# Laying Out Sparse Graphs with Provably Minimum Bandwidth

Alberto Caprara* Juan-José Salazar-González†

February 2003

**Abstract**

Finding a linear layout of a graph having minimum bandwidth is a combinatorial optimization problem that was studied since the 60s. Differently from other classical problems, the approach based on stating a suitable integer linear programming formulation and solving the associated linear programming relaxation seems to be useless in this case. This makes it nontrivial to design algorithms capable of solving to optimality instances of reasonable size. In this paper, we illustrate a new simple lower bound on the optimal bandwidth and its extension within an enumerative algorithm, leading to integer linear programming relaxations that can be solved efficiently and provide effective lower bounds if part of the layout is fixed. Keeping the integrality constraints in these relaxations is essential for this purpose. We show that the resulting method can solve to proven optimality 24 out of the 30 instances with less than 200 nodes from the literature, each within less than one minute on a personal computer. The new approach is also analyzed on randomly generated instances with up to 1000 nodes. Moreover, we propose for the first time a method to compute the well-known density lower bound on the optimal bandwidth, that succeeds in finding this bound within minutes for most literature instances with up to 250 nodes.

**Key words:** bandwidth, lower bound, integer linear programming relaxation, enumerative algorithm, computational results.

## 1 Introduction

The *bandwidth* problem, calling for a linear layout of a graph in which the maximum distance between adjacent nodes is minimized, is a classical problem in combinatorial optimization that finds its main application in the solution of systems of linear equations. Specifically, finding a minimum bandwidth layout of a graph is the same as permuting the rows and columns of a symmetric square matrix so as to minimize the distance of the nonzero elements from the main diagonal, and systems of linear equations are easier to solve if this distance is small [3].

One of the main characteristics of the bandwidth problem is that the methods which are successful for other important combinatorial optimization problems do not seem to be particularly useful for its solution. For instance, it is apparently useless to try to formulate the problem as an integer linear program and then apply exact or heuristic methods based on the resulting

*DEIS, University of Bologna, Italy (acaprara@deis.unibo.it)
†DEIOC, University of La Laguna, Spain (jjsalaza@ull.es)

linear programming relaxation, since the corresponding lower bounds are extremely bad. A well-known strong lower bound on the minimum bandwidth of a graph introduced in the early 70s [5] is the so-called *density* lower bound, that was shown to be on average almost equal to the minimum bandwidth (under an appropriate distribution of the instances) [19]. On the other hand, it is unclear how to compute this bound as well as how to use it within an enumerative algorithm to fathom subproblems for which part of the solution is fixed (typically, the first/last nodes in the layout). It is also known that the problem can be solved in polynomial time by dynamic programming if the bandwidth is bounded by a constant [16], but the exponent of the polynomial for both the time and the space complexity is given by the bandwidth itself, which makes the method impractical when this value is not very small.

As a consequence of the above situation, although successful ad hoc heuristics have been proposed for the problem (see e.g. [6, 14, 15, 9, 10, 18, 8]), no exact algorithms nor strong lower bounding procedures capable of tackling instances of reasonable size can be found in the literature. The only recent exception is [7], that illustrates an enumerative algorithm implicitly based on a very simple lower bound that can be computed in a very effective way. This algorithm works very well for dense graphs, essentially because for these graphs the simple bound coincides with other stronger bounds, whereas it is not effective for sparse instances as those that are encountered in practice. Note that the optimal bandwidth value is unknown for most instances in the "Matrix Market" collection, which is the main source of instances for the problem.

Although spending long time to minimize the bandwidth of a single system of linear equations is certainly not worth to speed-up the solution of the system itself, finding the exact value of the bandwidth for a relevant set of test instances is a challenging optimization problem in itself, and its outcome can be used to test the effectiveness of practical heuristics. Moreover, there are cases in which many systems of linear equations with the same nonzero pattern must be solved, in which case it is worth investing time in the corresponding (unique) bandwidth instance since this is going to speed-up the solution of all the systems.

In this work, we first analyze some lower bounds on the minimum bandwidth, discussing the complexity of their computation and some dominance relations among them. Specifically, we show that the computation of the density lower bound is $\mathcal{NP}$-hard, and introduce a new lower bound that can be found efficiently and is suited for use within an enumerative scheme. Interpreting this latter lower bound as the optimal value of a suitable integer linear programming relaxation of the problem leads naturally to a stronger bound to be used within enumeration, still efficient to compute. We stress that in this relaxation variables are restricted to be *integer* (if this condition was relaxed, the model would be useless). We further tighten this relaxation, showing that a better lower bound can be found within short time by solving a suitable *bilevel* integer linear program. The resulting branch-and-bound method is quite successful, in that it is capable of solving to proven optimality 24 out of the 30 instances from the literature associated with graphs having up to 200 nodes, each within less than one minute, whereas only the 10 easiest instances can be solved by the method of [7]. We also show how our method behaves

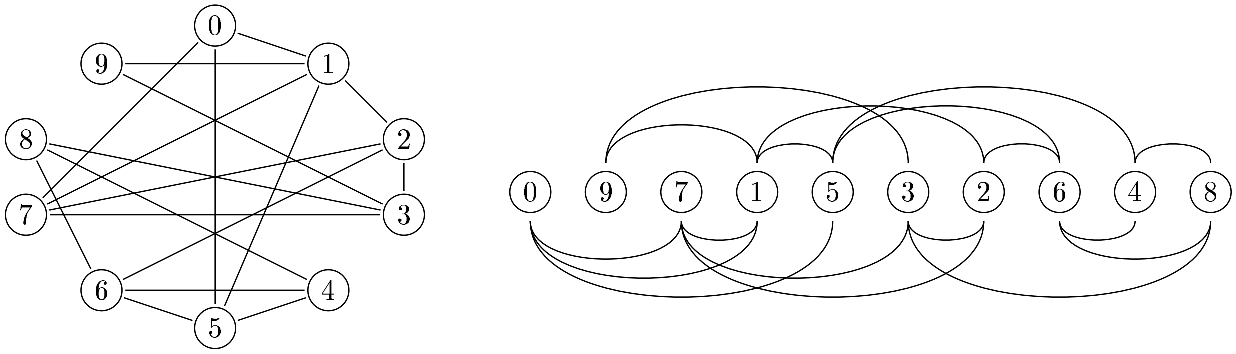


Figure 1: A graph and an associated optimal layout.

for random instances, confirming its effectiveness when the graph is sparse. Finally, we devise a simple but effective method to compute the density lower bound, whose value is reported for all instances in the literature with up to 250 nodes except one.

