the network with an edge-weighted graph $G = (V, E, w)$, where V denotes the set of players. There are $n = |V|$ players in the network. Moreover, it is supposed to be a *simple* graph, hence there is no loop and no multiple edges. The weight w_{uv} is defined for every edge $uv \in E$ as the utility both u and v receive if they interact. Note that $w_{uv} = w_{vu}$ by symmetry. In the general case, $w_{uv} \in \mathbb{Z} \cup \{-\infty\}$. In other words, for any two vertices $u, v \in V$, if $w_{uv} = -\infty$ then the players are *enemies* and they never interact. Given the natural dominance of weight $-\infty$ over all the other ones, we also say that $\{uv\}$ is a *conflict edge* when $w_{uv} = -\infty$.

Sometimes, the set of possible weights may be constrained, and we will denote it \mathcal{W} . Also, missing edges in the graph can be added with weight 0; so, without loss of generality, we assume that G is a complete graph, and we will simply write $G = (V, w)$ in the sequel. As an example, Figure 1(a) depicts a network with $\mathcal{W} = \{-\infty, 2, 3, 4\}$. Conflict edges are represented with dashed red lines, while edges with positive weight are represented with solid green lines.

Groups and Utilities We suppose in our model that the players are *partitioned* into n ordered sharing groups, some of them may be empty. Such a partition of the nodes is denoted $P = (X_1, X_2, \dots, X_n)$. For every node $u \in V$, we denote by $c_u(P) \in \llbracket 1, n \rrbracket$ the index of the group containing u . In other words, $u \in X_c$, where $c = c_u(P)$. The integer c is the *colour* of u in P .

Figure 1 presents three different partitions of the nodes for a graph with $\mathcal{W} = \{-\infty, 2, 3, 4\}$. Each of them has three non-empty groups (and so three empty groups).

For a given P , the *utility* for the player $u \in V$ is defined as $f_u(P) = \sum_{v \in X_c \setminus \{u\}} w_{uv}$, where $c = c_u(P)$ denotes the sharing group in P which u belongs to. The global utility, or social welfare, is $f(P) = \sum_{u \in V} f_u(P)$. For the partition of Figure 1(a), the respective utilities of $u_1, u_2, u_3, v_1, v_2, v_3$ are 7, 4, 0, 4, 5, 6 and the global utility is equal to 26.

Deviations Let P be any partition of the nodes. Given a coalition S with at most k players, we say that S is a *k-deviation*, or *k-set*, when all the players in S have an incentive to join the same (possibly empty) group in P , so that they *all* increase their individual utility in the process. This way, we aim at modeling situations where players can join a given group without any approval of its current members, *e.g.*, public communities in social networks, etc. The following definition formalizes the notion of *k-deviation*.

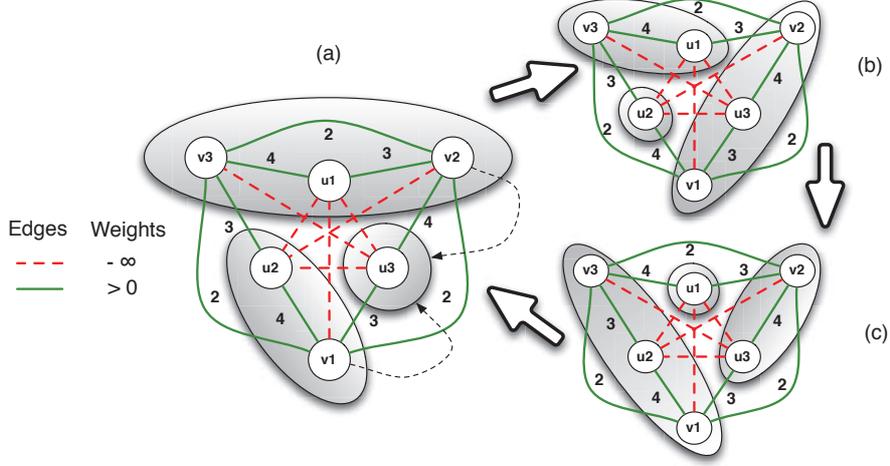


Figure 1: A network with set of weights $\mathcal{W} = \{-\infty, 2, 3, 4\}$ that does not admit a 2-stable partition. (a) 1-stable partition that is not 2-stable and that can be obtained after a 2-deviation in partition depicted in (c). (b) 1-stable partition that is not 2-stable and that can be obtained after a 2-deviation in partition depicted in (a). (c) 1-stable partition that is not 2-stable and that can be obtained after a 2-deviation in partition depicted in (b).

Definition 1 (*k*-deviation). Let $k \geq 1$ be any integer and let $P = (X_1, \dots, X_n)$ be any partition of V . A *k*-deviation is defined as a pair (S, j) , where $S \subseteq V$, $|S| \leq k$, $j \in \llbracket 1, n \rrbracket$, such that the partition $P' = (X'_1, \dots, X'_n)$, with $X'_i = X_i \setminus S$ if $i \neq j$ and $X'_j = X_j \cup S$ is such that $f_v(P') > f_v(P)$ for every $v \in S$.

As an illustration, consider the partition $P = (\{v_3, u_1, v_2\}, \{u_3\}, \{u_2, v_1\}, \{\}, \{\}, \{\})$ of Figure 1(a). There exists a 2-deviation $(\{v_2, v_1\}, 2)$ because v_2 and v_1 have an incentive to join the singleton group $\{u_3\}$. We indicate it by dashed arrows. Let $P' = (\{v_3, u_1\}, \{v_2, u_3, v_1\}, \{u_2\}, \{\}, \{\}, \{\})$ be the partition obtained after the 2-deviation $(\{v_2, v_1\}, 2)$ happens (Figure 1(b)). Observe that $f_{v_1}(P') > f_{v_1}(P)$ and $f_{v_2}(P') > f_{v_2}(P)$. Note that there is no 1-deviation for the partition of Figure 1(a).

Stability Let $k \geq 1$ be any integer and let P be any partition of the nodes. We say that P is a *k*-stable partition if and only if there is no *k*-deviation.

As an illustration, the three different partitions depicted in Figure 1 are 1-stable. However, none of them is 2-stable. Indeed, there is 2-deviation for the partition of Figure 1(a) that permits to obtain the partition of Figure 1(b); there is 2-deviation for the partition of Figure 1(b) that permits to obtain the partition of Figure 1(c), and there is 2-deviation for the partition of Figure 1(c) that permits to obtain the partition of Figure 1(a). The arrows in Figure 1 describes this "cycle" of partitions. It is possible to show for this graph, there is no 2-stable partition.

Dynamic of the game Players in the networks want to maximize their individual utility. Initially, none of them interact, and so, we only have singleton groups, that is $|X_j| = 1$ for every $j \in \llbracket 1, n \rrbracket$. Then we fix a constant parameter $k \geq 1$, and the colouring game starts. At each round i , we consider the current partition P_i of the nodes. In particular, there are only singleton groups in P_0 . When a *k*-deviation exists, we may allow any one of them to happen and, in so doing, we

Dynamic of the system (Algorithm 1)

Input: a positive integer $k \geq 1$, a set of weights \mathcal{W} , and a graph $G = (V, w)$.

Output: a partition k -stable for G .

- 1: Let P_0 be the partition composed of n singletons groups.
 - 2: Set $i = 0$.
 - 3: **while** there exists a k -deviation for P_i **do**
 - 4: Set $i = i + 1$.
 - 5: Compute the partition P_i after any k -deviation.
 - 6: **return** Partition P_i .
-

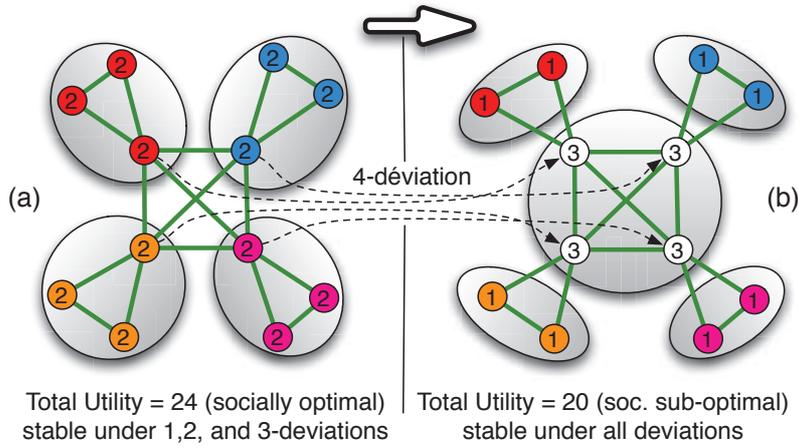


Figure 2: A network with set of weights $\mathcal{W} = \{-\infty, 1\}$. (a) 3-stable partition that is not 4-stable but it is optimal in terms of total utility. (b) k -stable partition for any $k \geq 1$ that is not optimal in terms of total utility.

break P_i , and we get a new partition P_{i+1} . If there is no k -deviation, the partition P_i is k -stable. An algorithmic presentation of the game is given in Algorithm 1.

For the graph of Figure 1, Algorithm 1 returns a k -stable partition if and only if $k = 1$ (otherwise the algorithm never ends because there is always a 2-deviation for this graph).

Example Consider the graph depicted in Figure 2 with $\mathcal{W} = \{-\infty, 1\}$. Figure 2(a) depicts a partition composed of 4 non-empty groups. The integers on the nodes represent their utilities. Observe that this partition is k -stable when $k \in \{1, 2, 3\}$. However, this partition is not 4-stable because there is a 4-deviation: the four central nodes can join an empty group (that corresponds to create a new group) and increase their utilities. The partition obtained after such a 4-deviation is depicted in Figure 2(b). This partition is k -stable for any $k \geq 1$. The utility of the four nodes that have joined the empty group is now 3 (instead of 2). However, the utility of the other nodes is now 1 (instead of 2). Thus, we deduce that this partition is not optimal in terms of total utility (the total utility has decreased from 24 to 20); but it is now stable under all deviations.

We will discuss in Section 5 several extensions of the model. We will show that most of our results still hold for these more general games.

3 The uniform case: are longest deviation sequences polynomial?

A colouring game is said *uniform* if, except for conflict edges, all edges have the same unit weight *i.e.*, $\mathcal{W} = \{-\infty, 1\}$. This game is entirely characterized by an unweighted and undirected *conflict graph* $G^- = (V, E)$ that contains all the conflict edges³. The complementary graph of G^- represents all the pairs of friends (with unit weight). Note also that given a partition P , for any player u , the individual utility $f_u(P)$ equals $|X_c| - 1$ with $c = c_u(P)$, which is how many players have the same colour as u . Recall that we only consider partition such that any two enemies do not belong to a same group. As shown in [15, 22] k -stable partitions always exist for any value of k . Their proof is algorithmic, but it does not compute a k -stable partition in polynomial-time *even if k is fixed*. As a first step toward a polynomial-time computation, we now prove that the better-response dynamic always terminates. Hence, the problem of computing a k -stable partition—in the uniform case—is in the complexity class PLS, for any fixed k (*e.g.*, see [19]).

In the following, given a partition P we define $\lambda_i(P)$ to be the number of groups of size i , and we denote by $\vec{\lambda}(P) = (\lambda_n(P), \dots, \lambda_1(P))$ the *partition vector*.

Lemma 2. *For any $k \geq 1$, for any conflict graph G^- , Algorithm 1 converges to a k -stable partition.*

Proof. Let P_i, P_{i+1} be two partitions for G^- such that P_{i+1} is obtained from P_i after a k -deviation. We prove that $\vec{\lambda}(P_i) <_L \vec{\lambda}(P_{i+1})$ where $<_L$ is the lexicographical ordering. To do so, let (S, j) be the k -deviation which breaks P_i . By definition, for any $u \in S$, we have $f_u(P_i) < f_u(P_{i+1})$. Furthermore the size of the group X_j has increased by $|S|$. Therefore, no vertex of S was in a group of size at least $|X_j| + |S|$ in P_i . Thus, we get $\vec{\Delta} = \vec{\lambda}(P_{i+1}) - \vec{\lambda}(P_i) = (0, \dots, 0, \Delta_{|X_j|+|S|} = 1, \dots)$, and so $\vec{\lambda}(P_i) <_L \vec{\lambda}(P_{i+1})$. Finally, as the number of possible vectors is finite, we obtain the convergence of Algorithm 1. \square

We can then define $L(k, n)$ as the size of a longest sequence of k -deviations among all the colouring games defined on a conflict graph with (at most) n nodes.

Let G^0 be the empty conflict graph. An instrumental observation for our next proofs is that:

Observation 1. *$L(k, n)$ is always attained in the colouring game defined on the empty conflict graph G^0 of order n , containing no conflict edges.*

Prior to this work, no lower bound on $L(k, n)$ was known, and the analysis was limited to potential function that only applies when $k = 1, 2$, and 3 [15, 22]. The analysis of the game becomes much more difficult as soon as 4-deviations are allowed. Table 1 summarizes our contributions:

Table 1: Previous Bounds and results we obtained on $L(k, n)$.

k	Prior to our work	Our results	
1	$O(n^2)$ [22]	exact analysis, which implies $L(k, n) \sim \frac{(2n)^{3/2}}{3}$	Theorem 8
2	$O(n^2)$ [22]	exact analysis, which implies $L(k, n) \sim \frac{(2n)^{3/2}}{3}$	Theorem 9
3	$O(n^3)$ [15, 22]	$\Omega(n^2)$	Theorem 12
≥ 4	$O(2^n)$ [22]	$\Omega(n^{\Theta(\ln(n))}), O(\exp(\pi\sqrt{2n/3})/n)$	Theorem 13

³The reader may wonder why we consider the conflict graph rather than the *friendship graph*: which is induced by the edges with unit weight. Our choice is motivated by the usual terminology for colouring games [22, 26], and because the conflict graph is the best suited to show the link between equilibria and vertex colouring.

3.1 Exact analysis for $k \leq 2$

In [22], the authors proved that the global utility increases for each k -deviation when $k \leq 2$. As that potential function is also upper bounded by $O(n^2)$, Algorithm 1 converges to a 2-stable partition in at most a quadratic time. We improve this result as we completely solve this case and give the exact (non-asymptotic) value of $L(k, n)$ when $k \leq 2$. The gist of the proof is to re-interpret sequences of deviations in the Dominance Lattice. This object has been widely used in theoretical physics and combinatorics to study systems in which the addition of one element (*e.g.*, a grain of sand) creates consequences in cascade (*e.g.*, the reconfiguration of a sand pile) [17]. Let us define an integer partition:

Definition 3 ([9]). *An integer partition of $n \geq 1$, is a non-increasing sequence of integers $Q = q_1 \geq q_2 \geq \dots \geq q_n \geq 0$ such that $\sum_{i=1}^n q_i = n$.*

Given any game with n players, there are as many partition vectors as there are integer partitions of n . Thus in the following, we will make no difference between a partition vector and the integer partition it represents. If we denote the number of integer partitions by p_n , then we know that Algorithm 1 reaches a stable partition in at most $p_n = \Theta((e^\pi \sqrt{\frac{2n}{3}})/n)$ steps (the upper-bound directly follows from Lemma 2 *i.e.*, the lexicographical ordering argument). This is already far less than 2^n , which was shown to be the best upper bound that one can obtain for $k \geq 4$ when using an additive potential function [22].

For $n \geq 6$ it can be seen that some partitions P of the players may never be in the same sequence of 1-deviations. It is hence important to deal with a partial ordering instead of a total one. Brylawski proved in [9] that dominance ordering creates a lattice on integer partitions, where successors and predecessors can be defined using a covering relation:

Definition 4. (*dominance ordering*) *Given two integer partitions of $n \geq 1$, denoted by $Q = q_1 \geq \dots \geq q_n$ and $Q' = q'_1 \geq \dots \geq q'_n$, we say that Q' dominates Q if $\sum_{j=1}^i q'_j \geq \sum_{j=1}^i q_j$, for all $1 \leq i \leq n$.*

Definition 5. (*covering*) *Given two integer partitions Q, Q' of $n \geq 1$, Q' covers Q if and only if Q' dominates Q and no other integer partition Q'' satisfies that Q' dominates Q'' and Q'' dominates Q .*

The following lemma characterizes when an integer partition covers another one.

Lemma 6 ([9]). *Given Q, Q', Q' covers Q if and only if there are j, k such that: (i) $q'_j = q_j + 1$; (ii) $q'_k = q_k - 1$; (iii) for all $i \notin \{j, k\}$, we have $q'_i = q_i$; (iv) either $k = j + 1$ or $q_j = q_k$.*

The main ingredient of our analysis is to exploit a strong relationship between covering and 1-deviations in a game, that holds as long as no conflict edges exist.

Lemma 7. *Assuming no conflict edges exist *i.e.*, $G^- = G^\emptyset$, let Q, Q' be two integer partitions of $n = |V|$. Then, Q' dominates Q if and only if there exist two partitions P, P' of the players such that: $\vec{\Lambda}(P') = Q'$, $\vec{\Lambda}(P) = Q$, and there is a valid sequence of 1-deviations from P to P' .*

Proof. (\Rightarrow) To prove the first direction (*a.k.a.*, the ‘only if part’), it suffices to prove the result whenever Q' covers Q . In such case, we have by Lemma 6 that there exist j, k satisfying: $q'_j = q_j + 1$, $q'_k = q_k - 1$, and for all i such that $i \notin \{j, k\}$, $q'_i = q_i$. Moreover, since $k = j + 1$ or $q_j = q_k$, we get $q_j \geq q_k$.

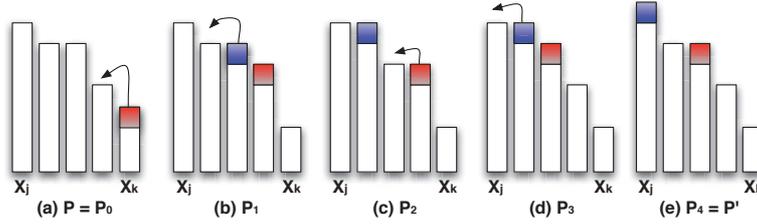


Figure 3: An example of decomposition for a 1-deviation from group X_k to group X_j .

Let P be any partition of the players such that $\vec{\Lambda}(P) = Q$. We can write w.l.o.g. $P = (X_1, \dots, X_n)$ with $|X_i| = q_i$ for all $1 \leq i \leq n$. We take a vertex $v \in X_k$, which exists because $|X_k| = q_k > 0$, and we claim that $(\{v\}, j)$ is a 1-deviation that breaks P . Indeed, we have that $|X_j| = q_j \geq q_k$, and no conflict edges exist by the hypothesis. So, we can break P by making $(\{v\}, j)$ happen and in so doing, we get a new partition $P' = (X'_1, \dots, X'_n)$ for the players such that: $|X'_j| = |X_j| + 1$, $|X'_k| = |X_k| - 1$, and for all i such that $i \notin \{j, k\}$, $|X'_i| = |X_i|$. In other words, $\vec{\Lambda}(P') = Q'$.

(\Leftarrow) To prove the converse direction, assume the existence of two partitions P, P' such that $\vec{\Lambda}(P) = Q$, $\vec{\Lambda}(P') = Q'$ and P' can be obtained from P after the 1-deviation $(\{v\}, j)$ happens. Then, v picks j as her new colour instead of her former colour k , with $|X_j| \geq |X_k|$ by the hypothesis. Furthermore, we can suppose w.l.o.g. that the n groups in P are ordered by decreasing size, up to a recolouring. So, we get either $|X_j| = |X_k|$, or $|X_j| > |X_k|$, hence $j \leq k - 1$. By a reordering of groups with equal size, we can also assume that X_j is the first group with size $|X_j|$, whereas X_k is the last group with size $|X_k|$ in P . Then:

- if $i \in \{1, \dots, j - 1\}$, then $\sum_{l=1}^i |X'_l| = \sum_{l=1}^i |X_l|$;
- if $i \in \{j, \dots, k - 1\}$, then $\sum_{l=1}^i |X'_l| = [\sum_{l=1}^i |X_l|] + 1$;
- if $i \in \{k, \dots, n\}$, then $\sum_{l=1}^i |X'_l| = \sum_{l=1}^i |X_l|$.

As a consequence, we have that Q' dominates Q by the hypothesis. \square

In other words, any sequence of 1-deviations, from a partition P to another partition P' , can be decomposed into more elementary 1-deviations, with the property that if there is such an elementary deviation from the partition P_i to the partition P_{i+1} , then the integer partition $\vec{\Lambda}(P_{i+1})$ covers $\vec{\Lambda}(P_i)$. Note that, by doing so, we may get another final partition $P'' \neq P'$ but it will have the same partition *vector*, as seen on an example in Figure 3. Since it has been proven in [18] that for $n = \frac{m(m+1)}{2} + r$, the longest chain in the Dominance Lattice has length $2\binom{m+1}{3} + mr$, we finally obtain the *exact* value for $L(1, n)$.

Theorem 8. *Let m and r be the unique non negative integers such that $n = \frac{m(m+1)}{2} + r$, and $0 \leq r \leq m$. We have $L(1, n) = 2\binom{m+1}{3} + mr$. This implies that $L(1, n) \sim \frac{2\sqrt{2}}{3}n\sqrt{n}$ as n gets large.*

Interestingly, we prove that 2-deviations have a similar action on partition vectors, which implies:

Theorem 9. $L(2, n) = L(1, n) = \Theta(n\sqrt{n})$.

Proof. Clearly, $L(2, n) \geq L(1, n)$.

For the other direction, let $G^\emptyset = (V, E)$ be the conflict graph such that $|V| = n$ and $|E| = 0$. Let P be any partition of the players for G^\emptyset such that there exists some 2-deviation $(\{u, v\}, j)$ that breaks P . In the following, let $c = c_u(P)$ and let $c' = c_v(P)$. If $|X_j| \geq |X_c|$ or $|X_j| \geq |X_{c'}|$, then the 2-deviation can be decomposed into 1-deviations. So, we suppose that $|X_j| = |X_c| - 1 = |X_{c'}| - 1$. There are two cases:

- Suppose $c = c'$. Then after the 2-deviation happens, the groups X_c, X_j are replaced with $X_c \setminus \{u, v\}, X_j \cup \{u, v\}$; equivalently, we obtain two groups of respective size $|X_c| - 2, |X_c| + 1$ instead of two groups of respective size $|X_c| - 1, |X_c|$. It thus follows that for any vertex $u_j \in X_j$, one can obtain the same partition vector by making the 1-deviation $(\{u_j\}, c)$ happen.
- Else, after the 2-deviation happens the groups $X_c, X_{c'}, X_j$ of respective size $|X_c|, |X_{c'}|, |X_c| - 1$ are replaced with the groups $X_c \setminus \{u\}, X_{c'} \setminus \{v\}, X_j \cup \{u, v\}$ of respective size $|X_c| - 1, |X_{c'}| - 1, |X_c| + 1$. Again, one can obtain the same partition vector, this time by making the 1-deviation $(\{v\}, c)$ happen.

Finally, any partition vector that is obtained from a 2-deviation may also be obtained from a sequence of 1-deviations. Thus, $L(2, n) \leq L(1, n)$.

Consequently, $L(2, n) = L(1, n) = \Theta(n\sqrt{n})$. □

3.2 Lower bounds for $k > 2$

On the one hand, the classical dominance ordering does not suffice to describe all k -deviations as soon as $k \geq 3$. It can be seen with the two following integer partitions of 10: $Q = (3, 3, 3, 1)$ and $Q' = (4, 2, 2, 2)$, that are incomparable, yet one can break Q to obtain Q' using a 3-deviation. On the other hand, we can reuse some techniques inspired from [18] so that we can obtain lower-bounds on $L(k, n)$. The method heavily relies on specific sequences of deviations that we will call *cascades*.

Overview To give a flavor of the method, we first observe there is only *one* k -stable partition in the empty conflict graph G^\emptyset namely, the one with partition vector (n) composed of only one summand. Then given two partitions P, P' of the players, we notice that if P' is obtained from P using a k -deviation, then we may have that $h(P') \geq h(P)$ as soon as $k \geq 3$, with $h(P)$ the length of a longest sequence in the Dominance Lattice from the integer partition $\vec{\Lambda}(P)$ to the integer partition (n) . Moreover, we also noticed that the larger k the larger $h(P') - h(P)$ may be. Thus it motivates the following strategy to lower-bound $L(k, n)$: at each step of Algorithm 1, one should break the current partition using a set S of maximum size⁴.

To ensure that many k -deviations can happen *consecutively*, we will seek for partitions containing a group of *each* size: from one to some value p . For instance, consider a partition with three groups of size p , and one group of size i for $1 \leq i \leq p - 2$. A first 3-deviation can happen so that a vertex in each of the three groups of size p changes her colour to join the group of size $p - 2$. In such case, we are left with a partition containing: one group of size $p + 1$, three groups of size $p - 1$, and one group of size i , for $0 \leq i \leq p - 3$. Hence, another 3-deviation can happen, so that a vertex in each of the three groups of size $p - 1$ changes her colour to join the group of size $p - 3$, and so on.

⁴We only consider *elementary* k -deviations *e.g.*, deviations such that the partition vector one obtains after they happen cannot be gotten using a sequence of smaller deviations.

Vectorial notations In order to define our cascades, we will rely on a *vectorial representation* of partitions and deviations. This novel representation allows us to describe, in simpler terms, the “patterns” that are applied recursively until we obtain a long sequence of k -deviations. Formally, given two partitions P, P' of the players, if P' is obtained from P using a k -deviation, then we represent the deviation by the vector $\vec{\Lambda}(P') - \vec{\Lambda}(P)$.

- In case of a 1-deviation, there are four possibilities. If some player leaves a group of size $q + 1$ for a group of size $p - 1$, hence $p \geq q + 2$, then the deviation is represented by a vector $\vec{\alpha}[p, q]^5$ whose entries all equal zero, except:
 - if $p = 2, q = 0$ then $\alpha_1 = -2, \alpha_2 = 1$;
 - if $p > 2, q = 0$ then $\alpha_1 = \alpha_{p-1} = -1, \alpha_p = 1$;
 - if $p = q + 2, q > 0$ then $\alpha_p = \alpha_q = 1$, and $\alpha_{p-1} = -2$;
 - if $p > q + 2, q > 0$ then $\alpha_{p-1} = \alpha_{q+1} = -1$, and $\alpha_p = \alpha_q = 1$.
- The case of 2-deviations can be ignored by Theorem 9.
- In case of a 3-deviation, we consider there are three groups of size $p - 1$ and a player leaves each group to join some group of (possibly null) size $p - 3$. Hence it is represented by a vector $\vec{\gamma}[p]^5$ whose entries all equal zero except: $\gamma_p = 1, \gamma_{p-1} = -3, \gamma_{p-2} = 3$, and if $p \neq 3$ $\gamma_{p-3} = -1$.
- Finally, in case of a 4-deviation, we consider there are four groups of size $p - 1$, one group of size $p - 4$, and a player in each group of size $p - 1$ changes her colour so that she joins the group of size $p - 4$. Therefore it is represented by a vector $\vec{\delta}[p]^5$ whose entries all equal zero, except: $\delta_p = 1, \delta_{p-1} = -4, \delta_{p-2} = 4$, and if $p \neq 4$ $\delta_{p-4} = -1$.

Throughout the remaining of the section, we will ignore all other k -deviations.

Properties To go further with our vectorial approach, one needs to check whether a given deviation $\vec{\varphi} = \vec{\Lambda}(P') - \vec{\Lambda}(P)$ is *valid i.e.*, the vector $\vec{\Lambda}(P) + \vec{\varphi}$ has no negative entries. Let us now introduce the notion of *balanced sequence*.

Definition 10. *Given any integer $h > 0$, let $\vec{\varphi}^1, \vec{\varphi}^2, \dots, \vec{\varphi}^t$ be vectors. We call this sequence h -balanced if, for any $1 \leq i \leq t$, the sum of the i first vectors, namely $\sum_{j=1}^i \vec{\varphi}^j$, has all its entries greater than or equal to $-h$.*

Given a h -balanced sequence $(\vec{\varphi}^1, \vec{\varphi}^2, \dots, \vec{\varphi}^t)$ of k -deviations, let $\vec{\Phi} = \sum_{i=1}^t \vec{\varphi}^i$ be the sum of all deviations, and let p_{\max} be the largest index j that satisfies $\vec{\Phi}_j \neq 0$. Equivalently, p_{\max} is the largest size of a group modified (hence created) after some deviation in the sequence happens (*i.e.*, $\forall l, \forall p > p_{\max}, \varphi_p^l = 0$). One can observe that a sufficient condition so that the sequence is valid is that it starts from a partition with at least h groups of each size j , for $1 \leq j \leq p_{\max}$. In the following, we will often make use of a *symmetric* property to find a balanced sequence:

Definition 11. *The minimum-size sub-vector that contains all non-zero entries of a vector is called the support of the vector. We say a vector has the symmetric property if, and only if, the support of the vector is symmetric. Equivalently, a vector has the symmetric property if, and only if, the coordinates of its support are invariant under the reverse permutation.*

⁵The index p represents the largest group created after the k -deviation happens.

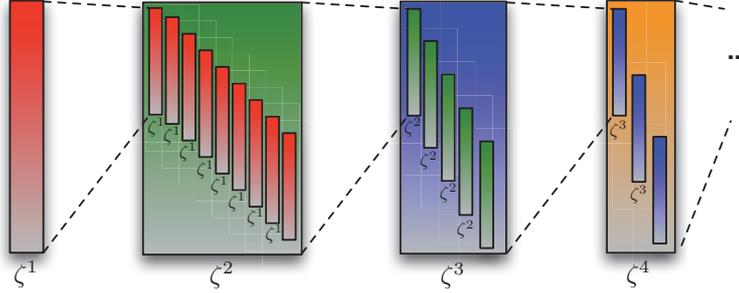


Figure 4: Long sequence using recursive cascades.

Given a vector, it might be useful in the following to notice that the size of the support is exactly $p_{\max} - p_{\min} + 1$, where p_{\max}, p_{\min} denote the largest non-zero index and the least non-zero index.

One can also notice every 1-deviation yields an elementary vector of the form $\vec{\alpha}[p, q]$ that has the symmetric property provided $p \geq q \geq 1$. But the property does not hold in general for k -deviations whenever $k \geq 3$.

Results Prior to our work, it was known $L(3, n) = O(n^3)$, which follows from another application of the potential function method [22]. But nothing proved that $L(3, n) > L(2, n)$, and in fact it was conjectured in [15] that both values are equal. Theorem 12 proves for the first time that deviations of multiple players can delay convergence and that the gap between $k = 2$ and $k = 3$ obtained from potential function is indeed justified. Using a considerable refinement of the cascade technique, we are able to prove in Theorem 13 a much more significant result: that 4-deviations are responsible for a sudden complexity increase, as we now prove that no polynomial bounds exist for $L(4, n)$.

Theorem 12. $L(3, n) = \Omega(n^2)$.

Theorem 13. $L(4, n) = \Omega(n^{\Theta(\ln(n))})$.

The rest of this section is mainly devoted to prove Theorem 13. At the end of this section, we also give a proof of Theorem 12 using the same techniques.

Theorem 13. As before, we assume no conflict edge exists. W.l.o.g. we also assume that the number of players is $n = cL(L + 1)/2$. The values c and L will be defined later, and we only assume for the moment that c is sufficiently large.

Starting from the partition with n singleton groups, let P^0 be such that $\vec{\Lambda}(P^0) = (0, \dots, 0, \lambda_L = c, \dots, \lambda_1 = c)$. Using our notations, one can obtain a group of size j with the sequence of 1-deviations defined by $\sum_{i=2}^j \vec{\alpha}[i, 0]$ and so, one can obtain P^0 using the sequence of 1-deviations defined by $c \cdot \left(\sum_{j=2}^L \sum_{i=2}^j \vec{\alpha}[i, 0] \right)$.

Our proof relies on a “shift” operator: given a vector $\vec{\varphi}$ whose support ranges between indices p_{\min}, p_{\max} , the vector $\text{tr}^{(i)} \vec{\varphi}$, $i < p_{\min}$, is a vector of the same size *and the same support* as $\vec{\varphi}$, but whose support ranges between indices $p_{\max} - i, p_{\min} - i$. For instance, we have $\text{tr}^{(1)}(0, 1, -2, 1, 0, 0, 0) = (0, 0, 1, -2, 1, 0, 0)$. In particular, if $\vec{\varphi}$ represents a k -deviation, then $\text{tr}^{(i)} \vec{\varphi}$ represents the same k -deviation, up to a decrease by i of all groups involved; that is, we have $\text{tr}^{(i)} \vec{\alpha}[p, q] = \vec{\alpha}[p - i, q - i]$, $\text{tr}^{(i)} \vec{\gamma}[p] = \vec{\gamma}[p - i]$, $\text{tr}^{(i)} \vec{\delta}[p] = \vec{\delta}[p - i]$.

One can extend the operator and its meaning to sequences of k -deviations as well. Formally, let $\vec{\varphi}^1, \dots, \vec{\varphi}^t$ be a sequence of k -deviations, and let $\vec{\Phi} = \sum_{l=1}^t \vec{\varphi}^l$. Then, if no group of size less than $i + 1$ is modified nor created by the sequence (i.e., $\forall l, \forall p \leq i, \varphi_p^l = 0$), we obtain by linearity of the operator that $\text{tr}^{(i)} \vec{\Phi} = \sum_{l=1}^t \text{tr}^{(i)} \vec{\varphi}^l$.

Interestingly, the so-called “shift” operator keeps the symmetric properties of a vector:

Claim 14. *Let $\vec{\phi}$ be any vector that has a support of size $s = p_{\max} - p_{\min} + 1$, and with the symmetric property. For any positive integers r and d such that $1 + (r - 1)d \leq p_{\min}$, the vector $\vec{\phi}' = \sum_{h=0}^{r-1} \text{tr}^{(hd)} \vec{\phi}$ also has the symmetric property.*

Proof. The support of vector $\vec{\phi}'$ has size $s' = (r - 1)d + s$. In the following, we will assume up to padding the vector $\vec{\phi}$ with additional null entries that it is unbounded i.e., it is indexed by \mathbb{Z} . By the hypothesis the vector $\vec{\phi}$ has the symmetric property and so, $\forall 1 \leq j \leq p_{\max} + p_{\min} - 1, \phi_j = \phi_{p_{\min} + p_{\max} - j}$. Let $0 \leq j \leq s'/2 - 1$. We have that:

$$\begin{aligned} \phi'_{p_{\max} - j} &= \sum_{h=0}^{r-1} \phi_{p_{\max} - j + hd} = \sum_{h=0}^{r-1} \phi_{p_{\max} + p_{\min} - (p_{\max} - j + hd)} \\ &= \sum_{h=0}^{r-1} \phi_{p_{\min} + j - (r-1-h)d} = \sum_{h=0}^{r-1} \phi_{p_{\min} - (r-1)d + j + hd} = \phi'_{p_{\min} - (r-1)d + j}. \end{aligned}$$

Thus, $\vec{\phi}'$ also has the symmetric property. \diamond

Let $t > 0$ and $T > 0$ be such that $2^{T-1}(2t^2 + 2) \leq L$. In order to prove Theorem 13, we construct sequences of deviations that we denote by $\vec{\zeta}^i$ for all $i = 1, \dots, T$. The construction is recursive. To construct the vector $\vec{\zeta}^{i+1}$ from $\vec{\zeta}^i$, we follow a particular construction that we will show valid and that is illustrated in Figure 4. The construction is composed of a repetition of the sequence defined by $\vec{\zeta}^i$ a certain number of times (linear in t) shifting the “starting point” of each sequence by the same value. The construction then adds 1-deviations in order to get a technical property, called *Good property* (e.g., see Definition 15 below).

Definition 15. *Let i be a positive integer. We will say the sequence $\vec{\zeta}^i$ has the Good Property if it has the symmetric property, its support has even size s_i , and there exist t_1^i, t_2^i satisfying:*

- $1 < t_1^i < t_2^i < 2t_1^i; t_2^i \leq 2^{i+1};$
- *and all entries of $\vec{\zeta}^i$ are null except for: $\zeta_L^i = \zeta_{L+1-s_i/2}^i = 1, \zeta_{L-t_1^i}^i = \zeta_{L-t_2^i}^i = -1,$ and symmetrically: $\zeta_{L-s_i+1}^i = \zeta_{L-s_i/2}^i = 1, \zeta_{L-s_i+t_1^i+1}^i = \zeta_{L-s_i+t_2^i+1}^i = -1.$*

Note that in terms of k -deviations, Definition 15 implies that L is the largest size of a group created after the sequence $\vec{\zeta}^i$ happens.

Let us construct the base-case $\vec{\zeta}^1$. It starts with a cascade of t^2 consecutive 4-deviations, namely $\sum_{i=0}^{t^2-1} \vec{\delta}[L - i]$. Then, a player in some group of size $L - 2$ picks a new colour so that she joins another group of size $L - 2$; another player leaves her group of size $L - t^2 - 1$ to join another one of the same size; a third player in a group of size $L - t^2 - 5$ picks a new colour so that she joins a group of size $L - t^2 - 4$; last, a fourth player leaves her group of size $L - 4$ for a group of size $L - 2$,

followed by a player leaving her group of size $L - 3$ for another one of the same size. Altogether, we obtain the sequence

$$\vec{\Phi}^1 = \sum_{i=0}^{t^2-1} \vec{\delta}[L-i] + \vec{\alpha}[L-1, L-3] + \vec{\alpha}[L-t^2, L-t^2-2] + \vec{\alpha}[L-t^2-3, L-t^2-6] + \vec{\alpha}[L-1, L-5] + \vec{\alpha}[L-2, L-4].$$

The sum-vector $\vec{\Phi}^1$ has all its entries equal to zero, except for: $\Phi_L^1 = \Phi_{L-5}^1 = \Phi_{L-t^2-1}^1 = \Phi_{L-t^2-6}^1 = 1$, and $\Phi_{L-1}^1 = \Phi_{L-2}^1 = \Phi_{L-t^2-4}^1 = \Phi_{L-t^2-5}^1 = -1$.

We finally construct $\vec{\zeta}^1$ by repeating the sequence⁶ $\vec{\Phi}^1$ many times, up to various shiftings, followed by a sequence of 1-deviations that yields:

$$\vec{\zeta}^1 = \sum_{i=0}^{t^2-5} \text{tr}^{(i)} \vec{\Phi}^1 + \sum_{i=0}^{t^2-4} \vec{\alpha}[L-4-i, L-2t^2+3+i] + \vec{\alpha}[L-t^2+4, L-t^2-5] + \vec{\alpha}[L-t^2+3, L-t^2-4].$$

Claim 16. *There is a constant h_1 such that the sequence defined by $\vec{\zeta}^1$ is h_1 -balanced.*

Proof. First, note that any k -deviation modifies a *constant* number of groups in the partition (at most $k + 1$). Thus, for any i , $0 \leq i \leq t^2 - 5$, the sequence defined by $\text{tr}^{(i)} \vec{\Phi}^1$ is balanced for some constant because, for any j , $1 \leq j \leq n$, the number of deviations that modifies some group of size j is constant as well. In addition, the vector $\text{tr}^{(i)} \vec{\Phi}^1$ contains a constant number of non-zero values and so, for any j , $1 \leq j \leq n$, the sequence defined by $\sum_{i=0}^{t^2-5} \text{tr}^{(i)} \vec{\Phi}^1$ also modifies a constant number of groups of size j . Since this sequence $\sum_{i=0}^{t^2-5} \text{tr}^{(i)} \vec{\Phi}^1$ is only completed with a subsequence of 1-deviations so that we obtain $\vec{\zeta}^1$, and that $\forall j$ there is also a constant number of such 1-deviations that modify or create some group of size j , we can safely conclude there exists a constant h_1 such that the sequence induced by $\vec{\zeta}^1$ is h_1 -balanced. \diamond

By the calculation, we obtain that all values in $\vec{\zeta}^1$ equal to zero, except for: $\zeta_L^1 = \zeta_{L-t^2}^1 = \zeta_{L-t^2-1}^1 = \zeta_{L-2t^2-1}^1 = 1$, and $\zeta_{L-2}^1 = \zeta_{L-3}^1 = \zeta_{L-2t^2+2}^1 = \zeta_{L-2t^2+1}^1 = -1$. This proves that $\vec{\zeta}^1$ satisfies the so-called Good Property of Definition 15 (with $t_1^1 = 2, t_2^1 = 3, s_1 = 2(t^2 + 1)$). Then, one can apply the following inductive step.

Claim 17. *Suppose that $\vec{\zeta}^i$ is defined, it has a support of size s_i and it has the Good Property. Then there exist two positive integers denoted by a_i, t_1^i , and there exists a sequence of 1-deviations denoted by $\vec{\xi}^{i+1}$ so that:*

$$\vec{\zeta}^{i+1} = \sum_{j=0}^{a_i} \text{tr}^{(jt_1^i)} \vec{\zeta}^i + \vec{\xi}^{i+1}$$

also has the Good Property.

In addition we have that if $\vec{\zeta}^i$ is h_i -balanced then $\vec{\zeta}^{i+1}$ is $(h_i + 1)$ -balanced, and it holds that $s_i \leq s_{i+1} < \frac{3}{2}s_i$ where s_{i+1} denotes the size of the support of $\vec{\zeta}^{i+1}$.

Proof. We first note that s_i is even by Definition 15. Let t_1^i, t_2^i be as defined in Definition 15. Also, let a_i be the largest even integer j such that $L - jt_1^i - t_2^i > L - s_i/2 + 1$.

We set $\vec{\Phi}^{i+1} = \sum_{j=0}^{a_i} \text{tr}^{(jt_1^i)} \vec{\zeta}^i$, that has the symmetric property by Claim 14.

⁶Note that we will often abuse of our “vector-sum” notation $\vec{\Phi} = \sum_{i=1}^t \vec{\varphi}^i$ to represent the whole sequence $\vec{\varphi}^1, \dots, \vec{\varphi}^t$.

In addition, let $t_1^{i+1} = t_2^i$ and $t_2^{i+1} = t_1^i + t_2^i$ ⁷. Since we have $1 < t_1^i < t_2^i < 2t_1^i$, $t_2^i \leq 2^{i+1}$ by the hypothesis, one obtains $1 < t_1^{i+1} < t_2^{i+1} < 2t_1^{i+1}$, $t_2^{i+1} < 2t_2^i \leq 2 \cdot 2^{i+1} = 2^{i+2}$.

We finally introduce $s_{i+1} = s_i + a_i t_1^i < \frac{3}{2}s_i$ the size of the support of $\vec{\Phi}^{i+1}$, and we observe that it is an even number because a_i, s_i are both even.

We aim to construct from $\vec{\Phi}^{i+1}$ a sequence $\vec{\zeta}^{i+1}$ that satisfies the Good Property w.r.t. $t_1^{i+1}, t_2^{i+1}, s_{i+1}$. To do so, we first introduce for any $0 \leq l \leq a_i$ the truncated sum-vector $\vec{\Psi}^l = \sum_{j=0}^l \text{tr}(j t_1^i) \vec{\zeta}^i$. In particular, we have $\vec{\Psi}^0 = \vec{\zeta}^i$ while $\vec{\Psi}^{a_i} = \vec{\Phi}^{i+1}$. Note that each $\vec{\Psi}^l$ has the symmetric property by Claim 14. Furthermore we have that all entries of $\vec{\Psi}^l$ are equal to zero, except for:

- $\Psi_L^l = \Psi_{L-s_i-lt_1^i+1}^l = 1; \forall 0 \leq j \leq l, \Psi_{L-s_i/2-jt_1^i+1}^l = \Psi_{L-s_i/2-(l-j)t_1^i}^l = 1;$
- $\Psi_{L-lt_1^i}^l = \Psi_{L-s_i+1}^l = -1; \forall 0 \leq j \leq l, \Psi_{L-jt_1^i-t_2^i}^l = \Psi_{L-s_i-(l-j)t_1^i+t_2^i}^l = -1.$

In particular, we have that: $\Phi_L^{i+1} = \Psi_L^{a_i} = 1; \Phi_{L-s_{i+1}/2+1}^{i+1} = \Psi_{L-s_i/2-\frac{a_i}{2}t_1^i+1}^{a_i} = 1; \Phi_{L-t_1^i}^{i+1} = \Psi_{L-t_2^i}^{a_i} = -1$, and $\Phi_{L-t_2^i}^{i+1} = \Psi_{L-t_1^i-t_2^i}^{a_i} = -1$. So, in order to obtain the so-called Good property of Definition 15, we are left to set to zero all other entries of the -sum-vector $\vec{\Phi}^{i+1}$, using 1-deviations. To achieve the result, let us partition the $4a_i$ indices we want to set to zero in 4-tuples $(L - j_1, L - j_2, L - s_{i+1} + 1 + j_2, L - s_{i+1} + 1 + j_1)$ such that: $j_1 < j_2$, and $\Phi_{L-j_1}^{i+1} = -1, \Phi_{L-j_2}^{i+1} = 1$. For each such 4-tuple, we add to the vector-sum $\vec{\Phi}^{i+1}$ the sequence of 1-deviations $\sum_{j=0}^{j_2-j_1-1} \vec{\alpha}[L - j_1 - j, L - s_{i+1} + 1 + j_1 + j]$ whose all entries are equal to zero, except for those indexed by the 4-tuple that are respectively equal to 1, -1, -1, 1.