## Basic definitions and notation

Consider an undirected connected graph $G = (V, E)$ and let $n := |V|$ and $m := |E|$. A *(linear) layout* of $G$ is a total ordering of the nodes of $G$, defined by a one-to-one function $\tau : V \to \{1, \dots, n\}$. For notational convenience, given $v \in V$, we will write $\tau_v$ for $\tau(v)$. Moreover, for a given layout, we will often imagine the nodes on a horizontal line, from *left to right* according to increasing values of $\tau$, saying that $u$ is *on the left* of $v$ if $\tau_u < \tau_v$. The *bandwidth of the layout*, denoted by $\phi_\tau(G)$, is the maximum difference between the numbers assigned to adjacent nodes, i.e. $\phi_\tau(G) := \max\{|\tau_u - \tau_v| : (u, v) \in E\}$. The *bandwidth of* $G$, denoted by $\phi(G)$, is the minimum bandwidth of a layout of $G$, i.e. $\phi(G) := \min\{\phi_\tau(G) : \tau \text{ layout of } G\}$. Figure 1 shows a graph $G$ with 10 nodes and an associated (optimal) layout $\tau$ with $\phi_\tau(G) = \phi(G) = 4$. For convenience, we will refer to the nodes of $G$ with the numbers from 0 to 9 as indicated in the figure.

A *partial layout* for node set $S \subseteq V$ is a function $\tau : S \to \{1, \dots, n\}$ such that $\tau_u \neq \tau_v$ for $u, v \in S, u \neq v$. Given a partial layout $\tau$, we will let $\phi_\tau(G)$ denote the minimum bandwidth of a layout obtained by extending $\tau$.

Given two nodes $u, v \in V$, we let $d(u, v)$ denote the *distance* between $u$ and $v$, i.e. the minimum number of edges in a path from $u$ to $v$. Given two (not necessarily disjoint) node sets $S, T \subseteq V$, we let

$$d(S, T) := \max\{d(u, v) : u \in S, v \in T\}$$

denote the maximum distance between a node in $S$ and a node in $T$. Value $d(S, S)$ is called the *diameter* of $S$, and will be denoted by $d(S)$. Moreover, we will write $d(u, T)$ for $d(\{u\}, T)$ and $d(S, v)$ for $d(S, \{v\})$.

For a node $u \in V$ and $k \leq d(u, V)$, we let

$$N_k(u) := \{v \in V : d(u, v) \leq k\}$$

denote the set of nodes at distance at most $k$ from $v$. The set $N_1(u)$ is the set of *neighbors* of $u$ (including $u$ itself). Note that *breadth-first search* finds in $O(m)$ time all sets $N_1(u), \ldots, N_{d(u,V)}(u)$ for a given node $u$. For $S \subseteq V$, we let $N_k(S) := \bigcup_{v \in S} N_k(v)$.

Finally, for a graph $G$, we let $\omega(G)$ denote the maximum cardinality of a *clique* of $G$, i.e.

$$\omega(G) := \max\{|S| : S \subseteq V, (u,v) \in E \text{ for all } u,v \in S, u \neq v\}.$$

# 2 Old and New Lower Bounds

In this section we will first present the most famous lower bound on $\phi(G)$, known as the *density lower bound*, showing that it is hard to compute and illustrating a related lower bound that can be found efficiently. Then, we present a new lower bound which is suited for use within an enumerative scheme (which does not seem to be the case for the density bound and its variation) and illustrate the use of this bound once a partial layout has been fixed.

## 2.1 The density lower bound

The density lower bound, apparently proposed first by Chvátal [5], is defined by

$$\beta(G) := \max_{S \subseteq V} \left\lceil \frac{|S|-1}{d(S)} \right\rceil. \tag{1}$$

For the graph $G$ in Figure 1 we have $\beta(G) = 3$, achieved e.g. for $S = \{0,1,2,5,7,9\}$.

The (very simple) proof of the following proposition is given explicitly since the same argument will be used to show the validity of other lower bounds.

**Proposition 1** *For every graph $G$, $\phi(G) \geq \beta(G)$.*

**Proof.** In any layout $\tau$ of $G$, the shortest path $P$ using only nodes in $S$ and joining the leftmost node $u$ in $S$ ($\tau_u = \min\{\tau_v : v \in S\}$) to the rightmost node $w$ in $S$ ($\tau_w = \max\{\tau_v : v \in S\}$) contains no more than $d(S)$ edges. This means that

$$d(S) \cdot \max\{|\tau_{v_1} - \tau_{v_2}| : (v_1,v_2) \in P\} \geq \sum\{|\tau_{v_1} - \tau_{v_2}| : (v_1,v_2) \in P\} \geq \tau_w - \tau_u \geq |S| - 1$$

and therefore $\max\{|\tau_{v_1} - \tau_{v_2}| : (v_1,v_2) \in P\} \geq \left\lceil \frac{|S|-1}{d(S)} \right\rceil$. $\square$

Although $\beta(G)$ is a very well known lower bound, we could not find any reference in which the complexity of its computation is proved. This is somehow surprising since this lower bound tends to be close to $\phi(G)$ in practice [6, 14], as it is also partially explained by probabilistic analysis [19]. Actually, it is easy to see the following:

**Proposition 2** *Computing $\beta(G)$ is $\mathcal{NP}$-hard.*

**Proof.** We give a simple reduction from the problem of finding the size $\omega(H)$ of a maximum clique of a graph $H = (W,F)$, which is well known to be $\mathcal{NP}$-hard [13]. Let $G$ be obtained from $H$ by adding $|W|$ new nodes to $W$, which are all pairwise connected as well as connected to all nodes in $W$. Clearly, $G$ has $2|W|$ nodes and $\omega(G) = \omega(H) + |W|$ (hence $\omega(G) - 1 \geq |W|$). Moreover, $\beta(G) = \omega(G) - 1$ since taking $S$ as a maximum clique of $G$ in (1) (with $d(S) = 1$) yields $\left\lceil \frac{|S|-1}{d(S)} \right\rceil = \omega(G) - 1 \geq |W|$, whereas taking $S$ as any set which is not a clique (with $d(S) \geq 2$) yields $\left\lceil \frac{|S|-1}{d(S)} \right\rceil \leq \left\lceil \frac{2|W|-1}{2} \right\rceil = |W|$. This implies that computing $\beta(G)$ is the same as computing $\omega(H)$, completing the proof. □

The following expression for $\beta(G)$, clearly equivalent to (1), points out the analogies and differences with the other lower bounds of this section

$$\beta(G) = \max_{S \subseteq V} \min_{v \in S} \left\lceil \frac{|S| - 1}{d(v, S)} \right\rceil . \tag{2}$$

The worst-case ratio between $\phi(G)$ and $\beta(G)$ is not known exactly. The following proposition summarizes the current state-of-the-art, the upper bound being from [11] and the lower bound from [4].

**Proposition 3** *For every graph $G$, $\phi(G) = O(\log^3 n \sqrt{\log n \log\log n}) \cdot \beta(G)$. Moreover, there exist graphs $G$ for which $\phi(G) = \Omega(\log n) \cdot \beta(G)$.*

Reference [1] illustrates a 1/2-approximation algorithm for the computation of $\beta(G)$. Actually, the algorithm computes another lower bound $\alpha(G)$ on $\phi(G)$ whose value is in $[\beta(G)/2, \beta(G)]$. Since the algorithm is only sketched in [1] and its performance guarantee not proved formally, we give a precise (short) description here, also considering that the new lower bound that we will propose in Section 2.3 is somehow related to $\alpha(G)$. This lower bound is defined by

$$\alpha(G) := \max_{v \in V} \max_{S \subseteq V : v \in S} \left\lceil \frac{|S| - 1}{2d(v, S)} \right\rceil . \tag{3}$$

For the graph $G$ in Figure 1 we have $\alpha(G) = 3 (= \beta(G))$, achieved again for $S = \{0, 1, 2, 5, 7, 9\}$ and $v = 1$ (with $d(v, S) = 1$).

The next proposition shows the relationship between $\alpha(G)$ and $\beta(G)$ along with a simple way to compute $\alpha(G)$, essentially due to [1].

**Proposition 4** *For every graph $G$, $\beta(G)/2 \leq \alpha(G) \leq \beta(G)$. Moreover, $\alpha(G)$ can be computed in $O(nm)$ time.*

**Proof.** We first derive an equivalent expression for $\alpha(G)$ that leads immediately to an efficient algorithm for its computation. Consider the node $v^* \in V$ and node set $S^* \subseteq V$ ($v^* \in S^*$) for which the maximum in (3) is attained, letting $k^* := d(v^*, S^*)$. $S^*$ is the largest set such that $d(v^*, S^*) = k^*$, i.e. $S^* = N_{k^*}(v^*)$. This implies

$$\alpha(G) = \max_{v \in V} \max_{k=1}^{d(v,V)} \left\lceil \frac{|N_k(v)| - 1}{2k} \right\rceil , \tag{4}$$

which means that $\alpha(G)$ can be computed in $O(nm)$ time by finding, for each node $v \in V$, sets $N_k(v)$ for $k = 1, \ldots, d(v, V)$ in $O(m)$ time.

Since all nodes in $S^*$ are at distance at most $k^*$ from $v^*$, we have $d(S^*) \leq 2k^*$, which means

$$\beta(G) \geq \left\lceil \frac{|S^*|-1}{d(S^*)} \right\rceil \geq \left\lceil \frac{|S^*|-1}{2k^*} \right\rceil = \alpha(G).$$

In order to show $\alpha(G) \geq \beta(G)/2$, consider now a set $S'$ for which the maximum in (1) is attained and let $v' \in S'$. Clearly, all nodes in $S'$ are at distance at most $k' := d(S')$ from $v'$, i.e. $S' \subseteq N_{k'}(v')$. Therefore

$$\alpha(G) \geq \left\lceil \frac{|N_{k'}(v')|-1}{2k'} \right\rceil \geq \left\lceil \frac{|S'|-1}{2d(S')} \right\rceil \geq \frac{\beta(G)}{2}.$$

□

If $G$ is complete, then $\beta(G) = n-1$ and $\alpha(G) = \lceil (n-1)/2 \rceil$, whereas if $G$ is a path $\beta(G) = \alpha(G) = 1$, showing that both inequalities in the statement of Proposition 4 may be tight.

The following trivial lower bound, which is weaker than $\alpha(G)$, is the starting lower bound used in the enumerative algorithm by [7]:

$$\varepsilon(G) = \max_{v \in V} \left\lceil \frac{|N_1(v)|-1}{2} \right\rceil. \tag{5}$$

For the graph $G$ in Figure 1, $\varepsilon(G) = 3$, achieved for $v = 1$.

## 2.2 Computing $\beta(G)$

Although $\beta(G)$ is by far the most famous lower bound on $\phi(G)$, we are not aware of any attempt to compute it exactly or even approximately with the exception of the computation of $\alpha(G)$ (never tested experimentally, to the best of our knowledge). Actually, there is apparently no reasonable ILP formulation for $\beta(G)$ and therefore, as is the case for $\phi(G)$, one has to resort to nonstandard methods.

Let $S^*$ be the set leading to the maximum in (1). As already noted before, if $d(S^*) = 1$, computing $\beta(G)$ is the same as finding the size $\omega(G)$ of a largest clique of $G$. Here we propose an enumerative scheme for the computation of $\beta(G)$ which is based on guessing the value $d$ of $d(S^*)$ and searching for a set $S$ with $d(S) = d$ and $|S|$ as large as possible. The key idea for an effective upper bound on the size of such an $S$ is given by the following proposition, whose proof is obvious. For $W \subseteq V$ and $d \in \{1, \dots, n-1\}$, define the *auxiliary graph* $A(d, W) := (W, D)$ whose edges represent the fact that the length $d_W(u, v)$ of the shortest path between two nodes $u, v \in W$ *using only nodes in* $W$ has length at most $d$, i.e. $D := \{(u, v) : u, v \in W, d_W(u, v) \leq d\}$.

**Proposition 5** *For every set $S$ with $d(S) = t$, $|S| \leq \omega(A(t, V))$.*

This simple result inspires the recursive enumerative procedure FIND_MAX_S given in Figure 2, which is initially called with $F := V$, $S := \emptyset$, and $S^* := \emptyset$, where $S$ is the current (partial) candidate set, $F$ is the set of nodes still not fixed either inside $S$ or outside $S$, and $S^*$ is the best set found so far. For a partial set $S$, we can impose outside $S$ all the nodes $v$ such

```
procedure FIND_MAX_S();
begin
    if F = ∅ then
        if |S| > |S*| then S* := S;
        return;
    end if;
    if ω(A(d, S ∪ F)) ≤ |S*| then return;
    for each v ∈ F do
        comment: impose v in S;
        F := F \ {v}; S := S ∪ {v};
        F' := F; F := F \ {w ∈ F : d_{S∪F}(v, w) > d};
        UPDATE_DISTANCES();
        FIND_MAX_S();
        comment: impose v outside S;
        F := F'; S := S \ {v};
        UPDATE_DISTANCES();
        FIND_MAX_S();
        F := F ∪ {v};
    end for each;
end.
```

Figure 2: Enumerative procedure to find $S^*$ with $d(S^*) \leq d$ and $|S^*|$ maximum.

that $d(v, S) > d$. Then, the upper bound test according to the proposition above amounts to checking whether the largest clique in $A(d, S \cup F)$ is larger than $|S^*|$ (if this is not the case we can backtrack). Note that such a clique will contain all the nodes in $S$. In practice, finding the size of a largest clique in a graph of relatively small size is quite fast by using combinatorial enumerative algorithms [17]. Actually, the computational bottleneck of the algorithm above turns out to be the re-computation of the distances $d_{S \cup F}(u, v)$ each time some nodes are imposed outside $S \cup F$, by procedure UPDATE_DISTANCES. The distances are then used to remove nodes from $F$ when another node is imposed in $S$.