Finally, let us assume $\vec{\zeta}^i$ is h_i -balanced. It remains to prove that $\vec{\zeta}^{i+1}$ is $(h_i + 1)$ -balanced. By the hypothesis, we have that $\forall j, \text{tr}(j) \vec{\zeta}^i$ is h_i -balanced, and $\forall l, p, \Psi_p^l \geq -1$, hence $\vec{\Phi}^{i+1}$ is $(h_i + 1)$ -balanced. Furthermore, we have by construction each subsequence of 1-deviations in the sequence $\vec{\zeta}^{i+1}$ is 1-balanced. As a result, $\vec{\Phi}^{i+1}$ is $(h_i + 1)$ -balanced implies that $\vec{\zeta}^{i+1}$ is $(h_i + 1)$ -balanced. \diamond

Recall that $t, T > 0$ are such that $2^{T-1}(2t^2 + 2) \leq L$, and $n = cL(L+1)/2$. Set $T = \lceil \log_2(t) + 1 \rceil$ and, without loss of generality, assume $2(t^3 + t) = L$. By the proof of Claim 17, one obtains a maximum shift of:

$$\sum_{i=1}^{T-1} a_i \cdot t_1^i < \sum_{i=1}^{T-1} \frac{s_i}{2} < \sum_{i=1}^{T-1} \left(\frac{3}{2}\right)^{i-1} \frac{s_1}{2} = \frac{2s_1}{3} \left(\left(\frac{3}{2}\right)^{T-1} - 1 \right) = O(t^{\log_2(3)+1}) = o(t^3)$$

for the range of the support of vector $\vec{\zeta}^T$ w.r.t. the range of the support of $\vec{\zeta}^1$. By comparison, we remind that the sequence $\vec{\zeta}^1$ solely modifies groups of size $\Omega(L) = \Omega(t^3)$. As a result, the vector $\vec{\zeta}^L$ indeed represents a sequence of k -deviations, and by Claim 17 it is $(h_1 + T - 1)$ -balanced. So, $n = (h_1 + T - 1)(2t^3 + 2t)(2t^3 + 2t + 1)/2 = (h_1 + \lceil \log_2(t) \rceil)(2t^3 + 2t)(2t^3 + 2t + 1)/2$ is sufficient to make this sequence valid. One can note that since h_1 is a universal constant, $n = O(t^6 \log_2(t))$.

Claim 18. *The sum-vector $\vec{\zeta}^{i+1}$, as it is defined in Claim 17, represents a sequence of at least $(\frac{s_i}{2^{i+2}} - 5)$ -times more deviations than in the sequence $\vec{\zeta}^i$, where s_i denotes the size of the support of $\vec{\zeta}^i$.*

⁷By induction, we get $t_1^i = F_{i+3}$ and $t_2^i = F_{i+4}$, where $F_i = \frac{1}{\sqrt{5}} \left(\left(\frac{1+\sqrt{5}}{2}\right)^i - \left(\frac{1-\sqrt{5}}{2}\right)^i \right)$ is the i^{th} Fibonacci number.

Proof. The sequence $\vec{\zeta}^i$ is repeated a_i times using the “shift” operator, with a_i the largest even integer j such that $L - jt_1^i - t_2^i > L - s_i/2 + 1$. Furthermore if $L - jt_1^i - t_2^i > L - s_i/2 + 1$ then $j \leq \frac{s_i - 4 - 2t_2^i}{2t_1^i}$. Since we have by Definition 15 that $t_1^i < t_2^i \leq 2^{i+1}$, then $\frac{s_i - 4 - 2t_2^i}{2t_1^i} > \frac{s_i}{2^{i+2}} - \frac{1}{2^i} - 1$ and so, $a_i \geq \lfloor \frac{s_i}{2^{i+2}} - \frac{1}{2^i} - 1 \rfloor - 2 \geq \frac{s_i}{2^{i+2}} - 5$. \diamond

Since the support of $\vec{\zeta}^1$ has size $2t^2 + 2$ by construction, it follows that $\vec{\zeta}^T$ represents a sequence of at least $\prod_{i=1}^{T-1} (\frac{2t^2+2}{2^{i+2}} - 5) \geq (\frac{2t^2+2}{2^{T+1}} - 5)^{T-1} \geq (\frac{2t^2+2}{4t} - 5)^{\log_2(t)} = \Omega(t^{\log_2(t)})$ deviations. As $n = O(t^6 \log_2(t))$, it proves Theorem 13. \square

We finally prove that $L(3, n) = \Omega(n^2)$.

Proof. [Theorem 12] As usual let $G^\emptyset = (V, E)$ be the empty conflict graph with $|V| = n$ and $|E| = 0$. The number of players is here assumed to be $n = O(cL^2)$, where c denotes a large constant integer. We start our sequence from any partition P^0 which satisfies $\vec{\lambda}(P^0) = (0, \dots, 0, \lambda_L = c, \dots, \lambda_1 = c)$. Note that one can reach such a partition using a sequence of 1-deviations, namely $c \cdot \left(\sum_{j=2}^L \sum_{i=2}^j \vec{\alpha}[i, 0] \right)$.

Let then $t = \frac{L-1}{4}$. We construct a sequence $\vec{\zeta}^1 = \sum_{i=0}^t \vec{\gamma}[L-i]$ of $t+1$ consecutive 3-deviations *a.k.a.*, a cascade. The sum-vector $\vec{\zeta}^1$ has all its entries equal to zero, except for: $\zeta_L^1 = \zeta_{L-2}^1 = 1$, $\zeta_{L-t-1}^1 = \zeta_{L-t-3}^1 = -1$, $\zeta_{L-1}^1 = -2$ and $\zeta_{L-t-2}^1 = 2$.

Using the so-called “shift” operator of Section 3.2, one can repeat the above sequence that yields $\vec{\zeta}^2 = \sum_{i=0}^{t-2} \text{tr}(i) \vec{\zeta}^1$. The sum-vector $\vec{\zeta}^2$ has all its entries equal to zero, except for: $\zeta_L^2 = \zeta_{L-t}^2 = \zeta_{L-t-2}^2 = \zeta_{L-2t}^2 = 1$, and $\zeta_{L-1}^2 = \zeta_{L-t+1}^2 = \zeta_{L-t-1}^2 = \zeta_{L-2t-1}^2 = -1$.

Again, we repeat the sequence $\vec{\zeta}^2$ using the so-called “shift” operator, and one obtains $\vec{\zeta}^3 = \sum_{i=0}^{t-4} \text{tr}(i) \vec{\zeta}^2$. The sum-vector $\vec{\zeta}^3$ has all its entries equal to zero, except for: $\zeta_L^3 = \zeta_{L-3t+3}^3 = \zeta_{L-3t+1}^3 = 1$, and $\zeta_{L-t+3}^3 = \zeta_{L-t+2}^3 = \zeta_{L-t}^3 = \zeta_{L-4t}^3 = -1$.

We finally repeat the sequence $\vec{\zeta}^3$ until we obtain $\vec{\zeta}^4 = \sum_{i=0}^{t-1} \text{tr}(i) \vec{\zeta}^3$. Note that since the sequence $\vec{\zeta}^1$ is a cascade, then it is h_1 -balanced for some constant h_1 (*e.g.*, see the proof of Claim 16) and so, since in addition each sequence $\vec{\zeta}^i$ has a constant number of non-zero entries, then it follows that the whole sequence $\vec{\zeta}^4$ is also h -balanced, for some larger constant $h \geq h_1$. Hence, it is a valid sequence whenever we start from P^0 and the constant c is large enough, and it represents a sequence of $\theta(t^4) = \theta(L^2) = \theta(n^2)$ deviations. As a result, we have $L(3, n) = \Omega(n^2)$. \square

4 The general case: Are games stable under deviations?

More generally, a colouring game may be defined with weights taking values in a larger set \mathcal{W} . Players then choose to interact with each others according to more complex preferences and the individual utility is not always related to the size of the sharing group. For the remaining of the section, let w_p be the largest positive weight in a graph, if any, and 0 otherwise.

Nash equilibria On the positive side, we first show the following result.

Theorem 19. *For any weighted graph, Algorithm 1 converges in $O(w_p n^2)$ steps to a 1-stable partition.*

The proof follows from a more general potential-function technique:

Lemma 20. *Given a graph $G = (V, w)$, let P, P' be partitions of the nodes. Let (S, j) be a k -deviation which breaks P , $S = \{u_1, u_2, \dots, u_k\}$. If one can obtain P' from P after this deviation happens then we have: $f(P') - f(P) \geq 2[k - \sum_{1 \leq i, l \leq k} w_{u_i u_l} + \sum_{\{u_i, u_l | c_{u_i}(P) = c_{u_l}(P)\}} w_{u_i u_l}]$.*

The proof of the lemma is deferred to the appendix. Theorem 19 follows from an application of Lemma 20 to the case of a 1-deviation.

Proof. [Theorem 19] Let P_i, P_{i+1} be two consecutive partitions of the players in Algorithm 1, and let $(\{u_1\}, j)$ be the 1-deviation that breaks P_i . By Lemma 20, we get that $f(P_{i+1}) - f(P_i) \geq 2[1 - 0 + 0] = 2$. Hence the global utility increases at each step of the dynamic, and it is upper bounded by an $O(w_p n^2)$. \square

Theorem 19 proves that *all* colouring games admit a 1-stable partition, or equivalently a Nash-equilibrium, and that one can be reached in pseudo-polynomial time. Note on the other hand that our dependency on the weights can be exponential. It is unlikely one can improve this analysis, as the problem of computing a 1-stable partition is PLS-complete⁸.

We now fix a set of weights \mathcal{W} for the analysis, and we look at the greatest value of k for which a k -stable partition *always* exists. In the following, we will denote this maximum value by $k(\mathcal{W})$, and we will establish it for various sets of weights. Finally, we will strengthen the aforementioned results by proving that deciding if a graph with weights in \mathcal{W} admits a k -stable partition is either trivial (*i.e.*, it is true for all such graphs) or NP-complete.

4.1 Games with a unique positive weight

We first focus on the subsets of $\{-\infty, 0, 1\} \cup -\mathbb{N}$. A good representative amongst these is the subset $\{-\infty, 0, 1\}$, which is the simplest set of weights extending the uniform case ($\mathcal{W} = \{-\infty, 1\}$) to accommodate *indifferent* edges. Surprisingly, we will show that introducing the null weight radically alters the stability properties of colouring games. Indeed, whereas $k(\{-\infty, 1\}) = \infty$, we will prove $k(\{-\infty, 0, 1\}) = 2$ after exhibiting a surprising counterexample for $k = 3$.

Let us first show that $k(\mathcal{W}) \geq 2$. In fact, Theorem 21 is a global stability result, which is more precise and uses structural properties of the graphs. Given a graph $G = (V, w)$, let us define the "friendship graph" $G^+ = (V, E^+)$ of G , where $E^+ = \{uv \in E : w_{uv} > 0\}$. We remind that the *girth* of a graph is the length of its shortest cycle. By definition, an acyclic graph has infinite girth.

Theorem 21. *Let k be a positive integer, and $G = (V, w)$ be constrained to $\mathcal{W} \subseteq \{-\infty, 0, 1\} \cup -\mathbb{N}$. If the girth of the friendship graph G^+ is at least $k + 1$, then Algorithm 1 always reaches a k -stable partition in $O(n^2)$ steps in that case.*

Proof. As in the case $k = 1$, let P_i, P_{i+1} be two consecutive partitions of the players in Algorithm 1, and let $(S = \{u_1, u_2, \dots, u_l\}, j)$ be the l -deviation that breaks P_i , with $l \leq k$. By Lemma 20, we get that $f(P_{i+1}) - f(P_i) \geq 2[l - \sum_{\{u_t, u_s | c_{u_t}(P_i) \neq c_{u_s}(P_i)\}} w_{u_t u_s}] \geq 2[l - |E^+ \cap S \times S|]$ because by the hypothesis the largest positive weight is 1. Furthermore, since the girth is at least $k + 1$ by the hypothesis, S induces a forest in G^+ , and so, $|E^+ \cap S \times S| \leq l - 1$. Hence, $f(P_{i+1}) - f(P_i) \geq 2$, and as the result the global utility increases at each step of the dynamic. The latter concludes the proof as the global utility is upper bounded by an $O(n^2)$. \square

Particularly, if G^+ is cycle-free, then we get there is a k -stable partition for G , for any $k \geq 1$; if G^+ is triangle-free, then there always exists a 3-stable partition for G . Furthermore, as the girth of any graph is at least 3, then there always exists a 2-stable partition.

⁸A simple reduction from cut games can be found *e.g.*, see [19].

Unlike the uniform case, one can show the quadratic bound on the number of steps is indeed tight. To show it, assume the set V can be partitioned in three distinct subsets V_1, V_2, V_3 . We assume the induced (friendship) subgraph $G^+[V_3]$ is the clique K_{p+1} , the induced subgraph $G^+[V_1 \cup V_2]$ is the complete bipartite graph $K_{p,p}$, and each node in V_2 is adjacent in G^+ to every node in V_3 . Note that $n = 3p + 1$. We obtain the graph $G = (V, w)$ by completing G^+ with all missing edges and setting their weight to zero. Here is a sequence one can obtain for the colouring game defined on G : in p consecutive 1-deviations, all nodes in V_3 pick the same colour, say 1; then, in p consecutive 1-deviations all nodes in V_1 pick the same colour as some node in V_2 , say 2, before the unique node in V_2 of colour 2 picks colour 1. By repeating the same process with all other $p - 1$ nodes in V_2 , one finally obtains a sequence of $(p + 1)^2 = \theta(n^2)$ 1-deviations.

Proposition 22. *There is a graph $G = (V, w)$ constrained to $\mathcal{W} = \{-\infty, 0, 1\}$ such that there does not exist a 3-stable partition for the colouring game defined on G .*

Figure 5 presents a graph with 18 vertices that does not admit any stable partition for $k = 3$. Note in particular the presence of 4 indifferent edges shown in dashed lines: without these edges a k -stable partition would exist for all values of k . The proof illustrates the complexity of sequences of 3-deviations, which are able to create infinite sequences of deviations at *constant* global utility! To keep the graph readable, we use conventions. (1) Some sets of nodes are grouped within a circle; an edge from another node to that circle denotes an edge to *all* elements of this set. (2) All nodes that are not connected by an edge on the Figure are enemies with weights $-\infty$. (3) Green solid edges represent edges with weight 1, whereas blue dashed edges represent edges with weight 0.

The proof of Proposition 22 relies on another structural result which we state as follows.

Definition 23. *Let $G = (V, w)$, and let $u, u' \in V$. We say that u and u' are quasi-twins if $w_{uu'} > 0$ and for all nodes $v \in V \setminus \{u, u'\}$ $w_{uv} = w_{u'v}$ except maybe for one v_0 for which $|w_{uv_0} - w_{u'v_0}| = 1$.*

Lemma 24. *Given a graph $G = (V, w)$, let P be a 1-stable partition for the colouring game defined on it. Then, $c_u(P) = c_{u'}(P)$ for all quasi-twin vertices u, u' .*

Proof. Without loss of generality we have that for all vertices $v \in V \setminus \{u, u'\}$, $w_{uv} \geq w_{u'v}$. Equivalently, either $w_{uv} = w_{u'v}$ for all $v \in V \setminus \{u, u'\}$, or there is a unique v_0 such that $w_{uv_0} = w_{u'v_0} + 1$ and $w_{uv} = w_{u'v}$ for all $v \in V \setminus \{u, u', v_0\}$. Suppose by contradiction $c_u(P) \neq c_{u'}(P)$. There are two cases to be considered.

- **Case $f_u(P) > f_{u'}(P)$.** Then $(u', c_u(P))$ breaks the partition because, after it happens we get a new partition P' so that $f_{u'}(P') \geq (f_u(P) - 1) + w_{uu'} \geq f_u(P) > f_{u'}(P)$.
- **Case $f_u(P) \leq f_{u'}(P)$.** Then $(u, c_{u'}(P))$ breaks the partition because, after it happens we get a new partition P' so that $f_u(P') \geq f_{u'}(P) + w_{uu'} > f_{u'}(P) \geq f_u(P)$.

There is a contradiction in both cases, hence $c_u(P) = c_{u'}(P)$. □

Proof. [Proposition 22] The set of vertices consists of four sets A_i , $0 \leq i \leq 3$, each of equal size $h \geq 2$ and with a special vertex a_i , plus four vertices b_i , $0 \leq i \leq 3$, and two vertices c_0 and c_1 . In what follows, indices are taken modulo 2 for c_j , $j \in \{0, 1\}$, and they are taken modulo 4 everywhere else. Figure 5 represents the example with $h = 3$. The friendship graph G^+ here consists of all the edges with weight 1; it contains:

1. all the edges between nodes in A_i ($0 \leq i \leq 3$);
2. edges between b_i and A_i ($0 \leq i \leq 3$);

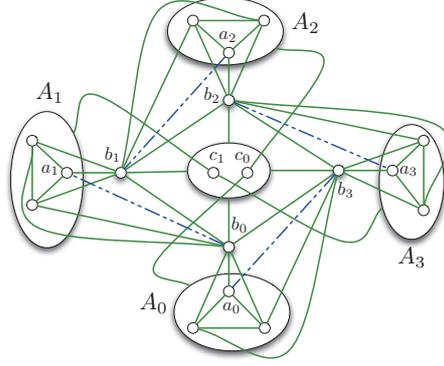


Figure 5: Graph that does not admit a 3-stable partition.

3. edges between b_i and $A_{i+1} \setminus \{a_{i+1}\}$ ($0 \leq i \leq 3$);
4. edges between b_i and b_{i-1} and b_{i+1} ($0 \leq i \leq 3$);
5. edges between c_0 and all the b_i , and edges between c_1 and all the b_i ;
6. edges between c_0 and $A_0 \cup A_2$, and edges between c_1 and $A_1 \cup A_3$.

Moreover, there are four edges with weight 0, namely the edges $b_i a_{i+1}$. All the remaining edges have weight $-\infty$. That is two nodes in different $A_i, A_{i'}$ are enemies; a user b_i is enemy of b_{i+2} and of the nodes in A_{i+2} and A_{i+3} ; c_0 and c_1 are enemies; c_0 is enemy of the nodes in A_1 and A_3 , and c_1 is enemy of the nodes in A_0 and A_2 . We now assume there exists a 3-stable partition P for the colouring game defined on $G = (V, w)$.

Claim 25. *Every node in A_i picks the same colour.*

Proof. It directly follows from Lemma 24 because all vertices in A_i are pairwise quasi-twins. \diamond

Claim 26. *b_i picks the same colour as nodes in A_i or nodes in A_{i+1} .*

Proof. Suppose it is not the case. Then $X_{c_{b_i}(P)}$ contains at most two other nodes: one of b_{i-1} and b_{i+1} (together enemies), and one of c_0 and c_1 (enemies). If $|X_{c_{b_i}(P)}| \leq 2$ or the group X_j containing A_i has size at least 3, then $(\{b_i\}, j)$ is a 1-deviation. So, we assume $|X_{c_{b_i}(P)}| = 3$, $h = 2$ and $A_i = X_j$. There are two cases. If $c_{b_i}(P) = c_{c_i}(P)$ then $(\{b_i, c_i\}, j)$ is a 2-deviation. Else, $c_{c_{i-1}}(P) = c_{b_i}(P)$, and $X_{c_{b_i}(P)} \cap \{b_{i-1}, b_{i+1}\} \neq \emptyset$; hence we can break P using $(X_{c_{b_i}(P)} \cap \{c_{i-1}, b_{i-1}, b_{i+1}\}, j')$, with $A_{i-1} \subseteq X_{j'}$ or $A_{i+1} \subseteq X_{j'}$. \diamond

Claim 27. *There is an i such that nodes A_i, b_i and b_{i-1} pick the same colour.*

Proof. Again, we show the claim by contradiction. We distinguish two cases:

Case 1: b_{i-1} is with A_i , but not b_i . So, as the claim is supposed to be false, b_i is with A_{i+1} , b_{i+1} is with A_{i+2} , and b_{i+2} is with A_{i+3} . Either $c_{b_{i-1}}(P) = c_{c_i}(P)$, hence $(\{b_i\}, c_{b_{i-1}}(P))$ breaks P , or $c_{b_{i-1}}(P) \neq c_{c_i}(P)$, hence $(\{b_i, c_i\}, c_{b_{i-1}}(P))$ breaks the partition.

Case 2: b_i is with A_i , but not b_{i-1} . So, as the claim is supposed to be false, b_{i-1} is with A_{i-1} , b_{i+1} is with A_{i+1} , and b_{i+2} is with A_{i+2} . Note either $c_{c_i}(P) = c_{b_i}(P)$ or $c_{c_i}(P) = c_{b_{i+2}}(P)$.

W.l.o.g., suppose $c_{c_i}(P) = c_{b_{i+2}}(P)$. Either $c_{b_{i-1}}(P) = c_{c_{i-1}}(P)$, hence $(\{b_i\}, c_{b_{i-1}}(P))$ breaks P , or $c_{b_{i-1}}(P) \neq c_{c_{i-1}}(P)$, hence it is $(\{b_i, c_{i-1}\}, c_{b_{i-1}}(P))$. \diamond

By Claim 27, it follows there is an i such that nodes in A_i, b_i, b_{i-1}, c_i all pick the same colour. Moreover, such a group is unique in P due to the conflict graph in (G, w) (induced by the conflict edges). By symmetry, we will assume $X_{c_{a_0}}(P) = \{b_0, b_3, c_0\} \cup A_0$.

Case 1: $c_{a_2}(P) = c_{b_1}(P) = c_{b_2}(P)$.

Either $c_{c_1}(P) = c_{a_1}(P)$ and $(\{b_1\}, c_{a_1}(P))$ breaks P , or $c_{c_1}(P) \neq c_{a_1}(P)$ and it is $(\{b_1, c_1\}, c_{a_1}(P))$.

Case 2: $c_{a_2}(P) = c_{b_2}(P) \neq c_{b_1}(P)$.

Either $c_{c_1}(P) = c_{a_3}(P)$ and $(\{b_2, b_3\}, c_{a_3}(P))$ breaks P , or $c_{c_1}(P) \neq c_{a_3}(P)$ and it is $(\{b_2, b_3, c_1\}, c_{a_3}(P))$.

Case 3: $c_{a_2}(P) = c_{b_1}(P) \neq c_{b_2}(P)$.

In that case, $(\{b_1\}, c_{a_1}(P))$ breaks P .

Case 4: $c_{a_2}(P) \neq c_{b_1}(P), c_{a_2}(P) \neq c_{b_2}(P)$, that implies $c_{b_2}(P) = c_{a_3}(P)$.

Either $c_{c_1}(P) = c_{a_3}(P)$ and $(\{b_3\}, c_{a_3}(P))$ breaks P , or $c_{c_1}(P) \neq c_{a_3}(P)$ and it is $(\{b_3, c_1\}, c_{a_3}(P))$.

Finally, there does not exist a 3-stable partition P for the colouring game defined on (G, w) . \square

We emphasize the following consequence of Proposition 22.

Observation 2. For $k > 1$, Algorithm 1 may not terminate even if a k -stable partition exists.

Proof. Let $G = (V, w)$ be the counter-example of Proposition 22 that does not admit a 3-stable partition. We construct the instance $G' = (V, w')$ from G by replacing every conflict edge by an edge with weight zero. On the one hand there exists a 3-stable partition for the colouring game defined on G' . On the other hand, one can construct an infinite sequence of steps by taking G as input for Algorithm 1, and it is a valid sequence of 3-deviations for G' as well. \square

4.2 Game with general weights

While the two results we obtained ($k(\mathcal{W}) \geq 1$ in general, $k(\mathcal{W}) = 2$ when \mathcal{W} contains a single positive weight) seem constrained, we now prove that these are the best results that one can hope for. Furthermore, we also want to consider in this section the case of “best friends” *i.e.*, two friends who want to interact unless there is an enemy of one of them to interfere. For this purpose, we put $w_{uv} = N$ where N denotes an arbitrarily large positive weight that is at least n times greater than any other (finite) weight in absolute value. Table 2 summarizes the most important values for $k(\mathcal{W})$, and it refers to the lemmas and proposition in the appendix where they are proved.

\mathcal{W}	$k(\mathcal{W})$	
$\{-\infty, a\}, a > 0$	∞	[22]
$\{-\infty, 0, a\}, a > 0$	2	Theorem 21, Proposition 22
$\{-\infty, a, b\}, b > a > 0$	1	Lemma 32
$\{-a, b\}, a > 0, b > 0$	$\leq 2 \cdot \lceil \frac{a+1}{b} \rceil + 1$	Lemma 35
$\{-\infty, N, -a\}, a > 0$	≤ 2	Corollary 34
$\mathbb{N} \cup \{N\}$ or $-\mathbb{N} \cup \{-\infty\}$	∞	(trivial)
$-\mathbb{N} \cup \{N\}$	∞	Proposition 36

Table 2: Values of $k(\mathcal{W})$ for different \mathcal{W} .

Finally we completely characterize the sets of weights \mathcal{W} which satisfy $k(\mathcal{W}) = \infty$.

Theorem 28. $k(\mathcal{W}) = \infty$ if, and only if: $\mathcal{W} \subseteq \mathbb{N} \cup \{N\}$; or $\mathcal{W} \subseteq -\mathbb{N} \cup \{-\infty\}$; or $\mathcal{W} = \{-\infty, a\}, a > 0$ (possibly $a = N$); or $\mathcal{W} \subseteq -\mathbb{N} \cup \{N\}$ (and so, $-\infty \notin \mathcal{W}$).

4.3 Intractability with conflict graphs

Under general weights, we have proved that not all games in general have a k -stable partition, even for relatively simple sets of weights and small values of k . On the other hand, one could argue that these results come from pathological cases which could be ruled out after checking a property of the graph. We prove it is unlikely to be the case: not only is the stability not guaranteed for a given game when $k > k(\mathcal{W})$, but it is computationally prohibitive to decide it. We first define:

Definition 29 (K-STABLE DECISION PROBLEM). *Let $k \geq 1$ and let \mathcal{W} be a (fixed) set of weights. Given a graph $G = (V, w)$ constrained to these weights, does there exist a k -stable partition for the colouring game defined on it ?*

Theorem 30. *For $k \geq 1$ and \mathcal{W} containing $-\infty$, either a k -stable partition always exists (i.e., $k \leq k(\mathcal{W})$); or the K-STABLE DECISION PROBLEM is NP-complete.*

The previous result of NP-hardness in [22] requires another kind of deviations in addition to the classical k -deviations we consider in our paper (see *gossip deviations* in Section 5). Moreover, the large positive weight N is essential in their proof, whereas ours overrules these strong constraints.

The rest of the section is devoted to prove Theorem 30. The K-STABLE DECISION PROBLEM is clearly in NP because one can decide whether a k -deviation exists in polynomial-time $n^{O(k)}$, for any fixed k . Informally, to prove the NP-hardness we will assume a counter-example to the K-STABLE DECISION PROBLEM exists, and we will build a supergraph of it that is arbitrarily large. We will characterize k -stable partitions for the colouring game defined on the supergraph. In particular, we will prove a necessary and sufficient condition so that a partition is k -stable is that one player from the counter-example picks the same colour as a large independent set from the supergraph. By doing so, we will be able to reduce the well-known MAXIMUM INDEPENDENT SET PROBLEM to the K-STABLE DECISION PROBLEM. The latter NP-complete problems seems to well encapsulate the difficulty of colouring games, as already observed in the uniform case [22].

For technical reasons, we will require one can lower-bound the utility of a player, in *any* 1-stable partition, by some positive constant. Intuitively, we make use of this lower-bound to ensure that in case a k -stable partition exists for the colouring game, then it means that some player from the counter-example picks the same colour as an independent set from the supergraph which is even larger. We now introduce two reductions so that one can obtain the lower-bound.

Reduction 1. *Let t be a positive integer, let \mathcal{W} be finite and such that $\mathcal{W} \cap (\mathbb{N} \cup \{N\}) \neq \emptyset$. We set $w_p = \max \mathcal{W}$, which is positive and may equal N .*

Given $G = (V, w)$ constrained to \mathcal{W} , and $n' \geq n = |V|$, we construct $G_{t, n'}$ as follows. We add to the graph n' distinct copies of the complete graph K_t whose edges are all weighted w_p . Then we add a conflict edge between any two nodes in two distinct copies of K_t , and an edge weighted w_p between any node in V and any node belonging to some copy of K_t .

Intuitively, Reduction 1 increases the minimum utility of the nodes to $w_p t$.

Reduction 2. *Let α be a positive integer, let \mathcal{W} be finite and such that $\mathcal{W} \cap (\mathbb{N} \cup \{N\}) \neq \emptyset$. We set $w_p = \max \mathcal{W}$, which is positive and may equal N .*

Given $G = (V, w)$ constrained to \mathcal{W} , we construct KG_α as follows. We replace every node $u \in V$ with a clique of α nodes $K_\alpha(u) \subseteq V(KG_\alpha)$. For all $u, v \in V$, every two nodes in $K_\alpha(u)$ are linked by an edge weighted w_p , and all nodes in $K_\alpha(u)$ are linked to all nodes in $K_\alpha(v)$ by edges weighted w_{uv} .

Said differently, we substitute every node in the graph by a clique. Our reduction will make use of the following lemma that we will prove in the appendix.

Lemma 31. *Let \mathcal{W} be finite and such that $\mathcal{W} \cap (\mathbb{N} \cup \{N\}) \neq \emptyset$. Given $G = (V, w)$, $n' \geq n = |V|$ and $t > n$, there exists a k -stable partition for the colouring game defined on G if, and only if, there exists a k -stable partition for the colouring game defined on $G_{t,n'}$.*

We are now able to prove Theorem 30:

Proof. [Theorem 30] Suppose there is $G_0 = (V_0, w^0)$ constrained to \mathcal{W} and such that it does not admit a k -stable partition. W.l.o.g. we constrain \mathcal{W} to the set of weights on the edges of G_0 so that it is finite. We notice that $\mathcal{W} \cap (\mathbb{N} \cup N) \neq \emptyset$ because otherwise, the partition with only singleton groups is k -stable for G_0 . One can also assume there exists some $x_0 \in V_0$ whose removal makes the existence of a k -stable partition for the gotten subgraph. Indeed, otherwise, we remove nodes sequentially until obtaining this property.

Let P^0 be a k -stable partition for the colouring game defined on $G_0 \setminus x_0$. The partition $P^0 \cup \{x_0\}$ is not k -stable by the hypothesis and so, let $f_0^{P^0}$ be the *maximum* utility node x_0 can obtain after any k -deviation happens which breaks $P^0 \cup \{x_0\}$. We define f_0 as the maximum value $f_0^{P^0}$, taken amongst all such k -stable partitions; if $f_0 \leq 0$, then we replace G_0 with \tilde{G}_{0t',n'_0} for t', n'_0 large enough (in such case, all above properties of the counter-example still hold by Lemma 31). By setting $w_p = \max \mathcal{W}$, one can define two other constants, namely $\alpha = \lceil \frac{f_0}{w_p} \rceil$ and $c_0 = 2n_0 + 1$, with $n_0 = |V_0|$.

We can now prove the NP-hardness by using a polynomial reduction for the MAXIMUM INDEPENDENT SET PROBLEM. Let $G = (V, E)$ be a graph, and let $c \geq c_0$ be an integer. We define $D_G = (V, w_G)$ such that $\forall uv \in E, w_{uv} = -\infty$ and $\forall uv \notin E, w_{uv} = w_p$. Let $t = \lfloor \alpha c - \frac{f_0}{w_p} \rfloor$, and let $G_1 = \tilde{G}_{0t,n_0}$, let $G_2 = KD_{G\alpha}$. Observe that $t > \alpha c - \frac{f_0}{w_p} - 1 \geq \alpha c - n_0 - 1 \geq c - n_0 - 1 \geq n_0$, because $f_0 \leq n_0 w_p$.

We finally build the graph H_G from G_1 and G_2 as follows. We add a conflict edge between any node of $G_1 \setminus x_0$ and any node of G_2 . All nodes of G_2 are linked to x_0 with an edge weighted w_p .

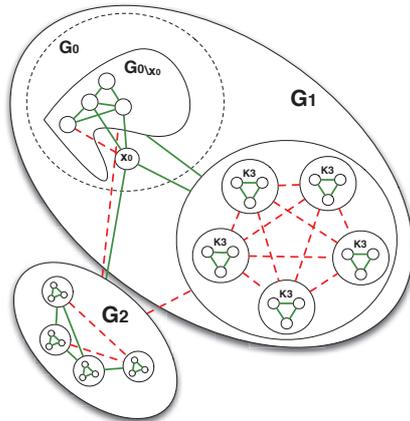


Figure 6: The transformation of an input.

The transformation above is illustrated in Figure 6. First assume that every independent set of G has a size lower than c . By contradiction, suppose there exists a k -stable partition for the colouring game defined on H_G . In such case, there can be no group with more than $\alpha(c-1)$ vertices of $V(G_2)$. Furthermore, nodes coloured as x_0 are either all in $V(G_1)$ or all in $V(G_2)$. Since $\alpha(c-1) = \alpha c - \lceil \frac{f_0}{w_p} \rceil \leq \alpha c - \frac{f_0}{w_p} - 1 < t$, it follows all such nodes belong to $V(G_1)$. Consequently, any k -stable partition for the colouring game defined on H_G , can be decomposed as follows: a k -stable partition for the colouring game defined on G_1 , and a k -stable partition for the colouring game defined on G_2 . Since G_0 does not admit a k -stable partition, hence G_1 does not admit one either by Lemma 31, it follows that H_G does not admit a k -stable partition.

Conversely, assume that there exists an independent set of G with size at least c . By [22], there exists a k -stable partition P^α for $G_2 \cup \{x_0\}$ which contains as a group X_j a maximum independent set of $G \cup \{x_0\}$. Moreover, $x_0 \in X_j$ because x_0 is an isolated vertex in the graph $G \cup \{x_0\}$. We also have that there exists a k -stable partition P^0 for $G_0 \setminus x_0$ and so, there exists a k -stable partition P^1 for $G_1 \setminus x_0 = (G_0 \setminus x_0)_{t, n_0}$ by Lemma 31. Last, we claim that $P^H = P^\alpha \cup P^1$ is a k -stable partition for H_G . Indeed, on the one hand we have that the utility of x_0 in P^α is at least $w_p \alpha c$. On the other hand, the maximum utility x_0 can get after a k -deviation that breaks P^1 is $f_0^{P^0} + w_p t \leq f_0 + w_p t = w_p(t + \frac{f_0}{w_p}) \leq w_p \alpha c$. We can conclude the NP-hardness, as our transformation is polynomial, and the MAXIMUM INDEPENDENT SET PROBLEM is NP-complete [21]. \square

5 Extensions of colouring games

All theoretical models of social dynamics should consider whether the overall behavior of the model is not too limited by some simplifying assumptions. We now discuss the way our results extend to account for various situations in the formation of social groups, including variants previously discussed and new ones. Full proofs of our results are provided in our technical report [14].

5.1 Gossiping

In this model, all k -deviations as previously defined are allowed. In addition two players in two distinct groups may “gossip” *i.e.*, both groups they are part of are merged. Obviously, and as before, this deviation will only take place if the two players benefit from the merge, but what is unique here is that the deviation does not require that other players in these groups benefit from the merge. They may even see a decrease in utility, but they are not given a choice to block the deviation — although they are in some sense seeing a modification of their group. This actually turns out to be equivalent to our model *in the uniform case*: one can check that there is a 1-deviation whenever there exists a gossip-deviation. Consequently all our results apply in that case, closing previously open problem with this model. In the general case, we prove that gossip creates instability even when a unique and fixed positive weight exists. We remind that without gossip, in such case a 2-stable partition always exists (see Theorem 21).

5.2 Asymmetry

Studying directed graphs rather than undirected graphs is a natural generalization. In this case, we may not have that $w_{uv} = w_{vu}$ for all players u, v . However, even if modest generalization of the model, asymmetrical weights lead to intractability. This can be seen with a simple digraph $D = (\{u, v\}, w)$ such that $w_{uv} > 0$ whereas $w_{vu} < 0$. Furthermore, the problem of deciding whether there exists a 1-stable partition is NP-hard. This result holds even when there can be no more than two groups in the partition.

5.3 Multichannel model and overlapping groups

One assumption of our model is that players form a partition, hence they are limited to a single channel to interact with their peers. In reality participants in social networks may engage in *multiple* groups. This motivates us to extend partitions into multisets, that we call 'configurations'. A configuration C is said to use q channels if each player participates (at most) to q groups. The utility of u depends on the number of groups that u shares with each peer:

$$f_u(C) = \sum_{v \in V} h(|X(u) \cap X(v)|, w_{uv}), \quad (1)$$

where $X(u)$ denotes the list of colours of u , and $h(g, w)$ is a function measuring the utility of sharing g groups with a player with weight w . Note that we assume, without loss of generality, that

$$\begin{aligned} h(0, \cdot) &= 0, \quad h(\cdot, 0) = 0 \quad \text{and} \quad \forall w \in \mathbb{Z}, \quad h(1, w) = w, \\ \forall g \in \mathbb{N}, \quad w &\mapsto h(g, w) \quad \text{is a non-decreasing function,} \\ \forall w \in \mathbb{Z}, \quad g &\mapsto w \cdot h(g, w) \quad \text{is a non-decreasing function.} \end{aligned}$$

The last property simply ensures that $h(g, w)$ increases with g when w is positive, and decreases with g when w is negative. Using the same potential function as before *e.g.*, global utility, it follows that there always exists a 1-stable configuration with q channels. It is now natural to wonder what is the behavior of $k_q(\mathcal{W})$ when the number of channels q is higher than 1. While it may have none or positive effects, we show that it is not always the case. Indeed, even for uniform games, we prove $k_2(\{-\infty, 1\}) \leq 2$ with 2 channels, while $k_1(\{-\infty, 1\}) = k(\{-\infty, 1\}) = \infty$.

5.4 Multi-modal relationship

Our model so far is heavily biased towards pairwise relationships, as the utility of a player depends on the sum of her interactions with all other members of the groups. In reality, more subtle interactions occur: one may be interested to interact with either friend u or v , but would not like to join a group where both of them are present. Our analysis also generalizes to this case. We use a hypergraph based model and we prove that, for instance, there always exists a 1-stable partition.

Acknowledgments

The authors would like to thank Jean-Claude Bermond and Julio Araujo for their useful comments, and Katrina Ligett for her valuable feedback.

References

- [1] R. Aumann. Acceptable Points in General Cooperative n-Person Games. In R. D. Luce and A. W. Tucker, editors, *Contributions to the Theory of Games IV, Annals of Mathematics Studies 40*, pages 287–324. Princeton University Press, Mar. 2005.
- [2] H. Aziz, F. Brandt, and P. Harrenstein. Pareto optimality in coalition formation. In *SAGT'11: Proceedings of the 4th international conference on Algorithmic game theory*. Springer-Verlag, Oct. 2011.
- [3] H. Aziz, F. Brandt, and H. G. Seedig. Computing desirable partitions in additively separable hedonic games. *Artificial Intelligence*, 195, Feb. 2013.

- [4] C. Ballester. NP-completeness in hedonic games. *Games and Economic Behavior*, 49(1):1–30, Oct. 2004.
- [5] S. Banerjee, H. Konishi, and T. Sönmez. Core in a simple coalition formation game. *Social Choice and Welfare*, 18(1):135–153, Jan. 2001.
- [6] B. D. Bernheim, B. Peleg, and M. D. Whinston. Coalition-proof nash equilibria i. concepts. *Journal of Economic Theory*, 42(1):1–12, 1987.
- [7] B. D. Bernheim and M. D. Whinston. Coalition-proof nash equilibria ii. applications. *Journal of Economic Theory*, 42(1):13–29, 1987.
- [8] A. Bogomolnaia and M. O. Jackson. The Stability of Hedonic Coalition Structures. *Games and Economic Behavior*, 38(2):201–230, Feb. 2002.
- [9] T. Brylawski. The lattice of integer partitions. *Discrete Mathematics*, 6(3):201 – 219, 1973.
- [10] N. Burani and W. S. Zwicker. Coalition formation games with separable preferences. *Mathematical Social Sciences*, 45(1):27–52, Feb. 2003.
- [11] I. Chatzigiannakis, C. Koninis, P. N. Panagopoulou, and P. G. Spirakis. Distributed game-theoretic vertex coloring. In *OPODIS’10: Proceedings of the 14th international conference on Principles of distributed systems*. Springer-Verlag, Dec. 2010.
- [12] X. Deng and C. Papadimitriou. On the Complexity of Cooperative Solution Concepts. *Mathematics of Operations Research*, 19(2):257–266, May 1994.
- [13] J. H. Drèze and J. Greenberg. Hedonic Coalitions: Optimality and Stability. *Econometrica*, 48(4):987–1003, May 1980.
- [14] G. Ducoffe, D. Mazaauric, and A. Chaintreau. Convergence of Coloring Games with Collusions. *Technical Report (available through arXiv)*, pages 1–10, Dec. 2012.
- [15] B. Escoffier, L. Gourvès, and J. Monnot. Strategic coloring of a graph. *Internet Mathematics*, 8(4):424–455, 2012.
- [16] M. Gairing and R. Savani. Computing Stable Outcomes in Hedonic Games. In *Algorithmic Game Theory*, pages 174–185. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.
- [17] E. Goles and M. A. Kiwi. Games on line graphs and sand piles. *Theoretical Computer Science*, 115(2):321 – 349, 1993.
- [18] C. Greene and D. J. Kleitman. Longest chains in the lattice of integer partitions ordered by majorization. *Eur. J. Comb.*, 7(1):1–10, Jan. 1986.
- [19] D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis. How easy is local search? *Journal of computer and system sciences*, 37(1):79–100, 1988.
- [20] M. Karakaya. Hedonic coalition formation games: A new stability notion. *Mathematical Social Sciences*, 61(3):157–165, May 2011.
- [21] R. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. 1972.

- [22] J. M. Kleinberg and K. Ligett. Information Sharing in Social Networks. *Games and Economic Behavior*, 82(C):702–716, Nov. 2013.
- [23] D. Moreno and J. Wooders. Coalition-proof equilibrium. *Games and Economic Behavior*, 17(1):80–112, 1996.
- [24] M. Olsen. Nash Stability in Additively Separable Hedonic Games and Community Structures. *Theory of Computing Systems*, 45(4):917–925, Jan. 2009.
- [25] M. Olsen, L. Bækgaard, and T. Tambo. On non-trivial Nash stable partitions in additive hedonic games with symmetric 0/1-utilities. *Information Processing Letters*, 112(23), Dec. 2012.
- [26] P. N. Panagopoulou and P. G. Spirakis. A game theoretic approach for efficient graph coloring. *The 19th International Symposium on Algorithms and Computation (ISAAC 2008)*, pages 183–195, 2008.
- [27] S. Pápai. Unique stability in simple coalition formation games. *Games and Economic Behavior*, 48(2):337–354, Aug. 2004.
- [28] I. Ray. Coalition-proof correlated equilibrium: A definition. *Games and Economic Behavior*, 17(1):56–79, 1996.
- [29] S.-C. Sung and D. Dimitrov. On Myopic Stability Concepts for Hedonic Games. *Theory and Decision*, 62(1):31–45, Dec. 2006.
- [30] S.-C. Sung and D. Dimitrov. Computational complexity in additive hedonic games. *European Journal of Operational Research*, 203(3):635–639, June 2010.
- [31] Y. Zick, E. Markakis, and E. Elkind. Arbitration and stability in cooperative games with overlapping coalitions. *Journal of Artificial Intelligence Research*, 50(1), May 2014.

.1 Proofs of Section 4

Proof. [Lemma 20] By the hypothesis, the utility of each vertex in S increases by at least 1. So, we get as the variation of the utility for the whole k -set: $\sum_{i=1}^k [f_{u_i}(P') - f_{u_i}(P)] \geq k$.

For every $1 \leq i \leq k$, we then define $\delta_i = \sum_{v \in X_{c_{u_i}(P)} \setminus S} w_{u_i v}$ and $\sigma_i = \sum_{u_l \in X_{c_{u_i}(P)}} w_{u_l u_i}$.

We define $\delta'_i = \sum_{v \in X_j} w_{u_i v}$ and $\sigma'_i = \sum_{l=1}^k w_{u_i u_l}$ in a similar way.

Then, $f_{u_i}(P) = \delta_i + \sigma_i$ and $f_{u_i}(P') = \delta'_i + \sigma'_i$. So, we get by summation that:

$$\sum_{i=1}^k f_{u_i}(P) = \sum_{i=1}^k \delta_i + 2 \sum_{\{u_i, u_l | c_{u_i}(P) = c_{u_l}(P)\}} w_{u_i u_l},$$

while $\sum_{i=1}^k f_{u_i}(P') = \sum_{i=1}^k \delta'_i + 2 \sum_{1 \leq i, l \leq k} w_{u_i u_l}$. Note that we have a factor 2 for any occurrence of $w_{u_i u_l}$, as it is counted once for u_i and once for u_l .

Furthermore, the variation of the global utility includes that of the nodes in S , that of the nodes in X_j , plus that of the nodes in $X_{c_{u_i}(P)} \setminus S$ for every $1 \leq i \leq k$. In other words, we get by symmetry that:

$$\begin{aligned} f(P') - f(P) &= \sum_{i=1}^k [f_{u_i}(P') - f_{u_i}(P)] + \sum_{i=1}^k [\delta'_i - \delta_i] \\ &= 2 \sum_{i=1}^k [f_{u_i}(P') - f_{u_i}(P)] - 2 \sum_{1 \leq i, l \leq k} w_{u_i u_l} + 2 \sum_{\{u_i, u_l | c_{u_i}(P) = c_{u_l}(P)\}} w_{u_i u_l} \\ &\geq 2k - 2 \sum_{1 \leq i, l \leq k} w_{u_i u_l} + 2 \sum_{\{u_i, u_l | c_{u_i}(P) = c_{u_l}(P)\}} w_{u_i u_l}. \end{aligned} \quad \square$$

Proof. [Lemma 31] We first remind that one obtains $\tilde{G}_{t, n'}$ from G using n' distinct copies of the complete graph K_t , that we will denote by $K_t^1, \dots, K_t^{n'}$ in the following.

First, let $P = (X_1, X_2, \dots, X_n)$ be a k -stable partition for the colouring game defined on G . We claim $P' = (X_1 \cup V(K_t^1), X_2 \cup V(K_t^2), \dots, X_n \cup V(K_t^{n'}), V(K_t^{n'+1}), \dots, V(K_t^{n'+n'}))$ is a k -stable partition for $\tilde{G}_{t, n'}$. By contradiction, let (S, j) be a k -deviation that breaks P' , with $1 \leq j \leq n + tn'$. Note that w.l.o.g., one can assume no node in S is coloured j in P , as $(S \setminus X_j, j)$ is still a k -deviation that breaks P' . In addition, we must have $S' = S \cap V(G) \neq \emptyset$. Let $1 \leq j' \leq n$ be such that $X_{j'} \cap (V(G) \setminus S') = \emptyset$. Either nodes in S pick a colour j no node in $V(G) \setminus S'$ was using and so, (S', j') is a k -deviation that breaks P ; or $S = S'$, and (S, j) also breaks P . A contradiction in both cases, because P is assumed to be k -stable.

Conversely, assume by contradiction that there is no k -stable partition for G , whereas there exists a k -stable partition $P' = (X'_1, \dots, X'_{n+tn'})$ for $\tilde{G}_{t, n'}$. By Lemma 24, for any i all nodes in K_t^i are coloured accordingly. Furthermore, we note by construction of $\tilde{G}_{t, n'}$ that for all nodes $u \in V(G)$, she has to be pick the same colour as some clique K_t^i (else, the partition P' would not even be 1-stable). Let P be the partition of $V(G)$ whose non-empty groups are those amongst $X'_1 \cap V(G), X'_2 \cap V(G), \dots, X'_{n+tn'} \cap V(G)$. By construction, P is not k -stable but then, every k -deviation (S, j) that breaks P also breaks P' . Consequently, there does not exist any k -stable partition for $\tilde{G}_{t, n'}$. \square

Counter-examples for stability Counter-examples with different sets of weights are presented in the following. We first make an observation which is the starting point of most of our counter-examples for stability: namely, the weights 2, 3, 4 in the counter-example of Figure 1 can be replaced by any weights w_1, w_2, w_3 such that $w_1 < w_2 < w_3$, and $w_1 + w_2 > w_3$. Especially, they can be replaced by $b, b + 1, b + 2$, for any $b \geq 2$. We now show through a counter-example no 2-stable partition exists in general when \mathcal{W} contains two positive elements, even when the zero is not present.

Lemma 32. *Let a, b be two positive integers such that $a < b$. There is a graph $G = (V, w)$ constrained to $\mathcal{W} = \{-\infty, a, b\}$ that does not admit a 2-stable partition.*

Proof. Users are partitioned into four sets $U_1 = \{x_1, x_2, x_3\}$, $U_2 = \{y_1, y_2, y_3\}$, $U_3 = \{z_1, z_2, z_3\}$ and $\{v_1, v_2, v_3\}$. Each of these sets is a clique with edges weighted b . In addition, each edge between a node in U_i and another node in $U_{i'}$, $i \neq i'$ is a conflict edge. Node v_1 is linked to nodes in U_2 with edges weighted b , to nodes in U_1 with conflict edges. In the same way, node v_2 is linked to nodes in U_3 with edges weighted b , to nodes in U_2 with conflict edges; node v_3 is linked to nodes in U_1 with edges weighted b , to nodes in U_3 with conflict edges. We finally set: $w_{v_1z_1} = w_{v_1z_2} = b$, $w_{v_1z_3} = a$; $w_{v_2x_1} = w_{v_2x_2} = b$, $w_{v_2x_3} = a$; $w_{v_3y_1} = w_{v_3y_2} = b$, $w_{v_3y_3} = a$.

Let us assume by contradiction there exists a 2-stable partition P .

Claim 33. *For any $1 \leq i \leq 3$, nodes in U_i are coloured accordingly.*

Proof. By symmetry, it suffices to show the claim for U_1 . First, nodes x_1, x_2 pick the same colour by Lemma 24. Furthermore, by construction there can be no node coloured $c_{x_1}(P)$ that is an enemy of x_3 . Since $\sum_{s \in V \setminus V_1} \max\{0, w_{x_3s}\} = b + a < 2b$, it follows $c_{x_3}(P) = c_{x_1}(P) = c_{x_2}(P)$ (otherwise, the partition would not even be 1-stable). \diamond

We can then substitute subsets U_i with nodes u_i through contractions, and by doing so, one obtains the variation of Figure 1 with weights $w_1 = b$, $w_2 = 2b + a$, $w_3 = 3b$. Let P' be the partition of the nodes one obtains from P in this new graph. Since one can always find a 2-deviation using nodes v_1, v_2, v_3 that breaks P' , one can deduce from such deviation a 2-deviation in G that breaks P . \square

Corollary 34. *If $\{-\infty, N\}$ is a strict subset of \mathcal{W} , then $k(\mathcal{W}) \leq 2$.*

Proof. First assume there exists a positive weight $a \in \mathcal{W} \setminus \{N\}$. Then we can apply Lemma 32 to the subset $\{-\infty, a, N\}$, and then one obtains $k(\mathcal{W}) = 1$. Else, let G_0 be the counter-example from Proposition 22, constrained to $\{-\infty, 0, 1\}$ and that does not admit a 3-stable partition. Let a non-negative be such that $-a \in \mathcal{W} \setminus \{N\}$. We construct G_1 from G_0 by replacing edges weighted 1 with edges weighted N , and null-weight edges with edges weighted $-a$. As there does not exist any 3-stable partition for G_0 , there does not exist any 3-stable partition for G_1 either. Thus, $k(\mathcal{W}) \leq 1$. \square

Lemma 35. *Let a, b be positive integers (not necessarily distinct). There is $G = (V, w)$ constrained to $\mathcal{W} = \{-a, b\}$ that does not admit a $2 \cdot (1 + \lceil \frac{a+1}{b} \rceil)$ -stable partition.*

Proof. Let x, y non-negative be such that $bx - ay = \gcd\{a, b\} = d$. The vertex-set is partitioned in $V_1, V_2, V_3, U_1^+, U_2^+, U_3^+, U_1^-, U_2^-, U_3^-$ plus three vertices u_1^-, u_2^-, u_3^- .

Any subset V_i has size $1 + \lceil \frac{a+1}{b} \rceil$, and there are constants k_1, k_2, k_3 that we assume sufficiently large

so that subsets U_i^+, U_i^- have respective size $\lceil \frac{1+b(1+\lceil \frac{a+1}{b} \rceil)}{d} \rceil \cdot x + k_i \cdot a$, $\lceil \frac{1+b(1+\lceil \frac{a+1}{b} \rceil)}{d} \rceil \cdot y + k_i \cdot b - 1$.

As for the weights, subsets $V_1 \cup V_2 \cup V_3$ and $U_i = U_i^+ \cup U_i^- \cup \{u_i^-\}$ all induce complete subgraphs of the friendship graph G^+ (here induced by edges weighted b). Any node of U_i^+ is also linked to all nodes in $(V_1 \cup V_2 \cup V_3) \setminus V_i$ with edges weighted positively. Last, node u_1 is adjacent in G^+ to all nodes in V_3 , and similarly node u_2 is adjacent in G^+ to all nodes in V_1 , node u_3 is adjacent in G^+ to all nodes in V_2 . Every remaining edge has weight $-a$.

Suppose by contradiction there is a $2(1 + \lceil \frac{a+1}{b} \rceil)$ -stable partition P for the colouring game defined on G . By Lemma 24, all nodes in subsets $V_1, V_2, V_3, U_1^+, U_2^+, U_3^+, U_1^-, U_2^-, U_3^-$ are coloured accordingly. Furthermore for any $i > j$, assuming $k_i \gg k_j \gg 1 + \lceil \frac{a+1}{b} \rceil$ we have that no node of $U_j \cup V_i$ picks the same colour as nodes in U_i^+ , nor the same colour as nodes in U_i^- (else, the partition would not even be 1-stable). Under similar assumptions, one obtains that all nodes in U_i are coloured accordingly. By contracting subsets V_i and U_i in the obvious way, one thus obtains

the variation of Figure 1 with weights $w_1 = (1 + \lceil \frac{a+1}{b} \rceil) \cdot b \geq b + a + 1$; $w_2 = \lceil \frac{1+b(1+\lceil \frac{a+1}{b} \rceil)}{d} \rceil \cdot d > w_1$; $w_3 = b + a + w_2 < w_1 + w_2$.

Let P' be the partition of the nodes one obtains from P in this new graph. Since one can always find a 2-deviation using nodes v_1, v_2, v_3 that breaks P' , one can deduce from such deviation a $2 \cdot (1 + \lceil \frac{a+1}{b} \rceil)$ -deviation in G that breaks P . \square

Positive result

Proposition 36. *If $\mathcal{W} \subseteq -\mathbb{N} \cup \{N\}$, then $k(\mathcal{W}) = \infty$.*

Proof. Let $G = (V, w)$ constrained to \mathcal{W} , and let G^+ be the friendship graph that is here induced by edges weighted N . We define P as the partition of the nodes whose non-empty groups are exactly the connected components of G^+ . Suppose (S, j) is a deviation that breaks P . On the one hand, no node has an incentive to decrease the number of best friends in her group, which means S is a collection of groups in P whose all vertices change their colour. On the other hand we have by the definition of P that no node in S can increase her number of best-friends by changing her colour. As a result, no such deviation can exist and so, P is a k -stable partition for any $k \geq 1$. \square

APPENDIX J

The parallel complexity of coloring games

The parallel complexity of coloring games [★]

Guillaume Ducoffe¹

Université Côte d’Azur, Inria, CNRS, I3S, France

Abstract. We wish to motivate the problem of finding *decentralized lower-bounds* on the complexity of computing a Nash equilibrium in graph games. While the centralized computation of an equilibrium in polynomial time is generally perceived as a positive result, this does not reflect well the reality of some applications where the game serves to implement distributed resource allocation algorithms, or to model the social choices of users with limited memory and computing power. As a case study, we investigate on the parallel complexity of a game-theoretic variation of graph coloring. These “coloring games” were shown to capture key properties of the more general welfare games and Hedonic games. On the positive side, it can be computed a Nash equilibrium in polynomial-time for any such game with a local search algorithm. However, the algorithm is time-consuming and it requires polynomial space. The latter questions the use of coloring games in the modeling of information-propagation in social networks. We prove that the problem of computing a Nash equilibrium in a given coloring game is PTIME-hard, and so, it is unlikely that one can be computed with an efficient distributed algorithm. The latter brings more insights on the complexity of these games.

1 Introduction

In algorithmic game theory, it is often the case that a problem is considered “tractable” when it can be solved in polynomial time, and “difficult” only when it is NP-hard or it is PLS-hard to find a solution. On the other hand, with the growing size of real networks, it has become a boiling topic in (non game-theoretic) algorithmic to study on the finer-grained complexity of polynomial problems [16]. In our opinion, the same should apply to graph games when they serve as a basis for new distributed algorithms. We propose to do so in some cases when it can be easily computed a Nash equilibrium in polynomial time. The following case study will make use of well-established parallel and space complexity classes to better understand the hardness of a given graph game.

Precisely, we investigate on a “coloring game”, first introduced in [14] in order to unify classical upper-bounds on the chromatic number. Since then it has been rediscovered many times, attracting attention on the way in the study of information propagation in wireless sensor networks [4] and in social networks [12]. We choose to consider this game since it is a good representative of the *separable welfare games* – proposed in [13] as a game-theoretic toolkit for distributed

[★] This work is partially supported by ANR project Stint under reference ANR-13-BS02-0007 and ANR program “Investments for the Future” under reference ANR-11-LABX-0031-01.

resource allocation algorithms – and the *additively separable symmetric Hedonic games* [3]. A coloring game is played on an undirected graph with each vertex being an agent (formal definitions will be given in the technical sections of the paper). Agents must choose a colour in order to construct a proper coloring of the graph. The individual goal of each agent is to maximize the number of agents with the same colour as hers. Furthermore, it can always be computed a Nash equilibrium in polynomial time with a simple local-search algorithm [6, 12, 14].

However, for n -vertex m -edge graphs, the above-mentioned algorithm has $\mathcal{O}(m + n\sqrt{n})$ -time complexity and $\mathcal{O}(n + m)$ -space complexity. Therefore, when the graph gets larger, potential applications of coloring games as a *computational mechanism design* (e.g., in order to assign frequencies in sensor networks in a distributed fashion, or to model the behaviour of social network users with limited power and storage) can be questioned. In particular, the authors in [11] report on the limited abilities of human subject networks to solve a coloring problem. In this note, we will investigate on the belonging of our problem – the computation of a Nash equilibrium in coloring games – to some complexity classes that are related to parallel and space complexity. Our goal in doing so is to bring more insights on the complexity of the problem.

Related work. Apart from lower-bounds in communication complexity [7], we are not aware of any analysis of decentralized complexity in game theory. Closest to our work are the studies on the sequential complexity of Hedonic games. Deciding whether a given Hedonic game admits a Nash equilibrium is NP-complete [1]. Every additively separable symmetric Hedonic games has a Nash equilibrium but it is PLS-complete to compute one [8]. Coloring games are a strict subclass where the local-search algorithm terminates on a Nash equilibrium within a polynomial number of steps. We will go one step further by considering their parallel complexity, something we think we are the first to study.

In [4], they introduced a distributed algorithm in order to compute the Nash equilibrium of a given coloring game. Their algorithm is a natural variation of the classical local-search algorithm for the problem, however, it does not speed up the computation of equilibria (at least theoretically). In addition, each agent needs to store locally the colouring of the graph at any given step, that implies *quadratic* space and communication complexity. Additional related work is [6, 12], where it is studied the number of steps of more elaborate local-search algorithms when up to k players are allowed to *collude* at each step. Informally, collusion means that the players can simultaneously change their colours for the same new colour provided they all benefit from the process (note that the classical local-search algorithm corresponds to the case $k = 1$).

Contributions. We prove that the problem of computing a Nash equilibrium in a given coloring game is PTIME-hard (Theorem 2). This is hint that the problem is inherently *sequential*, i.e., it is unlikely the computation of an equilibrium can be sped up significantly on a parallel machine with polynomially many processors. In particular, our negative result applies to the *distributed* setting since any distributed algorithm on graphs can be simulated on a parallel machine with

one processor per edge and per vertex. By a well-known relationship between space and parallel complexity [15], Theorem 2 also extends to show that no *space efficient* algorithm for the problem (say, within logarithmic workspace) can exist. Altogether, this may be hint that coloring games are a too powerful computational mechanism design for “lightweight” distributed applications.

Our reduction is from the standard MONOTONE CIRCUIT VALUE problem. However, the gadgets needed are technically challenging, and we will need to leverage nontrivial properties of coloring games in order to prove its correctness. Definitions and useful background will be given in Section 2. We will detail our reduction in Section 3 before concluding this paper in Section 4.

2 Definitions and notations

We use the graph terminology from [2]. Graphs in this study are finite, simple, and unweighted.

Coloring games. Let $G = (V, E)$ be a graph. A *coloring* of G assigns a positive integer, taken in the range $\{1, \dots, n\}$, to each of the n vertices in V . For every i , let L_i be the subset of vertices coloured i . We name L_i a *colour class* in what follows. Nonempty colour classes partition the vertex set V . The partition is a *proper coloring* when no two adjacent vertices are assigned the same colour, *i.e.*, for every $1 \leq i \leq n$ and for every $u, v \in L_i$, $\{u, v\} \notin E$.

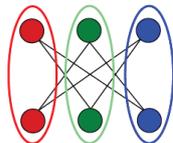


Fig. 1. Proper coloring of a graph G . Each colour class is represented by an ellipse. Every agent receives unit payoff.

Every graph G defines a *coloring game* whose n agents are the vertices in V . The strategy of an agent is her colour. Furthermore, every $v \in L_i$ receives payoff: -1 if there is $u \in L_i$ s.t. $\{u, v\} \in E$ (in which case, the coloring is not proper), and $|L_i| - 1$ otherwise. We refer to Figure 1 for an illustration. Finally, a *Nash equilibrium* of the coloring game is any coloring of G where no agent can increase her payoff by changing her strategy. In particular, the proper coloring in Figure 1 is a Nash equilibrium. More generally, observe that a Nash equilibrium in this game is always a proper coloring of G . In what follows, we will focus on the computation of Nash equilibria in coloring games.

Theorem 1 ([6, 12]). *For any coloring game that is specified by an n -vertex m -edge graph $G = (V, E)$, a Nash equilibrium can be computed in $\mathcal{O}(m + n\sqrt{n})$ -time and $\mathcal{O}(n + m)$ -space.*

Parallel complexity. Computations are performed on a parallel random-access machine (PRAM, see [9]) with an unlimited amount of processors. However, as

stated in the conclusion, our results also apply to more realistic parallel complexity classes. In what follows, we will use the fact that processors are numbered. We will handle with read/write conflicts between processors with the strategy CREW-PRAM (concurrent read, exclusive write). Let PTIME contain the decision problems that can be solved in *sequential* polynomial-time (that is, with a single processor). Problem A reduces to problem B if given an oracle to solve B, A can be solved in polylogarithmic-time with a polynomial number of processors. In particular, a problem B is PTIME-hard if every problem in PTIME reduces to B (this is formally defined as quasi-PTIME-hardness in [9]). Such reductions are finer-grained than the more standard logspace reductions.

3 Main Result

Theorem 2. *Computing a Nash equilibrium for coloring games is PTIME-hard.*

In order to prove Theorem 2, we will reduce from a variation of the well-known MONOTONE CIRCUIT VALUE problem, defined as follows.

Problem 1 (MONOTONE CIRCUIT VALUE).

Input: A boolean circuit \mathcal{C} with m gates and n entries, a word $w \in \{0, 1\}^n$ such that:

- the gates are either AND-gates or OR-gates;
- every gate has exactly two entries (in-degree two);
- a *topological ordering* of the gates is given, with the m^{th} gate being the output gate.

Question: Does \mathcal{C} output 1 when it takes w as input ?

MONOTONE CIRCUIT VALUE is proved to be PTIME-complete in [9].

3.1 The reduction

Let $\langle \mathcal{C}, w \rangle$ be any instance of MONOTONE CIRCUIT VALUE. We will reduce it to a coloring game as follows. Let $\mathcal{G} := (g_1, g_2, \dots, g_m)$ be the gates of the circuit, that are topologically ordered.

Construction of the gate-gadgets. For every $1 \leq j \leq m$, the j^{th} gate will be simulated by a subgraph $G_j = (V_j, E_j)$ with $12(n + j) - 9$ vertices. We refer to Figure 2 for an illustration. Let us give some intuition for the following construction of G_j . We aim at simulating the computation of the (binary) output of all the gates in \mathcal{C} when it takes w as input. To do that, given a supergraph G of G_j (to be defined later), and a fixed Nash equilibrium for the coloring game that is defined on G , we aim at guessing the output of the j^{th} gate from the subcoloring of G_j . More precisely, the subcoloring will encode a “local certificate” that indicates which values on the two entries of g_j cause the output.

Observe that to certify that an OR-gate outputs 1, it suffices to show that it receives 1 on any one of its two entries, whereas for an AND-gate it requires to

show that it outputs 1 on its two entries. Since by de Morgan’s laws, the negation of an AND-gate can be transformed into an OR-gate and vice-versa, therefore, we need to distinguish between three cases in order to certify the output of the gate. So, the vertices in V_j are partitioned in three subsets of equal size $4(n + j) - 3$, denoted by V_j^1, V_j^2, V_j^3 . Furthermore, for every $1 \leq t \leq 3$, every vertex in V_j^t is adjacent to every vertex in $V_j \setminus V_j^t$.

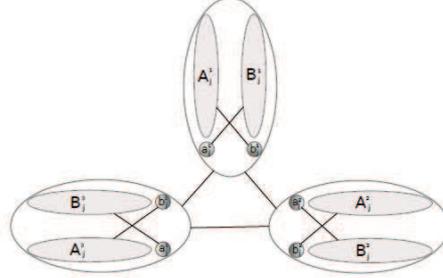


Fig. 2. Gadget subgraph G_j representing the j^{th} gate. An edge between two subsets of vertices (delimited by an ellipse) denotes the existence of a complete bipartite subgraph.

Let us now describe the structure of the three (isomorphic) subgraphs $G_j[V_j^t] = (V_j^t, E_j^t)$ with $1 \leq t \leq 3$. Informally, we will need this internal structure in order to ensure that every of the three subsets V_j^t will behave as a “truthful” certificate to decide on the output of the gate; *i.e.*, only a few vertices of V_j will be used to certify the output of the j^{th} gate, while all others will be divided into artificial aggregates that we name “private groups” whose role is to ensure “truthfulness” of the certificate (this will be made clearer in the following). There are two nonadjacent vertices $a_j^t, b_j^t \in V_j^t$ playing a special role. The other vertices in $V_j^t \setminus \{a_j^t, b_j^t\}$ are partitioned in two subsets A_j^t, B_j^t of respective size $2(n + j) - 3$ and $2(n + j) - 2$. The sets A_j^t, B_j^t are called the *private groups* of a_j^t, b_j^t . Furthermore, every vertex in A_j^t is adjacent to every vertex in $V_j^t \setminus (A_j^t \cup \{a_j^t\})$, similarly every vertex in B_j^t is adjacent to every vertex in $V_j^t \setminus (B_j^t \cup \{b_j^t\})$.

Since all edges are defined above independently the one from the other, the graph $G_j[V_j^1] = (V_j^1, E_j^1)$ (encoded by its adjacency lists) can be constructed with $|V_j^1| + |E_j^1| = 4(n + j)^2 - 2(n + j) - 2$ processors simply by assigning the construction of each vertex and each edge to a different processor. Note that each processor can decide on the vertex, resp. the edge, it needs to compute from its number. Overall, it takes $\mathcal{O}(\log(n + j))$ -time in order to construct $G_j[V_j^1]$ in parallel. The latter can be easily generalized in order to construct G_j in $\mathcal{O}(\log(n + j))$ -time with $|V_j| + |E_j|$ processors. Therefore, the graphs G_1, G_2, \dots, G_m can be constructed in parallel in $\mathcal{O}(\log(n + m))$ -time with $\sum_{j=1}^m (|V_j| + |E_j|)$ processors, that is polynomial in $n + m$.

Construction of the graph. Let $X = \{x_1, x'_1, \dots, x_i, x'_i, \dots, x_n, x'_n\}$ contain $2n$ nonadjacent vertices, that are two vertices per letter in the binary word w . The graph $G = (V, E)$ for the reduction has vertex-set $V = X \cup \left(\bigcup_{j=1}^m V_j \right)$. In particular, it has $2n - 9m + 6m(m + 2n + 1)$ vertices. Furthermore, $G[V_j]$ is

isomorphic to G_j for every $1 \leq j \leq m$. In order to complete our reduction, let us now describe how our gadgets are connected the one with the other.

For technical reasons, we will need to make adjacent every vertex in the private group A_j^t (resp. B_j^t), with $1 \leq j \leq m$ and $1 \leq t \leq 3$, to every vertex in $V \setminus V_j$. By doing so, note that every vertex in $V \setminus (A_j^t \cup \{a_j^t\})$ is adjacent to every vertex in A_j^t (resp., every vertex in $V \setminus (B_j^t \cup \{b_j^t\})$ is adjacent to every vertex in B_j^t). Furthermore, each edge is defined independently the one from the other. Hence, similarly as above, $\sum_{j=1}^m \sum_{t=1}^3 (|A_j^t| + |B_j^t|)|V \setminus V_j|$ processors are sufficient in order to construct these edges in $\mathcal{O}(\log(n + m))$ -time, that is polynomial in $n + m$.

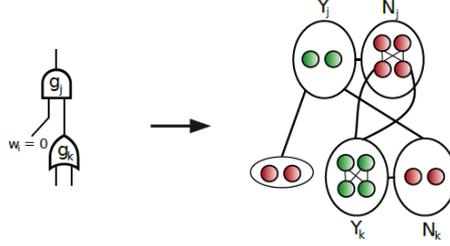


Fig. 3. Edges in G to simulate the two connections of an AND-gate in the circuit.

Finally, we recall that for every j , there are three cases to distinguish in order to decide on the output of the j^{th} gate, with each case being represented with some subset V_j^t . The union of subsets representing a *positive* certificate (output 1) is named Y_j , while the union of those representing a *negative* certificate (output 0) is named N_j . In particular, if the j^{th} gate is an OR-gate, let $Y_j := \{a_j^1, b_j^1, a_j^2, b_j^2\}$ and $N_j := \{a_j^3, b_j^3\}$ (it suffices to receive 1 on one input). Else, the j^{th} gate is an AND-gate, so, let $Y_j := \{a_j^1, b_j^1\}$ and $N_j := \{a_j^2, b_j^2, a_j^3, b_j^3\}$.

Suppose the j^{th} gate is an OR-gate (the case when it is an AND-gate follows by symmetry, up to interverting Y_j with N_j , see also Figure 3). Let us consider the first entry of the gate. There are two cases. Suppose that it is the i^{th} entry of the circuit, for some $1 \leq i \leq n$. If $w_i = 0$ then we make both x_i, x'_i adjacent to both a_j^1, b_j^1 ; else, $w_i = 1$, we make both x_i, x'_i adjacent to both a_j^3, b_j^3 . Else, the entry is some other gate of the circuit, and so, since gates are topologically ordered, it is the k^{th} gate for some $k < j$. We make every vertex in N_k adjacent to both a_j^1, b_j^1 , and we make every vertex in Y_k adjacent to both a_j^3, b_j^3 .

The second entry of the gate is similarly considered, up to replacing above the two vertices a_j^1, b_j^1 with a_j^2, b_j^2 . We refer to Figure 3 for an illustration. In particular, observe that there is only a constant number of edges that are added at this step for each gate. Furthermore, the construction of these new edges only requires to read the two in-neighbours of the gate in the circuit \mathcal{C} . As a result, the last step can be done in parallel in $\mathcal{O}(\log(n + m))$ -time with m processors.

3.2 Structure of a Nash equilibrium

The graph $G = (V, E)$ of our reduction (constructed in Section 3.1) defines a coloring game. Let us fix any Nash equilibrium for this game (that exists by

Theorem 1). We will show that it is sufficient to know the colour of every vertex in $Y_m \cup N_m$ in order to decide on the output of the circuit \mathcal{C} (recall that the m^{th} gate is the output gate). To prove it, we will need the following technical claims in order to gain more insights on the structure of the equilibrium.

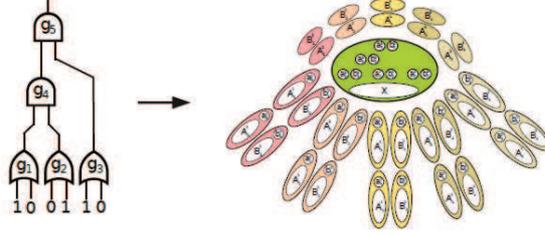


Fig. 4. A boolean circuit (left) with a Nash equilibrium of the coloring game from our reduction (right). Each colour class is represented with an ellipse. Intuitively, vertices in the central colour class simulate the computation of the output. Other colour classes contain a private group and they are “inactive”.

More precisely, we will prove that there are exactly $6m + 1$ colour classes, that are one colour class per private group A_j^t or B_j^t and an additional colour for the vertices in X . The intuition is that there are $2(n + m)$ vertices in one special colour class (including X) that simulates the computation of the output of \mathcal{C} , whereas all other vertices are “trapped” with the vertices in their respective private group. We refer to Figure 4 for an illustration.

Claim 1. *For every j , any colour class does not contain more than two vertices in every $Y_j \cup N_j$. Furthermore, if it contains exactly two vertices in $Y_j \cup N_j$ then these are a_j^t, b_j^t for some $1 \leq t \leq 3$.*

Proof. A Nash equilibrium is a proper coloring of G . Therefore, since any two vertices in different subsets among V_j^1, V_j^2, V_j^3 are adjacent by construction, they cannot have the same colour. Since $Y_j \cup N_j = \{a_j^1, b_j^1, a_j^2, b_j^2, a_j^3, b_j^3\}$ and $a_j^t, b_j^t \in V_j^t$ for every $1 \leq t \leq 3$, the claim follows directly. \diamond

Claim 2. *Any two vertices that are in a same private group have the same colour. Similarly, x_i and x'_i have the same colour for every $1 \leq i \leq n$.*

Proof. Let S be either a private group ($S = A_j^t$ or $S = B_j^t$ for some $1 \leq j \leq m$ and $1 \leq t \leq 3$), or a pair representing the same letter of word w (i.e., $S = \{x_i, x'_i\}$ for some $1 \leq i \leq n$). Let $v \in S$ maximize her payoff and let c be her colour. Note that v receives payoff $|L_c| - 1$ with L_c being the colour class composed of all the vertices with colour c . Furthermore, every $u \in S$ receives payoff lower than or equal to $|L_c| - 1$ by the choice of v . In such case, every $u \in S$ must be coloured c , or else, since the adjacency and the nonadjacency relations are the same for u and v (they are twins), furthermore u, v are nonadjacent, the agent u would increase her payoff to $|L_c|$ by choosing c as her new colour, thus contradicting the hypothesis that we are in a Nash equilibrium. \diamond

The argument we use in Claim 2 is that twin vertices must have the same colour. In what follows, we will use the same argument under different disguises.

Claim 3. *Let $1 \leq j \leq m$ and $1 \leq t \leq 3$. Either A_j^t or $A_j^t \cup \{a_j^t\}$ is a colour class, and in the same way either B_j^t or $B_j^t \cup \{b_j^t\}$ is a colour class. Furthermore, either $B_j^t \cup \{b_j^t\}$ is a colour class, or a_j^t and b_j^t have the same colour.*

Proof. Recall that a Nash equilibrium is a proper coloring of G . Since a_j^t is the only vertex in $V \setminus A_j^t$ that is nonadjacent to A_j^t , furthermore every two vertices in A_j^t have the same colour by Claim 2, therefore, either A_j^t or $A_j^t \cup \{a_j^t\}$ is a colour class. Similarly, either B_j^t or $B_j^t \cup \{b_j^t\}$ is a colour class. In particular, suppose that b_j^t does not have the same colour as her private group. Then, she must receive payoff at least $|B_j^t| = 2(n+j) - 2$ (else, she would increase her payoff by choosing the same colour as her private group, thus contradicting the hypothesis that we are in a Nash equilibrium). Furthermore, there can be only vertices in $V \setminus (A_j^t \cup B_j^t)$ with the same colour c as b_j^t . Suppose for the sake of contradiction that a_j^t does not have colour c . There are two cases to be considered.

Suppose that $A_j^t \cup \{a_j^t\}$ is a colour class. Then, a_j^t receives payoff $|A_j^t| = 2(n+j) - 3$. In such case, since a_j^t and b_j^t are twin vertices in $G \setminus (A_j^t \cup B_j^t)$, vertex a_j^t could increase her payoff to at least $2(n+j) - 1$ by choosing c as her new colour, thus contradicting the hypothesis that we are in a Nash equilibrium.

Else, a_j^t and b_j^t do not have the same colours as their respective private groups. In such case, A_j^t and B_j^t are colour classes, hence we can constrain ourselves to the subgraph $G \setminus (A_j^t \cup B_j^t)$. In particular, the restriction of the Nash equilibrium to the subgraph must be a Nash equilibrium of the coloring game defined on $G \setminus (A_j^t \cup B_j^t)$. Since a_j^t and b_j^t are twin vertices in $G \setminus (A_j^t \cup B_j^t)$, they must have the same colour by a similar argument as for Claim 2.

As a result, a_j^t must have colour c in both cases, that proves the claim. \diamond

We recall that we aim at simulating the computation of the output of all the gates in \mathcal{C} . To do that, we will prove the existence of a special colour class containing X and some pair in $Y_j \cup N_j$ for every j . Intuitively, the two vertices of $Y_j \cup N_j$ are used to certify the output of the j^{th} gate. However, this certificate is “local” in the sense that it assumes the output of the $j - 1$ smaller gates to be already certified. Therefore, we need to prove that there can be no “missing gate”, *i.e.*, every gate is represented in the special colour class.

Claim 4. *Let c be a colour such that $L_c \not\subseteq X$ and L_c does not intersect any private group (A_j^t or B_j^t for any $1 \leq j \leq m$ and $1 \leq t \leq 3$).*

Then, $X \subseteq L_c$ and there exists an index j_0 such that the following holds true: $|L_c \cap (Y_j \cup N_j)| = 2$ for every $1 \leq j \leq j_0$, and $L_c \cap (Y_j \cup N_j) = \emptyset$ for every $j_0 + 1 \leq j \leq m$.

Proof. By the hypothesis $L_c \not\subseteq X$ and L_c does not intersect any private group, so, there is at least one vertex of $\bigcup_{j=1}^m (Y_j \cup N_j)$ with colour c . Let j_0 be the largest index j such that there is a vertex in $Y_j \cup N_j$ with colour c . Since by Claim 1, there can be no more than two vertices of $Y_j \cup N_j$ that are in L_c for every j , therefore, by maximality of j_0 we get $|L_c| \leq |X| + 2j_0 = 2(n+j_0)$. In particular, observe that if $|L_c| = 2(n+j_0)$ then $X \subseteq L_c$ and for every $1 \leq j \leq j_0$ there are exactly two vertices in $Y_j \cup N_j$ with colour c . So, let us prove that $|L_c| = 2(n+j_0)$,

that will prove the claim. By the choice of j_0 , there is some $1 \leq t \leq 3$ such that $a_{j_0}^t \in L_c$ or $b_{j_0}^t \in L_c$. In particular, $|L_c| \geq \min\{|A_{j_0}^t|, |B_{j_0}^t|\} + 1 = 2(n + j_0) - 2$ or else, every vertex $v_{j_0}^t \in L_c \cap \{a_{j_0}^t, b_{j_0}^t\}$ would increase her payoff by choosing the colour of the vertices in her private group (that is a colour class by Claim 3), thus contradicting the hypothesis that we are in a Nash equilibrium.

We prove as an intermediate subclaim that for any $1 \leq j \leq j_0 - 1$ such that $L_c \cap (Y_j \cup N_j) \neq \emptyset$, there is some $1 \leq t' \leq 3$ such that $a_j^{t'}, b_j^{t'} \in L_c$. Indeed, in this situation, there is some t' such that $a_j^{t'} \in L_c$ or $b_j^{t'} \in L_c$. If $b_j^{t'} \in L_c$ then we are done as by Claim 3, $a_j^{t'} \in L_c$. Otherwise, $b_j^{t'} \notin L_c$ and we prove this case cannot happen. First observe that $a_j^{t'} \in L_c$ in this case. Furthermore, since $a_j^{t'}$ and $b_j^{t'}$ do not have the same colour we have by Claim 3 that $B_j^{t'} \cup \{b_j^{t'}\}$ is a colour class. In this situation, $b_j^{t'}$ receives payoff $2(n + j) - 2 \leq 2(n + j_0 - 1) - 2 < |L_c|$. Since in addition $a_j^{t'}$ and $b_j^{t'}$ are twins in $G \setminus (A_j^{t'} \cup B_j^{t'})$, vertex $b_j^{t'}$ could increase her payoff by choosing colour c , thus contradicting that we are in a Nash equilibrium. This proves $a_j^{t'}, b_j^{t'} \in L_c$, and so, the subclaim.

By the subclaim, there is an even number $2k$ of vertices in $\bigcup_{j=1}^{j_0-1} (Y_j \cup N_j)$ with colour c , for some $k \leq j_0 - 1$. Similarly, since by Claim 2 the vertices x_i, x'_i have the same colour for every $1 \leq i \leq n$, $|X \cap L_c| = 2n'$ for some $n' \leq n$. Now there are two cases to be considered.

Suppose that $b_{j_0}^t \in L_c$. Then, by Claim 3 $a_{j_0}^t \in L_c$. Furthermore $|L_c| \geq 2(n + j_0) - 1$ or else, vertex $b_{j_0}^t$ would increase her payoff by choosing the colour of the vertices in $B_{j_0}^t$ (that is a colour class by Claim 3), thus contradicting the hypothesis that we are in a Nash equilibrium. As a result, $|L_c| = 2(n' + k + 1) \geq 2(n + j_0) - 1$, that implies $n' + k \geq n + j_0 - 1$, and so, $|L_c| \geq 2(n + j_0)$, as desired.

Else, $b_{j_0}^t \notin L_c$ and we prove this case cannot happen. First observe that $a_{j_0}^t \in L_c$. Furthermore, $|L_c| = 2(n' + k) + 1 \geq 2(n + j_0) - 2$, that implies $n' + k \geq n + j_0 - 1$, and so, $|L_c| \geq 2(n + j_0) - 1$. However, since $a_{j_0}^t$ and $b_{j_0}^t$ do not have the same colour, $B_{j_0}^t \cup \{b_{j_0}^t\}$ is a colour class by Claim 3. In particular, $b_{j_0}^t$ receives payoff $2(n + j_0) - 2 < |L_c|$. Since $a_{j_0}^t, b_{j_0}^t$ are twins in $G \setminus (A_{j_0}^t \cup B_{j_0}^t)$, vertex $b_{j_0}^t$ could increase her payoff by choosing colour c , thus contradicting that we are in a Nash equilibrium.

Altogether, $|L_c| \geq 2(n + j_0)$, that proves the claim. \diamond

We point out that by combining Claim 1 with Claim 4, one obtains that for every $1 \leq j \leq m$, there are either zero or two vertices in $Y_j \cup N_j$ in each colour class not containing a private group, and in case there are two vertices then these are a_j^t, b_j^t for some $1 \leq t \leq 3$.

Claim 5. *Any two vertices in X have the same colour. Furthermore, for every $1 \leq j \leq m$, every vertex in $Y_j \cup N_j$ either has the same colour as vertices in X or as vertices in her private group.*

Proof. Let L_c be any colour class with at least one vertex in $\bigcup_{j=1}^m (Y_j \cup N_j)$. Let j_0 be the largest index j such that there is a vertex in $Y_j \cup N_j$ with colour c . In order to prove the claim, there are two cases to be considered. Suppose that

$L_c \neq A_{j_0}^t \cup \{a_{j_0}^t\}$ and $L_c \neq B_{j_0}^t \cup \{b_{j_0}^t\}$ for any $1 \leq t \leq 3$. We will prove that $X \subseteq L_c$, that will imply that L_c is unique in such a case, and so, will prove the claim. By the choice of colour c , $L_c \not\subseteq X$. Further, observe that there can be no private group with a vertex in L_c . As a result, this case follows directly from Claim 4.

Else, either $L_c = A_{j_0}^t \cup \{a_{j_0}^t\}$ or $L_c = B_{j_0}^t \cup \{b_{j_0}^t\}$ for some $1 \leq t \leq 3$, and we may assume that it is the case for any colour class L_c that contains at least one vertex in $\bigcup_{j=1}^m (Y_j \cup N_j)$ (or else, we are back to the previous case). So, let us constrain ourselves to the subgraph $G[X]$. In particular, the restriction of the Nash equilibrium to the subgraph must be a Nash equilibrium of the coloring game defined on $G[X]$. Since the vertices in X are pairwise nonadjacent, they must form a unique colour class in such case, that proves the claim. \diamond

We will need a “truthfulness” property to prove correctness of our reduction. Namely, the value of the output of any gate in the circuit must be correctly guessed from the agents with the same colour as vertices in X .

Claim 6. *Let $1 \leq j_0 \leq m$ such that for every $1 \leq j \leq j_0$, there is at least one vertex in $Y_j \cup N_j$ with the same colour c_0 as all vertices in X . Then for every $1 \leq j \leq j_0$, $L_{c_0} \cap Y_j \neq \emptyset$ if and only if the output of the j^{th} gate is 1.*

Proof. In order to prove the claim by contradiction, let $1 \leq j_1 \leq j_0$ be the smallest index j such that either $Y_j \cap L_{c_0} = \emptyset$ and the output of the j^{th} gate is 1 (false negative) or $Y_j \cap L_{c_0} \neq \emptyset$ and the output of the j^{th} gate is 0 (false positive). We will show that in such case, there is an edge with two endpoints of colour c_0 , hence the coloring is not proper, thus contradicting the hypothesis that we are in a Nash equilibrium. Note that since by de Morgan’s laws, the negation of an AND-gate can be transformed into an OR-gate and vice-versa, both cases are symmetrical, and so, we can assume w.l.o.g. that the j_1^{th} gate is an OR-gate. There are two subcases to be considered.

Suppose that the output of the j_1^{th} gate is 0 (false positive). In such case, $Y_{j_1} \cap L_{c_0} \neq \emptyset$. Let us consider the first entry of the gate. If it is the i^{th} entry of the circuit for some $1 \leq i \leq n$ then $w_i = 0$ (because the output of the j_1^{th} gate is 0) and so, by construction, $x_i, x'_i \in L_{c_0}$ are adjacent to $a_{j_1}^1, b_{j_1}^1$. Else, it is the k^{th} gate of the circuit for some $k < j_1$. By minimality of j_1 , since the output of the k^{th} gate must be 0 (because the output of the j_1^{th} gate is 0), $Y_k \cap L_{c_0} = \emptyset$, and so, $N_k \cap L_{c_0} \neq \emptyset$. By construction, every vertex in N_k is adjacent to $a_{j_1}^1, b_{j_1}^1$. As a result, $a_{j_1}^1, b_{j_1}^1$ have a neighbour in L_{c_0} in this subcase. We can prove similarly (by considering the second entry of the gate) that $a_{j_1}^2, b_{j_1}^2$ have a neighbour in L_{c_0} in this subcase. The latter implies the existence of an edge with both endpoints in L_{c_0} since $Y_{j_1} = \{a_{j_1}^1, b_{j_1}^1, a_{j_1}^2, b_{j_1}^2\}$.

Else, the output of the j_1^{th} gate is 1 (false negative). In such case, $Y_{j_1} \cap L_{c_0} = \emptyset$, hence $N_{j_1} \cap L_{c_0} \neq \emptyset$. Since the output of the gate is 1, there must be an entry of the gate such that: either it is the i^{th} entry of the circuit for some $1 \leq i \leq n$, and $w_i = 1$ (in which case, the two vertices $x_i, x'_i \in L_{c_0}$ are adjacent to both $a_{j_1}^3, b_{j_1}^3$ by construction); or it is the k^{th} gate of the circuit for some $k < j_1$ and

this gate outputs 1. In the latter case, by minimality of j_1 , $Y_k \cap L_{c_0} \neq \emptyset$. By construction, every vertex in Y_k is adjacent to $a_{j_1}^3, b_{j_1}^3$. As a result, $a_{j_1}^3, b_{j_1}^3$ have a neighbour in L_{c_0} in this subcase. The latter implies the existence of an edge with both endpoints in L_{c_0} since $N_{j_1} = \{a_{j_1}^3, b_{j_1}^3\}$. \diamond

3.3 Proof of Theorem 2

Proof of Theorem 2. Let $\langle \mathcal{C}, w \rangle$ be any instance of MONOTONE CIRCUIT VALUE. Let $G = (V, E)$ be the graph obtained with our reduction from Section 3.1, which can be constructed in polylogarithmic-time with a polynomial number of processors. The graph G defines a coloring game. We fix any Nash equilibrium for this game, that exists by Theorem 1. By Claim 5, any two vertices in X have the same colour c_0 . We will prove that there is at least one vertex in Y_m with colour c_0 if and only if the circuit \mathcal{C} outputs 1 when it takes w as input. Since MONOTONE CIRCUIT VALUE is PTIME-complete [9], the latter will prove that computing a Nash equilibrium for coloring games is PTIME-hard.

By Claim 6, we only need to prove that for every $1 \leq j \leq m$, there is at least one vertex in $Y_j \cup N_j$ with colour c_0 . To prove it by contradiction, let j_0 be the smallest index j such that no vertex in $Y_j \cup N_j$ has colour c_0 . By Claim 5, every vertex in $Y_{j_0} \cup N_{j_0}$ has the same colour as her private group. In particular, the three of $a_{j_0}^1, a_{j_0}^2, a_{j_0}^3$ receive payoff $2(n + j_0) - 3$. We will prove that one of these three agents could increase her payoff by choosing c_0 as her new colour, thus contradicting that we are in a Nash equilibrium. Indeed, by the minimality of j_0 , it follows by Claim 4 that for any $1 \leq j \leq j_0 - 1$, there are exactly two vertices of $Y_j \cup N_j$ with colour c_0 , while for every $j_0 \leq j \leq m$ there is no vertex in $Y_j \cup N_j$ with colour c_0 . As a result, $|L_{c_0}| = 2(n + j_0) - 2$. In particular, any agent among $a_{j_0}^1, a_{j_0}^2, a_{j_0}^3$ could increase her payoff by choosing c_0 as her new colour — provided she is nonadjacent to every vertex in L_{c_0} . We will show it is the case for at least one of the three vertices, that will conclude the proof of the theorem. Assume w.l.o.g. that the j_0^{th} gate is an OR-gate (indeed, since by de Morgan's laws, the negation of an AND-gate can be transformed into an OR-gate and vice-versa, both cases are symmetrical). There are two cases.