### 2.3 A new lower bound

We now present a new lower bound on $\phi(G)$, which can be found in $O(nm)$ time. The major advantage of this bound over $\beta(G)$ is that it is a natural bound to use within an enumerative scheme to find $\phi(G)$, as explained in the next section. The bound is

$$\gamma(G) := \min_{v \in V} \max_{S \subseteq V, v \in S} \left\lceil \frac{|S| - 1}{d(v, S)} \right\rceil . \tag{6}$$

For the graph $G$ in Figure 1 we have $\gamma(G) = 4$, achieved e.g. for $v = 4$ and $S = V \setminus \{7, 9\}$ (with $d(v, S) = 2$).

**Proposition 6** *For every graph $G$, $\phi(G) \geq \gamma(G)$. Moreover, $\gamma(G)$ can be computed in $O(nm)$ time.*

**Proof.** By the same observation as in the proof of Proposition 4, we have

$$\gamma(G) = \min_{v\in V} \max_{k=1}^{d(v,V)} \left\lceil \frac{|N_k(v)| - 1}{k} \right\rceil, \tag{7}$$

i.e. also the determination of $\gamma(G)$ requires the determination, in $O(nm)$ time, of sets $N_k(v)$ for all $v \in V$ and $k = 1, \ldots, d(v, V)$.

To show $\phi(G) \geq \gamma(G)$, let $u$ be the leftmost node in an optimal layout $\tau$ of $G$ ($\tau_u = 1$) and consider the set $S^*$ such that $u \in S^*$ and $\frac{|S^*|-1}{d(u,S^*)}$ is maximum. Letting $w$ be the rightmost node in $S^*$ in layout $\tau$, the same argument as in the proof of Proposition 1 shows that the bandwidth of $\tau$ is at least $\left\lceil \frac{|S^*|-1}{d(u,S^*)} \right\rceil$. This implies

$$\gamma(G) \leq \max_{S\subseteq V, u\in S} \left\lceil \frac{|S| - 1}{d(u,S)} \right\rceil = \left\lceil \frac{|S^*| - 1}{d(u,S^*)} \right\rceil \leq \phi_\tau(G) = \phi(G).$$

□

Note that if $G$ is complete $\gamma(G) = \beta(G) = 2\alpha(G) = n - 1$. Moreover, if $G$ is obtained by joining a complete graph $K_p$ on $p$ nodes with a path $u_0\ u_1 \ldots\ u_p$, identifying $u_p$ with one of the nodes in $K_p$, we have $n = 2p$, $\beta(G) = \phi(G) = p - 1$ and $\gamma(G) = \left\lceil \frac{2p-1}{p+1} \right\rceil \leq 2$ (the node for which the minimum in (6) is attained is $u_0$). In other words, we may have $\phi(G) = \Omega(n) \cdot \gamma(G)$, which is a big difference in terms of worst-case performance guarantee of $\gamma(G)$ with respect to $\beta(G)$ (see Proposition 3). Finally, note that the graph $G$ in Figure 1 shows that $\gamma(G) > \beta(G)$ is possible.

Although the worst-case performance of $\gamma(G)$ can be much worse than that of $\beta(G)$ (and $\alpha(G)$), $\gamma(G)$ is of comparable quality in practice and is a natural lower bound to use within enumerative schemes to compute $\phi(G)$, in which layouts are constructed from left to right. Of course, if $u$ is the leftmost node in the layout, the associated bandwidth cannot be less than $\max_{S\subseteq V, u\in S} \left\lceil \frac{|S|-1}{d(u,S)} \right\rceil$. A careful elaboration of this basic idea leads to much stronger lower bounds if more than one node is fixed in the layout, as discussed in Section 3.

Another lower bound on $\phi(G)$, which is obvious and implicitly used in the enumerative scheme for the problem proposed in [7], is obtained by considering in (7) only $k = 1$, namely the bound is

$$\delta(G) := \min_{v\in V} |N_1(v)| - 1. \tag{8}$$

For the graph $G$ in Figure 1 we have $\delta(G) = 2$, achieved e.g. for $v = 9$ (with $N_1(v) = \{1, 3, 9\}$).

The following is of course a corollary of Proposition 6, which is stated formally for future reference.

**Corollary 1** *For every graph $G$, $\phi(G) \geq \delta(G)$.*

### 2.4 Lower bounds for partial layouts

The practical use of lower bounds $\delta(G)$ and $\gamma(G)$ within an enumerative scheme is the following. Suppose enumeration has already fixed the, say, $k$ *leftmost* nodes in the layout $u_1\ u_2\ \ldots\ u_k$

with $\tau_{u_i} = i$ for $i = 1, \ldots, k$. Let $L := \{u_1, \ldots, u_k\}$, $F := V \setminus L$ be the set of *free* nodes, and recall that $\phi_\tau(G)$ denotes the minimum bandwidth of a layout obtained by extending the partial layout $\tau$. Of course,

$$\phi_\tau(G) \geq \max\{i - h : u_i, u_h \in L, (u_i, u_h) \in E\}. \tag{9}$$

This trivial lower bound, which is of course embedded into our implementation, will not be mentioned explicitly for the rest of the paper, with few exceptions.

The following are the simple extensions of Proposition 6 and Corollary 1 taking into account the partial layout. For $u_i \in L$, let $r_i := \min\{h : N_h(u_i) \cap F \neq \emptyset\}$.

**Proposition 7**

$$\phi_\tau(G) \geq \max_{i=1}^{k} \max_{h=r_i}^{d(u_i,F)} \left\lceil \frac{|N_h(u_i) \cap F| + k - i}{h} \right\rceil. \tag{10}$$

**Proof.** Consider some $u_i \in L$. We show that, for each $h \in \{r_i, \ldots, d(u_i, F)\}$,

$$\phi_\tau(G) \geq \left\lceil \frac{|N_h(u_i) \cap F| + k - i}{h} \right\rceil.$$

For $r_i = h = 1$, since $u_i$ is fixed in position $i$ and has nodes $u_{i+1}, \ldots, u_k$ fixed on the right, the rightmost node in the layout belonging to $N_1(u_i) \cap F$ will be in position $|N_1(u_i) \cap F| + k$ or higher. This shows $\phi_\tau(G) \geq |N_1(u_i) \cap F| + k - i$. The proof for higher values of $r_i$ and $h$ is similar, using considerations analogous to those in the proof of Proposition 6. □

Consider the partial layout with $k = 2$, $u_1 = 0$ and $u_2 = 2$ for the graph $G$ of Figure 1. The highest value in the right-hand-side of (10) is 4, achieved e.g. for $i = 2, h = 1$, for which $N_1(v_2) \cap F = \{1, 3, 6, 7\}$. This is also the maximum for the lower bound of the following corollary, which extends Corollary 1.

**Corollary 2**

$$\phi_\tau(G) \geq \max_{i=1}^{k} |N_1(u_i) \cap F| + k - i. \tag{11}$$

In the next section, we will show how to find lower bounds that are closely related to (and inspired by) the above, but significantly stronger. In order to define these bounds and show that they are stronger we will have to consider suitable relaxations of the problem.

# 3 Finding Optimal Layouts

The enumerative schemes that we will consider do not use the lower bounds of the previous section as they are, but only the (simple) ideas behind their definition in order to formulate the computation of $\phi(G)$ as an *Integer Linear Program* (ILP) and then solve suitable relaxations of this ILP, *always* keeping the integrality constraints in the relaxations (that otherwise would be useless).

Whenever needed, we will assume that the distances $d(u, v)$, $u, v \in V$, along with the sets $N_k(u)$, $u \in V, k = 1, \ldots, d(u, V)$, have initially been computed in $O(nm)$ time and stored.

### 3.1 ILP formulations and relaxations

The following is an easy ILP formulation for the determination of $\phi(G)$:

$$\begin{aligned} \min \phi, & \\ \phi \geq \tau_v - \tau_u, & \qquad (u,v) \in E, \\ \tau \in \Pi, & \end{aligned} \tag{12}$$

where $\Pi$ denotes the set of all permutations of $\{1, \ldots, n\}$, each one corresponding to a layout $\tau$ of $G$, and is mathematically described by:

$$\begin{aligned} \textstyle\sum_{u \in V} \tau_u = \frac{n(n+1)}{2}, & \\ \textstyle\sum_{u \in S} \tau_u \geq \frac{|S|(|S|+1)}{2}, & \qquad S \subset V, S \neq \emptyset, \\ \tau \text{ integer}. & \end{aligned} \tag{13}$$

Note that the continuous relaxation of formulation (12) (with (13) instead of $\tau \in \Pi$ and the integrality constraints removed) is completely useless since its optimal solution value is 0, achieved by setting $\tau_u := \frac{n+1}{2}$ for all $u \in V$. The fact that the feasible region of the continuous relaxation of (13) has only integer vertices does not help since the objective function in (12) is not a linear function of $\tau$.

Another ILP formulation (whose continuous relaxation is again useless) is given by:

$$\begin{aligned} \min \phi, & \\ d(u,v) \cdot \phi \geq \tau_v - \tau_u, & \qquad u, v \in V, \\ \tau \in \Pi. & \end{aligned} \tag{14}$$

Note that the additional constraints in (14) with respect to (12) are all linearly implied by those in (12). On the other hand, when some of the constraints are relaxed, (14) is significantly stronger than (12), as discussed below.

We first show how the above ILP formulations can be used to derive the same lower bounds as in the previous section within an enumerative scheme. Then, we derive stronger bounds based on stronger ILP relaxations.

As in the previous section, suppose enumeration has already fixed $\tau_{u_i} = i$ for $i = 1, \ldots, k$ and let $L := \{u_1, \ldots, u_k\}$ and $F := V \setminus L$. Lower bounds (11) and (10) coincide with the optimal solution value of suitable relaxations of the above, which take into account the fixing constraints. In particular, for $i = 1, \ldots, k$, consider the following relaxation of the formulation defined by (12) plus the fixing constraints:

$$\begin{aligned} \min \phi, & \\ \phi \geq \tau_v - \tau_{u_i}, & \qquad v \in F, (u_i, v) \in E, \\ \tau \in \Pi, & \\ \tau_{u_h} = h, & \qquad h = 1, \ldots, k. \end{aligned} \tag{15}$$

**Proposition 8** *The maximum value of (15) for $i = 1, \ldots, k$ is equal to the left-hand-side of (11).*

**Proof.** For $i = 1, \ldots, k$, the optimal solution of (15) is obtained by assigning the nodes in $N_1(u_i) \cap F$ to positions $k+1, \ldots, k + |N_1(u_i) \cap F|$ (and the remaining nodes to the remaining positions in an arbitrary way). The corresponding value is $|N_1(u_i) \cap F| + k - i$. $\square$

Consider now the following relaxation of (14) plus the fixing constraints, again for some $i = 1, \ldots, k$:

$$\begin{array}{rl} \min \phi, & \\ d(u_i, v) \cdot \phi \geq \tau_v - \tau_{u_i}, & v \in F, \\ \tau \in \Pi, & \\ \tau_{u_h} = h, & h = 1, \ldots, k. \end{array} \tag{16}$$

**Proposition 9** *The maximum value of (16) for $i = 1, \ldots, k$ is equal to the left-hand-side of (10).*

**Proof.** Analogous to the proof of Proposition 8. In particular, the optimal solution of (16) is obtained by laying out, starting from position $k$, first the nodes in $N_1(u_i) \cap F$, then the nodes in $(N_2(u_i) \setminus N_1(u_i)) \cap F$, and so on, the corresponding value being $\max_{h=r_i}^{d(u_i,F)} \left\lceil \frac{|N_h(u_i) \cap F| + k - i}{h} \right\rceil$. $\square$

### 3.2 Stronger relaxations

Stronger relaxations than (15) and (16) that can be solved efficiently are obtained by imposing constraints of the form $\phi \geq \tau_v - \tau_{u_i}$ or $d(u_i, v) \cdot \phi \geq \tau_v - \tau_{u_i}$ for *all* $u_i \in L$ at the same time.

First we show how to solve the following relaxation of (12) plus the fixing constraints, which is stronger than (15):

$$\begin{array}{rl} \min \phi, & \\ \phi \geq \tau_v - \tau_u, & u \in L, v \in F, (u, v) \in E, \\ \tau \in \Pi, & \\ \tau_{u_i} = i, & i = 1, \ldots, k, \end{array} \tag{17}$$

This is exactly the relaxation (implicitly) considered in [7], that is easily solved by assigning to positions $k+1, \ldots, n$ first the nodes in $F$ adjacent to $u_1$ (if any), then the nodes adjacent to $u_2$ (if any), and so on. This is stated formally in the following proposition.

**Proposition 10** *ILP relaxation (17) can be solved in $O(m)$ time.*

**Proof.** For each $v \in F$, let $d_v := \min\{i : u_i \in L \cap N_1(v)\}$ (defining $d_v := n$ if $L \cap N_1(v) = \emptyset$). All these values can easily be found in $O(m)$ time. Then, sort the nodes in $F$ according to increasing values of $d_v$ (breaking ties arbitrarily) and assign the nodes to positions $k+1, \ldots, n$ according to this order. This can be done in $O(n)$ time by *bucket sort* noting that the number of different $d_v$ values is $O(n)$. It is easy to check that the corresponding solution is optimal for (17). $\square$

Consider again the partial layout with $k = 2$, $u_1 = 0$ and $u_2 = 2$ for the graph $G$ of Figure 1. In an optimal solution of (17) nodes $1, 7, 5, 3, 6$ are laid out in this order in positions $3, 4, 5, 6, 7$

(and the remaining nodes in the other positions) and the corresponding value of $\phi$ is 5 (instead of 4, which is the optimal value of (15) and (16)).