Suppose that the output of the j_0^{th} gate is 1. In such case, there must be an entry of the gate such that: it is the i^{th} entry of the circuit, for some $1 \leq i \leq n$, and $w_i = 1$; or it is the k^{th} gate of the circuit for some $k < j_0$ and the output of that gate is 1. In the latter case, we have by Claim 6 that the two vertices of $Y_k \cup N_k$ with colour c_0 are in the set Y_k . Assume w.l.o.g. that the above-mentioned entry is the first entry of the gate. By construction, the two vertices $a_{j_0}^1, b_{j_0}^1$ are nonadjacent to every vertex in L_{c_0} . Else, the output of the j_0^{th} gate is 0. Therefore, for every entry of the gate: either it is the i^{th} entry of the circuit, for some $1 \leq i \leq n$, and $w_i = 0$; or it is the k^{th} gate of the circuit for some $k < j_0$ and the output of that gate is 0. In the latter case, we have by Claim 6 that the two vertices of $Y_k \cup N_k$ with colour c_0 are in the set N_k . By construction, the two vertices $a_{j_0}^3, b_{j_0}^3$ are nonadjacent to every vertex in L_{c_0} . In both cases, it contradicts that we are in a Nash equilibrium. \square

4 Conclusion and open perspectives

We suggest through this case study a more in-depth analysis of the complexity of computational mechanism designs. We would find it interesting to pursue similar investigations for other games. Experiments in the spirit of [11] could be helpful for our purposes. Further, we note that PRAM is seen by some as a too unrealistic model for parallel computation. Thus, one may argue that proving our reduction in this model casts a doubt on its reach. However, we can leverage on the stronger statement that MONOTONE CIRCUIT VALUE is *strictly* PTIME-hard [5]. It implies roughly that the sequential time and the parallel time to solve this problem cannot differ by more than a moderate polynomial-factor (unless the solving of *all* problems in PTIME can be sped up on a parallel machine by at least a polynomial-factor). Our reduction directly shows the same holds true for the problem of computing a Nash equilibrium in a given coloring game, that generalizes our hardness result to more recent parallel complexity classes (*e.g.*, [10]).

References

1. C. Ballester. NP-completeness in hedonic games. *Games and Economic Behavior*, 49(1):1–30, 2004.
2. J. A. Bondy and U. S. R. Murty. *Graph theory*. Grad. Texts in Math., 2008.
3. N. Burani and W. S. Zwicker. Coalition formation games with separable preferences. *Mathematical Social Sciences*, 45(1):27–52, 2003.
4. I. Chatzigiannakis, C. Koninidis, P. N. Panagopoulou, and P. G. Spirakis. Distributed game-theoretic vertex coloring. In *OPODIS'10*, pages 103–118.
5. A. Condon. A theory of strict P-completeness. *Computational Complexity*, 4(3):220–241, 1994.
6. G. Ducoffe, D. Mazauric, and A. Chaintreau. The complexity of hedonic coalitions under bounded cooperation. Submitted.
7. J. Feigenbaum, C. H. Papadimitriou, and S. Shenker. Sharing the cost of multicast transmissions. *J. Comput. Syst. Sci.*, 63(1):21–41, 2001.
8. M. Gairing and R. Savani. Computing stable outcomes in hedonic games. In *SAGT'10*, pages 174–185.
9. R. Greenlaw, H. J. Hoover, and W. L. Ruzzo. *Limits to parallel computation: P-completeness theory*. Oxford University Press, 1995.
10. H. Karloff, S. Suri, and S. Vassilvitskii. A model of computation for MapReduce. In *SODA'10*, pages 938–948.
11. M. Kearns, S. Suri, and N. Montfort. An experimental study of the coloring problem on human subject networks. *Science*, 313(5788):824–827, 2006.
12. J. Kleinberg and K. Ligett. Information-sharing in social networks. *Games and Economic Behavior*, 82:702–716, 2013.
13. J. R. Marden and A. Wierman. Distributed welfare games. *Operations Research*, 61(1):155–168, 2013.
14. P. N. Panagopoulou and P. G. Spirakis. A game theoretic approach for efficient graph coloring. In *ISAAC'08*, pages 183–195.
15. C. H. Papadimitriou. *Computational complexity*. John Wiley and Sons Ltd., 2003.
16. V. Vassilevska Williams. Fine-Grained Algorithms and Complexity (Invited Talk). In *STACS'16*.

Papers on Web's transparency

APPENDIX K

Xray: enhancing the Web's transparency with differential correlation

XRay: Enhancing the Web’s Transparency with Differential Correlation

Mathias Lécuyer, Guillaume Ducoffe, Francis Lan, Andrei Papancea, Theofilos Petsios,
Riley Spahn, Augustin Chaintreau, and Roxana Geambasu
Columbia University

Abstract

Today’s Web services – such as Google, Amazon, and Facebook – leverage user data for varied purposes, including personalizing recommendations, targeting advertisements, and adjusting prices. At present, users have little insight into how their data is being used. Hence, they cannot make informed choices about the services they choose.

To increase transparency, we developed *XRay*, the first fine-grained, robust, and scalable personal data tracking system for the Web. *XRay* predicts which data in an arbitrary Web account (such as emails, searches, or viewed products) is being used to target which outputs (such as ads, recommended products, or prices). *XRay*’s core functions are service agnostic and easy to instantiate for new services, and they can track data within and across services. To make predictions independent of the audited service, *XRay* relies on the following insight: by comparing outputs from different accounts with similar, but not identical, subsets of data, one can pinpoint targeting through correlation. We show both theoretically, and through experiments on Gmail, Amazon, and YouTube, that *XRay* achieves high precision and recall by correlating data from a surprisingly small number of extra accounts.

1 Introduction

We live in a “big data” world. Staggering amounts of personal data – our locations, search histories, emails, posts, and photos – are constantly collected and analyzed by Google, Amazon, Facebook, and a myriad of other Web services. This presents rich opportunities for marshaling big data to improve daily life and social well-being. For example, personal data improves the usability of applications by letting them predict and seamlessly adapt to future user needs and preferences. It improves business revenues by enabling effective product placement and targeted advertisements. Twitter data has been successfully applied to public health problems [36], crime prevention [44], and emergency response [22]. These beneficial uses have generated a big data frenzy, with Web services aggressively pursuing new ways to acquire and commercialize it.

Despite its innovative potential, the personal data frenzy has transformed the Web into an opaque and privacy-insensitive environment. Web services accumulate data, exploit it for varied and undisclosed purposes, retain it for extended periods of time, and possibly share it with others – *all without the data owner’s knowledge or consent*. Who has what data, and for what purposes is it used? Are the uses in the data owners’ best interests? Does the service adhere to its own privacy policy? How long is data used after its owner deletes it? Who shares data with whom?

At present, users lack answers to these questions, and investigators (such as FTC agents, journalists, or researchers) lack robust tools to track data in the ever-changing Web to provide the answers. Left unchecked, the exciting potential of big data threatens to become a breeding ground for data abuses, privacy vulnerabilities, and unfair or deceptive business practices. Examples of such practices have begun to surface. In a recent incident, Google was found to have used institutional emails from ad-free Google Apps for Education to target ads in users’ personal accounts [18, 37]. MySpace was found to have violated its privacy policy by leaking personally identifiable information to advertisers [25]. Several consumer sites, such as Orbitz and Staples, were found to have adjusted their product pricing based on user location [29, 43]. And Facebook’s 2010 ad targeting was shown to be vulnerable to micro-targeted ads specially crafted to reveal a user’s private profile data [23].

To increase transparency and provide checks and balances on data abuse, we argue that new, robust, and versatile tools are needed to effectively track the use of personal data on the Web. Tracking data in a *controlled environment*, such as a modified operating system, language, or runtime, is an old problem with a well-known solution: taint tracking systems [12, 16, 7, 48]. However, is it possible to track data in an *uncontrolled environment*, such as the Web? Can robust, generic mechanisms assist in doing so? What kinds of data uses are trackable and what are not? How would the mechanisms scale with the amount of data being tracked?

As a first step toward answering these questions, we built *XRay*, a personal data tracking system for the Web.

XRay correlates designated data *inputs* (be they emails, searches, or visited products) with data *outputs* results (such as ads, recommended products, or prices). Its correlation mechanism is service agnostic and easy to instantiate, and it can track data use within and across services. For example, it lets a data owners track how their emails, Google+, and YouTube activities are used to target ads in Gmail.

At its core, XRay relies on a *differential correlation* mechanism that pinpoints targeting by comparing outputs in different accounts with similar, but not identical, subsets of data inputs. To do so, it associates with every personal account a number of *shadow accounts*, each of which contains different data subsets. The correlation mechanism uses a simple Bayesian model to compute and rank scores for every data input that may have triggered a specific output. Intuitively, if an ad were seen in many accounts that share a certain email, and never in accounts that lack that email, then the email is likely to be responsible for a characteristic that triggers the ad. The email’s score for that ad would therefore be high. Conversely, if the ad were seen rarely in accounts with or lacking that email, that email’s score for this ad would be low.

Constructing a practical auditing system around differential correlation raises significant challenges. Chief among them is scalability with the number of data items. Theoretically, XRay requires a shadow account for each combination of data inputs to accurately pinpoint correlation. That would suggest an *exponential number of accounts*! Upon closer examination, however, we find that a few realistic assumptions and novel mechanisms let XRay reach high precision and recall with only a *logarithmic number of accounts in number of data inputs*. We deem this a major new result for the science of tracking data-targeting on the Web.

We built an XRay prototype and used it to correlate Gmail ads, Amazon product recommendations, and YouTube video suggestions to user emails, wish lists, and previously watched videos, respectively. While Amazon and YouTube provide detailed explanations of their targeting, Gmail does not, so we manually validated associations. For all cases, XRay achieved 80-90% precision and recall. Moreover, we integrated our Gmail and YouTube prototypes so we could track cross-service ad targeting. Although several prior measurement studies [10, 47, 21, 20, 31] used methodologies akin to differential correlation, we believe we are the first to build a generic, service agnostic, and scalable tool based on it. Overall, we make the following contributions:

1. The first general, versatile, and open system to track arbitrary personal Web data use by uncontrolled services. The code is available from our Web page <https://xray.cs.columbia.edu/>.

2. The first in-depth exploration into the scalability challenges of tracking personal data on the Web.
3. The design and implementation of robust mechanisms to address scaling, including data matching.
4. System instantiation to track data on three services (Gmail, Amazon, YouTube) and across services (YouTube to Gmail).
5. An evaluation of our system’s precision and recall on Gmail, Amazon, and YouTube. We show that XRay is accurate and scalable. Further, it reveals intriguing practices now in use by Web services and advertisers.

2 Motivation

This paper lays the algorithmic foundations for a new generation of scalable, robust, and versatile tools to lift the curtain on how personal data is being targeted. We underscore the need for such tools by describing potential usage scenarios inspired by real-life examples (§2.1). We do this not to point fingers at specific service providers; rather, we aim to show the many situations where transparency tools would be valuable for end-users and auditors alike. We conclude this section by briefly analyzing how current approaches fail to address these usage scenarios (§2.2).

2.1 Usage Scenarios

Scenario 1: Why This Ad? Ann often uses her Gmail ads to discover new retail offerings. Recently, she discussed her ad-clicking practices with her friend Tom, a computer security expert. Tom warned her about potential privacy implications of clicking on ads without knowing what data they target. For example, if she clicks on an ad targeting the keyword “gay” and then authenticates to purchase something from that vendor, she is unwittingly volunteering potentially sensitive information to the vendor. Tom tells Ann about two options to protect her privacy. She can either disable the ads altogether (using a system like Adblock [1]), or install the XRay Gmail plugin to uncover targeting against her data. Unwilling to give up the convenience of ads, Ann chooses the latter. XRay clearly annotates the ads in the Gmail UI with their target email or combination, if any. Ann now inspects this targeting before clicking on an ad and avoids clicking if highly sensitive emails are being targeted.

Scenario 2: They’re Targeting What? Bob, an FTC investigator, uses the XRay Gmail plugin for a different purpose: to study sensitive-data targeting practices by advertisers. He suspects a potentially unfair practice whereby companies use Google’s ad network to collect sensitive information about their customers. Therefore, Bob creates a number of emails containing keywords such as “cancer,” “AIDS,” “bankruptcy,” and “unemployment.” He refreshes the Gmail page many times, each time recording the targeted ads and XRay’s explanations

for them. The experiment reveals an interesting result: an online insurance company, TrustInUs.com, has targeted multiple ads against his illness-related emails. Bob hypothesizes that the company might use the data to set higher premiums for users reaching their site through a disease-targeted ad. He uses XRay results as initial evidence to open an investigation of TrustInUs.com.

Scenario 3: What’s With The New Policy?¹ Carla, an investigative journalist, has set up a watcher on privacy policies for major Web services. When a change occurs, the watcher notifies her of the difference. Recently, an important sentence in Google’s privacy policy has been scrapped:

If you are using Google Apps (free edition), email is scanned so we can display conceptually relevant advertising in some circumstances. Note that there is no ad-related scanning or processing in Google Apps for Education or Business with ads disabled.

To investigate scientifically whether this omission represents a shift in implemented policy, she obtains institutional accounts, connects them to personal accounts, and uses XRay to detect the correlation between emails in institutional accounts and ads in corresponding personal accounts. Finding a strong correlation, Carla writes an article to expose the policy change and its implications.

Scenario 4: Does Delete Mean Delete? Dan, a CS researcher, has seen the latest news that Snapchat, an ephemeral-image sharing Website, does not destroy users’ images after the requested timeout but instead just unlinks them [41]. He wonders whether the reasons for this are purely technical as the company has declared (e.g., flash wearing levels, undelete support, spam filtering) [39, 38] or whether these photos, or metadata drawn from them, are mined to target ads or other products on the Website. The answer will influence his decision about whether to continue using the service. Dan instantiates XRay to track the correlation between his expired Snapchat photos and ads.

2.2 Alternative Approaches

The preceding scenarios illustrate the importance of transparency in protecting privacy across a range of use cases. *We need robust, generic auditing tools to track the use of personal data at fine granularity (e.g., individual emails, photos) within and across arbitrary Web services.* At present, no such tools exist, and the science of tracking the use of personal Web data at a fine grain is largely non-existent.

¹In Feb. 2014, it was revealed based on court documents that Google could have used institutional emails to target ads in personal accounts [18]. In May 2014, Google committed to disable that feature [30]. Scenario 3 presents an XRay-based approach to investigate the original allegation.

Existing approaches can be broadly classified in two categories: *protection tools*, which prevent Web services’ acquisition or use of personal data, and (2) *auditing tools*, which uncover Web services’ acquisition or use of personal data. We discuss these approaches next; further related work is in §9.

Protection Tools. A variety of protection tools exist [11, 35, 1, 49]. For example, Ann could disable ads using an ad blocker [1]. Alternatively, she could encrypt her emails, particularly the sensitive ones, to prevent Google from using them to target ads. Dan could use a self-destructing data system, such as Vanish [14], to ensure the ephemerality of his Snapchat photos.

While we encourage the use of protection tools, they impose difficult tradeoffs that make them inapplicable in many cases. If Ann blocks all her ads, she cannot benefit from those she might find useful; if she encrypts all of her emails, she cannot search them; if she encrypts only her sensitive emails, she cannot protect any sensitive emails she neglected to encrypt in advance. Similarly, if Dan encrypts his Snapchat photos, sharing them becomes more difficult. While more sophisticated protection systems address certain limitations (e.g., searchable [5], homomorphic [15, 33], and attribute-based encryption [19], or privacy-preserving advertising [42, 13]), they are generally heavyweight [15], difficult to use [45], or require major service-side changes [15, 42, 13].

Auditing Tools. Given the limitations of protection tools, transparency is gaining increased attention [47, 12, 21]. If protecting data proves too cumbersome, limiting, or unresponsive of business needs, then users should at least be able to know: (1) *who is handling their data?*, and (2) *what is it being used for?*

Several tools developed in recent years partially address the first question by revealing where personal data flows from a local device [34, 12, 8]. TaintDroid [12] uses taint tracking to detect leakage of personal data from a mobile application to a service or third-party backend. ShareMeNot [34] and Mozilla’s Lightbeam Firefox add-on [27] identify third parties that are observing user activities across the Web. These systems track personal data – such as location, sensor data, Web searches, or visited sites – *until it leaves the user’s device.* Once the data is uploaded to Web services, it can be used or sold without a trace. In contrast, XRay’s tracking just begins: we aim to tell users how services use their data *once they have it.*

Several new tools and personalization measurement studies partially address the second question: what data is being used for [10, 47, 21, 20, 31]. In general, all existing tools are highly specialized, focusing on specific input types, outputs, or services. No general, principled foundation for data use auditing exists, that can be applied effectively to many services, a primary

motivation for this our work. For example, Bobble [47] reveals search result personalization based on user location (e.g., IP) and search history. Moreover, existing tools aim to discover only *whether* certain types of user inputs – such as search history, browsing history, IP, etc. – influence the output. None pinpoints at fine grain *which* specific input – which search query, which visited site, or which viewed product – or combination of inputs explain which output. XRay, whose goals we describe next, aims to do just that.

3 Goals and Models

Our overarching goal is to develop the core abstractions and mechanisms for tracking data within and across arbitrary Web sites. After describing specific goals (§3.1), we narrow our scope with a set of simplifying assumptions regarding the data uses that XRay is designed to audit (§3.2) and the threats it addresses (§3.3).

3.1 Goals

Three specific goals have guided XRay’s design:

Goal 1: Fine-Grained and Accurate Data Tracking. Detect which specific data inputs (e.g., emails) have likely triggered a particular output (e.g., an ad). While coarse-grained data use information (such as Gmail’s typical statement, “This ad is based on emails from your mailbox.”) may suffice at times, knowing the specifics can be revelatory, particularly when the input is highly sensitive and aggressively targeted.

Goal 2: Scalability. Make it practical to track significant amounts of data (e.g., past month’s emails). We aim to support the tracking of hundreds of inputs with reasonable costs in terms of shadow accounts. These accounts are generally scarce resource since their creation is being constrained by Web services. While we assume that users and auditors can obtain *some* accounts on the Web services they audit (e.g., a couple dozen), we strive to minimize the number required for accurate and fine-grained data tracking.

Goal 3: Extensibility, Generality, and Self-Tuning. Make XRay generic and easy to instantiate for many services and input/output types. Instantiating XRay to track data on new sites should be simple, although it may require some service-specific implementation of input/output monitoring. However, XRay’s correlation machinery – the conceptually challenging part of a scalable auditing tool – should be turn key and require no manual tuning.

3.2 Web Service Model

These goals may appear unsurmountable. An extremely heterogeneous environment, the Web has perhaps as many data uses as services. Moreover, data mining algorithms can be complex and proprietary. How can we abstract away this diversity and complexity to design robust and generic building blocks for scalable data

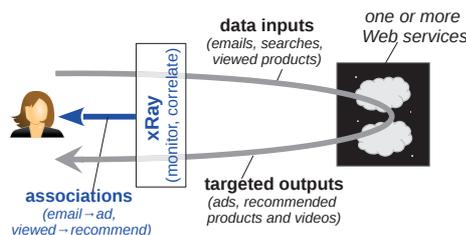


Figure 1: **XRay Conceptual View.** XRay views Web services as black boxes, monitors user *inputs* and *outputs* to/from them, and detects data use through correlation. It returns to the user or auditor *associations* of specific inputs and outputs.

tracking? Fortunately, we find that certain popular classes of Web data uses lend themselves to principled abstractions that facilitate scalable tracking.

Figure 1 shows XRay’s simplified view of Web services, and networks of services that exchange user data, are *black boxes* that receive personal data *inputs* from users – such as emails, pictures, search queries, locations, or purchases – and use them for varied purposes. Some uses materialize into *outputs* visible to users, such as ads, product or video recommendations, or prices. Others invisible to the users. XRay correlates some visible data inputs with some visible outputs by monitoring them, correlating them, and reporting strong *associations* to users. An example association is which email(s) contributed to the selection of a particular ad.

XRay relates only *strongly correlated* inputs with outputs. If an output is strongly correlated to an input (i.e., the input’s presence or absence changes the output), then XRay will likely be able to detect its use. If not (i.e., the monitored input plays but a small role in the output), then it may go undetected. XRay also relates small combinations of inputs with strongly correlated outputs.

Although simple, this model efficiently addresses several types of personal data functions, including product recommendations, price discriminations, and various personalization functions (e.g., search, news). We refer to such functions generically as *targeting functions* and focus XRay’s design on them.

Three popular forms of targeting are:

1. *Profile Targeting*, which leverages static or slowly evolving explicit information – such as age, gender, race, or location – that the user often supplies by filling a form. This type of targeting has been studied profusely [10, 47, 21, 20, 31]; we thus ignore it here.
2. *Contextual Targeting*, which leverages the content currently being displayed. In Gmail, this is the currently open email next to which the ad is shown. In Amazon or Youtube, the target is the product or video next to which the recommendation is shown.
3. *Behavioral Targeting*, which leverages a user’s past actions. An email sent or received today can trigger an ad tomorrow; a video watched now can trig-

ger a recommendation later. Use of histories makes it harder for users to track which data is being used, a key motivation for our development of XRay.

Theoretically, our differential correlation algorithms could be applied to all three forms of targeting. From a systems perspective, XRay’s design is geared towards *contextual targeting* and a *specific form of behavioral targeting*. The latter requires further attention. We observe that this broad targeting class subsumes multiple types of targeting that operate at different granularities. For example, a service could use as inputs a user’s most recent few emails to decide targeting. This would be similar to an extended context. Alternatively, a service could use historical input to learn a user’s coarse interests or characteristics and base its targeting on that.

XRay currently aims to disclose any targeting applied at the level of individual user data, or small combinations thereof. Our differential correlation algorithms could be applied to detect targeting that operates on a coarser granularity. However, the XRay system itself would require significant changes. Unless otherwise noted, we use *behavioral targeting* to denote the restricted form of behavioral targeting that XRay is designed to address. We formalize these restrictions in §4.2.

3.3 Threat Model

To further narrow our problem’s scope, we introduce threat assumptions. We assume that data owners (users and auditors) are trusted and do not attempt to leverage XRay to harm Web services or the Web ecosystem. While they trust Web services with their data, they wish to better understand how that data is being used. Data owners are thus assumed to upload the data in clear text to the Web services.

The threat models relevant for Web services depend on the use case. For example, Scenarios 1 and 2 in §2.1 assume Google is trusted, but its users wish to understand more about how advertisers target them through its ad platform. In contrast, in Scenarios 3 and 4, investigators may have reason to believe that Web services might intentionally frustrate auditing.

This paper assumes an *honest-but-curious* model for Web services: they try to use private data for financial or functional gains, but they do not try to frustrate our auditing mechanism, e.g., by identifying and disabling shadow accounts. The service might attempt to defend itself against more general types of attacks, such as spammers or DDoS attacks. For example, many Web services constrain the creation of accounts so as to limit spamming and false clicks. Similarly, Web services may rate limit or block the IPs of aggressive data collectors. XRay must be robust to such inherent defenses. We discuss challenges and potential approaches for stronger adversarial models in §7.

4 The XRay Architecture

XRay’s design addresses the preceding goals and assumptions. For concreteness, we draw examples from our three XRay instantiations: tracking email-to-ad targeting association within Gmail, attributing recommended videos to those already seen on YouTube, and identifying products in a wish list that generate a recommendation on Amazon.

4.1 Architectural Overview

XRay’s high-level architecture (Figure 2) consists of three components: (1) a *Browser Plugin*, which intercepts tracked inputs and outputs to/from an audited Web service and gives users visual feedback about any input/output associations, (2) a *Shadow Account Manager*, which populates shadow accounts with inputs from the plugin and collects outputs (e.g., ads) for each shadow account, and (3) the *Correlation Engine*, XRay’s core, which infers associations and provides them to the plugin for visualization. While the Browser Plugin and Shadow Account Manager are *service specific*, the Correlation Engine, which encapsulates the science of Web-data tracking, is *service agnostic*. After we describe each component, we focus on the design of the Correlation Engine.

Browser Plugin. The Browser Plugin intercepts designated inputs and outputs (i.e., *tracked inputs/outputs*) by recognizing specific DOM elements in an audited service’s Web pages. Other inputs and outputs may not be tracked by XRay (i.e., *untracked inputs/outputs*). The decision of what to track belongs to an investigator or developer who instantiates XRay to work on a specific service. For example, we configure the XRay Gmail Plugin to monitor a user’s emails as inputs and ads as outputs. When the Plugin gets a new tracked input (e.g., a new email), it forwards it both to the service and to the Shadow Account Manager. When the Plugin gets a new tracked output (e.g., an ad), it queries the Correlation Engine for associations with the user’s tracked inputs (message `get_assoc`).

Shadow Account Manager. This component: (1) populates the shadow accounts with subsets of a user account’s tracked inputs (denoted D_i), and (2) periodically retrieves outputs (denoted O_k) from the audited service for each shadow account. Both functions are service specific. For Gmail, they send emails with SMTP and call the ad API. For YouTube, they stream a video and scrape recommendations, and for Amazon, they place products in wish lists and scrape recommendations. The complexity of these tasks depends on the availability of APIs or the stability of a service’s page formats. Outputs collected from the Web service are placed into a *Correlation Database* (DB), which maps shadow accounts to their input sets and output observations. Figure 2 shows a par-

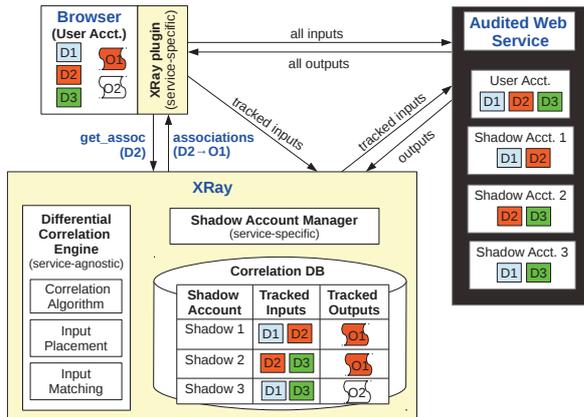


Figure 2: The XRay Architecture.

ticular assignment of tracked inputs across three shadow accounts. For example, Shadow 1 has inputs D_1 and D_2 . The figure also shows the outputs collected for each shadow account. Output O_1 appears in Shadows 1 and 2 but not in 3; output O_2 appears in Shadow 3 only.

Differential Correlation Engine. This engine, XRay’s service-agnostic “brain,” leverages the data collected in the Correlation DB to infer input/output associations. When new outputs from shadow accounts are added into the Correlation DB, the engine attempts to diagnose them using a *Correlation Algorithm*. We developed several such algorithms and describe them in §4.3. This process, potentially time-consuming process, is done as a background job, asynchronously from any user request. In Figure 2, differential correlation might conclude that D_2 triggers O_1 because O_1 appears consistently in accounts with that D_2 . It might also conclude that O_2 is *untargeted* given inconsistent observations. The engine saves these associations in the Correlation DB.

When the plugin makes a `get_assoc` request, the Correlation Engine looks up the specified output in its DB and returns any pre-computed association. If no output is found, then the engine replies *unknown* (e.g., if an ad never appeared in any shadow account or there is insufficient information). Periodic data collection, coupled with an online update of correlation model parameters, minimizes the number of unknown associations. Our experience shows that collecting shadow account outputs in Gmail every ten hours or so yielded few unknown ads.

While the preceding example is simple, XRay can handle complex challenges occurring in practice. First, outputs are never consistently seen across all shadow accounts containing the input they target. We call this the *limited-coverage* problem; XRay handles it by placing each data input in more shadow accounts. Second, an output may have been triggered by one of several targeted inputs (e.g., multiple emails on the same topic may cause related ads to appear), a problem we

refer to as *overlapping-inputs*. This exacerbates the number of accounts needed, since it diminishes the differential signal we receive from them. XRay uses robust, service-agnostic mechanisms and algorithms to match overlapping inputs, place them in the same accounts, and detects their use as a group.

Organization. The remainder of this section describes the Differential Correlation Engine. After constructing it for Gmail, we applied it as-is for Amazon and YouTube, where it achieved equally high accuracy and scalability despite observable differences in how targeting works on these three services. After establishing notations and formalizing our assumptions (§4.2), we describe multiple correlation algorithms, which build up to our self-tuning correlation algorithm that made this adaptation convenient (§4.3). §4.4 describes our input matching.

4.2 Notation and Assumptions

We use f to denote the black-box function that represents the service (e.g., Gmail) associating inputs D_i s (e.g., the emails received and sent) to targeted outputs O_k s (e.g., ads). Other inputs are either ignored by XRay, known only to the targeting system, or under no known control. We assume they are independent or fixed, captured in the randomness of f .

We assume that f decides targeting using: (1) a single input (e.g., show O_k if D_4 is in the account), (2) a conjunctive combination of inputs (e.g., show O_k if D_5 and D_8 are in the account), or (3) a disjunctive combination of the previous (e.g., show O_k if (D_5 and D_8) are in the account *or* if D_4 is in the account). We refer to conjunctive and disjunctive combinations as AND and OR combinations, respectively, and assume that their is bounded by a maximum *input size*, r . This corresponds to the preceding definition of behavioral targeting from §3.2. Contextual targeting will always be a single-input (size-one) combination.

Our goal is to decide whether f produced each output O_k as a reaction to a bounded-size combination of the D_i s. We define as *untargeted* any ad that is not targeted against any combination of D_i s, though in reality the ad could be targeted against untracked inputs. We denote untargeting as D_0 , meaning that the ad is targeted against the “void” email. Our algorithms compute the most likely combination from the N inputs that explains a particular set of observations, \vec{x} , obtained by XRay.

We define three probabilities upon which our algorithms and analyses depend. First, the *coverage*, p_{in} , is the probability that an account j containing the input D_i targeted by a particular ad, will see that ad at least once. Second, an account j' lacking input D_i will see the ad with a smaller probability, p_{out} . Third, if the ad is not behaviorally targeted, it will appear in each account with the same probability, p_0 . We assume that p_{in}, p_0, p_{out} are

constant across all emails, ads, and time, and that p_{out} is strictly smaller than p_{in} (bounded noise hypothesis).

Finally, we consider all outputs to be independent of each other across time. §8 discusses the implications.

4.3 Correlation Algorithms

A core contribution of this paper is our service-agnostic, self-tuning differential correlation algorithm, which requires only a logarithmic number of shadow accounts to achieve high accuracy. We wished not only to validate this result experimentally, but also to prove it theoretically in the context of our assumptions. This section constructs the algorithm in steps, starting with a naïve polynomial algorithm that illustrates the scaling challenges. We then define a base algorithm using set intersections and prove that it has the desired logarithmic scaling properties; it has parameters which, if not carefully chosen, can lead to poor results. We therefore extend this base algorithm into a self-tuning Bayesian model that automatically adjusts its parameters to maximize correctness.

4.3.1 Naïve Non-Logarithmic Algorithm

An intuitive approach to differential correlation is to create accounts for every combination of inputs, gathering maximum information about their behaviors. With a sufficient number of observations, one could expect to detect which accounts, and hence which subsets of inputs, target a particular ad. Unfortunately, this method requires a number of accounts that grows *exponentially* as the number of items N to track grows. When restricting the size of combinations to r , as we do in XRay, the number of accounts needed is *polynomial* (in $O(N^r)$), or *linear* if we study unique inputs only. Even a linear number of accounts in the number N of inputs remains impractical to scale to large input sizes (e.g., a mailbox).

4.3.2 Threshold Set Intersection

We now show that it is possible to infer behavioral targeting using no more than a *logarithmic* number of accounts as a function of the number of inputs. Specifically, we prove the following theorem:

Theorem 1 *Under §4.2 assumptions, for any $\epsilon > 0$ there exists an algorithm that requires $C \times \ln(N)$ accounts to correctly identify the inputs of a targeted ad with probability $(1 - \epsilon)$. The constant C depends on ϵ and the maximum size of combinations r ($O(r2^r \log(\frac{1}{\epsilon}))$).*

To demonstrate the theorem, we define the *Set Intersection Algorithm* and prove that it has the correctness and scaling properties specified in the theorem. Given that outputs will appear more often in accounts containing the targeting inputs, the core of the algorithm is to determine the set of inputs appearing in the highest number of accounts that also see a given ad. This paper describes a basic version of the algorithm that makes

```

// Set Intersection Algo:
// Runs with each collected ad.
In: Output  $O_k$  (e.g. an ad).
Params: MIN_ACTIVE_ACCTS, THRESHOLD.
Out: Targeted input combination.
// Step 1: Compute active accounts.
 $A_k$  = the accounts that see ad  $O_k$ .
if  $|A_k| < \text{MIN\_ACTIVE\_ACCTS}$ 
    return  $\emptyset$ 
end
// Step 2: Create input combination hypothesis.
targeted_set =  $\emptyset$ 
foreach input  $D_i$  do
    if  $\frac{\text{number of } A_k \text{ containing } D_i}{|A_k|} > \text{THRESHOLD}$ 
        targeted_set +=  $D_i$ 
    end
end
// Step 3: Verify it is a real combination.
if  $\frac{\text{number of } A_k \text{ containing entire targeted\_set}}{|A_k|} < \text{THRESHOLD}$ 
    return  $\emptyset$ 
end
// targeted_set triggered the output.
return targeted_set

```

Figure 3: **The Set Intersection Algorithm.** Can be proven to predict targeting correctly under certain assumptions with a logarithmic number of accounts.

some simplifying assumptions and provides a brief proof sketch. The detailed proof and complete algorithm are described in our technical report [26].

Algorithm. The algorithm relies on a randomized placement of inputs into shadow accounts, with some redundancy to cope with imperfect coverage. We thus pick a probability, $0 < \alpha < 1$, create $C \ln(N)$ shadow accounts, and place each input D_i randomly into each account with probability α . Figure 3 shows the Set Intersection algorithm for a set of observations, \vec{x} . Given an output O_k collected from the user account, we compute the set of *active accounts*, A_k , as those shadow accounts that have seen the output (Step 1). We then compute the set of inputs that appear in at least a threshold fraction of active accounts; this set is our candidate for the combination being targeted by the ad (Step 2). Finally, we check that the entire combination is in a threshold fraction of the active accounts (Step 3). Theoretically, we prove that there exists a threshold for which the algorithm is arbitrarily correct with the available $C \ln(N)$ accounts. Practically, this threshold must be tuned experimentally to achieve good accuracy on every service – a key reason for our Bayesian enhancement in §4.3.3.

Correctness Proof Sketch. The proof shows that if there were targeting, every non-targeting input would have a vanishingly small probability to be in a significant fraction of the active accounts. Let us call S the set of inputs

<pre> // Bayesian Prediction Alg: // Runs with each collected ad. In: Output O_k (e.g. an ad). Out: Targeted input. // Compute probabilities. foreach input D_i do $\mathbb{P}[D_i \vec{x}] = \text{bayes}(\mathbb{P}[\vec{x} D_i])$ end // Compute untargeted prob. $\mathbb{P}[D_\emptyset \vec{x}] = \text{bayes}(\mathbb{P}[\vec{x} D_\emptyset])$ // Return event with max prob. return D_i with max $\mathbb{P}[D_i \vec{x}]$ </pre>	<pre> // Parameter Learning Alg: // Runs periodically. // Initialize params (arbitrary) $p_{in} = .7, p_{out} = .01, p_\emptyset = .1$ do foreach output O_k do Run Bayesian Prediction. end Update $p_{in}, p_{out}, p_\emptyset$ from predictions. until $p_{in}, p_{out}, p_\emptyset$ converge end </pre>
---	--

Figure 4: **Bayesian Correlation.** Left: Bayesian prediction algorithm for behavioral targeting. Right: typical iterative inference process to learn parameters.

contained in a significant fraction of the active accounts. Without targeting, these inputs would be present in the accounts by mere chance. Since inputs are independently distributed into the accounts, we show that the probability of S not being empty decreases exponentially with the number of active accounts (through Chernoff bounds). With targeting, we show that with high probability no other input than the explaining combination is in S , because of the bounded noise hypothesis. Appendix A.2 provides further proof details.

The proofs and algorithm included in this paper work only for conjunctive combinations (e.g., D_1 and D_2 , see §4.2). The theory, however, can be extended to disjunctive combinations (e.g., (D_1 and D_2) or D_5), but the algorithm for detecting such combinations is more complex and relies on a recursive argument: if we find one combination from the disjunction, then the active accounts that include this combination define a context where the combination appears non-targeting because it is everywhere. If we recursively apply our algorithm in this context, we can detect the second combination in the disjunction, then the third, etc (see technical report [26]).

4.3.3 Self-Tuning Bayesian Algorithm

The Set Intersection algorithm provides a good theoretical foundation; however, it requires parameters be tuned and applies only to behavioral targeting, not contextual targeting. Thus, we include in XRay a more robust, self-tuning version that leverages a Bayesian algorithm to adjust parameters automatically through iterated inference. Our algorithm relies on three models: one that predicts behavioral targeting, one that predicts contextual targeting, and one that combines the two.

Behavioral Targeting. The Bayesian behavioral targeting model uses the same random assignment as the Set Intersection algorithm, and it leverages the same information from the shadow account observations, \vec{x} . It counts the observations x_j of ad O_k in an account j as a binary signal: if the ad has appeared at least once in

account j , we count it once; otherwise we do not count it. Briefly, the Bayesian model is a simple generative model that simulates the audited service given some targeting associations (e.g., D_i triggers O_k). It computes the probability for this model to generate the outputs we do observe for every targeting association. The most likely association will be the one XRay returns.

In more detail if the ad were targeted towards D_i , then an account j containing D_i would see this ad at least once with a *coverage* probability p_{in} ; otherwise, it would miss it with probability $(1 - p_{in})$. An account j' without input D_i would see the ad with a smaller probability, p_{out} , missing it with probability $(1 - p_{out})$. If the ad were not behaviorally targeted, it would appear in each account with the same probability, p_\emptyset . If we define A_k as the set of active accounts that have seen the ad, and \bar{A}_i as the set of accounts that contain email D_i , then we have the following definitions for the probabilities:

$$\begin{aligned} \mathbb{P}[\vec{x} | D_i] &= (p_{in})^{|\bar{A}_i \cap A_k|} (1 - p_{in})^{|\bar{A}_i \cap \bar{A}_k|} \\ &\quad \times (p_{out})^{|\bar{A}_i \cap A_k|} (1 - p_{out})^{|\bar{A}_i \cap \bar{A}_k|}, \\ \mathbb{P}[\vec{x} | D_\emptyset] &= (p_\emptyset)^{|A_k|} (1 - p_\emptyset)^{|\bar{A}_k|}, \end{aligned}$$

where D_\emptyset designates the untargeted prediction.

The preceding formula has an interesting interpretation that is visible if placed in the equivalent form:

$$\begin{aligned} \mathbb{P}[\vec{x} | D_i] &= (p_{in})^{|\bar{A}_k|} (1 - p_{out})^{|\bar{A}_k|} \\ &\quad \times \left(\frac{1 - p_{in}}{1 - p_{out}} \right)^{|\bar{A}_i \cap \bar{A}_k|} \left(\frac{p_{out}}{p_{in}} \right)^{|\bar{A}_i \cap A_k|} \end{aligned}$$

From the point of view of the event D_i , an account found in $\bar{A}_i \cap \bar{A}_k$ is a false positive (an ad was expected but was not shown). This should lower the probability, especially when the *coverage* p_{in} is close to 1. Inversely, an account found in $\bar{A}_i \cap A_k$ acts as a false negative (we observed an ad where we did not expect it), which should decrease the probability, especially when p_{out} is close to 0.

These formulas let us infer the likelihood of event D_i according to Bayes' rule: $\mathbb{P}[A | B] = \frac{\mathbb{P}[B | A] \times \mathbb{P}[A]}{\mathbb{P}[B]}$. Figure 4 shows two algorithms. First, the prediction algorithm (left) predicts the targeting of O_k by computing the probabilities defined above, applying Bayes' rule, and returning the input with the maximum probability. Second, the parameter learning algorithm (right) computes the variables that those probabilities depend upon (p_{in} , p_{out} , and p_\emptyset) using an iterative process. It repeatedly runs the prediction algorithm for all outputs and re-computes p_{in} , p_{out} , and p_\emptyset based on the predictions. It stops when the variables converge (i.e., their variation from one iteration to another is small).

Contextual Targeting. Contextual targeting is more straightforward since it uses content shown next to the ad. XRay also uses Bayesian inference and defines the observations as how many times ad O_k is seen next to

email D_i . Our causal model assumes imperfect coverage: if this ad were contextually targeted towards D_i , it would occur next to that email with probability $p_{\text{in}} < 1$ and next to any other email with probability p_{out} . Alternatively, if the ad were untargeted, our model predicts it would be shown next to any email with probability p_\emptyset . Hence, $\mathbb{P}[\bar{x}|D_i] = (p_{\text{in}})^{x_i} (p_{\text{out}})^{\sum_{i' \neq i} x_{i'}}$, $\mathbb{P}[\bar{x}|D_\emptyset] = (p_\emptyset)^{\sum_i x_i}$. For this model, parameters are also automatically computed by iterated inference.

Composite Model (XRay). The contextual and behavioral mechanisms were designed to detect different types of targeting. To detect both types, XRay must combine the two scores. We experimented with multiple combination functions, including a decision tree and the arithmetic average, and concluded that the arithmetic average yields sufficiently good results. XRay thus defines the *composite model* that averages scores from individual models, and we demonstrate in §6.3 that doing so yields higher recall for no loss in precision.

4.4 Input Matching and Placement

Our design of differential correlation, along with our logarithmic results for random input placement, relies on the fundamental assumption that the probability of getting an ad O_1 targeted at an input D_1 in a shadow account that lacks D_1 is vanishingly small. However, when inputs attract the same ads (a.k.a., overlapping inputs), a naive input placement can contradict this assumption. Imagine a Gmail account with multiple emails related to a Caribbean trip. If placement includes Caribbean emails in every available shadow account, related ads will appear in groups of accounts with no email object in common. XRay will thus classify them as untargeted. Our Amazon experiments showed XRay’s recall dropping from 97% to 30% with overlapping inputs (§6.5).

To address this problem, XRay’s Input Matching module identifies similar inputs and directs the Placement Module to co-locate them in the same shadow accounts. The key challenge is to identify similar inputs. One method is to use content analysis (e.g., keywords matching), but this has limitations. First, it is not service agnostic; one needs to reverse engineer complex and ever-changing matching schemes. Second, it is hard to apply to non-textual media, such as YouTube videos.

In XRay, we opt for a more robust, systems technique rooted in the key insight that we can deduce similar inputs from contextual targeting. Intuitively, inputs that trigger similar targeting from the Web service should attract similar outputs in their context. The Input Matching module builds and compares inputs’ *contextual signatures*. Contextual signature similarity is the distance between inputs (e.g., email) in a Euclidean space, where each output (e.g., ad) is a dimension. The coordinate of an email in this dimension is the number of

times the ad was seen in the context of the email. XRay then forwards close inputs to the same shadow accounts. Once the placement is done, behavioral targeting against that email’s group can be inferred effectively.

This input matching mechanism differs fundamentally from any content analysis technique, such as keyword matching, because it groups inputs *the same way the Web service does*.² It is robust and very general: we used it on both Gmail and Amazon without changing a single line of code to change.

5 XRay-based Tools

To evaluate XRay’s extensibility, we instantiated it on Gmail, YouTube, and Amazon. The engine, about 3,000 lines of Ruby, was first developed for Gmail. We then extended it to YouTube and Amazon, without any changes to its correlation algorithms. We did need to do minor code re-structuring, but the experience felt turn key when integrating a new service into the correlation machinery.

Building the full toolset required non-trivial coding effort, however. Instantiating XRay for a specific Web service is a three-step process. First, the developer instantiates appropriate data models (less than 20 code lines for our prototypes). Second, she implements a service-specific shadow account manager and plugin; care must be taken not to be too aggressive to avoid adversarial service reactions. While these implementations are conceptually simple, they require some coding; our Amazon and YouTube account managers were built by two graduate students new to the project, and have around 500 lines of code. Third, the developer creates a few shadow accounts for the audited service and runs a small exploratory experiment to determine the service’s coverage. XRay uses the coverage to estimate the number of shadow accounts needed for a given input size. All other parameters are self-tuned at runtime.

6 Evaluation

We evaluated XRay with experiments on Gmail, Amazon, and YouTube. While Amazon and YouTube provide ground truth for their targeting, Gmail does not. We therefore manually labeled ads on Gmail and measured XRay’s accuracy, as described in §6.1 and validated in §6.2. We sought answers to four questions:

- Q1 *How accurate are XRay’s inference models?* (§6.3)
- Q2 *How does XRay scale with input size?* (§6.4)
- Q3 *Can input matching manage overlap?* (§6.5)
- Q4 *How useful is XRay in practice?* (§6.6)

²We call this method “monkey see, monkey do” because we watch how the service groups inputs and group them similarly.

Ad Keyword	Targeted Email	Detected by XRay?	XRay Scores	# Accounts & Displays
Chaldean Poetry	Like Chaldean Poetry?	Yes	0.99, 1.0	13/13, 1588/1622
Steampunk	Fan of Steampunk?	Yes	0.99, 1.0	13/13, 888/912
Cosplay	Discover Cosplay.	Yes	0.99, 1.0	13/13, 440/442
Falconry	Learn about Falconry.	Yes	0.99, 1.0	13/13, 1569/1608

Figure 5: **Self-Targeted Ads.** Fourth column shows XRay’s correlation scores X, Y, (Bayesian) Behavioral and Contextual scores, respectively. Fifth column shows raw behavioral and contextual data for interpretation: X/Y, Z/T means that the ad was seen in X active accounts that contain the targeted email out of a total of Y active accounts; the ad was shown Z times in the context of the targeted email out of a total of T times.

6.1 Methodology

We evaluated XRay with experiments on Gmail, Amazon, and YouTube. For inputs, we created a workload for each service by selecting topics from well-defined categories relevant for that service. For Gmail and YouTube, we crafted emails and selected videos based on AdSense categories [17]; for Amazon, we selected products from its own product categories [2]. We used these categories for most of our experiments (§6.3–§6.5). We used these categories to create two types of workloads: (1) a non-overlapping workload, in which each data item belonged to a distinct category, and (2) an overlapping workload, with multiple data items per category (described in §6.5).

To assess XRay’s accuracy, we needed the ground truth for associations. Amazon and YouTube provide it for their recommendations. For instance, Amazon provides a link “Why recommended?” which explicitly explains the recommendation. For Gmail, we manually labeled ads based on our personal assessment. The ads for different experiments were labeled by different people, generally project members. A non-computer scientist labeled the largest experiment (51 emails).

We evaluate two metrics: (1) *recall*, the fraction of positive associations labeled as such, and (2) *precision*, the fraction of correct associations. We define *high accuracy* as having both high recall and high precision.

6.2 Sanity-Check Experiment

To build intuition into XRay’s functioning, we ran a simple sanity-check experiment on Gmail. Recall that, unlike Amazon and YouTube, Gmail does not provide any ground truth, requiring us to manually label associations, a process that can be itself faulty. Before measuring XRay’s accuracy against labeled associations, we checked that XRay can detect associations for our own ads, whose targeting we control. For this, we strayed away from the aforementioned methodology to create a highly controlled experiment. We posted four Google AdWords campaigns targeted on very specific keywords (Chaldean Poetry, Steampunk, Cosplay, and

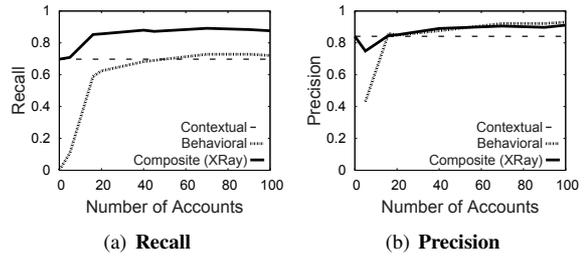


Figure 6: **Bayesian Model Accuracy.** Recall and precision for each of the three Bayesian models vs. shadow account number, using the Bayesian algorithm. XRay needed 16 accounts to reach the “knee” with high recall and precision.

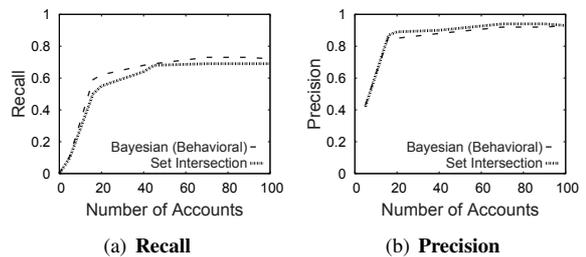


Figure 7: **Bayesian vs. Set Intersection Comparison.** Recall and precision for detecting *behavioral* targeting with each algo.

Falconry), crafted an inbox that included one email per keyword, and used XRay to recover the associations between our ads and those emails. In total, we saw our ads 1622, 912, 442, and 1608 times, respectively, across all accounts (shadows and master). Figure 5 shows our results. After one round of ad collection (which involved 50 refreshes per email), XRay correctly associated all four ads with the targeted email. It did so with very high confidence: composite model scores were 0.99 in all cases, with very high scores for both contextual and behavioral models. The figure also shows some of the raw contextual/behavioral data, which provides intuition into XRay’s perfect precision and recall in this controlled experiment. We next turn to evaluating XRay in less controlled environments, for which we use the workloads and labeling methodology described in §6.1.

6.3 Accuracy of XRay’s Inference Models (Q1)

To assess the accuracy of XRay’s key correlation mechanisms (Bayesian behavioral, contextual, and composite), we measured their recall and precision under non-overlapping workloads. Figures 6(a) and 6(b) show how these two metrics varied with the number of shadow accounts for a 20-email experiment on Gmail. The results indicate two effects. First, both contextual and behavioral models were required for high recall. Of the 193 distinct ads seen in the user account, 121 (62%) were targeted, and XRay found 109 (90%) of them, a recall we deem high. Of the associations XRay found, 37% were found by only one of the models: 15

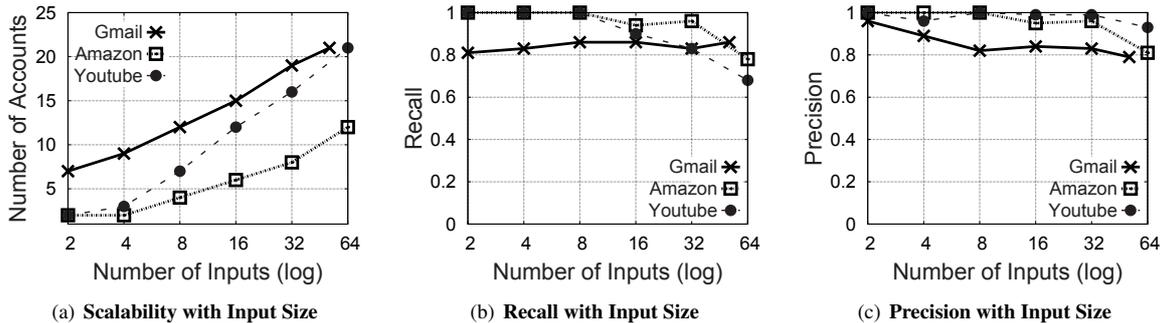


Figure 8: **Scalability.** (a) Number of accounts required to achieve the knee accuracy for varied numbers of inputs. (b), (c) Recall/precision achievable with the number of accounts in (a). Behavioral uses the Bayesian algorithm.

by the contextual model only, and 24 by the behavioral model only. Thus, both models were necessary, and composing them yielded high recall. Our Amazon and YouTube experiments (which provide ground truth) yielded very similar results: on a 20-input experiment, we reached over 90% recall and precision with only 8 and 12 accounts, respectively.

Second, the composite model’s recall exhibited a knee-shaped curve for increasing shadow account numbers, with a rapid improvement at the beginning and slow growth thereafter. With 16 accounts, XRay exceeded 85% recall; increasing the number of accounts to 100 yielded a 1.9% improvement. Precision also remained high (over 84%) past 16 accounts. We define the *knee* as the minimum number of accounts needed to reap most of the achievable recall and precision.

We also wished to compare the accuracy of the Bayesian algorithm, which conveniently self-tunes its parameters, to the parameterized Set Intersection algorithm. We manually tuned the latter as best as we could. Figures 7(a) and 7(b) show the recall and precision for detecting behavioral targeting with the two methods for a non-overlapping workload. The two algorithms performed similarly, with the Bayesian staying within 5% of the manually tuned algorithm. We also tested the algorithms on an Amazon dataset, and using a version of the Set Intersection algorithm with empirical optimizations. The conclusion holds: the Bayesian algorithm, with self-tuned parameters, performs as well as the Set Intersection technique with manually tuned parameters. We focus the remainder of this evaluation on the Bayesian algorithm.

6.4 Scalability of XRay with Input Size (Q2)

A main contribution of this paper is the realization that, under certain assumptions, the number of accounts needed to achieve high accuracy for XRay scales logarithmically with the number of tracked inputs. We have proven that under certain assumptions, the Set Intersection algorithm scales logarithmically. This

theoretical result is hard to extend to the Bayesian algorithm, so we evaluated it experimentally by studying three metrics with growing input size: the number of accounts required to reach the recall knee and the value of recall/precision at this knee. Figures 8(a), 8(b) and 8(c) show the corresponding results for Gmail, YouTube and Amazon. For Gmail, the number of accounts necessary to reach the knee increased less than 3-fold (from 8 to 21) as input size increased more than 25-fold (from 2 to 51). For Amazon and YouTube, the increases in accounts were 6- and 8-fold respectively, for a 32-fold increase in input size. In general, the roughly linear shapes of the log-x-scale graphs in Figure 8(a) confirm the logarithmic increase in the number of accounts required to handle different inputs. Figure 8(b) and 8(c) confirm that the “knee number” of accounts achieved high recall and precision (over 80%).

What accounts for the large gap between the number of accounts needed for high accuracy in Gmail versus Amazon? For example, tracking a mere two emails in Gmail required 8 accounts, while tracking two viewed products in Amazon needed 2 accounts. The distinction corresponds to the difference in coverage exhibited by the two services. In Gmail, a targeted ad was typically seen in a smaller fraction of the relevant accounts compared to a recommended product in Amazon. XRay adapted its parameters to lower coverage automatically, but it needed more accounts to do so.

Overall, these results confirm that our theoretical scalability results hold for real-world systems given carefully crafted, non-overlapping input workloads. We next investigate how more realistic overlapping input workloads challenge the accuracy of our theoretical models and how input matching – a purely systems technique – helps address this challenge.

6.5 Input Matching Effectiveness (Q3)

To evaluate XRay’s accuracy with overlapping inputs, we infused our workloads with multiple items from the same category. (e.g., multiple emails targeting the

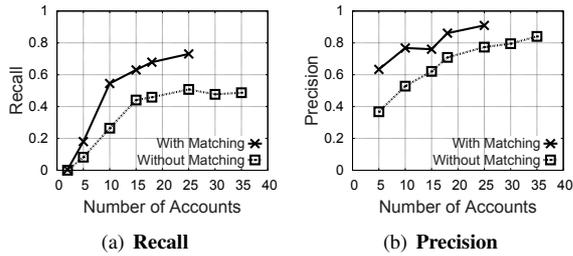


Figure 9: **Input Matching effectiveness.** Behavioral (Bayesian) recall and precision in Gmail with overlapping inputs, with and without Matching.

same topic on Gmail and multiple products in the same category in Amazon). For the Gmail experiments, we (as users) could not tell when Gmail targeted a specific email from a group of similar emails. We therefore ran two different types of experiments. First, a controlled, albeit unrealistic, one for Gmail. We replicated various emails *identically* in a user’s inbox: 1 email was replicated 4 times, 2 emails 3 times, 4 emails 2 times, and 12 were single, for a total of 30 emails. This end-of-a-spectrum workload demonstrates how matching works ideally. XRay matched *all* redundant emails correctly. More importantly, Figures 9(a) and 9(b) show XRay’s precision/recall with and without matching-aware placement for XRay’s behavioral model, the only model improved by matching. Without input matching, XRay struggled to find differential signals: even with 35 shadow accounts for a 30-email experiment, recall was only 48%. With input matching, XRay’s correlation model drew a stronger signal from each account and attained close to 70% recall for 16 accounts.

Second, for Amazon, we created a more realistic overlapping workload by selecting three *distinct* products in each of six product categories (e.g., from the Outdoor & Cycling category, we selected a helmet, pedals, and shoes). With a total workload of 18 products, XRay’s input matching matched all but one item (shoes) into its correct group. With the new grouping, XRay’s recall improved by a factor of 3 (from 30% to 93%) compared to the no-matching case for 18 products with 10 accounts; precision was 2.6 times higher (from 34% to 88%).

These results demonstrate that XRay’s matching scheme is both portable across Web services and essential for high accuracy with overlapping workloads.

6.6 Anecdotal Use Experience (Q4)

To gain intuition into XRay’s practical value, we ran a small-scale, anecdotal experiment that fished for Gmail ads targeted against a few specific topics. We created emails focused on topics such as cancer, Alzheimer, depression, HIV, race, homosexuality, pregnancy, divorce, and debt. Each email consisted of keywords closely related to one topic (e.g., the depression-related email

Topic	Targeted Ads	XRay Scores	# Accounts & Displays
Alzheimer	Black Mold Allergy Symptoms?	0.99,	9/9,
	Expert to remove Black Mold.	0.05	61/198
	Adult Assisted Living.	0.99,	8/8,
Cancer	Affordable Assisted Living.	0.99	12/14
	Ford Warriors in Pink. Join The Fight.	0.96,	9/9,
	Rosen Method Bodywork for physical or emotional pain.	0.98,	1022/1106
Depression	Shamanic healing over the phone.	0.98,	7/7,
	Text Coach - Get the girl you want and Desire.	0.99,	24/598
		0.99,	16/16,
African American		0.99,	117/117
	Racial Harassment? Learn your rights now.	0.93,	7/7,
	Racial Harassment, Hearing racial slurs?	0.04	31/276
Homosexuality		0.99,	10/10,
	SF Gay Pride Hotel.	0.2	851/5808
	Luxury Waterfront.	0.99,	10/10,
Pregnancy	Cedars Hotel Loughborough, 36 Bedrooms, Restaurant, Bar.	0.2	627/7172
	Find Baby Shower Invitations.	0.99,	9/9,
	Get Up To (60% Off) Here!	1.0	50/99
	Ralph Lauren Apparel.	0.96,	8/8,
	Official Online Store.	1.0	36/43
	Clothing Label-USA.	0.99,	9/9,
Best Custom Woven Labels.	1.0	14/14	
Divorce	Bonobos Official Site.	0.99,	9/9,
	Your Closet Will Thank You.	0.99	64/71
	Law Attorneys specializing in special needs kids education.	0.99,	9/9,
Debt	Cerbone Law Firm, Helping Good People Thru Bad Times	0.99,	9/9,
	Take a New Toyota Test Drive, Get a \$50 Gift Card On The Spot.	0.99,	635/666
	Great Credit Cards Search.	0.99,	10/10,
	Apply for VISA, MasterCard...	0.9	94/94
	Stop Creditor Harassment, End the Harassing Calls.	0.9	7/7,
		0.9	58/65
		0.99,	9/9,
		0.0	151/2358
		0.99,	8/8,
		0.96	256/373

Figure 10: **Example of Targeted Ads.** Columns three and four show the same data as columns four and five in Figure 5.

included *depression*, *depressed*, and *sad*; the homosexuality email included *gay*, *homosexual*, and *lesbian*). We then launched XRay’s Gmail ad collection and examined the targeting associations. We acknowledge that a much larger-scale experiment is needed to reach statistically-meaningful conclusions. Hence, we relate our experience by example.

Figure 10 shows ads that XRay associated with each topic, with its confidence scores. Conservatively, we only consider ads with high scores. We make two observations. First, our small-scale experiment confirms that it is possible to target sensitive topics in users’ inboxes. All disease-related emails, except for the HIV one, are strongly correlated with a number of ads. A “Shamanic healing” ad appears exclusively in accounts containing the depression-related email, and many times in its context; ads for assisted living services target the Alzheimer email; and a Ford campaign to fight breast cancer targets the cancer email. Race, homosexuality, pregnancy, divorce, and debt also attract plenty of ads. For example, the pregnancy email is strongly targeted by an ad for baby-shower invitations (shown in the figure), maternity- and lactation-related ads (not shown), and, interestingly, a number of ads for general-purpose clothing

(shown). As another example, the debt email is strongly targeted by a car dealership ad that entices the targeted users to take a Toyota test drive using a \$50 gift offering. Discussing the morality of targeting such sensitive topics is beyond our statute, however we believe that the lack of transparency, coupled with sensitive-topic targeting, opens users to subtle dangers, a topic we discuss next.

Second, for many ads, the association with the targeted email is not obvious at all. Nothing in the “Shamanic healing” ad suggests targeting against depression; nothing in the general-purpose clothing ads suggest targeting against pregnancy; and nothing in the “Cedars hotel” ad suggests an orientation toward the homosexuality email. If no keyword in the ad suggests relation with sensitive topics, a user clicking on the ad may not realize that they could be disclosing private information to advertisers. Imagine an insurance company wanted to gain insight into pre-existing conditions of its customers before signing them up. It could create two ad campaigns – one that targets cancer and another youth – and assign different URLs to each campaign. It could then offer higher premium quotes to visitors who come through the cancer-related ads to discourage them from signing up while offering lower premium quotes to those who come through youth-related ads. We believe that the potential for this attack illustrates the urgent need for increased transparency in ad targeting.

6.7 Summary

Our evaluation results show that XRay supports fine-grained, accurate data tracking in popular Web services, scales well with the size of data being tracked, is general and flexible enough to work efficiently for three Web services, and robustly uses systems techniques to discover associations when ad contents provide no indication of them. We next discuss how XRay meets its last goal: robustness against honest-but-curious attackers.

7 Security Analysis

As stated in §3.3, two threat models are relevant for XRay and applicable to different use cases. First, an *honest-but-curious* Web service does not attempt to frustrate XRay, but it could incorporate defenses against typical Web attacks, such as DDoS or spam, that might interfere with XRay’s functioning. Second, a *malicious service* takes an adversarial stand toward XRay, seeking to prevent or otherwise disrupt its correlations. Our current XRay prototype is robust against the former threat and can be extended to be so against the latter. In either case, third-party advertisers can attempt to frustrate XRay’s auditing. We discuss each threat in turn.

Non-Malicious Web Services. Many services incorporate protections against specific automated behaviors. For example, Google makes it hard to create new ac-

counts, although doing so remains within reach. Moreover, many services actively try to identify spammers and click fraud. Gmail includes sophisticated spam filtering mechanisms, while YouTube rate limits video viewing to prevent spam video promotion. Finally, many services rate limit access from the same IP address.

XRay-based tools must be aware of these mechanisms and scale back their activities to avoid raising red flags. For example, our prototype for Gmail, YouTube, and Amazon rate limit their output collection in the shadow accounts. Moreover, XRay’s very design is sensitive to these challenges: by requiring as few accounts as possible, we minimize: (1) the load on the service imposed by auditing, and (2) the amount of input replication across shadow accounts. Moreover, XRay’s workloads are often atypical of spam workloads. Our XRay Gmail plugin sends emails from one to a few other accounts, while spam is sent from one account to many other accounts.

Malicious Third-Party Advertisers. Third-party advertisers have many ways to obfuscate their targeting from XRay, particularly if it may arouse a public outcry. First, an advertiser could purposefully weaken its targeting by, for example, targeting the same ad 50% on one topic and 50% on another topic. This weakens input/output correlation and may cause XRay to infer untargeting. However, it also makes the advertisers’ targeting less effective and potentially more ambiguous if their goal is to learn specific sensitive information about users. Second, an advertiser might target complex combinations of inputs that XRay’s basic design cannot discover. Our accompanying technical report shows an example of how advertisers might achieve this [26]. It also extends our theoretical models so they can detect targeting on linear combinations with only a constant factor increase in the number of accounts. We plan to incorporate and evaluate these extensions in a future prototype.

Malicious Web Services. A malicious service could identify and disable shadow accounts. Identification could be based on abnormal traffic (successive reloads of email pages), data distribution within accounts (several accounts with subsets of one account), and perhaps more. XRay could be extended to add randomness and deception (e.g., fake emails, varying copies). More importantly, a collaborative approach to auditing, in which users contribute their ads and input topics in a privacy-preserving way is a promising direction for strengthening robustness against attacks. Web services cannot, after all, disable legitimate user accounts to frustrate auditing. We plan to pursue this direction in future work.

8 Discussion

XRay takes a significant step toward providing data management transparency in Web services. As an initial effort, it has a number of limitations. First, both the Set

Intersection and Bayesian algorithms assume independent targeting across accounts and over time. In reality, ad targeting is not always independent across either. For example, advertisers set daily ad budgets. When the budget runs out, an ad can stop appearing in accounts mid-experiment even though it has the targeted attributes. The system might incorrectly assume that no targeting is taking place, when it could resume the next day. XRay takes reduced coverage into account, but differences between ads can let some targeting pass unnoticed. XRay does not currently account for these dependencies, but estimating their impact is an important goal for future work.

Second, we assume that targeting noise is bounded and smaller than the targeting signal. While this condition seems to hold on the evaluated services, other services making more local decisions may be harder to audit. For example, Facebook might target ads based on friends' information, potentially creating noise that is as high as the targeting signal. A future solution might imitate the social network in shadow accounts.

Third, XRay uses Web services atypically. To the best of our knowledge, it does not violate any terms of service. It does, however, collect ads paid for by advertisers to detect correlation. Ad payment is per impression and pay per click. The former is vastly less expensive than the latter [32]. XRay creates false impressions only but never clicks on ads. A back-of-the-envelope calculation using impression pricing from [32] of \$0.6/thousand impressions reveals that XRay's cost should be minimal: at most 50 cents per ad for our largest experiments.

Despite these limitations, XRay has proven itself useful for many needs, particularly in an auditing context. An auditor can craft inputs that avoid many of these limitations. For example, emails can be written to avoid as much overlap as possible and keep the size of inputs used for targeting within reasonable bounds. We hope that XRay's solid correlation components will streamline much-needed investigations – by researchers, journalists, or the FTC – into how personal data is being used.

9 Related Work

While §2.2 covered Web data protection and auditing related works, we next cover other related topics. Our work relates to recent efforts to measure various forms of personalization, such as search [21, 47], pricing [31], and ad discrimination [40]. They generally employ a methodology similar in spirit to differential correlation, but their goals differ from ours. They aim to quantify *how much* output is personalized and what *type* of information is used overall. In contrast, XRay seeks to provide fine-grained diagnosis of which *input data* generates which personalized results. Through its scaling mechanisms – unique in the personalization and

data tracking literature – XRay scales well even when the relevant inputs are many and unknown in advance.

Our work also relates to a growing body of research measuring advertising networks. These networks, notably complex and difficult to crawl [3], are rendered opaque by the need to combat click fraud [9], and have been shown to be susceptible to leakage [24] and profile reconstruction attacks [6]. As for other personalization, prior studies focused mostly on macroscopic trends (e.g., What fraction of ads are targeted?) [3] or qualitative trends (e.g., Which ads are targeted toward gay males?) [20]. Various studies showed traces – but not a prevalence – of potential abuse through concealed targeting [20] and data exchange between services [46]. These works primarily focus on display advertising, and each distinguishes contextual advertising using a specific classifier with semantic categories obtained from Google's Ad Preferences Managers or another public API [28].

XRay departs significantly from these works. First, since it entirely ignores the content and even the domain of targeting, it is readily applied as-is to ads in Gmail, product recommendations, and videos. Second, while previous methods label ads as “behavioral” in bulk once other explanations fail [28], XRay remains grounded on positive evidence, and determines to *which* inputs an output should be attributed. Third, XRay's mechanisms to avoid exponential input placement and deal with overlapping inputs are unprecedented in the Web-data-tracking context. While they resemble *black box* software testing [4], the specific targeting assumption we leverage have, to our knowledge, no prior equivalent.

10 Conclusions

The tracking of personal data usage poses unique challenges. XRay shows for the first time that accurate, *fine-grained* tracking need not compromise portability and scalability. For users who care about *which* piece of their data has been targeted, it offers a unique level of precision and protection. Our work calls for and promotes the best practice of voluntary transparency, while at the same time empowering investigators and watchdogs with a significant new tool for increased vigilance.

11 Acknowledgements

We thank our shepherd, Dan Boneh, the anonymous reviewers, and numerous colleagues (Jonathan Bell, Sandra Kaplan, Michael Keller, Yoshi Kohno, Hank Levy, Yang Tang, Nicolas Viennot, and Junfeng Yang) for their valuable feedback. This work was supported by DARPA Contract FA8650-11-C-7190, NSF CNS-1351089 and CNS-1254035, Google, and Microsoft.

References

- [1] Adblock plus. <https://adblockplus.org>.

- [2] Amazon. Product categories. <http://services.amazon.com/services/soa-approval-category.htm>.
- [3] P. Barford, I. Canadi, D. Krushevskaja, Q. Ma, and S. Muthukrishnan. Adscape: Harvesting and Analyzing Online Display Ads. In *Proc. of the 23rd International Conference on WWW*, 2014.
- [4] B. Beizer. *Black-Box Testing. Techniques for Functional Testing of Software and Systems*. John Wiley & Sons, May 1995.
- [5] D. Boneh, G. Crescenzo, R. Ostrovsky, and G. Persiano. Public Key Encryption with Keyword Search. In *Proc. of the ACM European Conference on Computer Systems (EuroSys)*, pages 506–522. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.
- [6] C. Castelluccia, M. A. Kaafar, and M. Tran. Betrayed by your ads! *PETS'12: Proceedings of the 12th International Conference on Privacy Enhancing Technologies*, 2012.
- [7] W. Cheng, Q. Zhao, B. Yu, and S. Hiroshige. Tainttrace: Efficient flow tracing with dynamic binary rewriting. In *Proc. of the 11th IEEE Symposium on Computers and Communications*, 2006.
- [8] Chrome web store - collusion for chrome. <https://chrome.google.com/webstore/detail/collusion-for-chrome/ganlifbpcplnldliibcbegplfmcfigp>.
- [9] V. Dave, S. Guha, and Y. Zhang. Measuring and fingerprinting click-spam in ad networks. In *SIGCOMM '12: Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Crchitectures, and Protocols for Computer Communication*. ACM Request Permissions, Aug. 2012.
- [10] N. Diakopoulos. Algorithmic accountability reporting: On the investigation of black boxes. Tow Center for Digital Journalism, Columbia University. February, 2014.
- [11] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. *Technical Report*, 2004.
- [12] W. Enck, P. Gilbert, B. gon Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth. TaintDroid: An information-flow tracking system for realtime privacy monitoring on smartphones. In *Proc. of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2010.
- [13] M. Fredrikson and B. Livshits. RePriv: Re-imagining Content Personalization and In-browser Privacy. *2011 IEEE Symposium on Security and Privacy*, pages 131–146, 2011.
- [14] R. Geambasu, T. Kohno, A. Levy, and H. M. Levy. Vanish: Increasing data privacy with self-destructing data. In *Proc. of USENIX Security*, 2009.
- [15] C. Gentry. Fhe using ideal lattices. In *Proc. of the ACM Symposium on Theory of Computing (STOC)*, 2009.
- [16] D. B. Giffin, A. Levy, D. Stefan, D. Terei, D. Mazières, J. C. Mitchell, and A. Russo. Hails: Protecting data privacy in untrusted web applications. In *In Proc. of the 10th USENIX Conference on Operating Systems Design and Implementation*, 2012.
- [17] Google. Adsense categories. <https://support.google.com/adsense/answer/3016459>.
- [18] J. Gould. SafeGov.org - Google admits data mining student emails in its free education apps, 2014.
- [19] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proc. of the ACM Conference on Computer and Communications Security (CCS)*, 2006.
- [20] S. Guha, B. Cheng, and P. Francis. Challenges in measuring online advertising systems. In *IMC '10: Proceedings of the 10th Annual Conference on Internet Measurement*, 2010.
- [21] A. Hannak, P. Sapiezynski, A. M. Kakhki, B. Krishnamurthy, D. Lazer, A. Mislove, and C. Wilson. Measuring personalization of web search. In *WWW '13: Proceedings of the 22nd International Conference on World Wide Web*, 2013.
- [22] A. L. Hughes and L. Palen. Twitter adoption and use in mass convergence and emergency events. *International Journal of Emergency Management*, 2009.
- [23] A. Korolova. Privacy Violations Using Microtargeted Ads: A Case Study. In *ICDM Workshops*, 2010.
- [24] A. Korolova. Privacy violations using microtargeted ads: A case study. *Data Mining Workshops (ICDMW), 2010 IEEE International Conference on*, pages 474–482, 2010.
- [25] B. Krishnamurthy and C. E. Wills. On the leakage of personally identifiable information via online social networks. In *Proc. of the 2nd ACM Workshop on Online Social Networks*, 2009.
- [26] M. Lecuyer, G. Ducoffe, F. Lan, A. Papancea, T. Petsios, R. Spahn, A. Chaintreau, and R. Geambasu. XRay: Enhancing the Web's Transparency with Differential Correlation. Technical report, CS Department, Columbia University, 2014.
- [27] Lightbeam. <http://www.mozilla.org/lightbeam/>.
- [28] B. Liu, A. Sheth, U. Weinsberg, J. Chandrashekar, and R. Govindan. AdReveal: improving transparency into online targeted advertising. In *Proc. of the Twelfth ACM Workshop on Hot Topics in Networks*, 2013.
- [29] D. Mattioli. WSJ.com - On Orbitz, Mac Users Steered to Pricier Hotels, 2012.
- [30] V. McKalin. Techtimes.com - google: We promise not to spy on student email accounts to deliver ads, 2014.
- [31] J. Mikians, L. Gyarmati, V. Erramilli, and N. Laoutaris. Detecting price and search discrimination on the internet. In *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*, pages 79–84, 2012.
- [32] L. Olejnik, T. Minh-Dung, C. Castelluccia, et al. Selling off privacy at auction. In *In Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2013.
- [33] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan. Cryptdb: Protecting confidentiality with encrypted query processing. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles, SOSP '11*, pages 85–100, 2011.
- [34] F. Roesner. sharemenot.cs.washington.edu.
- [35] F. Roesner, T. Kohno, and D. Wetherall. Detecting and defending against third-party tracking on the web. In *NSDI'12: Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*. USENIX Association, Apr. 2012.
- [36] A. Sadilek and H. Kautz. Modeling the impact of lifestyle on health at scale. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, 2013.
- [37] SafeGov.org. Declaration of Kyle C. Wong in Support of Google Inc.'s Opposition to Plaintiffs' Motion for Class Certification, 2013.
- [38] Snapchat. <http://blog.snapchat.com/>.
- [39] Snapchat blog - how snaps are stored and deleted.
- [40] L. Sweeney. Discrimination in online ad delivery. *Communications of the ACM*, 56(5), May 2013.
- [41] The Guardian. Snapchat's expired snaps are not deleted, just hidden, 2014.
- [42] V. Toubiana, A. Narayanan, and D. Boneh. Adnostic: Privacy preserving targeted advertising. *Proc. NDSS*, 2010.
- [43] J. Valentino-Devries, J. Singer-Vine, and A. Soltani. WSJ.com - Websites Vary Prices, Deals Based on Users' Information, 2012.
- [44] X. Wang, M. Gerber, and D. Brown. Automatic crime prediction using events extracted from twitter posts. In S. Yang, A. Greenberg, and M. Endsley, editors, *Social Computing, Behavioral - Cultural Modeling and Prediction*, volume 7227 of *Lecture Notes in Computer Science*, pages 231–238. 2012.
- [45] A. Whitten and J. D. Tygar. Why Johnny can't encrypt: A usability evaluation of PGP 5.0. In *Proc. of USENIX Security*, 1999.
- [46] C. E. Wills and C. Tatar. Understanding What They Do with What They Know. *WPES '12: Proceedings of the 12th Annual ACM Workshop on Privacy in the Electronic Society*, 2012.
- [47] X. Xing, W. Meng, D. Doozan, N. Feamster, W. Lee, and A. C. Snoeren. Exposing Inconsistent Web Search Results with Bobble. *Passive and Active Measurements Conference*, 2014.
- [48] Y. Zhu, J. Jung, D. Song, T. Kohno, and D. Wetherall. Privacy scope: A precise information flow tracking system for finding application leaks. Technical Report UCB/Eecs-2009-145, 2009.
- [49] P. R. Zimmermann. The official PGP user's guide. 1995.

A Proof of Theorem 1

A.1 Targeting functions, Axioms and Core Family

A combination \mathcal{C} of order r , also called r -combination, is a subset of r elements among the N inputs.

Each given ad is associated with a *targeting function* defined as a mapping f from any subset \mathcal{C} of the N inputs into $\{0, 1\}$, where $f(\mathcal{C}) = 1$ denotes that an account containing \mathcal{C} as inputs should be targeted. By convention, untargeted ads are associated with the null function $f(\cdot) = 0$. Any targeting function f satisfies two axioms:

- **monotonicity:** $\mathcal{C} \subseteq \mathcal{C}' \implies f(\mathcal{C}) \leq f(\mathcal{C}')$.
 - **input-sensitivity:** $\exists \mathcal{C}, \mathcal{C}'$ s.t. $f(\mathcal{C}) = 0, f(\mathcal{C}') = 1$.
- Monotonicity simply reflects that an account with strictly more interest or hobbies should in theory be relevant to more ads, and never to less. Input sensitivity prevents the degenerate case where a targeting function is constant.

A family S of size l is any collection of l distinct combinations. The *order* of this family is defined as the largest order of a combination it contains. For any family S , one can define a targeting function that takes value 1 whenever the subset contains at least one combination in S . Indeed, as shown in [26], the converse is true:

Lemma 1 *For each monotone, input-sensitive targeting function there exists a unique family S satisfying:*

- S has size l and order r and it explains f , which means $f(\mathcal{C}) = 1$ holds if and only if $\exists \mathcal{C}' \in S, \mathcal{C}' \subseteq \mathcal{C}$.
- No family of size $l' < l$ explains f .
- No family of order $r' < r$ explains f .

Hence, associated with each ad and therefore each targeting function is a unique family of input combinations that are targeted, called the ad's *core family*, and we now sketch why it is correctly identified by our algorithm.

A.2 Algorithm and Correctness

For any family of subsets S and fraction $0 \leq x \leq 1$, we say a subset of inputs \mathcal{C} is an x -intersecting subset of S if x subsets in S have at least one input in \mathcal{C} . Our proof exploits an original connection between small intersecting subsets (that can be found efficiently) to show how they can reveal a core family. One way to understand why is the following: say, for instance, that the targeting function f takes value 1 exactly when one of the inputs within \mathcal{C} is found in the account. Then \mathcal{C} is exactly the union of inputs found in the core family and intersects all accounts within scope, i.e., forms a large fraction of those receiving the ad.

The key property to explain our algorithm is random subsets. We can show under the conditions of the theorem that there exists $0 < x < 1$ that satisfies two properties related to the inputs of accounts receiving the ads: (1) if targeting does not occur, then with a large probability we cannot find a subset of l inputs that meets at least a fraction x of the accounts seeing the ad, and (2) if targeting does occur, we have accounts receiving the ads for

various reasons, within and outside the targeting scope. But we can show with high probability that at least a fraction x of them are within scope and hence must include one combination in the core family. Since with each core family of size l one can associate an intersecting subset that contains at most l elements, checking the existence of such a subset reveals the presence of targeting.

This explains why an algorithm can qualitatively conclude whether targeting occurs or not, but it does not explain how the core family can be computed. However, leveraging stronger results of random subsets allows to apply the same rule recursively, offering multiple ways to determine exactly the core family even with a polynomial number of operations.

More formally, we define: A random *Bernoulli subset*, denoted by $B(n, p)$, is a subset such that any of n elements is contained with probability p independently of all others. A random *Bernoulli family* of size m is a collection of m independent Bernoulli subsets. We first show property (1) above more formally:

Lemma 2 *Let $x > 0$, $s \in \mathbb{N}$, $p < 1 - (1-x)^{\frac{1}{s}}$, and a Bernoulli family $B_1(n, p), B_2(n, p), \dots, B_m(n, p)$. For any $\varepsilon > 0$ and polynomial P of degree $\leq r$, there exists $A > 0$ such that with probability $(1 - \frac{\varepsilon}{P(n)})$ no x -intersection subset exists of size s whenever we have:*

$$m \geq A \cdot ((s+r) \ln(n) + \ln(1/\varepsilon)).$$

To prove property (2), we need to bound, among accounts receiving an ad, the fraction that is outside the scope of targeting but still receives the ads because $p_{\text{out}} > 0$. Formally, we have:

Lemma 3 *Let $x > 0$, $\alpha > 0$, and a core family of size l and order r $p_{\text{in}}, p_{\text{out}}$ where we have $p_{\text{out}}/p_{\text{in}} < \frac{1-x}{x} \frac{\alpha^r}{1-\alpha^r}$. Let \mathcal{C} be a combination of order r .*

For any $\varepsilon > 0$ and polynomial P of degree $\leq r$, there exists $A > 0$ such that with probability $(1 - \varepsilon/P(n))$ the following holds: Among accounts containing \mathcal{C} and receiving the ad, at least x fraction of them is within the targeting scope whenever we have:

$$m \geq A \cdot (r \ln(n) + \ln(1/\varepsilon)).$$

The two lemmas above (proved in [26]) can be combined whenever α satisfies the inequality for p in the first lemma, which shows that an algorithm can detect the presence of targeting whenever

$$p_{\text{out}}/p_{\text{in}} < \frac{1-x}{x} \frac{(1-(1-x)^{\frac{1}{l}})^r}{1-(1-(1-x)^{\frac{1}{l}})^r}.$$

A naive exponential algorithm could be used to exhaustively search for a core family using this brick. We also show that a polynomial algorithm can refine this analysis to compute the core family at the expense of a more complex recursion in [26].

APPENDIX L

Web Transparency for Complex Targeting: Algorithms, Limits and Tradeoffs

Web Transparency for Complex Targeting: Algorithms, Limits, and Tradeoffs

Guillaume Ducoffe, Mathias Lécuyer, Augustin Chaintreau, Roxana Geambasu
Inria & U. Nice Sophia Antipolis, Columbia U.
guillaume.ducoffe@inria.fr, {mathias,augustin,roxana}@cs.columbia.edu

Big Data promises important societal progress but exacerbates the need for due process and accountability. Companies and institutions can now discriminate between users at an individual level using collected data or past behavior. Worse, today they can do so in near perfect opacity. The nascent field of *web transparency* aims to develop the tools and methods necessary to reveal how information is used, however today it lacks robust tools that let users and investigators identify targeting using multiple inputs.

Here, we formalize for the first time the problem of detecting and identifying targeting on *combinations of inputs* and provide the first algorithm that is asymptotically exact. This algorithm is designed to serve as a theoretical foundational block to build future scalable and robust web transparency tools. It offers three key properties. First, our algorithm is *service agnostic* and applies to a variety of settings under a broad set of assumptions. Second, our algorithm's analysis delineates a *theoretical detection limit* that characterizes which forms of targeting can be distinguished from noise and which cannot. Third, our algorithm establishes *fundamental tradeoffs* that lead the way to new metrics for the science of web transparency. Understanding the tradeoff between effective targeting and targeting concealment lets us determine under which conditions predatory targeting can be made unprofitable by transparency tools.

1. QUICK OVERVIEW

A primer on web transparency tools.

To address the big-data web's untenable opacity, a new set of transparency tools have been proposed recently [7, 4, 5, 6, 3, 9]. Generally speaking, they assume no insider information about how the data-driven web service operates and instead rely on a specific form of *black box testing* [2] to detect data use. Briefly, a transparency tool works as follows. First, it collects the results of a series of tests in which *inputs* vary, *e.g.*, browsing history [7], search history [4], emails [6], locations [9], or explicit profile information [3]. Second, by

examining the observed *outputs* – *e.g.*, search results [4, 9], ads seen [3, 6], recommendations [6, 5], or prices [7, 5] – the tool deduces how the system personalizes its behavior based on this input. Finally, the tool's deductions are used as *hypotheses* that are further analyzed for implications by the tool's users, such as end users, journalists, privacy watchdogs, or federal investigators.

To be valuable to their users, transparency tools strive to meet three requirements:

1. **Scalability:** Each test may involve multiple preliminary steps to open a new web account and populating its inputs. These steps cannot always be automated and may be expensive (*e.g.*, creating a Google account requires buying a new phone number). It is also important to keep resources to a minimum as the size of outputs/inputs grows.
2. **Accuracy:** The deduction that the tool provides should be *sound*, which means that it can be trusted not to originate from noise or other limitations of the experiments. The tool should also be *complete* which specifies that it rarely misses an important deduction.
3. **Broad Applicability:** Ideally, the same tool should apply not only to many different services but also various forms of data usage within those services, with only minor and intellectually simple changes.

Perhaps unsurprisingly, the first two requirements are often in conflict. The third makes the problem extremely challenging, and has barely been considered to date. With few exceptions [6, 3], previous transparency tools were designed for a specific service or usage in order to detect a particularly sensitive topic: price discrimination [7], search results personalization [4], censorship [9]. Only recently has development of widely-applicable, generic tools begun to be considered to allow generic data collections [5] and service-agnostic detection methods [6, 3]. Despite appearances, however, we find that even the latest transparency tools are limited in the kind of data uses they can support accurately and scalably. We believe that the biggest roadblock is the lack of support for detecting complex, multi-input targeting, which we find mandatory for building scalable, accurate, and broadly applicable tools.

Our new findings.

We prove that targeting that uses one or several combinations of N inputs can be detected and identified with asymptotically perfect accuracy, and that this only requires

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage, and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s). Copyright is held by the author/owner(s).

SIGMETRICS'15, June 15–19, 2015, Portland, OR, USA.

ACM 978-1-4503-3486-0/15/06.

<http://dx.doi.org/10.1145/2745844.2745896>.

a logarithmic order $O(\ln(N))$ of experimental observations. To place our contribution in respect to prior work, this shows that a web transparency tool can remain scalable and accurate, without being strictly restricted to single input targeting. However, and this is where our contribution contrasts the most with previous results, it comes at a cost: the intensity of targeting (defined below) needs to be sufficiently large. In other words, web transparency need not always be blind to combined targeting, however as the inherent complexity of targeting increases, targeting becomes easier to conceal. Our results open a new chapter in the understanding of Big Data: to determine sufficient and necessary conditions under which one can prevent its opacity.

2. PROBLEM FORMULATION & RESULTS

We formalize the following intuitive problem: given a set of N inputs representing possible information items present in a user's account (such as emails or searches), we wish to determine how they affect occurrence of one particular output of interest (such as an ad or a recommendation).

Our main assumption is that the output is affected through an unknown targeting function f of the inputs, to be determined. The function f is defined separately for each output. The targeting function f is a mapping from the set of all combinations to $\{0; 1\}$. By convention, $f(C) = 1$ indicates that an account containing C is targeted, and we denote $f(\cdot) = 0$ if the ad is untargeted.

Experiments and outcome properties.

Because in practice we have no access to the targeting function, we rely on experiments to observe its reaction to various inputs. Intuitively, these experiments collect outputs from a set of accounts that contain subsets of the inputs and produce a set of observations of f . For example, experiments could collect ads for accounts with different subsets of emails. More formally, the experimental infrastructure we assume is similar to an *oracle* from function learning theory [8, 1]. We assume that our experimental oracle satisfies the following axiom. There exist two probabilities p_{in} , p_{out} such that:

$$\mathbb{P}[\mathcal{O}(C_i) = 1 | f(C_i) = 1] \geq p_{in} > p_{out} \geq \mathbb{P}[\mathcal{O}(C_j) = 1 | f(C_j) = 0].$$

where p_{in} is a minimal bound on the probability that an account receives an output that is relevant for it and p_{out} is a maximal bound on the probability that an account receives an output that is not relevant for it. This axiom properly states that f is related to the outcome we study. It allows the variables to also depend on other factors: hidden inputs that are not in the set of N we study, external sources of randomness such as availability of ad-slot, competition. One experimental design used in practice [6, 3] and that fits this axiom is to populate each account randomly so that an input independently appears with probability α .

Under the assumptions above, we say that an algorithm using m observations solves the targeting detection problem if it can correctly decide whether $f(\cdot) \neq 0$ and hence that the output is targeted using at most m queries to \mathcal{O} . Going further, an algorithm solves the targeting identification problem if it correctly returns the function f . Naturally, both problems rely on random observations and hence our goal is to design algorithms whose detection/identification error is arbitrarily small for large N .

Since one should distinguish (at least) between N inputs, it seems that a minimum of $\Omega(\ln(N))$ binary observations are absolutely necessary at least for the identification. This is hence what we assume and we aim at keeping it at this absolute minimum.

THEOREM 1. *Assuming that f is a monotone DNF with size at most s and width at most w , and that ratio p_{out}/p_{in} is below a predetermined bound, we provide a targeting detection algorithm that for any $\varepsilon > 0$ requires $O(\ln(N/\varepsilon))$ observations, $O(N^s \ln(N/\varepsilon))$ operations and is correct with probability $(1 - \varepsilon/N)$.*

THEOREM 2. *Under the same assumption, we provide a targeting identification algorithm that for any $\varepsilon > 0$ requires $O(\ln(N/\varepsilon))$ observations, $O(N^{s+w} \ln(N/\varepsilon))$ operations and is correct with probability $(1 - \varepsilon/N)$.*

3. ACKNOWLEDGEMENTS

This work was supported by DARPA Contract FA8650-11-C-7190, NSF CNS-1351089 and CNS-1254035, Google, and Microsoft.

4. REFERENCES

- [1] D. Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, Apr. 1988.
- [2] B. Beizer. *Black-Box Testing. Techniques for Functional Testing of Software and Systems*. John Wiley & Sons, May 1995.
- [3] A. Datta, M. C. Tschantz, and A. Datta. Automated Experiments on Ad Privacy Settings: A Tale of Opacity, Choice, and Discrimination. *arXiv.org*, Aug. 2014.
- [4] A. Hannak, P. Sapiezynski, A. M. Kakhki, B. Krishnamurthy, D. Lazer, A. Mislove, and C. Wilson. Measuring personalization of web search. In *WWW '13: Proceedings of the 22nd international conference on World Wide Web*. International World Wide Web Conferences Steering Committee, May 2013.
- [5] A. Hannak, G. Soeller, D. Lazer, A. Mislove, and C. Wilson. Measuring Price Discrimination and Steering on E-commerce Web Sites. In *IMC '14: Proceedings of the 2014 Conference on Internet Measurement Conference*. ACM Request Permissions, Nov. 2014.
- [6] M. Lecuyer, G. Ducoffe, F. Lan, A. Papancea, T. Petsios, R. Spahn, A. Chaintreau, and R. Geambasu. XRay: Enhancing the Web's Transparency with Differential Correlation. In *23rd USENIX Security Symposium (USENIX Security 14)*, San Diego, CA, 2014. USENIX Association.
- [7] J. Mikians, L. Gyarmati, V. Erramilli, and N. Laoutaris. Detecting price and search discrimination on the internet. In *HotNets-XI: Proceedings of the 11th ACM Workshop on Hot Topics in Networks*. ACM Request Permissions, Oct. 2012.
- [8] R. A. Servedio. On learning monotone DNF under product distributions. *Information and Computation*, 193(1):57–74, Aug. 2004.
- [9] X. Xing, W. Meng, D. Doozan, N. Feamster, W. Lee, and A. C. Snoeren. Exposing Inconsistent Web Search Results with Bobble. *Passive and Active Measurements Conference*, 2014.

APPENDIX M

Can Web Transparency Tools Cope with Complex Targeting?

Corresponding Author*, Second Author, Third Author, and Fourth Author

Can Web Transparency Tools Cope with Complex Targeting?