Actually, the repeated solution of (17) within an enumerative scheme, each time with an additional node in $L$, requires on average $\Theta(\frac{m}{n})$ time per subproblem (details are omitted here), which makes the method proposed in [7] extremely fast in exploring subproblems.

A completely analogous approach can be used to compute a lower bound which is much tighter in practice for sparse graphs with a computational effort that is not much higher. This is the main advantage of our approach with respect to the one in [7]. The relaxation, which is stronger than (16), is:

$$\begin{aligned} &\min \phi, \\ d(u,v)\cdot\phi &\geq \tau_v - \tau_u, \qquad && u \in L, v \in F, \\ &\tau \in \Pi, \\ \tau_{u_i} &= i, && i = 1,\dots,k. \end{aligned} \tag{18}$$

For each node $v \in V$, let

$$s_v := \min_{u \in L} d(u,v) \tag{19}$$

denote the shortest distance between a node in $L$ and $v$. Moreover, let

$$N_1^L(v) := \{w \in N_1(v) : s_w = s_v - 1\} \tag{20}$$

denote the set of neighbors of $v$ whose shortest distance from a node in $L$ is one unit smaller than that of $v$ (if $s_v = 1$, $N_1^L(v)$ is the set of neighbors of $V$ in $L$).

**Proposition 11** *ILP relaxation (18) can be solved in $O(m \log n)$ time.*

**Proof.** We show how to check in $O(m)$ time the feasibility of the problem obtained from (18) by fixing $\phi$ to a value in $\{1,\dots,n\}$. Then, performing binary search on $\phi$ yields the overall complexity.

A solution of (18) has value at most $\phi$ if and only if each node $v \in F$ having distance $h$ from a node $u_i \in L$ is assigned to a position $\tau_v \leq h\cdot\phi + i$. Specifically, we must have $\tau_v \leq \ell_v$, where

$$\ell_v := \min\{h\cdot\phi + i : u_i \in L \cap N_h(v)\}. \tag{21}$$

Value $\ell_v$ can be thought of as the *last* feasible position for node $v$ in a solution of (18) of value at most $\phi$. Note that

$$\ell_v = \min\{\phi + \ell_w : w \in N_1^L(v)\}, \tag{22}$$

letting $\ell_{u_i} := i$ for $i = 1,\dots,k$.

To test if a solution with $\tau_v \leq \ell_v$ for $v \in F$ exists, we can compute $\ell_v$ for $v \in F$ in $O(m)$ time from the node distances, according to (22), sort the nodes $v \in F$ according to increasing values of $\ell_v$ in $O(n)$ time by bucket sort, and, again in $O(n)$ time, assign the nodes in $F$ to positions $k+1,\dots,n$ according to this order, checking if the corresponding layout satisfies $\tau_v \leq \ell_v$ for $v \in F$. $\square$

| $v$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| $\ell_v$ | 9 | 5 | 9 | 5 | 6 | 9 | 6 | 9 |

Table 1: Values of $\ell$ for the graph in Figure 1 with $k = 2$, $u_1 = 9$, $u_2 = 8$.

Consider the partial layout with $k = 2$, $u_1 = 9$ and $u_2 = 8$ for the graph $G$ of Figure 1. It is easy to check that the corresponding optimal value of (16) and (17) is $\phi = 4$. The constraints in (18) yield the values for $\ell_v$ given in Table 1, and it is immediate to see that the relaxation is infeasible (a trivial necessary condition for feasibility is $\ell_v \geq n = 10$ for at least one node $v \in F$). In fact, the optimal value of (18) is $\phi = 5$.

We now present a generalization of the above relaxation which can be used when, besides the $k$ leftmost nodes in the layout, also the $q$ *rightmost* nodes in the layout $v_1\ v_2\ \ldots\ v_q$ have been fixed, with $\tau_{u_i} = i$ for $i = 1, \ldots, k$ (as before) and $\tau_{v_i} = n - i + 1$ for $i = 1, \ldots, q$. Let $R := \{v_1, \ldots, v_q\}$ and $F := V \setminus (L \cup R)$. The relaxation is now:

$$
\begin{array}{rll}
\min \phi, & & \\
d(u,v) \cdot \phi \geq \tau_v - \tau_u, & u \in L, v \in F, & \\
d(u,v) \cdot \phi \geq \tau_u - \tau_v, & u \in R, v \in F, & \\
\tau \in \Pi, & & \\
\tau_{u_i} = i, & i = 1, \ldots, k, & \\
\tau_{v_i} = n - i + 1, & i = 1, \ldots, q. &
\end{array} \tag{23}
$$

**function** FEASIBLE_POSITIONS();
**begin**
  **for** $i := k + 1$ **to** $n - q$ **do**
    **if** $\{v \in F : f_v \leq i\} = \emptyset$ **then return**(FALSE);
    $w := \arg\min\{\ell_v : v \in F, f_v \leq i\}$;
    **if** $\ell_w < i$ **then return**(FALSE);
    $\tau_w := i$; $F := F \setminus \{w\}$;
  **end for**;
  **return**(TRUE);
**end**.

Figure 3: Procedure to find (if any) a feasible solution to relaxation (23) for a given value of $\phi$.

**Proposition 12** *ILP relaxation (23) can be solved in $O(m \log n + n \log^2 n)$ time.*

**Proof.** As in the proof of Proposition 11, we perform binary search on the value of $\phi$. For $v \in F$, let $\ell_v$ be defined by (21) and

$$f_v := \max\{(n - i + 1) - h \cdot \phi : v_i \in R \cap N_h(v)\} \tag{24}$$

($f_v$ can be thought of as the *first* feasible position for node $v$ in a solution of (23) of value at most $\phi$). A solution of (23) has value at most $\phi$ if and only if, for each node $v \in F$, we have $f_v \leq \tau_v \leq \ell_v$.

| $v$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 9 |
|---|---|---|---|---|---|---|---|---|
| $f_v$ | 2 | 2 | 6 | 6 | 2 | 6 | 2 | 2 |
| $\ell_v$ | 5 | 9 | 9 | 9 | 5 | 9 | 5 | 9 |
| $\tau_v$ | 2 | 5 | 7 | 8 | 3 | 9 | 4 | 6 |

Table 2: Values of $f$ and $\ell$ for the graph in Figure 1 with $k = q = 1$, $u_1 = 0$, $v_1 = 8$ and the corresponding layout found by FEASIBLE_POSITIONS.

Accordingly, we can compute $\ell_v$ and $f_v$ for $v \in F$ in $O(m)$ time from the node distances as in the proof of Proposition 11, and then, for $i = k+1, \ldots, n-q$, assign to the current position $i$ the node with smallest $\ell_v$ such that $f_v \leq i$. The corresponding procedure, called FEASIBLE_POSITIONS and illustrated in Figure 3, can be implemented to run in time $O(n \log n)$ by using a *heap* to dynamically store the set $\{v \in F : f_v \leq i\}$ in order of $\ell_v$, since every node is inserted and removed at most once from the heap. Initially, only the nodes with $f_v \leq k+1$ are inserted in the heap, and, each time $i$ is increased, the nodes with $f_v = i$ are added to the heap.

Combined with the $O(\log n)$ steps for the binary search of the correct value of $\phi$, the above discussion shows the overall complexity of $O(m \log n + n \log^2 n)$. □

As an illustration of procedure FEASIBLE_POSITIONS, consider the partial layout with $k = q = 1$, $u_1 = 0$ and $v_1 = 8$ for the graph $G$ of Figure 1. Table 2 gives the corresponding values of $\ell_v$, $f_v$, and the resulting layout $\tau$ for $\phi = 4$ (for which relaxation (23) is of course feasible – see Figure 1).

We remark that in the parametric solution of (18) and (23) when a new node has been added to $L$, recomputation of $l_v$ and $f_v$ can be done in $O(n)$ time (see Section 3.4), whereas the complexity of the remaining operations is essentially unchanged. This yields parametric complexities of $O(n \log n)$ and $O(n \log^2 n)$, respectively.

Finally, we point out that when the value of $\phi(G)$ is close to $n$ (which is often the case for relatively dense graphs) the lower bound provided by (18) tends to be the same as the one associated with (17), which makes this latter relaxation much more effective (being faster to solve). Specifically, note that for $\phi \geq n/2$ we have $\ell_v = n$ for all $v \in F$ such that $L \cap N_1(v) = \emptyset$, which means that a solution of value $\phi$ exists for (18) if and only if it exists for (17). Of course, the same applies considering (23) and its "bidirectional"counterpart obtained from (17).

### 3.3 Further strengthening

In this section, we discuss a further strengthening of the relaxations in the previous section. Recall the vectors $\ell$ and $f$ in the proof of Proposition 12, representing, respectively, the first and last feasible positions of each node $v \in F$ in a layout of bandwidth at most $\phi$ (if any). The strengthening in this section amounts to computing tighter values for $\ell_v$ and $f_v$ – afterward the solution of the relaxations proceeds as in the previous section.

Assume first only the leftmost part of the layout has been fixed, letting again $L = \{u_1, \ldots, u_k\}$, $\tau_{u_i} = i$ $(i = 1, \ldots, k)$ and $F := V \setminus L$. Recall the definition of $N_1^L(v)$ given in (20). Moreover, for $p \in \{1, \ldots, n\}$, let $\Pi_p$ denote the set of all $p$-tuples of *distinct* integer values in $\{1, \ldots, n\}$ (noting that $\Pi_n \equiv \Pi$). Consider the following *bilevel* ILP relaxation:

$$
\begin{array}{lr}
\min \phi, & \\
\tau_v \le \ell_v, & v \in F, \\
\tau_{u_h} = h, & h = 1, \ldots, k, \\
\tau \in \Pi, & \\
\ell_{u_h} = h, & h = 1, \ldots, k, \\
\phi \ge i - h, & (u_i, u_h) \in E, \\
\ell_v = \left\{ \begin{array}{rl} \max \pi_v^v, & \\ \phi \ge \pi_v^v - \pi_u^v, & u \in N_1^L(v), \\ \pi_u^v \le \ell_u, & u \in N_1^L(v), \\ \pi^v \in \Pi_{|N_1^L(v) \cup \{v\}|}, & \end{array} \right\}, & v \in F.
\end{array}
\tag{25}
$$

Here, the value of $\ell_v$ is determined by the inner ILP (one for each $v \in F$), generally referred to as the *lower level* ILP, in which a layout $\pi^v$ of the nodes in $N_1^L(v) \cup \{v\}$ compatible with $\ell$ and such that $\pi_v^v$ is as large as possible is found. The outer ILP is called the *outer level ILP*. Note that $\phi$ and $\ell$ appear both in the inner and outer level ILP. The optimal solution of (25) yields a lower bound on the bandwidth of any layout $\tau$ with $\tau_{u_h} = h$ for $h = 1, \ldots, k$, since, given such a layout $\tau$, setting $\phi := \phi_\tau$, $\ell := \tau$, and $\pi_u^v := \tau_u$ for $u \in N_1^L(v) \cup \{v\}$ in each lower level ILP yields a feasible solution of (25).

Consider again the partial layout with $k = 2$, $u_1 = 9$ and $u_2 = 8$ for the graph $G$ of Figure 1, already considered in the previous section and for which the values of $\ell_v$ for (18) associated with $\phi = 4$ are given in Table 1. Consider node 2. We have $N_1^L(2) = \{1, 3, 6\}$. In the corresponding optimal solution of the lower level ILP for $\phi = 4$, we have $\pi_6^2 = 6$, $\pi_3^2 = 5$, $\pi_1^2 = 4$ (or $\pi_6^2 = 6$, $\pi_3^2 = 4$, $\pi_1^2 = 5$), leading to $\pi_2^2 = \ell_2 = 8$ (instead of 9). Analogously, $\ell_5 = \ell_7 = 8$. Although infeasibility for $\phi = 4$ is already shown by (18), the example shows the improvement in the $\ell_v$ values (and suggests that in practice the optimal solution value of (25) may often be strictly larger than that of (18)).

Before discussing how to solve (25), we show that it is stronger than (18).

**Proposition 13** *The optimal value of ILP relaxation (25) is at least as large as the one of ILP relaxation (18).*

**Proof.** Letting $\phi, \tau, \ell, \pi$ be a feasible solution of (25), we show that $\phi, \tau$ is feasible for (18). This amounts to proving

$$d(u, v) \cdot \phi \ge \tau_v - \tau_u, \quad u \in L, v \in F. \tag{26}$$

Actually, we will show

$$\ell_v \le d(u, v) \cdot \phi + \tau_u, \quad u \in L, v \in F, \tag{27}$$

that implies (26) together with $\tau_v \le \ell_v$ for $v \in F$.

We prove first (27) for $d(u, v) = s_v$, recalling (19) and using induction on $s_v$. If $s_v = d(u, v) = 1$, i.e. $u \in N_1^L(v)$, the lower level ILP in (25) for $v$ implies $\ell_v = \pi_v^v \le \phi + \pi_u^v = \phi + \tau_u$. This proves the basis of the induction.