Abstract: Big Data promises important societal progress but exacerbates the need for algorithmic accountability as more and more decisions affecting millions of users are being automated using personal and private information. The recent area of *web transparency* has developed generic methods to reveal *which* information item or input generates personalization and differentiated treatments. We surveyed 13 transparency tools and found that they all make stringent assumptions on personalization; unless an algorithm exploits a *single input*, it can be *undetected* – or to be more precise, ensuring its detection incurs a *prohibitive exhaustive search*. The ever increasing set of relevant inputs to monitor exacerbates the difficulty of this task. As personalization becomes ever more complex, we see the familiar conditions of an arms race, where transparency and evasion techniques grow more sophisticated. Until now, we understand little of it, let alone how to best address it to make the web more transparent.

In this paper, we present the first algorithms for transparency enhancing tools that are efficient and correct even with general complex targeting. To take a simple example, ad-targeting can usually depend on multiple inputs in a way that may not satisfy previous properties like linearity. Here we formalize it for the first time, and show how to combine simple heuristics into transparency algorithms that are provably correct while keeping the cost of queries and computations minimum (actually matching computational lower bounds). We introduce a key metric, the *targeting lift*, and show it dictates the conditions for algorithms with different computation cost to succeed. Small scale experiments with 20 or 50 inputs, and large scale simulations with up to 100,000 inputs complement our theoretical analysis: they prove that our algorithms expand the scope of targeting found beyond the state of the art, that they exhibit excellent scaling properties as the set of inputs grows, and that they resist even challenging targeting lift values.

Keywords: transparency enhancing tools, behavioural targeting, blackbox analysis, computational learning, analysis of random algorithms

1 Introduction

“Just as neighborhoods can serve as a proxy for racial or ethnic identity, there are new worries that big data technologies could be used to digitally redline unwanted groups, either as customers, employees, tenants, or recipients of credit.”

– Executive Office of the President, *Big Data: Seizing Opportunities, Preserving Values*, [16].

Big Data – a set of measurement and decision tools leveraging large sets of records such as emails, web visits, voice calls, credit card transactions, tweets, and geo-tagged content – is becoming central to any business today. Tomorrow it promises to unlock opportunities in myriad public services, promoting healthy behaviors [18], environmental sustainability [17], preventing crime [21], mitigating congestion and pollution in transportation [14], improving disaster recovery [13], and economic development [9]. However, the availability and use of this information can also challenge our society’s values, as illustrated by recent cases of online discrimination found in advertising [20], pricing [8, 15], and hiring [1]. Our society must be able to celebrate Big Data’s potential for progress *and* perpetuate our commitment to equal opportunities for all: We should be ready to guard all from the intended or unintended consequences that differentiation enables.

Today, most systems handling personal data, such as web services and mobile applications, behave as *black-boxes*, which no one outside the service provider – neither the end-users nor privacy watchdogs or federal investigators like the Federal Trade Commission – can understand and monitor. *Web transparency* is a nascent area that takes an engineering approach to promote better accountability. Its goal is to enable everyone to answer critical questions about *how* personal data is being used. However, this is a complex question to answer and

*Corresponding Author: Corresponding Author: Affil,

E-mail: email@email.edu

Second Author: Affil, E-mail: email@email.edu

Third Author: Affil, E-mail: email@email.edu

Fourth Author: Affil, E-mail: email@email.edu

current tools quickly show their limits. For example, we found that state of the art tools [6, 11] have been designed and evaluated primarily to detect when a single input is responsible for an action. But personal data is growing in size and complexity. For example, as we present later in three potential scenarios of data use, it could be a *combination* of inputs (*i.e.*, multiple inputs together) that explain personalized outputs. One may then ask “Can transparency enhancing tools extend beyond single-input personalization? or would they meet a computational barrier or restriction which impedes their practicality? If the success of these tools depends on the environment, which metrics matter?”

This paper answers the aforementioned questions as it provides a formal definition of the detection and the identification problem posed by web transparency under complex targeting. We introduce multiple new algorithms, and a new metric that is shown through rigorous proof and experimental/simulated evidence to control the accuracy (or lack thereof) for those tools. Most importantly we show that web transparency continues to scale even with complex targeting, but that it poses new combinatorial challenges and computational tradeoffs. Several of our results suggest future research avenues to design future robust transparency enhancing tools.

We now present the following contributions.

- After reviewing 13 transparency tools, we identify multi-input targeting as a key missing feature, motivated by multiple scenarios in which it may be relevant. We introduce a formal model of online targeting that allows us to pose detection and identification in rigorous terms. This model encompasses arbitrarily more complex targeting and introduces a key metric: the targeting lift. (§2)
- We then introduce an heuristic algorithm which leverages intuitive facts about random subsets to detect when multi-input targeting is taking place. In a small experiment comparing this heuristic to the state of the art, we find that it significantly increases the coverage of transparency tools. This observation even holds when an approximation is used in lieu of the exact suggested heuristic and it motivates a more thorough analysis to prove it rigorously. (§3)
- By a mathematical analysis of properties of random subsets, we show that those detection algorithms (exact and approximate) are provably correct under a specific condition that relates to the targeting lift we introduced above. Moreover, we prove these algorithms can be used iteratively or recursively not

only to detect but to exactly identify the function that is being applied to the inputs. All the algorithms we present scale remarkably well as the number of inputs grows, *i.e.*, they require only $O(\ln N)$ observations. While the first algorithm we introduce is polynomial in computation cost, we prove a (much more involved) approximation exists that correctly identifies the function with linear computation cost, matching a trivial lower bound. (§4)

- Finally, we come back to analyze more thoroughly the impact of the targeting lift, the key metric enabling web transparency. First, our analysis shows that transparency enhancing tools exhibit a tradeoff where lighter forms of targeting can - at least in theory - only be found by more expensive algorithms. Second, we run large numerical simulations proving that even moderate numbers of accounts (around 300) allow us to decipher complex multi-input targeting among 100,000 of inputs, while remaining robust to noise in the targeting lift. These reports demonstrate the possibility of running transparency with 1000 times more inputs than any experiment to date. We show that the greedy heuristic we introduced is robust and efficient, and typically succeeds for much higher levels of noise than what our theoretical bounds would suggest. Finally, we discuss the practical consequences of our results: we define the price of opacity which measures the added cost incurred by an advertiser that aims at running illegal targeting under the radar of current transparency-enhancing tools. (§5)

Our results greatly enhance the complexity that can be handled by transparency tools, and also delineate where they may fail. This is the first formal definition of algorithms that can provably exactly identify targeting in the general case we present here. Before this analysis, the only previous mention of input combination appears as an unvalidated extension in the XRay paper [11], which centers on a different Bayesian algorithm restricted to single input targeting (used for comparison in our small experiment). The authors present no formal proof beyond some initial steps (resembling (Lemma 1, 2 and 3 here) reproduced here to keep this paper self-contained. All other results, proofs and algorithms involved are new, and only partly appeared as part of an unpublished appendix in a technical report [REMOVED FOR ANONYMITY].

2 Problem formulation

2.1 Requirements of transparency tools

To address the big-data web’s untenable opacity, a new set of transparency tools have been proposed recently [6–8, 11, 15, 22]. Generally speaking, they assume no insider information about how the data-driven web service operates and instead rely on a specific form of *black box testing* [4] to detect data use. Briefly, a transparency tool works as follows. First, it collects the results of a series of tests in which *inputs* vary, *e.g.*, browsing history [15], search history [7], emails [11], locations [22], or explicit profile information [6]. Second, by examining the observed *outputs* – *e.g.*, search results [7, 22], ads seen [6, 11], recommendations [8, 11], or prices [8, 15] – the tool deduces how the system personalizes its behavior based on this input. Finally, the tool’s deductions are used as *hypotheses* that are further analyzed for implications by the tool’s users, such as end users, journalists, privacy watchdogs, or federal investigators. The deductions are generally not considered as definite proof, but as quantified hypotheses that require further investigation and reasoning. For example, an FTC investigator might use a transparency tool’s deductions to glean at potential predatory targeting against some vulnerable population, but will open his/her own detailed investigation to validate whether the targeting was intentional or automatic, which of many potential parties applied this targeting, etc.

To be valuable to their users, transparency tools strive to meet three requirements:

1. **Scalability:** Each test may involve multiple preliminary steps to open a new web account and populate its inputs. These steps cannot always be automated and may be expensive (*e.g.*, creating a Google account requires buying a new phone number). It is also important to keep resources to a minimum as the size of outputs/inputs grows.
2. **Accuracy:** The deduction that the tool provides should be *sound*, which means that it can be trusted not to originate from noise or other limitations of the experiments. The tool should also be *complete* which specifies that it rarely misses an important deduction.
3. **Broad Applicability:** Ideally, the same tool should apply not only to many different services but also various forms of data usage within those services, with only minor and intellectually simple changes.

Perhaps unsurprisingly, the first two requirements are often in conflict. The third makes the problem extremely challenging, and has barely been considered to date. With few exceptions [6, 11, 12], previous transparency tools were designed for a specific service or usage in order to detect a particularly sensitive topic, such as price discrimination [15], search personalization [7] or censorship [22]. Only recently has development of widely-applicable, generic tools begun to be considered to allow generic data collections [8] and service-agnostic detection methods [6, 11, 12]. However, we find that even the latest transparency tools are limited in the kind of data usage they can detect or identify in large scale examples. We believe that the biggest roadblock is the lack of support for detecting complex, multi-input targeting, which we argue must be addressed for building scalable, accurate, and broadly applicable tools. We motivate this need next.

2.2 Why does multi-input targeting matter?

Without exception, all existing transparency tools are designed to detect situations in which a *single input* is responsible for an algorithmic decision. AdFisher [6] builds for a given individual input (*i.e.*, gender, or race) a classifier that *retroactively* predicts the presence of an input based on the outputs observed for this account. This allows sound predictions of whether individual inputs are treated differently by the audited system. The XRay tool [11] relies on a Bayesian inference procedure that determines the likelihood that a given input explains the distribution of ads seen across accounts. Sunlight [12] introduces a machine learning technique that approximates targeting functions in a linear model, with no proven guarantee, especially when input combinations are found to affect the targeting function (see below).

We first argue that targeting function that depends on multiple inputs are *far from unusual*. We provide three scenarios where those can be found:

- For years, search engines have combined previous and current queries for input *disambiguation* (*e.g.*, deciding whether a user querying the word “apple” is looking for fruits or electronics). Similarly, advertisers today can define their targets by combining multiple keywords together, and even jointly with given demographics or location [10].
- Even if the target is specified by a unique term or topic, it is common for inputs to *overlap* (*e.g.*, two

emails may trigger the same output because they both contain the same important keyword). In fact, ad-targeting platforms like adwords often leverage relation between inputs, such as different keywords, as suggestions for how to expand the target of a given campaign. Some overlap is common even among a small number of inputs, as evidence of our experiments prove.

- Third, using multiple terms could be a way to *conceal* targeting. There are situations in which sensitive information such as race, gender or sexual orientation are forbidden to be used explicitly for targeting. But the nature of big data makes it possible to exploit other inputs that correlate with the intended target to obtain a similar effect. A collection of first names, for instance, was shown to be used as a proxy for race [20].

We now would like to highlight how those scenarios illustrate the limitations of current transparency tools. For example, if multiple inputs overlap and they all cause a personalized ads, XRay’s tests [11] conclude that no targeting takes place since all hypotheses associated with a single input yield poor likelihood. Ad-Fisher [6] is more robust as it can detect that an input plays a partial role and causes increased likelihood of an ad. But that knowledge applies only if the experimenter knows a priori which input to test and conduct an experiment about. For instance, in the concealed targeting case, monitoring the use of N inputs as potential proxy requires an exhaustive search with queries growing linearly with N . Sunlight [12] approximates targeting functions through linear combinations of inputs. In the disambiguation case, when different subsets of inputs cause targeting only when they are found together, Sunlight can only return a linear approximation of the targeting function. While extensions may be considered with an exhaustive list of subsets as new dimensions of an input space, this appears costly and has not been considered or validated.

In contrast to the above, our approach presents transparency algorithms directly designed for the complex case where the targeting function depends on multiple inputs. We would like, also in contrast to the above, for our algorithm to provide a formal theoretical guarantee under a small set of conditions. To do so, we first need to formally define the problem.

2.3 A formal model of complex targeting

We formalize the following intuitive problem: given a set of N inputs representing possible information items present in a user’s account (such as emails, searches, or other examples above), we wish to determine how they affect a web service’s choice of one particular output of interest (such as an ad or a recommendation).

Our main assumption is that the output is affected through an unknown targeting function f of the inputs, to be determined. The function f is defined separately for each output (i.e., it should be called f_{output} but we elide the subscript for simplicity). Formally, we define a combination \mathcal{C} of width r as a subset of r elements chosen among the N inputs. The targeting function f is a mapping from the set of all combinations to $\{0; 1\}$. We distinguish between two types of functions. Those that are untargeted will randomly assign their output with a probability that is independent of the tracked inputs. Those that are targeted will assign their output based on the presence of certain inputs in the account. For these targeted functions, we say an account \mathcal{C} is inside the target if $f(\mathcal{C}) = 1$, and outside the target otherwise.

Any targeting function $f \neq 0$ should satisfy two assumptions (a.k.a., axioms):

A1 **monotonicity**: $\mathcal{C} \subseteq \mathcal{C}' \implies f(\mathcal{C}) \leq f(\mathcal{C}')$.

A2 **input-sensitivity**: $\exists \mathcal{C}, \mathcal{C}'$ s.t. $f(\mathcal{C}) \neq f(\mathcal{C}')$.

Monotonicity simply reflects that an account with strictly more inputs that may indicate interest or hobbies should in theory be relevant to more ads, and never to less. Input sensitivity prevents the degenerate case where a targeting function is constant.

In practice we have no access to the targeting function directly. Instead we rely on experiments to observe how various inputs are treated. Intuitively, these experiments collect outputs from a set of accounts that contain subsets of the inputs and produce a set of noisy observations of f . For example, experiments could collect ads for accounts with different subsets of emails. More formally, the experimental infrastructure we assume is similar to a *noisy oracle* \mathcal{O} as defined in computational learning theory [2, 3, 19]. It can be interpreted as a collection of random variables $\mathcal{O}(\mathcal{C})$ associated with each combination of input \mathcal{C} .

We assume that this oracle \mathcal{O} (i.e., the family of random variables it defines) satisfies the following third axiom:

A3 **targeting lift**: There exist p_{in}, p_{out} such that

$$P[\mathcal{O}(\mathcal{C})=1|f(\mathcal{C})=1] = p_{in} > p_{out} = P[\mathcal{O}(\mathcal{C}')=1|f(\mathcal{C}')=0].$$

The targeting lift axiom properly states that f is related to the outcome we observe. It allows the variables to also depend on other factors: hidden inputs that are not in the set of N we study, external sources of randomness such as availability of ad-slot, competition. This axiom is also here to avoid the degenerate case where the values taken by $\mathcal{O}(\mathcal{C})$ for each \mathcal{C} are simply not affected by $f(\mathcal{C})$. We note that $p_{\text{in}}/p_{\text{out}} > 1$ and we call this ratio the targeting lift which denotes how much more likely are the ads to be shown for an account in-target. A function is more likely to be easy to detect or identify when the targeting lift is high, while for moderate or small targeting lift, the presence of ads in many accounts outside the target makes this more difficult.

2.3.1 Core family

A family S of size l is any collection of l distinct combinations. The width of the family is defined as the largest width of a combination it contains. Interestingly, there is a duality between families and targeting functions. On the one hand, one can define for any family S a targeting function $f : \mathcal{C} \rightarrow \max_{S \in \mathcal{S}} I_{\{S \subseteq \mathcal{C}\}}$ that takes value $f(\mathcal{C}) = 1$ whenever the subset \mathcal{C} contains at least one combination in S . On the other hand, we show (proof omitted due to space constraints) that the converse holds: for a targeting function f , there is a *unique* family with the property that it is of both minimum width and minimum size:

Lemma 1. *For each monotone, input-sensitive targeting function, f , there exists a unique family S satisfying:*

- (i) S has size l and width r and it explains f , i.e.,

$$f(\mathcal{C}) = 1 \text{ holds if and only if } \exists \mathcal{C}' \in S, \mathcal{C}' \subseteq \mathcal{C}.$$
- (ii) No family of size $l' < l$ explains f .
- (iii) No family of width $r' < r$ explains f .

For each targeting function, this family is the core family. Similarly, the size and the width of f is defined by its core family. Lemma 1 follows from the monotonicity axiom and does *not* hold for non-monotonic functions.

For example, with $N = 4$, let $S = \{\{1, 3\}, \{4\}\}$, and $S' = \{\{1, 2, 3\}, \{4\}, \{2, 4\}, \{1, 3\}\}$ and consider the function $f : \mathcal{C} \mapsto \max_{S \in \mathcal{S}'} I_{\{S \subseteq \mathcal{C}\}}$. We see that S' explains f by definition, and S also explains f . Intuitively, if S explains f , then if we were to observe that all combinations in S' receive an ad, this could in theory be explained by the hypothesis that the ad is targeted

at accounts which contain any of the combinations of inputs in S . Alternatively, if S does not explain S' , then it shows that S is not sufficient on its own to interpret this observation.

2.3.2 Targeting Detection, Targeting Identification

Under the assumptions above, an algorithm using m observations solves the targeting detection problem if it can correctly decide whether the output is the product of a targeted or untargeted function. Going further, an algorithm solves the targeting identification problem if it correctly returns the core family defining f . Naturally, both problems rely on random observations and hence our goal is to design algorithms whose detection/identification error is arbitrarily small for large N .

Let us quickly discuss the computational cost that we can hope to achieve. Since one should distinguish (at least) between N inputs, it seems that a minimum of $\Omega(\ln(N))$ binary observations are absolutely necessary at least for the identification. Similarly each input needs to be considered so a minimum of $\Omega(N)$ elementary computations are to be performed. This is hence what we assume and we aim at keeping it at this absolute minimum. Note that those condition imply that we cannot detect function of any arbitrary size and width (e.g., the function $f : \mathcal{C} \mapsto I_{\{|\mathcal{C}| \geq N/2\}}$ has a core family with a size growing *exponentially* in N). So our goal is to solve detection/identification given bounded size and width, a priori known¹, while carefully analyzing how those bounds affect the complexity and correctness of the algorithm. In fact, at the end of the next section we show that an algorithm solving identification exists that matches the computational lower bounds above, provided that the targeting lift is sufficiently strong.

3 Transparency Algorithms

In this section, we present our new transparency algorithm that we justify informally through an intuitive heuristic argument. We then present immediate empirical evidence, from a small scale experiment, that algorithms based on this heuristic are promising to ex-

¹ If the bounds are unknown, one could run algorithms with increasing values of maximum size and width and stop when targeting is correctly detected and/or identified.

pand the scope of transparency enhancing tools. All the claims we made here are formally proved in the next section, but can be found here without superfluous technicalities.

3.1 Heuristic for targeting detection

We assume that a collection of m accounts are created. For each of those accounts, every one of the N inputs we study is independently and randomly included with probability α . For a given advertisement we wish to study, we observe multiple accounts that receive it in our experiment, which we generally call the *active accounts*.

Let us first discuss how one could design a test to distinguish untargeted ads from ads with infinite targeting lift ($p_{in} > 0$ and $p_{out} = 0$). The essential argument follows from a simple yet powerful intuition: First, if on the one hand the ad is targeted with an infinite targeting lift, then any combinations of inputs \mathcal{C} for an active account must verify $f(\mathcal{C}) = 1$ and hence it must contain one combination from the core-family of f . In the later case, let us denote by l the size of the core family of f . Then there exists a subset \mathcal{C}' of l inputs among N such that each active account contains at least one element of \mathcal{C}' . Basically \mathcal{C}' is a small subset that intersects all set of inputs found in all active account. On the other hand, if we assume that none of the inputs that we included play any role in the appearance of this particular ad - *i.e.*, if the ad is not targeted - then the inputs of the active accounts are the sole results of our random choices. In that case, having a small subset that intersects all of those random choices is a very unlikely event.

This informally suggests that looking for a small subset intersecting the inputs of all (or most) active accounts in which the ads appear may potentially serve as a good test that the ad is targeted. Note that when it is the case, the inputs found in this small intersecting subset are primary candidates to be the ones defining the target. With this in mind, we introduce a simple heuristic algorithm for detection:

Note that, while this simple heuristic appears natural, it requires in the worst case to test all subsets of size up to l to find one intersecting a large number of the active accounts. This can be overly expensive, so one may decide to simplify this procedure by simply testing the subset returned by a simple greedy dynamic, which may often be a good approximation.

Finally, one may decide to build a simple identification algorithm which, instead of returning a simple binary value, actually returns the set \mathcal{C}' that was found

Heuristic Detection-x Algorithm

Known parameters: m , maximum size l , probability α .

Entries: a threshold parameter $x > \alpha$.

Result: 1 if the ad is targeted, 0 otherwise.

- 1: Create m accounts, include each input independently with probability α .
 - 2: Construct the family $S^{(ad)}$ containing all combinations of inputs in accounts seeing the ad.
 - 3: **for** \mathcal{C}' of l inputs **do**
 - 4: **if** $|\{\mathcal{C} \in S^{(ad)} | \mathcal{C} \cap \mathcal{C}' \neq \emptyset\}| > x \cdot |S^{(ad)}|$ **then**
 - 5: **return** 1.
 - 6: **end if**
 - 7: **end for**
 - 8: **return** 0.
-

Heuristic Greedy Detection-x Algorithm

Known parameters: m , maximum size l , probability α .

Entries: a threshold parameter $x > \alpha$.

Result: 1 if the ad is targeted, 0 otherwise.

- 1: Create m accounts, include each input independently with probability α .
 - 2: Construct the family $S^{(ad)}$ containing all combinations of inputs in accounts seeing the ad.
 - 3: $\mathcal{C}' = \emptyset$,
 - 4: **while** $|\mathcal{C}'| < l$ **do**
 - 5: $\mathcal{C}' \leftarrow \mathcal{C}' \cup \operatorname{argmax}_i |\{\mathcal{C} \in S^{(ad)} | \mathcal{C} \cap (\mathcal{C}' \cup \{i\})\}|$
 - 6: **if** $|\{\mathcal{C} \in S^{(ad)} | \mathcal{C} \cap \mathcal{C}' \neq \emptyset\}| > x \cdot |S^{(ad)}|$ **then**
 - 7: **return** 1.
 - 8: **end if**
 - 9: **end while**
 - 10: **return** 0.
-

in this procedure. One can hope under some conditions (for instance, if the width of the targeting is known to be 1) that this procedure returns the correct core family of f .

However naive these procedures may seem, we prove later that under some conditions they are essentially correct. Moreover they can be used effectively as building blocks for exact identification in the general case. But before proving that formally, we first test them in a practical setting.

3.2 Report from a small scale experiment

We run two small scale experiments to gain experience and inform our model of online advertising and the presence of ad targeting leveraging multiple inputs. The full content of those preliminary experiments will be made available. For each experiment, we first manually created one *master* Gmail account as well as 100 other *shadow* Gmail accounts. Inputs in these experi-

ments are emails, which are all included in the master account, and independently assigned to each shadow account with probability 1/2. In the first experiment, we used 20 emails on unrelated topics (*e.g.*, the purchase of a new car, TV, college applications). In the second experiment, after creating different accounts we used 50 emails, including 15 groups with 2,3 or 4 emails on the same topic (*e.g.*, two emails that are distinct but both relate to biking), and 12 separate unrelated emails. The outputs are ads shown to each account, collected by opening and refreshing multiple times each email.

To study targeting, we reproduced the state of the art Bayesian algorithm from [11] as well as a simple greedy algorithm. The former was previously designed and evaluated to accurately identify single input targeting. The latter is new and picks sequentially, among emails included in accounts seeing the ads, the email that is found in the most remaining number of accounts (this design is defined formally and analyzed later, see Section 4). In the latter, we conclude that targeting takes place when the fraction of accounts containing one email in the subset is sufficiently large. For simplicity we used here fixed thresholds: an ad is classified as targeted if either an email is found in at least 70% of the accounts with ads, or two emails are found so that either of them is included in at least 90% of those accounts. This last case in particular indicates that subset of two inputs are simultaneously targeted.

Our first experiment was a simple sanity check. Using the same ground truth as in [11], we consider 108 ads seen in at least 30 accounts including the master account, and found that for single input targeting, Bayesian has 95% precision and 86% recall, whereas the greedy algorithm finds 95% precision and 88% recall. As expected, emails are on unrelated topics, so considering subset of size 2 yields no specific improvement. In the second experiment, with more inputs and some overlap, results are quite different: Among 230 ads seen in at least 30 accounts including the master, we find that 81 (35%) are classified as targeted on a single input by Bayes, and are all found by the Greedy algorithm (by manually checking the content of the ads, as in previous ground truth, we found the vast majority of those, 85%, are correct classified). More interestingly, 79 ad (34%) are found *in addition* by the Greedy algorithm: another manual check based on the ad content proved that this classification is correct 70% of the time. We present some representative examples in Figure 1. Note that some multi-input targeting are direct consequences of the experiment design (*e.g.*, the left example with two emails with almost identical topic). This proves

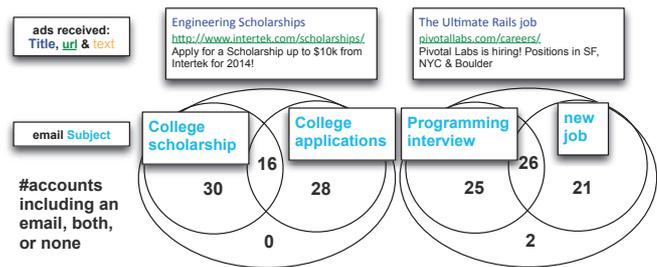


Fig. 1. Multi-input targeting in Exper. 2.

the limit of previous methods, since each email alone appears insufficient to explain targeting. Interestingly, some examples are less obvious (*e.g.*, the right example with subject “programming interview” and “new job” were not initially considered to be on the same topic, but happened to target similar ads). We believe this situation would be common in experiments with many emails. Note finally that, although we found quite a lot of multi-input targeting (at least 25% of the ads we analyzed), we cannot a priori conclude that this occurs with such frequency as it typically depends on many factors including the input sets. At best we can expect that it is not infrequent, and that our experience of exploiting simple heuristics can be made more systematic.

In this paper, motivated by this preliminary evidence, we wish to study the problem of multi-input targeting more formally. In particular, are heuristics like those we used above sound and complete to find this form of targeting when it takes place? Since multi-input targeting raises a combinatorial difficulty, how likely are the properties that make simple detection easy or hard? To answer those questions, we turn to a systematic mathematical analysis of a model reproducing the essential ingredients that we observed.

4 Proof of Correctness

Now that we found algorithms based on intersecting subsets to be intuitively justified, at least for detection, and practically useful, it remains to determine under which conditions those are rigorously correct. Moreover, beyond mere detection it remains to be shown that those test could be effectively used for exact identification. In this section, we do not only prove that, but also that this can be done while matching the computational lower bound of a linear number of operation.

4.1 Polynomial Targeting Detection

The first detection test we introduced runs naively using a polynomial number of operations to test for small subsets intersecting many active accounts. We first determine conditions under which it is provably correct.

Theorem 1. *Assuming $S^{(\text{core})}$ has size at most l and width at most r , and $\frac{p_{\text{out}}}{p_{\text{in}}} < \varphi_{l,r}(\alpha) = \frac{\alpha^r}{1-\alpha^r} \frac{(1-\alpha)^l}{1-(1-\alpha)^l}$, then there exists $x > \alpha$ and $C > 0$ satisfying:*

For any $\varepsilon > 0$, if $m \geq C \cdot (\ln(N) + \ln(1/\varepsilon))$, Heuristic detection- x is correct with probability $(1 - \frac{\varepsilon}{N})$.

Proof. A subset of inputs \mathcal{C} is an x -intersecting subset of a family S (for $0 \leq x \leq 1$) if at least a fraction x of the subsets in S intersect \mathcal{C} (i.e., each contains an input chosen in \mathcal{C}):

$$|\{ S \in S \mid \exists D_i \in \mathcal{C}, D_i \in S \}| \geq x \cdot |S|.$$

Similarly, we say that S' is an x -intersecting family of a family S if at least a fraction x of the subsets contained in S contain a combination chosen in S' :

$$|\{ S \in S \mid \exists \mathcal{C} \in S', \mathcal{C} \subseteq S \}| \geq x \cdot |S|.$$

If S' satisfies this property, one can choose for each combination of S' one given input in this combination. Those inputs together form an x -intersecting subset, proving:

Fact 1. *Let S' be an x -intersecting family of S , there exists \mathcal{C} an x -intersecting subset of S with $|\mathcal{C}| \leq |S'|$.*

Given a particular advertisement an account is said active if it has received the advertisement during the experiment. We denote by $S^{(\text{ad})}$ the active family that is built after considering all active accounts and adding for each of them the combination of its inputs to $S^{(\text{ad})}$. Finally, as seen in the previous section, if an advertisement is targeted it admits a unique core family. We then denote it by $S^{(\text{core})}$ and observe that it is never empty. By convention the case where the advertisement is non-targeted is denoted by $S^{(\text{core})} = \emptyset$.

Our first preliminary result a property of random subsets and ensures that our test is *sound*, i.e., it almost never returns targeting when it's not the case.

- A random Bernoulli subset, denoted by $B(N, \alpha)$, is a subset such that any of N elements is contained with probability α independently of all others.
- A random Bernoulli family of size m is a collection of m independent Bernoulli subsets.

Since Bernoulli subsets and families derive from many independent decisions to include or not a single element,

we can use concentration inequalities on the distribution of sum of binary variables. Indeed such variable remains close to its expectation (i.e., up to a constant multiplicative factor) except on an event of polynomially small probability. This holds as soon as its expectation is at least logarithmic and it implies:

Lemma 2. *Let $1 > x > 0$, $l \in \mathbb{N}$, $\alpha < 1 - (1-x)^{\frac{1}{l}}$, and a Bernoulli family $B_1(N, \alpha), B_2(N, \alpha), \dots, B_m(N, \alpha)$. There exists $C > 0$ such that for any $\varepsilon > 0$ and polynomial P , if $m \geq C \cdot (l \ln(N) + \ln P(N) + \ln(1/\varepsilon))$, then with probability $(1 - \varepsilon/P(N))$ no x -intersecting subset exists of size l for this Bernoulli family.*

Lemma 3. *Assume $x, \alpha > 0$, and $\frac{p_{\text{out}}}{p_{\text{in}}} < \frac{1-x}{x} \frac{\alpha^r}{1-\alpha^r}$, where no combination in the core family has width larger than r .*

There exists $C > 0$ such that for any $\varepsilon > 0$ and polynomial P , for any combination \mathcal{C} , whenever we have

$$m \geq \alpha^{-|\mathcal{C}|} \cdot C \cdot (\ln P(N) + \ln(1/\varepsilon)).$$

then with probability $(1 - \varepsilon/P(N))$ the following holds: among accounts containing \mathcal{C} and seeing the ad, at least a fraction x of them is within the targeting scope $S^{(\text{in})}$,

$$\text{i.e., } \frac{|\{ S \in S^{(\text{ad}, \text{in})} \mid \mathcal{C} \subseteq S \}|}{|\{ S \in S^{(\text{ad})} \mid \mathcal{C} \subseteq S \}|} \geq x,$$

where $S^{(\text{in})}$ denotes the family of combinations \mathcal{C} such that $f(\mathcal{C}) = 1$ and $S^{(\text{ad}, \text{in})} = S^{(\text{ad})} \cap S^{(\text{in})}$.

As a consequence, we see that the targeting lift does affect the inputs found in $S^{(\text{ad})}$ as it is responsible for a fraction of them. Under the condition of this lemma, since we saw that a 1 -intersecting set exists for $S^{(\text{ad}, \text{in})} = S^{(\text{ad})} \cap S^{(\text{in})}$ that is constructed using $S^{(\text{core})}$, it is also an x -intersecting subset of $S^{(\text{ad})}$.

Based on the above results, we can expect that with high probability two claims hold when m has order $O(\ln(N))$:

- **Soundness:** If targeting does not occur, i.e., if we have $S^{(\text{core})} = \emptyset$, then $S^{(\text{ad})}$ has no x -intersecting subset of size l .
- **Completeness:** If targeting occurs, i.e., if we have $S^{(\text{core})} \neq \emptyset$, an x -intersecting subset for $S^{(\text{ad})}$ of size $|S^{(\text{core})}|$ exists.

However, it is important that we prove that the same value of x is used simultaneously for both claims, which is why a careful analysis is required.

In order to apply both claims for the same x , we need that $\alpha < 1 - (1-x)^{1/l}$ and $p_{\text{out}}/p_{\text{in}} < \frac{1-x}{x} \frac{\alpha^r}{1-\alpha^r}$

are verified. Fortunately it is easy to show that such x exists as soon as $\frac{p_{\text{out}}}{p_{\text{in}}} < \varphi_{l,r}(\alpha) = \frac{\alpha^r}{1-\alpha^r} \frac{(1-\alpha)^l}{1-(1-\alpha)^l}$ which completes the proof. \square

4.2 Detection with Linear Cost

In this section, we prove that, under stricter condition on the ratio $p_{\text{out}}/p_{\text{in}}$, it is possible to prove the correctness of the approximate detection test that uses a greedy heuristic. This shows that targeting detection is feasible in linear cost. We first introduce

$$\hat{\varphi}_{l,r}^{(\text{gr})}(\alpha) = \frac{(1-\alpha)^l - \left(1 - \frac{1}{l}\right)^l}{1 - (1-\alpha)^l} \frac{\alpha^r}{1-\alpha^r}. \quad (1)$$

Theorem 2. *Assuming $S^{(\text{core})}$ has size at most l and width at most r , and $\frac{p_{\text{out}}}{p_{\text{in}}} < \hat{\varphi}_{l,r}^{(\text{gr})}(\alpha)$, then there exists $x > \alpha$ and $C > 0$ satisfying:*

For any $\varepsilon > 0$, if $m \geq C \cdot (\ln(N) + \ln(1/\varepsilon))$, Heuristic Greedy detection- x is correct with probability $(1 - \frac{\varepsilon}{N})$.

Proof. Let us first formalize our greedy building block. For any family S , we introduce the function that counts for any combination \mathcal{S} how many elements of S it intersects: $g^S : \mathcal{S} \mapsto |\{ \mathcal{S}' \in S \mid \mathcal{S} \cap \mathcal{S}' \neq \emptyset \}|$. Note that \mathcal{S} is an x -intersecting subset if and only if $g^S(\mathcal{S}) \geq x \cdot |\mathcal{S}|$. Our tests relate to maxima of g^S under constraints.

This function is submodular, non-decreasing, non-negative. Hence one way to find \mathcal{S} maximizing its value is to follow an iterative greedy algorithm, starting with $\mathcal{S}_0 = \emptyset$ that constructs $\mathcal{S}_1, \dots, \mathcal{S}_l$ as follows:

$$\mathcal{S}_{i+1} := \operatorname{argmax} \left\{ g^S(\mathcal{S}') \mid \mathcal{S}_i \subseteq \mathcal{S}', |\mathcal{S}'| = |\mathcal{S}_i| + 1 \right\}.$$

Using standard argument on maximum of such submodular functions, one can show that for *any* subset \mathcal{S} of size l , we have $g^S(\mathcal{S}_l) \geq (1 - (1 - \frac{1}{l})^{1/l}) g^S(\mathcal{S})$. This immediately implies that if an x -intersecting subset is to exist (which we can show when targeting occurs with high probability), then the greedy heuristic necessarily returns an $(1 - (1 - \frac{1}{l})^{1/l}) \cdot x$ intersecting subset.

Following the same argument as the proof of the detection test in the previous section, this implies that the greedy algorithm offers a detection test that asymptotically correct under a more restricted conditions on the value of $p_{\text{out}}/p_{\text{in}}$. \square

Note that the condition in the theorem is stronger than $p_{\text{out}}/p_{\text{in}} < \varphi_{l,r}(\alpha)$ as we have for all $l > 1$:

$$\forall \alpha > 0, \frac{(1-\alpha)^l - \frac{1}{e}}{(1-\alpha)^l} \cdot \varphi_{l,r}(\alpha) < \hat{\varphi}_{l,r}^{(\text{gr})}(\alpha) < \varphi_{l,r}(\alpha). \quad (2)$$

This implies (see §5) that some targeting found by the exact exhaustive algorithm is missed by the greedy version.

4.3 From Detection to Identification

Here we show rigorously that the previous heuristic of intersecting subsets can be leveraged to provide exact identification of the targeting function.

Theorem 3. *Assuming $S^{(\text{core})}$ has size at most l and width at most r , and $\frac{p_{\text{out}}}{p_{\text{in}}} < \varphi_{l,r}(\alpha)$, then there exists $x > \alpha$ and $C > 0$ satisfying:*

For any $\varepsilon > 0$, if $m \geq C \cdot (\ln(N) + \ln(1/\varepsilon))$, an identification algorithm using no more than $O(N^{l+r} \ln(N))$ operations is correct with probability $(1 - \frac{\varepsilon}{N})$.

We prove this result in the rest of this section. Relating back to the requirements formulated for web transparency tools in, our results show that targeting on bounded-size combinations can be detected and identified *accurately* with very few resources (*scalably*). Our entire formalism is kept on purpose generic and its treatment of noisy combined targeting makes it more *broadly applicable* than prior systems.

Previously in this section, we designed a provably approximately correct *targeting detection* test that produces a provable certificate: an x -intersecting subset obtained under specific conditions. To go beyond mere detection, one would like to identify exactly *which* inputs are being targeted, a much more difficult task. Those are *a priori* related to inputs found in the intersecting subset. However, this intersecting subset is not unique, it may contain only a subset on relevant inputs of the core family.

Key observation: How targeting behaves “beyond” some combination We first answer to the following key question: For a combination \mathcal{C} , what can be said about the set of accounts that contain it and those among them that are active, $\{\mathcal{S} \in S^{(\text{ad})} \mid \mathcal{C} \subseteq \mathcal{S}\}$? By definition, all of those subsets contain all inputs in \mathcal{C} so we are interested in understanding how other inputs affect them. This is where exactly two cases emerge: Firstly, if we assume that \mathcal{C} does contain a combination of the core, it automatically implies that independently of any other inputs, they all receive the ad with the same probability. Secondly, if we assume on the other hand the opposite, then among all accounts including \mathcal{C} , there will be specific sets of inputs that may complete a combination from the core family and hence be

targeted more heavily than others. This latter case resembles the situation of a targeting lift which can be detected. The former case resembles a situation where ads appear randomly. We can therefore design a new test as follows, that is sound and complete to determine in which case we are.

Definitions and main building block We remind that we denote $S^{(\text{in})}$ the set of combinations \mathcal{C} that satisfy $f(\mathcal{C}) = 1$. One can observe these are all supersets of combinations in the core family $S^{(\text{core})}$. Our test will determine whether a given combination \mathcal{C} belongs to this family. Indeed, we now show how to leverage the above observations to determine whether $\mathcal{C} \in S^{(\text{in})}$ using intersecting subsets. As it will be useful later to generalize this test to apply to intersecting subsets of smaller size $k \leq l$, we directly write the most general case. Note that this requires a new condition:

$$\hat{\varphi}_{l,r}^{(k)}(\alpha) = \frac{1 - \frac{l}{k}(1 - (1 - \alpha)^l)}{\frac{l}{k}(1 - (1 - \alpha)^l)} \frac{(1 - (1 - \alpha)^{l/k})^r}{1 - (1 - (1 - \alpha)^{l/k})^r}. \quad (3)$$

When k is chosen equal to l , this condition is equivalent to previous one as the one above. The case where $k < l$, also proved here, will be of importance to design a linear identification algorithms in the next section.

We introduce the following key proposition

Proposition 1. *Assuming $S^{(\text{core})}$ has size at most l and width at most r , $\frac{p_{\text{out}}}{p_{\text{in}}} < \hat{\varphi}_{l,r}^{(k)}(\alpha)$, for $k \leq l$, then there are $x, C > 0$ satisfying:*

For any $\varepsilon > 0$, polynomial P , and combination \mathcal{C} , when $m \geq \alpha^{-|\mathcal{C}|} \cdot C \cdot (\ln(N) + \ln P(N) + \ln(1/\varepsilon))$, then with probability $(1 - \varepsilon/P(N))$ exactly one of the following claims holds:

- (i) \mathcal{C} contains a core combination, i.e., it is in $S^{(\text{in})}$.
- (ii) an x -intersecting subset of size k exists for

$$\Delta^{(\text{ad})}(\mathcal{C}) = \left\{ \mathcal{S} \cap \bar{\mathcal{C}} \mid \mathcal{S} \in S^{(\text{ad})}, \mathcal{C} \subseteq \mathcal{S} \right\}.$$

Identification using exhaustive local search Assume we are equipped with the test of Prop 1 and that the targeting function to detect has width at most r . A first (costly) algorithm to compute the core family searches for the results of all tests and in the worst case it examines all the $O(N^r)$ possible combinations.

This algorithm maintains a current core $S^{(\text{core})}$ that is initially empty. At each step $i \geq 1$ it considers all combinations of size i that are not strict supersets of a combination already found in $S^{(\text{core})}$. In particular, note that it considers all the possible N singletons for $i = 1$. For each combination \mathcal{C} considered at step i (i.e., $|\mathcal{C}| = i$), we apply to \mathcal{C} the recognition test from Prop 1 (using

$k = l$). If $\mathcal{C} \in S^{(\text{in})}$, then \mathcal{C} is indeed in the core of the function by inclusion wise minimality and so, we put the combination in $S^{(\text{core})}$. Otherwise we drop it. When the algorithm stops after r steps, all core combinations have been necessarily identified.

Some final remarks are required to prove the correctness of the above algorithm: First, to perform the recognition test of Prop 1 (using $k = l$), we need a bound on the size of $S^{(\text{core})}$ and that the bound on the targeting lift is satisfied. We also need to observe that, since all tested combinations have width at most r , the value of $\alpha^{-|\mathcal{C}|}$ that appears in the bound for m indeed remains bounded. Finally, since we conduct $O(N^r)$ tests, we need to ensure that *none* of them fail. Using the fact that by choosing C large enough, Prop 1 holds for any polynomial P , we can choose $P(N) = N^r$ and applies the union bound to all tests. As a final remark, since each test boils down to identifying an intersecting subset of size l among m accounts, it uses at most $O(N^l m) = O(N^l \ln(N))$ operations so, the total number of operations in the algorithm is $O(N^{l+r} \ln(N))$.

4.4 Identification with linear cost

Getting rid of the exhaustive local search step in the identification of the targeting function is much more difficult. This requires each test to be computationally cheaper and more importantly to conduct a much smaller number of them. However, surprisingly, it can be done using more properties of intersecting subsets and a significantly more elaborate algorithm.

The gist: getting more from each test Given \mathcal{C} , §3 presents a test to determine if it belongs to the target $S^{(\text{in})}$, but more importantly if this answer is negative it produces a certificate which is an x -intersecting subset, or even the family of all such subsets. We first extract more information from this output.

Lemma 4. *Assuming $S^{(\text{core})}$ has size at most l and width at most r , that $k \leq l$ and $\frac{p_{\text{out}}}{p_{\text{in}}} < \sup_{\alpha \in]0, \frac{1}{2}[} \hat{\varphi}_{l,r}^{(k)}(\alpha)$, then there are $x, \alpha, C > 0$ satisfying:*

For any $\varepsilon > 0$, polynomial P , and combination \mathcal{C} , when $m \geq \alpha^{-|\mathcal{C}|} \cdot C \cdot (\ln(N) + \ln P(N) + \ln(1/\varepsilon))$, then with probability $(1 - \varepsilon/P(N))$ the following two claims hold:

- (i) All combinations in $S_{x,k}$ intersect $\bigcup_{\mathcal{S} \in S^{(\text{core})}} \mathcal{S}$.
- (ii) $\mathcal{C} \cup \bigcup_{\mathcal{S} \in S_{x,k}} \mathcal{S}$ is empty or contains a core comb.

$S_{x,k} = \left\{ \mathcal{S} \mid \mathcal{S} \text{ } x\text{-intersecting for } \Delta^{(ad)}(\mathcal{C}), |\mathcal{S}| \leq k \right\}$. *GSI algorithm identifies the core family with probability $1 - \varepsilon/P(N)$*

For the rest of this section, we will apply this lemma with $k = 1$; it then has a quasi-linear cost. Under this condition there is targeting if, and only if, there is an x -intersecting subset of size 1. Fact (i) above implies that all such x -intersecting subsets are included in $\bigcup_{\mathcal{S} \in S^{(core)}} \mathcal{S}$, hence we have at most lr of them.

A recursive identification test using eliminations. It seems *a priori* promising to operate by elimination for the following reason: assuming \mathcal{C} contains a core combination, we can remove elements from \mathcal{C} one by one and only keep those whose removal changes the outcome of the test from Prop. 1. By inclusion wise minimality, the resulting combination belongs to the core family. Applying Lemma 4 to the empty combination provides a first detection test of targeting (*i.e.*, if this one fails, the algorithm returns $S^{(core)} = \emptyset$) and a superset of a core combination (Line 1). By the elimination process aforementioned, we obtain a core combination (Line 3-7). We then recursively apply the same algorithm to multiple scenarios where we carefully remove exactly one input from the first combination (Line 9-12). This ensures that future combinations obtained by the same methods are different.

Generalized-Set-Intersection(GSI) Algorithm

Entries: a family $S^{(ad)}$; and a threshold parameter x .

Result: The core-explaining family $S^{(core)}$.

```

1:  $S^{(core)} \leftarrow \{\}; \mathcal{S} \leftarrow \{D_i \mid \frac{|\{A_j \in S^{(ad)} \mid D_i \in A_j\}|}{|S^{(ad)}|} \geq x\}$ ;
2: if  $\mathcal{S} \neq \emptyset$  then
3:   for all  $D_i \in \mathcal{S}$ ,  $\hat{S} = S \setminus D_i$  do
4:     if  $\{D'_i \notin \hat{S} \mid \frac{|\{A_j \in S^{(ad)} \mid \hat{S} \cup \{D'_i\} \subseteq A_j\}|}{|\{A_j \in S^{(ad)} \mid \hat{S} \subseteq A_j\}|} \geq x\} = \emptyset$  then
5:        $S \leftarrow S \setminus D_i$ .
6:     end if
7:   end for
8:    $S^{(core)} \leftarrow S^{(core)} \cup \{\mathcal{S}\}$ .
9:   for all  $D_i \in \mathcal{S}$  do
10:     $S_i \leftarrow \text{GSI}(S^{(ad)} \setminus \{A_j \mid D_i \in A_j\}, x)$ .
11:     $S^{(core)} \leftarrow S^{(core)} \cup S_i$ .
12:   end for
13: end if

```

Proposition 2. *Assume $S^{(core)}$ has size at most l and width at most r , and $\frac{p_{out}}{p_{in}} < \frac{1-lx}{l} \frac{x^{r-1}}{1-x^r}$, for some $0 < x < 1/l$.*

There exist α, C satisfying $\forall \varepsilon > 0$, for all polynomial P , whenever $m \geq C \cdot (\ln P(N) + \ln \frac{1}{\varepsilon})$, the

Analysis of complexity It is straightforward to prove that GSI terminates, because the number of data inputs decreases at each recursive call to the algorithm. To upper-bound its time complexity in the worst-case, we introduce the so-called “relevant set” \mathcal{I} . Its definition is inductive, following the trace of some execution of the algorithm. Namely, let \mathcal{S} be the union of all x -intersecting subsets of size 1. If $\mathcal{S} = \emptyset$, then we set $\mathcal{I} = \emptyset$. Else, for each recursive call $\text{GSI}(\mathcal{A} \setminus \{A_j \mid D_i \in A_j\}, x)$ of the algorithm, we name \mathcal{I}_i its relevant set and we set $\mathcal{I} = \mathcal{C} \cup (\bigcup_i \mathcal{I}_i)$. Note that we have by construction $\forall i, |\mathcal{I}_i| \leq |\mathcal{I}| - 1$.

We state below (proof omitted) that GSI is Fixed-Parameter Tractable (FPT) when the size $|\mathcal{I}|$ of the relevant set is fixed.

Lemma 5. *GSI can be implemented to run in a time bounded by $O(|\mathcal{I}|! \cdot N \cdot |S^{(ad)}|)$.*

Proposition 3. *Under the assumptions of Prop. 2 GSI terminates in $O((lr)! \cdot Nm)$ -time with probability $1 - \varepsilon/P(N)$.*

5 Impact of Targeting Lift

5.1 Targeting lift defines in theory a limit to web transparency

As seen above, the tests we design can dramatically expand the scope of web transparency tools. We proved that observations from inputs and outputs can reveal how a targeting function affects what is shown to a population based on combination of multiple inputs. Since the ability to discriminate using such functions is fundamental to Big Data, and since our tests made minimal assumptions, our results can be readily used in many more applications to come beyond ad-targeting, our primary motivation.

This poses an important question: Is there a fundamental limit to web transparency? That would be a situation in which all tests so far fail no matter how parameters such as x, α and C are chosen. We show it is the case, and we compute the exact conditions under which all known tests fail. We also measure the detection power of other simpler tests to understand how they affect those limitations.

We showed tests always exists to successfully detect and identify a targeting of size l and order r such that

$$\frac{p_{\text{out}}}{p_{\text{in}}} \leq \sup_{\alpha \in]0;1[} \varphi_{l,r}(\alpha) = \sup_{\alpha \in]0;1[} \frac{\alpha^r}{1 - \alpha^r} \frac{(1 - \alpha)^l}{1 - (1 - \alpha)^l}.$$

This condition leads to a closed form.

Lemma 6. *Let $M_{l,r} = \sup_{\alpha \in]0;1[} \varphi_{l,r}(\alpha)$, we have*

$$\begin{cases} \text{if } l = 1, & M_{1,r} = 1/r, \\ \text{if } r = 1, & M_{l,1} = 1/l, \\ \text{for all } r, l, & \frac{1}{(2^{\max(l,r)} - 1)^2} \leq M_{l,r} \leq \frac{1}{(2^{\min(l,r)} - 1)^2}, \\ \text{for all } r, l, & M_{l,r} = M_{r,l}, \\ \text{if } r = l = n, & M_{n,n} = 1/(2^n - 1)^2. \end{cases}$$

If $r > 1, l > 1$, $M_{l,r} = \frac{(\alpha^*)^r}{1 - (\alpha^*)^r} \frac{(1 - \alpha^*)^l}{1 - (1 - \alpha^*)^l}$ where α^* is the only solution in $]0;1[$ of

$$r\alpha^{r+1} - l(1 - \alpha)^{l+1} - (r + l)\alpha + r = 0.$$

Consequence on the concealment of targeting

From our analysis important observations arise: First, when a single input is used for targeting *i.e.*, $l = 1, r = 1$, we have $M_{l,r} = 1$ and the condition above is always verified except for the trivial case where $p_{\text{out}} = p_{\text{in}}$, impossible by definition. This confirms that *single input targeting never can stay hidden from all transparency tests*. However, all other forms of targeting that leverage multiple inputs stand sharply in contrast. Indeed any function whose core family is not a single input exhibits a non-trivial *undetected targeting lift*: it can in practice discriminate users to some extent without being detected by any of the tests known. Note that, in practice, a web transparency tool can restrict itself to test a few assumptions on carefully selected inputs, and it may in some cases detect their use, but it becomes a very different task as no method is shown to work *in general*.

It is particularly informative to see how this *undetected targeting lift* grows with the complexity of the formula used. Figure 2 presents the value of the function $\varphi_{l,r}$ that defines it (it is drawn up to a change of variable to make it easier to read). As proved in the lemma and shown in the Figure, if the targeting solely uses disjunction (resp. conjunction) of inputs, *i.e.*, $r = 1$ (resp. $l = 1$) as shown in Figure 2 (left), the targeting lift behaves as $1/l$ and hence it is polynomial. This polynomial expansion remains if l grows while r remains small. The proof is omitted but Figure 2 (middle) presents an example. In contrast, when l and r grow simultaneously, *e.g.*, if they are equal as shown in Figure 2 (right) one

can show that targeting lift can be *exponential* and *undetected*. While this demonstrate the hardness of web-transparency, we note that such forms of complex targeting combining so many inputs to decide may be relatively rare in practice.

Limits of web transparency for simpler tests

We present in Figure 3 along with other conditions the condition that allows our linear identification algorithm to be correct, *i.e.*, $\hat{\varphi}_{l,r}^{(k)}$ with $k = 1$, for the case $l = r = 2$. It illustrates some important observations: First, the conditions to apply the linear identification algorithm are more restricted (In this case, the targeting lift must be approximately 10 times larger). This points to another qualitative interesting insight: There exist instances of targeting for which a linear algorithm may be able to detect them without necessarily being able to completely identify them.

5.2 Simulations Targeting lift in large scale systems

Synthetic Experiments Both the Bayesian algorithm from [11] and the greedy algorithms here have several parameters upon which the accuracy of their predictions depends. While the XRay paper presents sufficient conditions for the success of the bayesian algorithm, the performance of the algorithm outside these conditions is not well understood. For the greedy algorithm, no such conditions are known. While there are known optimization methods for computing the ideal parameters for these algorithms, these methods rely critically on the availability of ground-truth data. For many important systems, this data is unlikely to ever be made available to anybody outside the proprietor. Thus, in the interest of measuring the boundaries and limiting conditions for these algorithms, as well as determining their efficacy beyond the theoretical guarantees provided in their definitions, we have measured their accuracy through a series of experiments that simulate our model of online interactions. These findings are intended to provide guidelines for what can and cannot be done, and to inform on the cost associated to particular endeavors.

Experimental Setup We will analyze four algorithms. The *Exhaustive Detection Algorithm*, the *Exhaustive Identification Algorithm*, and a pair of *Greedy Detection/Identification Algorithms*. The core components of the exhaustive algorithms are Proposition 1 and Proposition 2 from [11] respectively. All of these algorithms can roughly be divided into a *collection phase* and a *computation phase*.

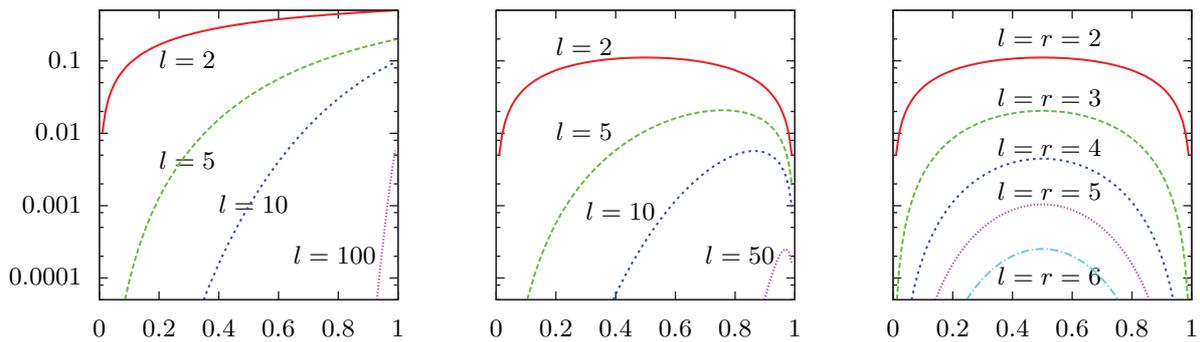


Fig. 2. Value of $\varphi_{l,r}$ as a function of $1 - \alpha$ for $r = 1$ (left), $r = 2$ (middle) and $r = l$ (right).

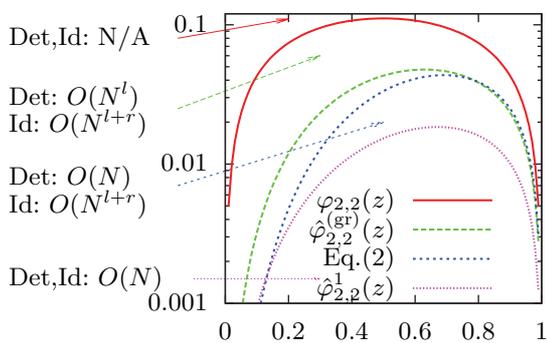


Fig. 3. Comparing the detection/identification region of various algorithms for $l = r = 2$.

During the collection phase, we take a pool of N emails and create m accounts. Each account contains each email with independent probability α . These accounts are then exposed to the targeting system and allowed to collect ads. In lieu of an actual targeting system, our synthetic experiments simulate the ad-targeting algorithm with a boolean stochastic targeting function T_C . This function has an underlying core family C . In the noiseless case, $T_C(A)$ is true if and only if A matches C . In order to simulate untargeted ads, we have a special function T_\emptyset which randomly targets an account with fixed probability β . For our synthetic experiments, we have used $\beta = 0.5$.

During the computation phase, our algorithm no longer has access to T_C , and only sees body of ads which have been targeted by T_C . This mirrors the state of available information that is available to the researcher when attempting to test the targeting properties of a particular system. In this phase, an identification algorithm is required to produce a core family C' that matches C with high probability. For detection, we are merely required to produce a bit which matches, with high probability, the truth value of “ $C \neq \emptyset$ ”.

Our model admits two noise rates, each affecting the negative and positive instances of T_C , respectively. If A matches C , then $T_C(A)$ will be positive with probability p_{in} , otherwise, $T_C(A)$ will be positive with probability p_{out} . We will generally be interested in *symmetric noise*, where there is a certain noise value η , such that $p_{out} = \eta$ and $p_{in} = 1 - \eta$. For this type of noise, the *targeting lift* τ is given by $\tau = p_{out}/p_{in} = \eta/(1 - \eta)$.

To measure the success of an algorithm, we independently evaluate its performance in both targeted and untargeted scenarios, producing measures for targeted accuracy a_t and untargeted accuracy a_u . In any application, the actual accuracy of the algorithm will be given by $a_{real} = pa_t + (1 - p)a_u$, where p is the prevalence of targeted ads. However, since this parameter p is unknown to us a priori, and indeed may differ for each ad, we will assume it is chosen by an adversary who wishes to minimize our total accuracy. This provides us with a unified accuracy measure which we call min-accuracy a_m , which provides a lower bound for a_{real} and is equivalent to $a_m = \min\{a_t, a_u\}$.

For each algorithm under consideration, our primary interest is to find the maximum value of τ that can be tolerated, such that with a proper setting of the other parameters, we can still achieve a min-accuracy of 99%. We know that the optimal value for α is the one that guarantees $P[T_C(A)] = 0.5$ over a random choice of A . With sufficient information about the targeting function, this value can be determined algebraically. The proper value for x can then be found through a simple optimization procedure. First note that for all algorithms under consideration, a_t is monotonically decreasing in x , while a_u is monotonically increasing with x . Further note that a_u is totally independent of τ , while a_t is monotonically decreasing in τ . Thus, it suffices to find the smallest value of x such that $a_u > 99\%$ when $\tau = 0$, which can be achieved via binary search. We may

then use this value of x and find the maximum value of τ that will still guarantee $a_t > 99\%$, also via binary search. Per our prior observations, we know that this combination of α , x and τ will achieve $a_m > 99\%$.

Experimental Results

In Fig. 4(left) we may observe that the greedy algorithm is noise-tolerant, even for relatively high levels of τ . Noise-tolerance decays almost linearly, even as N increases exponentially, indicating a logarithmic dependence on N . Furthermore, we may note that while for $L = 1$ the difficulty of identification and detection are comparable, as L increases the cost of precise identification of the targeting function becomes steeper, and the decrease in noise-tolerance for increasing values of N is more pronounced. This suggests for very complex targeting, only detection is possible when noise levels are expected to be high and accounts are limited or the inputs under consideration are abundant.

Given the high cost associated with obtaining new accounts, and the deliberate barriers placed by content providers on automation of this task, the number of available accounts is often the limiting reagent in targeting inversion problems. Observing Fig. 4(middle), we note that accuracy can be boosted significantly by allowing ourselves access to a greater number of accounts. This allows the greedy algorithm to attain a noise-tolerance level much greater than the theoretical bound. Moreover, as we can see in Fig. 4(right), the greedy algorithm can very nearly approximate the noise tolerance of the optimal exhaustive search procedure.

These results highlight the importance of the greedy algorithm, which can be seen as applying the well-known greedy heuristic to solve a version of the optimal set cover problem. In particular, the algorithm achieves high noise-tolerance, approximates the exhaustive algorithm when l and r are small, and reduces the computations from an polynomial function using l and r to a linear one. For extremely large values of N , such as those often encountered in practice, the greedy algorithm is the only computationally tractable option.

5.3 An application: The price of opacity

We specified the theoretical limits to targeting detection. However, to complete the picture one must wonder whether it is economically feasible for an advertiser to conceal her targeting. To answer this question, we introduce a simplistic ad purchase model, in which the advertiser has to pay a fixed cost $C > 0$ for every account receiving the ad and she gains on average a fixed

positive revenue $R > C$ for each active account *within her scope* which encompasses a fraction q of the users. We assume that through various bids, this advertiser controls the values of $p_{\text{in}}, p_{\text{out}}$ that users experience.

In case the advertiser is fully transparent, she can minimize the cost of her targeting campaign simply by setting p_{out} to 0. She would then potentially earn $p_{\text{in}}q(R - C)$ during that campaign. However, if she wishes to keep her advertising method concealed from transparency tools, she needs to increase p_{out} at least to $p_{\text{in}} \cdot M_{l,r}$. We immediately deduce that her cost increases by $p_{\text{out}}(1 - q)C = p_{\text{in}} \cdot M_{l,r}(1 - q)C$. It is hence multiplied by a factor $(1 + M_{l,r} \frac{1-q}{q})$, which denotes the *Price of Opacity*. As an example, for $l = 2, r = 2$ and $q = 0.05$, an advertiser already spent *three times* more on advertising just to avoid detection, and this price is particularly damaging when the target is small. To remain opaque she loses at least a fraction $\frac{1-q}{q} \frac{C}{R-C} \cdot M_{l,r}$ of her revenue. A necessary condition for the advertiser to remain profitable is that $\frac{1-q}{q} \frac{C}{R-C} \cdot M_{l,r} < 1$. Note that, with the exception of l and r , these parameters are not set by the advertisers.

In face of the above result, an advertiser might be tempted to decrease her targeting scope artificially so that a small value of p_{out} is enough to avoid detection. Formally, such an attack consists in replacing the core family of her targeting function by another one with larger dimensions $l' > l, r' > r$. Our preliminary analysis (omitted due to lack of space) shows that the decrease in p_{out} is offset by the opportunity loss of being too restricted in the definition of the target. It can quickly become economically unviable, especially when fixed costs are present. The analysis of this attack and how to best cope with it remains beyond the scope of our work.

6 Related Work

Our formulation resembles identification of monotone DNF formulae from an oracle (*e.g.*, see [2, 3, 19]) and more generally the theory of formal learning. However, it is quite distinct for three main reasons: First, and most importantly, we only care about *proper exact learning*, by which an algorithm is required not to produce a function closely approximating the monotone formula (as in [5, 19]), but to compute an exact representation of the formula itself. Second, previous proper exact learning techniques make adaptive queries [2, 3], which is impractical as typically web transparency in-

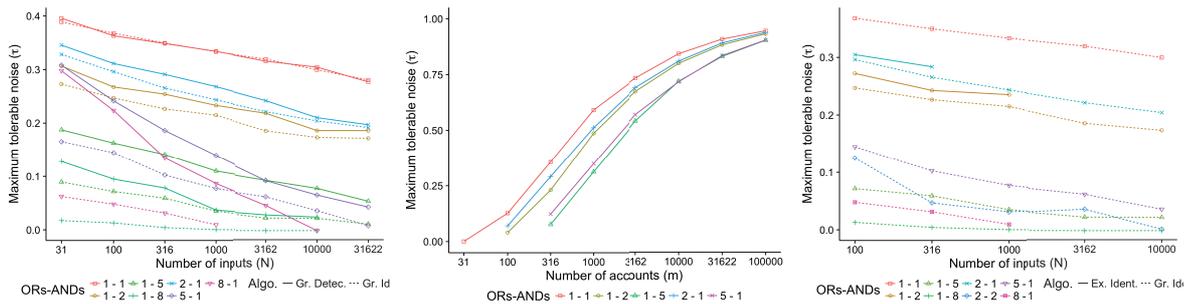


Fig. 4. Greedy Algorithms: Identification vs. Detection (left) using increasing accounts (middle) compares with exhaustive (right).

volves a large data collection that take significant time. Our algorithm in contrast makes non-adaptive queries: the set of combination to test is fixed and data are collected prior to any steps of the algorithm. Finally, since each query is resource intensive, we require the number of queries to grows as a logarithm of the number of variables. We are not aware of any previous algorithm – or extension of previously known techniques – that can fit all of these stringent conditions.

Another parameter of learning model often introduced is the VC dimension. However, in the case we study, it grows linearly² with N . This makes such algorithms impractical in this case.

Our work relates to recent efforts to measure various forms of personalization [7, 15, 20, 22]. They aim to quantify *how much* output is personalized and what *type* of information is used overall. In contrast, we seek to analyze fine-grained diagnosis of which combinations of *data inputs* generate which personalized results. Our analyses of scaling properties and tradeoffs are unique in the personalization literature.

Closest to our work are XRay [11] and AdFisher [6] – to our knowledge the very first two systems that fit in the emerging vision of data use transparency (described in Section 2). XRay aims to meet all the scalability, accuracy, and broad applicability requirements of web transparency, however the tool it currently provides lacks support for combinations – uniquely addressed in this paper through three new algorithms. We hope this can remove a critical roadblock to achieving any or all of these properties in practice. AdFisher aims to meet the accuracy and in some respect the broad applicability requirements, but does not consider scaling as a core requirement. This paper uniquely considers the theoret-

ical underpinnings of tools that aim to support all three requirements at once. We believe that support for linear combinations of inputs is a crucial step toward achieving those requirements.

7 Conclusion

Web transparency – a nascent and critical field to deploy big data while protecting us against its abuse – poses brand new challenge to the analysis of personalization algorithms. By providing the first theoretical analysis of web transparency for general targeting functions, and the promise of simple random algorithms, we show that web transparency can indeed be successful at scale. However, our research clearly indicates that transparency will also be governed by theoretical limits of detection and inherent tradeoffs.

We believe that our results create many opportunities for further research at the frontier of stochastic models, distributed systems, and algorithms handling personal data. Beyond understanding how to design efficient personalization algorithms, one hopes in addition to decipher their operations from the outside using the minimum number of information and operations available. Our works suggest many open problems that this field will grow to encompass, such as new methods to improve detection when other conditions are satisfied by the targeting functions, a careful analysis of computational lower-bounds that could leverage information theory, or new algorithms achieving better tradeoffs.

References

- [1] A. Acquisti and C. M. Fong. An Experiment in Hiring Discrimination. *Available at SSRN 2031979*, Apr. 2012.
- [2] D. Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, Apr. 1988.

² Indeed, if the targeting only depends on at most k inputs, then it is easy to prove that the $N - k$ inputs of which the function f does not depend on form a set that is shattered by the function. Hence the VC dimension of f is at least $N - k$ in such case

- [3] D. Angluin and D. K. Slonim. Randomly fallible teachers: Learning monotone DNF with an incomplete membership oracle. *Machine Learning*, 14(1):7–26, Jan. 1994.
- [4] B. Beizer. *Black-Box Testing. Techniques for Functional Testing of Software and Systems*. John Wiley & Sons, May 1995.
- [5] N. H. Bshouty and C. Tamon. On the Fourier spectrum of monotone functions. *Journal of the ACM*, 43(4):747–770, July 1996.
- [6] A. Datta, M. C. Tschantz, and A. Datta. Automated experiments on Ad privacy settings. In *Proceedings on Privacy Enhancing Technologies*, 2015.
- [7] A. Hannak, P. Sapiezynski, A. M. Kakhki, B. Krishnamurthy, D. Lazer, A. Mislove, and C. Wilson. Measuring personalization of web search. In *WWW '13: Proceedings of the 22nd international conference on World Wide Web*. International World Wide Web Conferences Steering Committee, May 2013.
- [8] A. Hannak, G. Soeller, D. Lazer, A. Mislove, and C. Wilson. Measuring Price Discrimination and Steering on E-commerce Web Sites. In *IMC '15: Proceedings of the 2015 Conference on Internet Measurement Conference*. ACM Request Permissions, Nov. 2014.
- [9] R. Heeks. Development 2.0: the IT-enabled transformation of international development. *Communications of the ACM*, 53(4):22–24, 2010.
- [10] A. Korolova. Privacy violations using microtargeted ads: A case study. *Data Mining Workshops (ICDMW), 2010 IEEE International Conference on*, pages 474–482, 2010.
- [11] M. Lécuyer, G. Ducoffe, F. Lan, A. Papancea, T. Petsios, R. Spahn, A. Chaintreau, and R. Geambasu. XRay: Enhancing the Web's Transparency with Differential Correlation. *23rd USENIX Security Symposium (USENIX Security 14)*, 2014.
- [12] M. Lecuyer, R. Spahn, and Y. Spiliopolous. Sunlight: Fine-grained Targeting Detection at Scale with Statistical Confidence. In *CCS '15 Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 554–566, New York, New York, USA, 2015. ACM Press.
- [13] A. Majchrzak and P. H. B. More. Emergency! Web 2.0 to the rescue! *Communications of the ACM*, 54(4):125, Apr. 2011.
- [14] D. Merugu, B. S. Prabhakar, and N. S. Rama. An incentive mechanism for decongesting the roads: A pilot program in Bangalore. In *Proceedings of the 2009 Workshop on Economics of Networks, Systems, and Computation (NetEcon '09)*, 2009.
- [15] J. Mikians, L. Gyarmati, V. Erramilli, and N. Laoutaris. Detecting price and search discrimination on the internet. In *HotNets-XI: Proceedings of the 11th ACM Workshop on Hot Topics in Networks*, pages 79–84, New York, New York, USA, Oct. 2012. ACM Request Permissions.
- [16] J. Podesta, P. Pritzker, E. J. Moniz, J. Holdren, and J. Zients. Big Data: Seizing Opportunities, Preserving Values. *Executive Office of the President*, pages 1–85, May 2014.
- [17] A. Rial and G. Danezis. Privacy-preserving smart metering. In *WPES '11: Proceedings of the 10th annual ACM workshop on Privacy in the electronic society*. ACM Request Permissions, Oct. 2011.
- [18] A. Sadilek and H. Kautz. Modeling the impact of lifestyle on health at scale. In *WSDM '13: Proceedings of the sixth ACM international conference on Web search and data mining*. ACM Request Permissions, Feb. 2013.
- [19] R. A. Servedio. On learning monotone DNF under product distributions. *Information and Computation*, 193(1):57–74, Aug. 2004.
- [20] L. Sweeney. Discrimination in online ad delivery. *Communications of the ACM*, 56(5):44–54, May 2013.
- [21] X. Wang, M. S. Gerber, and D. E. Brown. Automatic crime prediction using events extracted from twitter posts. In *SBP'12: Proceedings of the 5th international conference on Social Computing, Behavioral-Cultural Modeling and Prediction*. Springer-Verlag, Apr. 2012.
- [22] X. Xing, W. Meng, D. Doozan, N. Feamster, W. Lee, and A. C. Snoeren. Exposing Inconsistent Web Search Results with Bobble. In *PAM '14: Proceedings of the Passive and Active Measurements Conference*, 2014.

8 Proofs

8.1 Proof of Lemma 1

Lemma 1. We define $S^{(\text{in})}$ the set of all combinations for which f takes value 1. Let \vec{D}_f be the digraph with vertex-set $S^{(\text{in})}$ and with arc-set $\{(\mathcal{C}, \mathcal{C}') \mid \mathcal{C} \subsetneq \mathcal{C}'\}$. We have that \vec{D}_f is a DAG because the subset-containment relation defines a partial order. So, let S be the non-empty set of combinations with null in-degree in \vec{D}_f . By construction, each combination in $S^{(\text{in})}$ contains some combination of S and $S \subseteq S^{(\text{in})}$, hence S explains f . Furthermore, we claim that S is contained in *any* family S' explaining f : indeed, since S' is required to contain a subset of any combination $\mathcal{C} \in S$, and no combination of $S^{(\text{in})}$ is strictly contained in \mathcal{C} , then it must contain \mathcal{C} . This shows that S satisfies all conditions of Lemma 1. Finally, since another family explaining f needs to include S , then it will necessarily have a higher size l , hence S is the unique with both minimum size and order. \square

8.2 Chernoff bound

Lemma 7. If Y is a sum of independent binary variables, let $\mu = E[Y]$, we have for any $0 < \delta \leq 1$:

$$P[Y \geq (1 + \delta)\mu] \leq \exp\left(-\frac{\delta^2 \mu}{3}\right), \text{ and}$$

$$P[Y \leq (1 - \delta)\mu] \leq \exp\left(-\frac{\delta^2 \mu}{2}\right)$$

Thus, for any polynomial P , integer N and value $\varepsilon > 0$,

$$\mu \geq \frac{3}{\delta^2} \ln \left(\frac{2P(N)}{\varepsilon} \right) \implies P[|Y - \mu| \leq \delta\mu] \geq 1 - \frac{\varepsilon}{P(N)}.$$

8.3 Proof of Lemma 2

Lemma 2. Let us consider an arbitrary combination \mathcal{C} of size l . We introduce Y the variable counting how many Bernoulli subsets \mathcal{C} intersects, and we note that \mathcal{C} is an x -intersecting subset exactly if $Y \geq xm$. We also observe that Y is a sum of binary independent variables and so, since the probability that \mathcal{C} intersects an arbitrary Bernoulli subset is $1 - (1 - \alpha)^l$, it has expectation $\mu = (1 - (1 - \alpha)^l) m$. Assuming $\alpha < 1 - (1 - x)^{\frac{1}{l}}$ as we do, μ is multiplicatively smaller than xm . Hence we can apply Chernoff Bound to conclude that $P[Y \geq xm] \leq \frac{\varepsilon}{P(N)^N}$ when

$$m \geq C \cdot \ln \left(N^l P(N) / \varepsilon \right) \text{ with } C = \frac{3(1 - (1 - \alpha)^l)}{\left(x - (1 - (1 - \alpha)^l)\right)^2}.$$

Since there are $\binom{N}{l} \leq N^l$ choices of \mathcal{C} , by the union bound the probability that at least one of them is an x -intersecting subset is at most $\frac{\varepsilon}{P(N)}$. \square

8.4 Proof of Lemma 3

Lemma 3. We introduce the set of inputs associated with each account $\mathcal{S}_1 = B_1(N, \alpha), \dots, \mathcal{S}_m = B_m(N, \alpha)$, and for each of them we define Y_j a variable with the following value:

$$\begin{cases} 1 & \text{if } \mathcal{S}_j \text{ is in target, sees the ad, and } \mathcal{C} \subseteq \mathcal{S}_j, \\ -\frac{x}{1-x} & \text{if } \mathcal{S}_j \text{ not in target, sees the ad, and } \mathcal{C} \subseteq \mathcal{S}_j, \\ 0 & \text{otherwise} \end{cases}$$

We introduce $Y = \sum_{j=1}^m Y_j$, it is a sum of binary independent variables. We also note that the property of the theorem holds exactly if $Y \geq 0$. It is then sufficient to prove that this occurs with high probability.

First by the linearity of expectation we have that:

$$\begin{aligned} \mathbb{E}[Y] &= \sum_{j=1}^m \left(\alpha^{|\mathcal{C}|} q_C p_{in} - \frac{x}{1-x} \alpha^{|\mathcal{C}|} (1 - q_C) p_{out} \right) \\ &= \left(q_C p_{in} - \frac{x}{1-x} (1 - q_C) p_{out} \right) \alpha^{|\mathcal{C}|} m, \end{aligned}$$

where q_C denotes the probability for an account to be within scope knowing that it contains \mathcal{C} . This expectation is positive as long as it holds that $p_{out}/p_{in} <$

$\frac{1-x}{x} \frac{q_C}{1-q_C}$. Moreover, the above upper-bound is monotonically increasing with q_C , which is at least α^r because it suffices to complete \mathcal{C} with any combination of the core to be within scope. As a result, it always holds that $E[Y] > 0$ (with respect to our assumption about the ratio p_{out}/p_{in} for the lemma).

Hence $P[Y \geq 0] \geq 1 - \frac{\varepsilon}{P(N)}$ whenever $m \geq \alpha^{-|\mathcal{C}|} \cdot C$.

$$\ln(P(N)/\varepsilon) \quad \text{with } C \leq \frac{2}{\alpha^r p_{in}} \left(1 - \frac{x}{1-x} \frac{1 - \alpha^r}{\alpha^r} \frac{p_{out}}{p_{in}} \right)^{-1}.$$

8.5 Proof of Proposition 1

Proposition 1. Pick α so that $\frac{p_{out}}{p_{in}} < \hat{\varphi}_{l,r}^{(k)}(\alpha)$, x arbitrarily close to $1 - (1 - \alpha)^k$. First we prove (i) implies (ii) cannot hold which is the easy part of the result. Indeed, if a combination of the core is contained in \mathcal{C} , then any account that contains \mathcal{C} as part of its input is in the target $\mathcal{S}^{(in)}$, hence it sees the ad with probability p_{in} , and this holds irrespective of all other inputs. One deduces that $\Delta^{(ad)}(\mathcal{C})$ in that case is a Bernoulli family of expected size $\alpha^{|\mathcal{C}|} p_{in} m$, thus we can apply Lemma 2 if $\alpha < 1 - (1 - x)^{1/k}$, and conclude that (ii) may only occur with small probability $\varepsilon/P(N)$.

To show that if (i) does not hold, then (ii) does, we define the two following subsets.

$$\text{Let } \Delta^{(ad,in)}(\mathcal{C}) = \left\{ \mathcal{S} \cap \bar{\mathcal{C}} \mid \mathcal{S} \in \mathcal{S}^{(ad,in)}, \mathcal{C} \subseteq \mathcal{S} \right\},$$

$$\text{and } \Delta^{(core)}(\mathcal{C}) = \left\{ \mathcal{S} \cap \bar{\mathcal{C}} \mid \mathcal{S} \in \mathcal{S}^{(core)} \right\}.$$

Note that since no combination of the core family is included in \mathcal{C} , no element of $\Delta^{(core)}(\mathcal{C})$ is empty. Furthermore, observe that by definition a combination in $\mathcal{S}^{(ad,in)} = \mathcal{S}^{(ad)} \cap \mathcal{S}^{(in)}$ should contain a combination of the core. This directly implies that a combination in $\Delta^{(ad,in)}(\mathcal{C})$ necessarily contains a combination from $\Delta^{(core)}(\mathcal{C})$, which is by consequence a 1-intersecting family of $\Delta^{(ad,in)}(\mathcal{C})$.

We can apply Lemma 3 not using x but $(l/k) \cdot x$, as Eq.(3) ensures the condition of the lemma is satisfied for this value and α . This shows that with probability at least $(1 - \varepsilon/P(N))$, $|\Delta^{(ad,in)}(\mathcal{C})|/|\Delta^{(ad)}(\mathcal{C})| \geq (l/k) \cdot x$.

The family $\Delta^{(core)}(\mathcal{C})$ contains at most l elements, and it intersects all combinations in $\Delta^{(ad,in)}(\mathcal{C})$. Moreover:

$$\forall (X_i)_{i=1}^I, \max_{i_1, \dots, i_a} |X_{i_1} \cup \dots \cup X_{i_a}| \geq \frac{a}{I} \left| \bigcup_{i=1}^I X_i \right|.$$

Hence there exists a subset of $k \leq l$ combinations chosen in $\Delta^{(core)}(\mathcal{C})$ that collectively intersect at least $(k/l) \cdot$

$|\Delta^{(\text{ad, in})}(\mathcal{C})|$ elements. It is hence an x -intersecting family of $\Delta^{(\text{ad})}(\mathcal{C})$ with probability $(1 - \varepsilon/P(N))$, proving (ii). \square

8.6 Proof of Lemma 4

Lemma 4. The expected size of $\Delta^{(\text{ad})}(\mathcal{C})$ is $\alpha^{|\mathcal{C}|} q_{\mathcal{C}} m$ with

$$q_{\mathcal{C}} = P \left[B_j(N, \alpha) \in S^{(\text{in})} \mid \mathcal{C} \in B_j(N, \alpha) \right] \geq \alpha^r.$$

Hence by Chernoff bound there exists C_0 such that whenever $m \geq \alpha^{-|\mathcal{C}|} \cdot C_0 \cdot \log \frac{P(N)}{\varepsilon}$, with high probability we have $|\Delta^{(\text{ad})}(\mathcal{C})| \geq \alpha^{|\mathcal{C}|} q_{\mathcal{C}} m / 2$. If we constrain all combinations of $\Delta^{(\text{ad})}(\mathcal{C})$ to the $N' < N$ data inputs that are not part of $\bigcup_{S \in S^{(\text{core})}} \mathcal{S}$, we thus obtain a random Bernoulli family of large size $\geq \alpha^{|\mathcal{C}|} q_{\mathcal{C}} m / 2$ with parameters N', α . Assuming $\alpha < 1 - (1 - x)^{1/k}$, this family does not admit an x -intersecting subset of size $\leq k$ w.h.p. by Lemma 2, hence no combination in $S_{x,k}$ can be fully contained into it.

Moreover if it holds that $\frac{p_{\text{out}}}{p_{\text{in}}} < \sup_{\alpha} \hat{\varphi}_{l,r}^{(k)}(\alpha)$, then by the proof of Proposition 1 there is an x -intersecting family composed of $\leq k$ combinations from $\Delta^{(\text{core})}(\mathcal{C})$, hence w.h.p. there are $\leq k$ combinations from $S^{(\text{core})}$ fully contained in $\mathcal{C} \cup (\bigcup_{S \in S_{x,k}} S)$. \square

8.7 Proof of Proposition 2

Proposition 2. We will prove the proposition by induction on the size $s \leq l$ of the core family. If $s = 0$, then we are done as by Proposition 1 we have w.h.p. that $\{D_i \mid \frac{|\{A_j \in S^{(\text{ad})} \mid D_i \in A_j\}|}{|S^{(\text{ad})}|} \geq x\} = \emptyset$ whenever $m \geq C_0 \cdot \log \frac{P(N)}{\varepsilon}$, for some constant C_0 .

Suppose by the induction hypothesis that for all $0 \leq s' \leq s - 1$, there exists a constant $C_{s'}$ such that GSI is correct w.h.p. whenever $m \geq C_{s'} \cdot \log \frac{P(N)}{\varepsilon}$ and the core family has size s' . Let $S^{(\text{core})}$ be of size s . By the dichotomy result from Proposition 1, there is C' such that $\mathcal{S} = \{D_i \mid \frac{|\{A_j \in S^{(\text{ad})} \mid D_i \in A_j\}|}{|S^{(\text{ad})}|} \geq x\} \neq \emptyset$ w.h.p. whenever $m \geq C' \cdot \log \frac{P(N)}{\varepsilon}$. Furthermore by Lemma 4, this set contains a combination from the core family. GSI iterates over the data inputs $D_i \in \mathcal{S}$, it removes each temporarily and it applies the recognition test from Prop.1 to decide whether $\mathcal{S} \setminus D_i$ is within target. If so, then it still contains a combination of the core family and we can delete D_i from \mathcal{S} permanently. Else, the data input D_i is *critical i.e.*, it intersects all combinations of $S^{(\text{core})}$ contained into \mathcal{S} , and we put it back in \mathcal{S} . As

a result, the remaining data inputs in \mathcal{S} at the end of the iteration are a combination from $S^{(\text{core})}$ that we will denote by \mathcal{C} in the following.

Let $D_i \in \mathcal{C}$ be fixed, and let \mathcal{A}_i be the active accounts amongst those in the random Bernoulli subfamily \mathcal{B}_i , defined as all the $B_j(N, \alpha)$ not containing this data input. Note that \mathcal{B}_i has size $(1 - \alpha)m$ on expectation, and that f constrained to \mathcal{B}_i is equivalent to the targeting function f_i whose core family is $S_i = S^{(\text{core})} \setminus \{\mathcal{S} \mid D_i \in \mathcal{S}\}$. Thus by the induction hypothesis, there is C_{s_i} , with $s_i = |S_i|$, such that w.h.p. the output of the recursive call $\text{GSI}(S^{(\text{ad})} \setminus \{A_j \mid D_i \in A_j\}, x)$ is exactly the core subfamily S_i if $|\mathcal{B}_i| \geq C_{s_i} \cdot \log \frac{P(N)}{\varepsilon}$. Furthermore, there is C^i such that $|\mathcal{B}_i| \geq (1 - \alpha)m/2$ w.h.p. if $m \geq C^i \cdot \log \frac{P(N)}{\varepsilon}$.

Consequently, the output of the algorithm is with high probability $\{\mathcal{C}\} \cup (\bigcup_{D_i \in \mathcal{C}} S_i)$ if $m \geq C_s \cdot \log \frac{P(N)}{\varepsilon}$, with

$$C_s \geq \max\{C'\} \cup \{C^i \mid D_i \in \mathcal{C}\} \cup \{2C_{s_i}(1 - \alpha)^{-1} \mid D_i \in \mathcal{C}\}.$$

This concludes the proof because $S^{(\text{core})} = \{\mathcal{C}\} \cup (\bigcup_{D_i \in \mathcal{C}} S_i)$ by the monotonicity assumption. \square

8.8 Proof of Lemma 5

Lemma 5. Given any fixed subset \mathcal{D} of data inputs, we can compute in $O(|S^{(\text{ad})}| + \sum_{j=1}^k |A_j|) = O(N \cdot |S^{(\text{ad})}|)$ -time the set of accounts $\{A_j \in \mathcal{A} \mid \mathcal{D} \subseteq A_j\}$ and so, $\{D'_i \notin \mathcal{D} \mid \frac{|\{A_j \in \mathcal{A} \mid \mathcal{D} \cup \{D'_i\} \subseteq A_j\}|}{|\{A_j \in \mathcal{A} \mid \mathcal{D} \subseteq A_j\}|} \geq x\}$. As a result, Lines 1-7 (til the first for-loop) can be executed in $O(|\mathcal{C}| \cdot N \cdot |S^{(\text{ad})}|) = O(|\mathcal{I}| \cdot N |S^{(\text{ad})}|)$ -time. Moreover, at each step of the second for-loop it holds that S, S_i are subsets of $\mathcal{P}(\mathcal{I})$ and so, the merge $S \cup S_i$ can be executed in $O(|\mathcal{I}| 2^{|\mathcal{I}|})$ -time. Let us finally denote by $T_{\mathcal{I}}$ an upper-bound on the computational cost of any recursive call to the algorithm.

We have that GSI can be executed in $O(|\mathcal{I}| \cdot (N \cdot |S^{(\text{ad})}| + |\mathcal{I}| 2^{|\mathcal{I}|} + T_{\mathcal{I}}))$ -time. By induction, the time-complexity is $O(|\mathcal{I}|! \cdot N \cdot |S^{(\text{ad})}| + \sum_{j=1}^{|\mathcal{I}|-1} \frac{|\mathcal{I}|!}{(|\mathcal{I}| - j)!} (|\mathcal{I}| - j + 1) 2^{|\mathcal{I} - j + 1})$.

$\Gamma(|\mathcal{I}|)$
This concludes the proof as we have:

$$\Gamma(|\mathcal{I}|) = \sum_{j=1}^{|\mathcal{I}|-1} \frac{|\mathcal{I}|!}{j!} (j + 1) 2^{j+1} \leq 2 \cdot |\mathcal{I}|! \left[\sum_{j=0}^{\infty} (j + 1) \frac{2^j}{j!} \right] \leq 6e^2 \cdot |\mathcal{I}|! \quad \square$$

8.9 Proof of Proposition 3

Proposition 3. Let $\mathcal{S}^{(core)} = \{D_i \mid \exists C \in \mathcal{S}^{(core)} \text{ s.t. } D_i \in C\}$, and let $\mathcal{S} = \{D_i \mid \frac{|A_j \in \mathcal{S}^{(ad)} \mid D_i \in A_j|}{|\mathcal{S}^{(ad)}|} \geq x\}$. By Lemma 4, there is C' such that $\forall \varepsilon > 0$, for all polynomial P , we have $\mathcal{S} \subseteq \mathcal{S}^{(core)}$ with probability $1 - \frac{\varepsilon}{P(N)}$ whenever $m \geq C' \cdot \log \frac{P(N)}{\varepsilon}$. Applying this argument recursively, we obtain that the depth of the recursive calls is upper-bounded by lr and so, that we also have $\mathcal{I} \subseteq \mathcal{S}^{(core)}$ with probability $1 - \frac{\varepsilon}{P(N)}$ whenever $m \geq C'(1 - \alpha)^{-lr} \cdot \log \frac{P(N)}{\varepsilon}$. In such case, our set-intersection algorithm can be implemented to run in $O(|\mathcal{I}| \cdot N \cdot |\mathcal{S}^{(ad)}|) = O((lr)! \cdot Nm)$ -time by Lemma 5.

The expected running-time is therefore upper-bounded by an $O\left(\left[\left(1 - \frac{\varepsilon}{P(N)}\right) (lr)! + \frac{\varepsilon}{P(N)} N!\right] Nm\right)$. By setting $\frac{\varepsilon}{P(N)} = \frac{1}{N!}$, we conclude that it is $O((lr)! \cdot Nm)$ whenever $m = \Omega(N \log N)$. \square

8.10 Proof of Lemma 6

Lemma 6. When $l = 1$ one can easily see that $\varphi_{1,r}$ is strictly increasing on this interval and computes its limit as x approaches 1. A similar argument holds for $r = 1$.

Whenever $r > 1$ and $l > 1$, introducing the new variable $z = (1 - x)^{1/l}$ we first observe:

$$\varphi_{l,r}(z) = f_l(z) \cdot f_r(1 - z), \text{ where } f_n(z) = \frac{z^n}{1 - z^n}.$$

This symmetry immediately implies that $M_{l,r} = M_{r,l}$. Note that the form of the function also directly yields that

$$M_{\max(l,r), \max(l,r)} \leq M_{l,r} \leq M_{\min(l,r), \min(l,r)}.$$

We have $\varphi'_{l,r}(z) = f'_l(z) \cdot f_r(1 - z) - f_l(z) \cdot f'_r(1 - z)$, and observe that this derivative becomes null whenever we have $f'_l(z)/f_l(z) = f'_r(1 - z)/f_r(1 - z)$. Moreover, it holds that

$$f'_n(z) = -\frac{lz^{n-1}}{(1 - z^n)^2} \text{ hence } f'_n(z)/f_n(z) = \frac{n}{z(1 - z^n)}$$

so that the condition is $\frac{l}{z(1 - z^l)} = \frac{r}{(1 - z)(1 - (1 - z)^r)}$ which yields the value of z reaching the maximum.

To conclude, we just need to observe that there is a unique solution in $]0; 1[$. We can immediately observe, when $r > 1$ and $l > 1$ that the product $f_l(z) \cdot f_r(1 - z)$ has null limits on both sides, and a derivative that is positive near 0^+ and 1^- . Since its third derivative is strictly positive, its second derivative increases and can

only be null once. We deduce that the derivative cannot cancel twice between 0 and 1 since it would create two inflection points.

Finally, when $r = l = n$, since the product is symmetric in z and it has a unique maximum on $]0; 1[$ it has to be in $z = \frac{1}{2}$ which yields the result. \square

APPENDIX N

A theory for ad targeting identification

A theory for ad targeting identification

Abstract

The problem of learning juntas has been studied under different notions of learning. We here study a new model, where the goal is “exact” learning (with high probability) of a given Boolean function with membership queries. The main novelty in the model is that queries are subject to asymmetric classification noise and limited cross-unit effects. Furthermore, it extends (and is inspired by) the theory behind two already well established *web transparency* tools. Such tools aim to put emphasis on any form of misuse of our personal data by the institutions and companies on-line. However, they have so far accounted only for targeting on a *monotone* function, with the constrictive assumption that users are targeted *independently* the one from the other. In contrast with prior work, we prove that *any* function can be learnt in this extended model — conditioned on one assumption about the noise, that will be proved to be necessary. More precisely, we show that for any k , the function can be learnt in $N^{\mathcal{O}(k)}$ -time and $\text{polylog}(N)$ queries with k -juntas as hypotheses. Our algorithms build upon some variation of a known greedy heuristic which reduces to Set Cover in order to infer the relevant variables. Finally, we show that there is a given (monotone) 2-junta which cannot be learnt within our model when it is made no assumption about the noise.

Keywords: web transparency; negative targeting; cross-unit effect; exact learning; juntas.

1 The model

1.1 Targeting functions

Our problem formulation extends the one in [7]. So, in particular, it differs from (but can be easily shown to be equivalent to) standard terminology in the literature of Boolean function learning. Let $\mathcal{D} = \{D_1, D_2, \dots, D_N\}$ be a set of N inputs representing individual information from a given user (typically, emails in an account, see also [4]). Our main objective is to identify how these inputs affect a given output of interest (say, an ad or a recommendation).

In order to achieve the goal, we here assume that each output is affected through an unknown targeting function f_{output} , that we simply denote by f in the following. More precisely, let any $\mathcal{C} \subseteq \mathcal{D}$ be called a combination. The targeting function f is a mapping from the family of all combinations to the Boolean set $\{0; 1\}$. By convention, $f(\mathcal{C}) = 1$ indicates that an account *exactly* containing the inputs in \mathcal{C} is targeted, and we denote $f(\cdot) = 0$ if the ad is untargeted. In particular unlike [5], we do not assume here that f is *monotone* (*i.e.*, $f(\mathcal{C}) = 1$ and $\mathcal{C} \subseteq \mathcal{C}' \not\Rightarrow f(\mathcal{C}') = 1$). Hence, the targeting function f can be any Boolean formula with a subset of \mathcal{D} as its variables. However in practice, we assume that f only depends on a small number k of inputs, with k being a universal constant. Such targeting functions are called k -juntas in the literature [3]. We are particularly interested in how the value of k affects the complexity of the targeting detection and identification problems, *i.e.*, what is their parameterized complexity ?

In this note, we adopt the classical approach in order to learn the targeting function f , that is, we seek to learn the class $\mathcal{S}^{(\text{in})}$ of all combinations \mathcal{C} such that $f(\mathcal{C}) = 1$.

1.2 Outcome properties

In order to learn the targeting function, we are bound to rely on experiments — to see how it reacts to various inputs. For instance, in [7] these experiments consist in collecting the ads for Gmail accounts with different subsets of emails. We model this as an oracle from function learning theory [1], denoted by \mathcal{O}_f . Formally, \mathcal{O}_f is a membership oracle with (asymmetric) classification noise. That is, it outputs the Boolean $f(\mathcal{C})$ for any combination \mathcal{C} with some probability to flip the result. Unlike prior work [2], we do not assume the classification noise to be symmetric, *i.e.*, the oracle may flip the result with some probability *depending on the combination*. Nonetheless, we will assume a few properties for the noise distribution. To our best knowledge, the following assumptions that are made on this probability have not been studied before in the literature.

Histories. Experiments in [9] have evidenced that the noise distribution is subject to cross-unit effects. So, in order to handle with these correlations, we find it more suitable to generalize our oracle \mathcal{O}_f so that it can take *families of combinations* as inputs. More precisely, let a family be any vector of combinations, denoted by $\mathcal{F} = \langle A_1, A_2, \dots, A_t \rangle$. The outcome $\mathcal{O}_f(\mathcal{F})$ is simply defined as the binary vector $\mathcal{O}_f(\mathcal{F}) = \langle \mathcal{O}_f(A_1), \mathcal{O}_f(A_2), \dots, \mathcal{O}_f(A_t) \rangle$. Furthermore, let the pair $H_{\mathcal{F}} = (\mathcal{F}; \mathcal{O}_f(\mathcal{F}))$ be the history of f .

As usual, let $\mathcal{F}_{-i} = \langle A_1, \dots, A_{i-1}, A_{i+1}, \dots, A_t \rangle$. We will assume that each individual outcome $\mathcal{O}_f(A_i)$ may be correlated to the partial history $H_{\mathcal{F}_{-i}}$. However, it may and must be the case that some natural properties hold independently from any history, that we now detail as follows:

Assumption 1 (targeting lift). *There exists a universal constant $\varphi \in]0; 1[$, called the targeting lift and such that for any $\mathcal{C}_0, \mathcal{C}_1$ with $f(\mathcal{C}_0) = 0, f(\mathcal{C}_1) = 1$:*

$$\Pr[\mathcal{O}_f(A_i) = 1 \mid A_i = \mathcal{C}_0, H_{\mathcal{F}_{-i}}] < \varphi \cdot \Pr[\mathcal{O}_f(A_i) \mid A_i = \mathcal{C}_1, H_{\mathcal{F}_{-i}}].$$

Assumption 1 is local and it simply ensures that it is more likely for \mathcal{O}_f to output 1 on accounts A_i within scope, that is, for which $f(A_i) = 1$. In particular, it implies that the targeting function f is related to the outcome we study. Note that a similar assumption was made in [5, 7], but in the less realistic case when no cross-unit effect occurs, and so, each account is targeted independently. However, the following two global assumptions are new (if there is no cross-unit effect then they can be proven to be true by using standard concentration inequalities and independence, see [7]).

Assumption 2 (polynomial-growth). *There exist positive universal constant α, β, γ with $\alpha \leq 1$ and such that:*

$$\Pr\left[\sum_{i=1}^t \mathcal{O}_f(A_i) < \left(\beta \cdot |\mathcal{F} \cap \mathcal{S}^{(in)}|^{\alpha}\right)\right] \leq e^{-\gamma \cdot t}$$

We properly state with Assumption 2 that there must be a significant fraction of the account population *within scope* being targeted, except on some small event with low probability like, for instance, when the targeting campaign runs out of budget. Note that in [5], it was assumed that there is some minimum *constant* probability p_{in} for an account within scope to be targeted, so, Assumption 2 was satisfied for $\alpha = 1$. By considering the case $\alpha \leq 1$, we may consider the case when this minimum probability slowly tends to zero, say, $p_{in} \sim p_o / \log^{\mathcal{O}(1)}(N)$ where p_o is a constant. This case was observed to happen in practice [7].

Assumption 3 (noninterference). *Let the targeting function f only depend on inputs in $V \subseteq \mathcal{D}$. Furthermore, let $A'_i = A_i \cap V$ and let $\mathcal{F}' = \langle A'_1, \dots, A'_t \rangle$.*

$$\Pr[\mathcal{O}_f(\mathcal{F})] = \Pr[\mathcal{O}_f(\mathcal{F}')].$$

Finally, we formalize with Assumption 3 that none of the input that does not affect the targeting can impact on the outcome.

1.3 Experiments and random families

The model in Section 1.2 supports *adaptive queries*. However in practice, the methodologies for web transparency [9] recommend to use so-called “exchangeable” accounts in the experiments. In particular, one experimental design used in practice [4, 7] is to populate each account randomly so that an input independently appears with same probability — that will be taken equal to 1/2 in the remaining of the paper. So, we will constrain our queries on pairwise independent random accounts in the following. We will name a random Bernoulli family any family of such random accounts.

2 Preliminaries

Prior work [5, 7] has made extensive use of concentration inequalities in the analysis of the algorithms, *i.e.*, Chernoff bounds. Standard bounds apply to the sum of *independent* variables, so, they cannot be used in our setting directly. The following is a tedious (but classical) analysis where we show how to adapt Chernoff bounds to our needs.

Lemma 1. *Let X_1, \dots, X_m be random Boolean variables satisfying:*

$$p_{\min} \leq \mathbb{P}[X_i = 1 \mid X_1, \dots, X_{i-1}] \leq p_{\max}$$

for some constant p_{\min}, p_{\max} . Then the following hold for any $0 < \delta < 1$:

$$\mathbb{P}\left[\sum_{i=1}^m X_i \geq (1 + \delta) \cdot p_{\max} \cdot m\right] \leq e^{-\delta^2 m p_{\max}/3}$$

$$\mathbb{P}\left[\sum_{i=1}^m X_i \leq (1 - \delta) \cdot p_{\min} \cdot m\right] \leq e^{-\delta^2 m p_{\min}/2}$$

Proof. By symmetry, we will only consider the first inequality. Let $t > 0$. Let us show that:

$$\mathbb{E}\left[\prod_{i=1}^m e^{t \cdot X_i}\right] \leq (p_{\max}(e^t - 1) + 1)^m.$$

The proof is by induction. By the hypothesis,

$$\mathbb{E}[e^{t \cdot X_m} \mid X_1, \dots, X_{m-1}] = e^t \cdot \mathbb{P}[X_m = 1 \mid X_1, \dots, X_{m-1}] + 1 \cdot \mathbb{P}[X_m = 0 \mid X_1, \dots, X_{m-1}] \leq p_{\max}(e^t - 1) + 1,$$

that is the base case. Suppose for the induction hypothesis that:

$$\mathbb{E}\left[\prod_{j=i+1}^m e^{t \cdot X_j} \mid X_1, \dots, X_i\right] \leq (p_{\max}(e^t - 1) + 1)^{m-i}.$$

Then by the law of total probability:

$$\begin{aligned} \mathbb{E}\left[\prod_{j=i}^m e^{t \cdot X_j} \mid X_1, \dots, X_{i-1}\right] &= e^t \cdot \mathbb{P}[X_i = 1 \mid X_1, \dots, X_{i-1}] \cdot \mathbb{E}\left[\prod_{j=i+1}^m e^{t \cdot X_j} \mid X_1, \dots, X_{i-1}, X_i = 1\right] \\ &\quad + 1 \cdot \mathbb{P}[X_i = 0 \mid X_1, \dots, X_{i-1}] \cdot \mathbb{E}\left[\prod_{j=i+1}^m e^{t \cdot X_j} \mid X_1, \dots, X_{i-1}, X_i = 0\right] \\ &\leq (\mathbb{P}[X_i = 1 \mid X_1, \dots, X_{i-1}] \cdot (e^t - 1) + 1) \cdot (p_{\max}(e^t - 1) + 1)^{m-i} \leq (p_{\max}(e^t - 1) + 1)^{m-i+1}, \end{aligned}$$

which proves the induction hypothesis. The remaining of the proof is now classical computation of Chernoff Bound. By Markoff inequality:

$$\begin{aligned} \mathbb{P}\left[\sum_{i=1}^m X_i \geq (1 + \delta) \cdot p_{\max} \cdot m\right] &= \mathbb{P}\left[e^{t \cdot \sum_{i=1}^m X_i} \geq e^{t \cdot (1 + \delta) \cdot p_{\max} \cdot m}\right] \leq \mathbb{E}\left[e^{t \cdot \sum_{i=1}^m X_i}\right] / e^{t \cdot (1 + \delta) \cdot p_{\max} \cdot m} \\ &= e^{-t \cdot (1 + \delta) \cdot p_{\max} \cdot m} \cdot \mathbb{E}\left[\prod_{i=1}^m e^{t \cdot X_i}\right] \leq e^{-t \cdot (1 + \delta) \cdot p_{\max} \cdot m} \cdot (p_{\max}(e^t - 1) + 1)^m \\ &\leq e^{-t \cdot (1 + \delta) \cdot p_{\max} \cdot m} \cdot e^{p_{\max}(e^t - 1) \cdot m} = e^{p_{\max} \cdot m \cdot (e^t - 1 - t \cdot (1 + \delta))} \end{aligned}$$

Finally, set $t = \ln(1 + \delta)$. One obtains:

$$\mathbb{P}r\left[\sum_{i=1}^m X_i \geq (1 + \delta) \cdot p_{\max} \cdot m\right] \leq \left(\frac{e^\delta}{(1 + \delta)^{1+\delta}}\right)^{mp_{\max}} \leq e^{-\delta^2 mp_{\max}/3}.$$

□

3 Building block algorithm: the case $k=1$

We first describe a simple algorithm with 1-juntas as hypotheses. A simpler variation of the following Algorithm 1 is the core algorithm of the Xray prototype [7]. Here, we extend the algorithm to the case of negative targeting, and we prove its correctness under our more general assumptions.

Input: a family \mathcal{F} ; threshold parameters x, y .

Output: intersecting families S_x, S_y .

$S_x \leftarrow \{D_j \in \mathcal{D} \mid D_j \text{ appears in } \geq x \cdot |\mathcal{F}| \text{ accounts in } \mathcal{F}\};$

$S_y \leftarrow \{D_j \in \mathcal{D} \mid D_j \text{ appears in } \leq y \cdot |\mathcal{F}| \text{ accounts in } \mathcal{F}\};$

Algorithm 1: Set-intersection algorithm.

The reader may observe that Algorithm 1 requires two parameters x, y as inputs. For simplicity, we will assume that a good estimate on the *targeting lift* (Assumption 1) is given, and we will show in the following that the latter information is enough in order to tune x, y . Nonetheless, we point out that finding these two parameters in practice may be cumbersome. We refer the reader to [6, 7] for experimental and theoretical methods in order to tune x, y .

Lemma 2. *Fix any polynomial P . Let \mathcal{B} be a random Bernoulli family. There is a constant $a_{x,y}$ such that if $y < 1/2 < x$, \mathcal{F} contains the targeted accounts in \mathcal{B} and has size $|\mathcal{F}| \geq a_{x,y} \cdot \log(2 \cdot P(N) \cdot N/\varepsilon)$, then the following holds with probability $\geq 1 - \varepsilon/P(N)$: the targeting function f depends on any input in S_x or S_y .*

Proof. Set $a_{x,y} = \frac{6}{(\min\{1-2y, 2x-1\})^2}$. We will identify $\leq N$ events so that, if it is the case that f does not depend on some input in $S_x \cup S_y$, then one of these events must fail. So, in order to prove the lemma, we will prove that each event fails with probability $\leq \varepsilon/(P(N) \cdot N)$ — in which case, it is easy to conclude by taking a union bound.

Before this, we claim that every input D_j of which the targeting function f does not depend on appears in the combinations of \mathcal{F} *independently* and with probability $1/2$. Our proof is combinatorial. Let $I \subseteq \{1, \dots, |\mathcal{B}|\}$ be fixed. The set I will denote the indices of combinations $A_i \in \mathcal{B}$ that are targeted, *i.e.*, $\mathcal{O}_f(A_i) = 1$ if and only if $i \in I$. For any $i \in I$, let X_i be the random Boolean variable denoting whether $D_j \in A_i$. In order to prove the claim, we must prove that variables X_i are mutually independent. So, let $J \subset I$ and $i \in I \setminus J$. Let us fix X_q for every $q \in J$. Finally, consider the set Ω_J of all histories $(\mathcal{B}, \mathcal{O}_I)$ satisfying: \mathcal{O}_I is the binary vector with its nonzero entries indexed by I , and for every $q \in J$, $D_j \in A_q$ if and only if $X_q = 1$. Since $h_i : \langle A_1, \dots, A_{i-1}, A_i, A_{i+1}, \dots, A_{|\mathcal{B}|} \rangle \rightarrow \langle A_1, \dots, A_{i-1}, A_i \Delta \{D_j\}, A_{i+1}, \dots, A_{|\mathcal{B}|} \rangle$ is one-to-one, there is half of the histories in Ω_J that satisfy X_i . Furthermore, since f does not depend on D_j , for any $(\mathcal{B}, \mathcal{O}_I) \in \Omega_J$, $\mathbb{P}r[\mathcal{O}_f(\mathcal{B}) = \mathcal{O}_I] = \mathbb{P}r[\mathcal{O}_f(h_i(\mathcal{B})) = \mathcal{O}_I]$ by Assumption 3. As a result, $\mathbb{P}r[X_i \mid X_{q_1}, \dots, X_{q_{|J|}}] = 1/2$, with $J = \{q_1, \dots, q_{|J|}\}$, that proves the claim.

Finally, for every irrelevant input D_j (there are $\leq N$ such inputs), since $|\mathcal{F}|$ is large enough we have by Chernoff bound (applying Lemma 1 with $p_{\min} = p_{\max} = 1/2$) that with probability $\geq 1 - \varepsilon/(P(N) \cdot N)$ the irrelevant input D_j appears in a fraction $y < \cdot < x$ of the accounts in \mathcal{F} . □

By Lemma 2, every input detected by the Set-intersection algorithm is relevant. However, the main drawback with the above Algorithm 1 is that it may fail in identifying *all* the relevant inputs for the targeting. We now prove that Algorithm 1 can be used in order to learn the targeting function f , in the special case when it depends on a unique input (either positively, or negatively).

Theorem 1. Set $\varphi/(1+\varphi) < y < 1/2 < x < 1/(1+\varphi)$. Furthermore, fix any polynomial Q . Let \mathcal{B} be a random Bernoulli family of size $|\mathcal{B}| = m$.

There is a constant $b_{x,y}$ such that if $m \geq b_{x,y} \cdot \log^{1/\alpha}(8 \cdot Q(N) \cdot N/\varepsilon)$ and \mathcal{F} contains the targeted accounts in \mathcal{B} , then the following holds with probability $\geq 1 - \varepsilon/Q(N)$:

- $S_x = \{D_j\}$, $S_y = \{\}$ if $f : \mathcal{C} \rightarrow \mathbb{I}_{\{D_j \in \mathcal{C}\}}$;
- $S_x = \{\}$, $S_y = \{D_j\}$ if $f : \mathcal{C} \rightarrow \mathbb{I}_{\{D_j \notin \mathcal{C}\}}$.

Proof. Let $a_{x,y}$ be the constant defined in Lemma 2. Set $b_1 = 12$, $b_2 = 1/\gamma$, $b_3 = 4 \left(\frac{a_{x,y}}{\beta}\right)^{1/\alpha}$, $b_4 = 4 \left(\frac{3}{\beta \cdot (1-x)(1+\varphi)^2}\right)^{1/\alpha}$ and $b_5 = 4 \left(\frac{3\varphi^2}{\beta \cdot (y-\varphi(1-y))^2}\right)^{1/\alpha}$. Finally, set $b_{x,y} = \max_{1 \leq i \leq 5} b_i$.

By symmetry we only need to consider the case when $f : \mathcal{C} \rightarrow \mathbb{I}_{\{D_j \in \mathcal{C}\}}$ for some fixed j . In this case, since \mathcal{B} is a random Bernoulli family, $\mathbb{E}[|\mathcal{B} \cap \mathcal{S}^{(\text{in})}|] = |\mathcal{B}|/2$. Thus, by Chernoff bound, since $m \geq b_1 \cdot \log(8 \cdot Q(N)/\varepsilon)$ we have that $|\mathcal{B} \cap \mathcal{S}^{(\text{in})}| \geq |\mathcal{B}|/4$ with probability $\geq 1 - \varepsilon/(4 \cdot Q(N))$. Furthermore, by Assumption 2, since $m \geq b_2 \cdot \log(8 \cdot Q(N)/\varepsilon)$ we have that $|\mathcal{F}| \geq \beta \cdot (|\mathcal{B} \cap \mathcal{S}^{(\text{in})}|)^\alpha$ with probability $\geq 1 - \varepsilon/(4 \cdot Q(N))$.

First, let us show that irrelevant inputs can be ignored. Since we can assume that $|\mathcal{F}| \geq \beta \cdot (b_3/4)^\alpha \cdot \log(8 \cdot Q(N) \cdot N/\varepsilon)$, it follows that $|\mathcal{F}| \geq a_{x,y} \cdot \log(8 \cdot Q(N) \cdot N/\varepsilon)$ with probability $\geq 1 - \varepsilon/(2 \cdot Q(N))$. Furthermore, setting $P(N) = 4 \cdot Q(N)$, one obtains by Lemma 2 that with probability $\geq 1 - \varepsilon/(4 \cdot Q(N))$, either $D_j \in S_x$ or $S_x = \emptyset$, similarly either $D_j \in S_y$ or $S_y = \emptyset$.

Second, let $\mathcal{F} = \langle A_{i_1}, \dots, A_{i_t} \rangle$. Define X_l to be the random Boolean variable denoting whether $D_j \in A_{i_l}$. We will prove that $\Pr[X_l = 1 \mid X_1, \dots, X_{l-1}] \geq 1/(1+\varphi)$. To prove it, fix $I \subseteq \{1, \dots, |\mathcal{B}|\}$, with $|I| = t$, that will represent the indices of the combinations $A_{i_l} \in \mathcal{B}$ that are within \mathcal{F} . Let $i_l \in I$ and let X_1, \dots, X_{l-1} be fixed. Consider the set Ω of all histories $(\mathcal{B}, \mathcal{O}_I)$ where \mathcal{O}_I denotes the binary vector with its nonzero entries being indexed by I and for any $p < l$, $D_j \in A_{i_p}$ if and only if $X_p = 1$. Since $h_l : \langle A_1, \dots, A_{i_{l-1}}, A_{i_l}, A_{i_{l+1}}, \dots, A_m \rangle \rightarrow \langle A_1, \dots, A_{i_{l-1}}, A_{i_l} \Delta \{D_j\}, A_{i_{l+1}}, \dots, A_m \rangle$ is one-to-one, there is half of the histories in Ω satisfying X_l . Furthermore, for any $(\mathcal{B}, \mathcal{O}_I)$ satisfying X_l , by Assumption 1:

$$\frac{\Pr[\mathcal{O}_f(\mathcal{B}) = \mathcal{O}_I]}{\Pr[\mathcal{O}_f(h_l(\mathcal{B})) = \mathcal{O}_I]} = \frac{\Pr[\mathcal{O}_f(A_{i_l}) = 1 \mid D_j \in A_{i_l}, \mathbf{H}_{\mathcal{B}-i_l} = \mathcal{O}_{I-i_l}]}{\Pr[\mathcal{O}_f(A_{i_l}) = 1 \mid D_j \notin A_{i_l}, \mathbf{H}_{\mathcal{B}-i_l} = \mathcal{O}_{I-i_l}]} > 1/\varphi.$$

As a result, $\Pr[X_l = 1 \mid X_1, \dots, X_{l-1}] \geq 1/(1+\varphi)$, that proves the claim. By Lemma 1 (with $p_{\min} = 1/(1+\varphi)$), since we can assume that $|\mathcal{F}| \geq \beta \cdot (b_4/4)^\alpha \cdot \log(8 \cdot Q(N)/\varepsilon)$ one obtains that $D_j \in S_x$ with probability $\geq 1 - \varepsilon/(4 \cdot Q(N))$. Therefore, we can prove Theorem 1 simply by taking a union bound. \square

4 Application: targeting identification

Equipped with Algorithm 1, we will show how it can be used as a routine in order to learn *any* targeting function f with k -juntas as hypotheses. Our algorithm proceeds in two main steps. It first computes the (at most k) relevant inputs of which f depends on, that is the dominant part of the complexity of our algorithm (Section 4.1). Then, it guesses from these inputs the truth table of f (Section 4.2). Furthermore, we will show that the second step requires an additional hypothesis about the noise, that will be proved to be necessary in Section 5.

4.1 Finding the relevant inputs

The relevant inputs will be inferred by virtually “fixing” $k-1$ inputs from the set \mathcal{D} . Such removal will reduce the problem to the identification of a given 1-junta, and so, Algorithm 1 can be used. This is formalized with Algorithm 2.

Input: a family \mathcal{F} ; a lower-bound lb ; threshold parameters x, y .

Output: the set of relevant inputs V .

$V \leftarrow \{\}$;

foreach $\langle \mathcal{C}_{in}, \mathcal{C}_{out} \rangle$ **with** $|\mathcal{C}_{in}| + |\mathcal{C}_{out}| \leq k - 1$ **do**

$\hat{\mathcal{F}} \leftarrow \{A_p \in \mathcal{F} \mid \mathcal{C}_{in} \subseteq A_p \text{ and } A_p \cap \mathcal{C}_{out} = \emptyset\}$;

if $|\hat{\mathcal{F}}| \geq lb$ **then**

$(S_x, S_y) \leftarrow \text{Set-intersection}(\hat{\mathcal{F}}; x, y)$;

$\hat{S} \leftarrow (S_x \cup S_y) \setminus \mathcal{C}_{in}$;

$V \leftarrow V \cup \hat{S}$;

end

end

Algorithm 2: Inference algorithm for the relevant inputs.

Theorem 2. Set $\varphi/(1 + \varphi) < y < 1/2 < x < 1/(1 + \varphi)$. Furthermore, fix any polynomial R . Let \mathcal{B} be a random Bernoulli family of size $|\mathcal{B}| = m$.

There are constant $a_{x,y}, c_{x,y}$ such that if:

- f is a k -junta;
- $lb = a_{x,y} \cdot \log(3 \cdot 2^k \cdot R(N) \cdot N^k / \varepsilon)$;
- $m \geq c_{x,y} \cdot \log^{1/\alpha}(3 \cdot k \cdot 2^{k+2} \cdot R(N) \cdot N^k / \varepsilon)$, and \mathcal{F} contains the targeted accounts in \mathcal{B} ;

then Algorithm 2 is correct with probability $\geq 1 - \varepsilon/R(N)$.

Proof. Let $a_{x,y}, b_{x,y}$ as defined in Lemma 2 and Theorem 1. Set $c_{x,y} = 3 \cdot 2^{k+2} \cdot b_{x,y}$. Finally, let $\langle \mathcal{C}_{in}, \mathcal{C}_{out} \rangle$ be fixed, with $|\mathcal{C}_{in}| + |\mathcal{C}_{out}| \leq k - 1$.

Define $\hat{\mathcal{B}} \subseteq \mathcal{B}$ as the family of all combinations that both contain \mathcal{C}_{in} and do not intersect \mathcal{C}_{out} . Let $\hat{\mathcal{D}} = \mathcal{D} \setminus (\mathcal{C}_{in} \cup \mathcal{C}_{out})$. Furthermore, let $\hat{f}(\hat{\mathcal{C}}) = f(\hat{\mathcal{C}} \cup \mathcal{C}_{in})$ and $\hat{\mathcal{O}}_f(\hat{\mathcal{C}}) = \mathcal{O}_f(\hat{\mathcal{C}} \cup \mathcal{C}_{in})$ for every combination $\hat{\mathcal{C}} \subseteq \hat{\mathcal{D}}$. In order to reuse the results from Section 3, we will base on the property that $\hat{\mathcal{O}}_f$ “almost” behaves like an oracle for the targeting function \hat{f} . That is, it satisfies Assumptions 1 and 2 (trivially), but it only satisfies Assumption 3 partially. More precisely, if f (and not \hat{f}) only depends on some inputs in $\hat{V} \cup \mathcal{C}_{in} \cup \mathcal{C}_{out}$, then Assumption 3 applies for \hat{V} . So, applying Lemma 2 to $\hat{\mathcal{B}}$, since lb is large enough and $\hat{\mathcal{O}}_f$ satisfies the above weaker version of Assumption 3, one obtains that f depends on any input in \hat{S} with probability $\geq 1 - \varepsilon/(3 \cdot 2^{k-1} \cdot R(N) \cdot N^{k-1})$. By taking a union bound over all possible pairs $\langle \mathcal{C}_{in}, \mathcal{C}_{out} \rangle$, it follows that f depends on any input in V with probability $\geq 1 - \varepsilon/(3 \cdot R(N))$.

In order to complete the proof of the theorem, let D_j be any input on which f depends on. Since D_j is relevant, there is a bipartition $\langle \mathcal{C}_{in}, \mathcal{C}_{out} \rangle$ of the relevant inputs from $\mathcal{D} \setminus D_j$ so that $f(\mathcal{C}_{in} \cup \{D_j\}) \neq f(\mathcal{C}_{in})$. So, let us fix any such bipartition $\langle \mathcal{C}_{in}, \mathcal{C}_{out} \rangle$. In such case, \hat{f} is a 1-junta that only depends on D_j , furthermore $\hat{\mathcal{O}}_f$ satisfies Assumption 3 for \hat{f} . The average size of $\hat{\mathcal{B}}$ is $|\mathcal{B}|/2^{k-1}$. So, by Chernoff bound (with $\delta = 1/2$), $\hat{\mathcal{B}}$ has size $\geq b_{x,y} \cdot \log^{1/\alpha}(3 \cdot k \cdot 2^{k+2} \cdot R(N) \cdot N^k / \varepsilon)$ with probability $\geq 1 - \varepsilon/(3 \cdot k \cdot R(N))$. In such case, by Theorem 1, D_j is placed in \hat{S} with probability $\geq 1 - \varepsilon/(3 \cdot R(N) \cdot k)$. So, by taking a union bound over the relevant inputs, every input on which f depends on is in V with probability $\geq 1 - 2\varepsilon/(3 \cdot R(N))$. \square

4.2 Filtering routine

Suppose that we are given the relevant inputs for the targeting function f . In order to learn $\mathcal{S}^{(in)}$, it suffices to learn all the subsets \mathcal{C} on these (at most k) inputs so that $f(\mathcal{C}) = 1$. Intuitively, this can be achieved by comparing any two combinations $\mathcal{C}_0, \mathcal{C}_1$ and testing whether containing one of these two subsets, say, \mathcal{C}_1 , increases the chance to be targeted (compared to \mathcal{C}_0). One may expect that the latter certifies $f(\mathcal{C}_1) = 1$ and $f(\mathcal{C}_0) = 0$. Algorithm 3 (introduced next) builds upon this intuition.

Input: a set of inputs V ; a family \mathcal{F} ; a threshold parameter t .

Output: the class \mathcal{T}_k of all bipartitions $\langle \mathcal{C}_{in}, \mathcal{C}_{out} \rangle$ of V s.t. $\langle \mathcal{C}_{in}, \mathcal{C}_{out} \rangle \in \mathcal{S}^{(in)}$.

$k \leftarrow \lfloor |V| \rfloor$;

Partition \mathcal{F} into $\mathcal{F}_1, \mathcal{F}_2, \dots, \mathcal{F}_{2^k}$ s.t.:

- $\forall 1 \leq i < 2^k, |\mathcal{F}_i| \geq |\mathcal{F}_{i+1}|$;
- $\forall A_p, A_q \in \mathcal{F}, A_p \cap V = A_q \cap V \iff A_p, A_q \in \mathcal{F}_i$ for some i ;

for $i \in \{1, \dots, 2^k\}$ **do**

$V_i \leftarrow A_p \cap V$ with $A_p \in \mathcal{F}_i$;

end

$i_{\text{lim}} \leftarrow \min\{1 \leq i \leq 2^k \mid |\mathcal{F}_i| \geq t \cdot |\mathcal{F}_{i+1}|\}$;

$\mathcal{T}_k \leftarrow \{\langle V_i, V \setminus V_i \rangle \mid 1 \leq i \leq i_{\text{lim}}\}$;

Algorithm 3: Recognition algorithm for the targeting function.

It turns out that subtle complications occur which may prevent Algorithm 3 to be correct with high probability. These can be best defined by introducing a new parameter, that we define next.

Definition 1. *The oracle has positive variance ψ if for any family $\mathcal{F} = \langle A_1, \dots, A_t \rangle$ the following holds for any $\mathcal{C}_1, \mathcal{C}'_1 \in \mathcal{S}^{(in)}$:*

$$\Pr[\mathcal{O}_f(A_i) = 1 \mid A_i = \mathcal{C}_1, \mathbf{H}_{\mathcal{F}_{-i}}] \geq \psi \cdot \Pr[\mathcal{O}_f(A_i) = 1 \mid A_i = \mathcal{C}'_1, \mathbf{H}_{\mathcal{F}_{-i}}].$$

We will show in Section 5 that it is, roughly, the comparison between the positive variance and the *targeting lift* (cf. Assumption 1) that determines what can be learnt within our model.

Theorem 3. *Suppose that the oracle has positive variance $1 \geq \psi > \varphi$, where φ denotes the targeting lift. Set $1/\psi < t < 1/\varphi$, and let \mathcal{B} be a random Bernoulli family of size m .*

There is a constant d_t such that if f is a k -junta that exactly depends on the k inputs in V , $m \geq d_t \cdot \log^{1/\alpha}(3 \cdot 2^{2k+1}/\varepsilon)$, and \mathcal{F} contains the targeted accounts in \mathcal{B} , then Algorithm 3 outputs f with probability $\geq 1 - \varepsilon$.

Proof. Set $d_1 = 12, d_2 = \frac{1}{\gamma}$ and $d_3 = 2^{k+1} \cdot \left(\frac{3 \cdot (1+t)^2}{\beta \cdot (1+\varphi) \cdot (1-t\varphi)^2}\right)^{1/\alpha}, d_4 = 2^{k+1} \cdot \left(\frac{3 \cdot (1+t)^2}{\beta \cdot (1+\psi) \cdot (t\psi-1)^2}\right)^{1/\alpha}$. Finally, set $d_t = \max_{1 \leq i \leq 4} d_i$.

Fix any pair $V_i, V_j \subseteq V$ so that $f(V_i) = 1$ (there are $\leq 4^k$ such pairs). Define \mathcal{B}_i , resp. \mathcal{B}_j , as the subfamily of all accounts $A_p \in \mathcal{B}$ so that $A_p \cap V = V_i$, resp. $A_p \cap V = V_j$. Since $\mathbb{E}[|\mathcal{B}_i|] = |\mathcal{B}|/2^k$ and $m = |\mathcal{B}|$ is large enough, by Chernoff bound (Lemma 1 with $\delta = 1/2$), we have that $|\mathcal{B}_i| \geq |\mathcal{B}|/2^{k+1}$ with probability $1 - \varepsilon/(3 \cdot 4^k)$. The latter implies by Assumption 2, $|\mathcal{F}_i| \geq \beta(\max\{d_3, d_4\}/2^{k+1})^\alpha \cdot \log(3 \cdot 2^{2k+1}/\varepsilon)$ with probability $\geq 1 - \varepsilon/(3 \cdot 4^k)$. We will show that, with high probability, $|\mathcal{F}_i| \geq t \cdot |\mathcal{F}_j|$ if and only if $f(V_j) = 0$, that will prove the theorem.

Denote by $\langle A_{p_1}, A_{p_2}, \dots, A_{p_s} \rangle$ the subfamily of all accounts in $\mathcal{F}_i \cup \mathcal{F}_j$. Let X_l be the random Boolean variable denoting whether $A_{p_l} \in \mathcal{F}_i$. Fix the set of indices I of all accounts within $\mathcal{B}_i \cup \mathcal{B}_j$. Similarly, fix the subset $J \subseteq I$ of the s accounts within $\mathcal{F}_i \cup \mathcal{F}_j$, and fix the variables X_1, X_2, \dots, X_{l-1} . Consider the set Ω of the histories $(\mathcal{B}, \mathcal{O})$ satisfying:

- all the entries of \mathcal{O} that are indexed by J are nonzero (there may be other nonzero entries);
- all the entries of \mathcal{O} that are indexed by $I \setminus J$ are equal to zero;
- for every $A_p \in \mathcal{B}$, we have that $A_p \in \mathcal{F}_i \cup \mathcal{F}_j$ if and only if $p \in I$;

- for every $1 \leq q \leq l-1$, $A_{p_q} \in \mathcal{F}_i$ if and only if $X_q = 1$.

Let $h_l : \langle A_1, \dots, A_{p_l-1}, A_{p_l}, A_{p_l+1}, \dots, A_m \rangle \rightarrow \langle A_1, \dots, A_{p_l-1}, A'_{p_l}, A_{p_l+1}, \dots, A_m \rangle$ where $A_{p_l} \setminus V = A'_{p_l} \setminus V$ and $\{A_{p_l} \cap V, A'_{p_l} \cap V\} = \{V_i, V_j\}$. Since $(\mathcal{B}, \mathcal{O}) \rightarrow (h_l(\mathcal{B}), \mathcal{O})$ is one-to-one, there is half of the histories in Ω which satisfy X_l .

On the one direction, assume $f(V_j) = 0$. For every $(\mathcal{B}, \mathcal{O}) \in \Omega$ such that X_l is satisfied, by Assumption 1:

$$\frac{\Pr[\mathcal{O}_f(\mathcal{B}) = \mathcal{O}]}{\Pr[\mathcal{O}_f(h_l(\mathcal{B})) = \mathcal{O}]} = \frac{\Pr[\mathcal{O}_f(A_{p_l}) = 1 \mid A_{p_l} \cap V = V_i, \mathbf{H}_{\mathcal{B}-p_l} = \mathcal{O}_{-p_l}]}{\Pr[\mathcal{O}_f(A_{p_l}) = 1 \mid A_{p_l} \cap V = V_j, \mathbf{H}_{\mathcal{B}-p_l} = \mathcal{O}_{-p_l}]} > 1/\varphi.$$

As a result, $\Pr[X_l = 1 \mid X_1, \dots, X_{l-1}] \geq 1/(1+\varphi) > t/(1+t)$. By Lemma 1 (with $p_{\min} = 1/(1+\varphi)$), since $|\mathcal{F}_i| \leq |\mathcal{F}_i| + |\mathcal{F}_j|$ is large enough, we have that that $|\mathcal{F}_i| \geq \frac{t}{1+t} \cdot (|\mathcal{F}_i| + |\mathcal{F}_j|)$ with probability $\geq 1 - \varepsilon/(3 \cdot 4^k)$, and so, $|\mathcal{F}_i| \geq t \cdot |\mathcal{F}_j|$ with high probability.

On the other direction, assume $f(V_j) = 1$. For every $(\mathcal{B}, \mathcal{O}) \in \Omega$ such that X_l is satisfied, by Definition 1:

$$\frac{\Pr[\mathcal{O}_f(\mathcal{B}) = \mathcal{O}]}{\Pr[\mathcal{O}_f(h_l(\mathcal{B})) = \mathcal{O}]} = \frac{\Pr[\mathcal{O}_f(A_{p_l}) = 1 \mid A_{p_l} \cap V = V_i, \mathbf{H}_{\mathcal{B}-p_l} = \mathcal{O}_{-p_l}]}{\Pr[\mathcal{O}_f(A_{p_l}) = 1 \mid A_{p_l} \cap V = V_j, \mathbf{H}_{\mathcal{B}-p_l} = \mathcal{O}_{-p_l}]} \leq 1/\psi.$$

As a result, $\Pr[X_l = 1 \mid X_1, \dots, X_{l-1}] \leq 1/(1+\psi) < t/(1+t)$. By Lemma 1 (with $p_{\max} = 1/(1+\psi)$), since $|\mathcal{F}_i| \leq |\mathcal{F}_i| + |\mathcal{F}_j|$ is large enough, we have that that $|\mathcal{F}_i| < \frac{t}{1+t} \cdot (|\mathcal{F}_i| + |\mathcal{F}_j|)$ with probability $\geq 1 - \varepsilon/(3 \cdot 4^k)$, and so, $|\mathcal{F}_i| < t \cdot |\mathcal{F}_j|$ with high probability. \square

Note that in [7], the oracle has positive variance 1. Therefore, it follows from Theorem 3 that *any* targeting function is learnable in this simpler model.

Furthermore, we notice that Algorithm 3 (or slight variations of it) can be proved correct under other assumptions that are similar in spirit as Definition 1. For instance, the oracle has *negative variance* ψ' if for any family $\mathcal{F} = \langle A_1, \dots, A_t \rangle$ the following holds for any $\mathcal{C}_0, \mathcal{C}'_0 \notin \mathcal{S}^{(\text{in})}$:

$$\Pr[\mathcal{O}_f(A_i) = 1 \mid A_i = \mathcal{C}_0, \mathbf{H}_{\mathcal{F}-i}] \geq \psi' \cdot \Pr[\mathcal{O}_f(A_i) = 1 \mid A_i = \mathcal{C}'_0, \mathbf{H}_{\mathcal{F}-i}].$$

When the oracle has negative variance $\psi' > \varphi$ the targeting function can be learnt by replacing line 3 in Algorithm 3 with $i_{\text{lim}} \leftarrow \max\{1 \leq i \leq 2^k \mid |\mathcal{F}_i| \geq t \cdot |\mathcal{F}_{i+1}|\}$. Nonetheless, it remains elusive to learn the targeting function *without* any additional information. We will explain in the subsequent section why no such algorithm exists.

5 Impossibility results

Intuitively, the targeting function f can be learnt only if the *targeting lift* can be detected (cf. Assumption 1). Our additional assumptions on the positive, resp. negative, variance in Section 4.2 are ways to detect the lift using a simple leftmost, resp. rightmost, approach. In the general case when no such assumption holds, ambiguity may occur that prevents from detecting f with high certainty.

Proposition 1. *It is impossible to learn the targeting function f in general. In particular, there is a given monotone 2-junta that cannot be learnt even if the targeting lift is arbitrarily small.*

Proof. In order to prove the result, we will construct an oracle \mathcal{O}_f that satisfies Assumptions 1, 2 and 3 for two distinct targeting functions. The latter is enough to prove the proposition since in such case, \mathcal{O}_f could be used in our model for any of the two functions, and so, these cannot be distinguished with high probability. More precisely, fix $0 < p_0 < 1/5$. Let us define \mathcal{O}_f such that for any combination \mathcal{C} :

$$\Pr[\mathcal{O}_f(\mathcal{C}) = 1] = p_0 \cdot (1 + 2 \cdot \mathbb{I}_{\{D_1 \in \mathcal{C}\}} + 2 \cdot \mathbb{I}_{\{D_2 \in \mathcal{C}\}}).$$

Since every combination has positive probability to be targeted and the above oracle considers the combinations independently, by Chernoff bound, \mathcal{O}_f satisfies Assumption 2 with $\alpha = 1$ for any targeting

function. Furthermore, \mathcal{O}_f satisfies Assumption 3 for any targeting function that only depends on the two inputs D_1, D_2 . In particular, let $f_1(\mathcal{C}) = \max\{\mathbb{I}_{\{D_1 \in \mathcal{C}\}}, \mathbb{I}_{\{D_2 \in \mathcal{C}\}}\}$ and let $f_2(\mathcal{C}) = \mathbb{I}_{\{D_1 \in \mathcal{C}\}} \cdot \mathbb{I}_{\{D_2 \in \mathcal{C}\}}$. These two functions are monotone. In fact, they are linear combinations since $f_1(\mathcal{C}) = 1 \iff \mathbb{I}_{\{D_1 \in \mathcal{C}\}} + \mathbb{I}_{\{D_2 \in \mathcal{C}\}} \geq 1$, and similarly $f_2(\mathcal{C}) = 1 \iff \mathbb{I}_{\{D_1 \in \mathcal{C}\}} + \mathbb{I}_{\{D_2 \in \mathcal{C}\}} \geq 2$. For both functions, the oracle \mathcal{O}_f satisfies Assumption 1 with any targeting lift $\varphi > 1/2$.

Finally, note that in order to extend this negative result to lifts arbitrarily smaller than $1/2$, one may just consider a slightly more complex oracle for the above two functions f_1, f_2 , namely:

$$\Pr[\mathcal{O}_f(\mathcal{C}) = 1] = q_0^{3 - \mathbb{I}_{\{D_1 \in \mathcal{C}\}} - \mathbb{I}_{\{D_2 \in \mathcal{C}\}}},$$

for some nonzero probability q_0 that can be taken arbitrarily small. \square

We point out that the negative result of Proposition 1 applies to the particular case when the targeting function is a linear combination of inputs. Hence, it applies to the theory behind the tools XRay [7] and Sunlight [8], that assume the targeting function is monotone or a linear combination of inputs respectively.

References

- [1] D. Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342.
- [2] J. Arpe and R. Reischuk. Learning juntas in the presence of noise. *Theoretical Computer Science*, 384(1):2–21, 2007.
- [3] A. Blum. Relevant examples and relevant features: Thoughts from computational learning theory. In *AAAI Fall Symposium on Relevance*, volume 5, 1994.
- [4] A. Datta, M. Tschantz, and A. Datta. Automated experiments on ad privacy settings. *Proceedings on Privacy Enhancing Technologies*, 2015(1):92–112, 2015.
- [5] G. Ducoffe, M. Lécuyer, A. Chaintreau, and R. Geambasu. Web transparency for complex targeting: Algorithms, limits, and tradeoffs. In *Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS ’15, pages 465–466, New York, NY, USA, 2015. ACM.
- [6] G. Ducoffe, M. Tucker, and A. Chaintreau. Can web transparency cope with complex targeting? Submitted., 2016.
- [7] M. Lécuyer, G. Ducoffe, F. Lan, A. Papancea, T. Petsios, R. Spahn, A. Chaintreau, and R. Geambasu. Xray: Enhancing the web’s transparency with differential correlation. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 49–64, San Diego, CA, Aug. 2014. USENIX Association.
- [8] M. Lecuyer, R. Spahn, Y. Spiliopolous, A. Chaintreau, R. Geambasu, and D. Hsu. Sunlight: Fine-grained targeting detection at scale with statistical confidence. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security, CCS ’15*, pages 554–566, New York, NY, USA, 2015. ACM.
- [9] M. Tschantz, A. Datta, A. Datta, and J. Wing. A methodology for information flow experiments. In *Computer Security Foundations Symposium (CSF), 2015 IEEE 28th*, pages 554–568, July 2015.