Now suppose (27) is valid for $s_v = d(u,v) = 1, \ldots, p-1$. Consider nodes $u \in L, v \in F$ with $s_v = d(u,v) = p \geq 2$ along with a shortest path from $u$ to $v$ in $G$. Let $w$ be the node preceding $v$ in the shortest path, noting that $w \in F$, $s_w = s_v - 1$ and $d(u,v) = d(u,w) + 1$. Consider the lower level ILP in (25) for $v$. Since $w \in N_1^L(v)$, we have by the induction hypothesis

$$\ell_v = \pi_v^v \leq \phi + \pi_w^v \leq \phi + \ell_w \leq \phi + d(u,w) \cdot \phi + \tau_u = d(u,v) \cdot \phi + \tau_u.$$

The proof is completed by showing (27) for $d(u,v) > s_v$. Let $L_v := \{w \in L : d(w,v) = s_v\}$ and consider again a shortest path from $u$ to $v$ in $G$. If this path visits a node $w \in L_v$, (27) follows from $\ell_v \leq d(w,v) \cdot \phi + \tau_w$ and

$$\tau_w = \ell_w \leq d(u,w) \cdot \phi + \tau_u, \tag{28}$$

that holds thanks to constraint $\phi \geq i - h$ for $(u_i, u_h) \in E$ in (25). Otherwise, let $w$ be the last node in $L$ in the shortest path and $t \in L_v$. We have $d(u,v) \geq d(u,w) + 1 + d(t,v)$, that implies (27) together with (28), $\ell_v \leq d(t,v) \cdot \phi + \tau_t$, and $\tau_t \leq \phi + \tau_w$, the latter holding since $w$ has by definition a neighbor in $F$. □

We next address the solution of (25).

**Proposition 14** *ILP relaxation (25) can be solved in $O(m \log n)$ time.*

**Proof.** We show how to check feasibility for a given value of $\phi$. Computation of $s_v$ as defined in (19) for each node $v \in F$ takes $O(m)$ time (and can be done once for all before searching the correct $\phi$ value). Then, the lower level ILPs in (25) can be solved one after the other by considering the nodes $v$ in $F$ according to increasing values of $s_v$, noting that the value of $\ell_v$ is not affected by that of $\ell_w$ if $s_w \geq s_v$.

When the lower level ILP for node $v$ is considered, the values of $\ell_w$ for each $w \in N_1^L(v)$ have already been determined by the solution of the previous problems. Then, the optimal solution of the lower level ILP is obtained by proceeding as in the solution of (17) (see Proposition 10), considering the nodes in $N_1^L(v)$ according to decreasing values of $\ell$, say $N_1^L(v) = \{w_1, \ldots, w_q\}$ with $\ell_{w_1} \geq \ldots \geq \ell_{w_q}$. We define $\pi_{w_1}^v := \ell_{w_1}$ and then, for $i = 2, \ldots, q$, $\pi_{w_i}^v := \min\{\ell_{w_i}, \pi_{w_{i-1}}^v - 1\}$. The value of $\pi_v^v = \ell_v$ is then given by $\pi_v^v = \phi + \pi_{w_q}^v$.

The time to solve the lower level ILP for $v$ is clearly proportional to $|N_1^L(v)|$ (in particular, sorting the nodes in $N_1^L(v)$ according to decreasing values of $\ell_w$ can be done in linear time by bucket sort), which leads to an overall $O(m)$ complexity for this stage, including the determination of $N_1^L(v)$ for $v \in F$. Once all $\ell_v$ values have been computed, feasibility of the outer level problem can be checked in $O(n)$ time as in the solution of (18), illustrated in the proof of Proposition 11. □

We note that a stronger version of (25) could be defined by imposing in the lower level ILP constraint $\phi \geq \pi_v^v - \pi_u^v$ for all $u \in N_1(v)$ (instead of $N_1^L(v)$). This relaxation could be solved iteratively in a time which is provably polynomial but considerably higher than the one needed for (25). Since the lower bound provided in practice by this variant was exactly the same as the one given by (25) for all instances we tried (for all subproblems in the enumeration), we decided to stick to (25) in our implementation.

Another strengthening of (25) is given by defining the following lower level ILP:

$$\ell_v = \left\{ \begin{array}{rl} \max \pi_v^v, & \\ d(u,v) \cdot \phi \geq \pi_v^v - \pi_u^v, & u \in N^L(v), \\ \pi_u^v \leq \ell_u, & u \in N^L(v), \\ \pi^v \in \Pi_{|N^L(v) \cup \{v\}|}, & \end{array} \right\}, \qquad v \in F,$$

where $N^L(v) := \{u \in V : s_u < s_w\}$ is the set of nodes which are closer to $L$ than $v$ is. Each lower level ILP here can be solved by assigning the values $\pi_w^v$ to each $w \in N^L(v)$, as for the solution of (25), with the difference that this assignment is now completely analogous to the one in Figure 3. Namely, the first position available is $\ell^* := \max_{w \in N^L(v)} \ell_w$, and we set $\pi_{w_1}^v := \ell^*$ for the node $w_1 \in N^L(v)$ with $\ell_{w_1} = \ell^*$ and $d(w_1, v)$ *minimum*. Then, the next position available is $\ell^* := \min\{\pi_{w_1}^v - 1, \max_{w \in N^L(v) \setminus \{w_1\}} \ell_w\}$, and we set $\pi_{w_2}^v := \ell^*$ for the node $w_2 \in N^L(v) \setminus \{w_1\}$ with $\ell_{w_2} = \ell^*$ and $d(w_2, v)$ *minimum*, and so on. Accordingly, the time for the solution of each lower level ILP is now $O(n \log n)$ (noting that $|N^L(v)|$ may be $\Omega(n)$), and therefore the overall solution time for the relaxation $O(n^2 \log^2 n)$. In our computational experiments, this new relaxation yielded better lower bounds than the original (25) in some cases, but the corresponding smaller number of subproblems in the enumeration did not compensate the higher running time required for its solution. This confirmed our choice of (25).

In (25), the role of the lower level ILPs is to provide stronger (lower) values of $\ell_v$ than in (18). In a completely analogous way, one can define (independent) lower level ILPs that provide stronger (higher) values of $f_v$. The resulting ILP relaxation, which is to (23) what (25) is to (18), reads:

$$\begin{array}{ll} \min \phi, & \\ f_v \leq \tau_v \leq \ell_v, & v \in F, \\ \tau_{u_h} = h, & h = 1, \ldots, k, \\ \tau_{v_i} = n - i + 1, & i = 1, \ldots, q, \\ \tau \in \Pi, & \\ \ell_{u_h} = h, & h = 1, \ldots, k, \\ \phi \geq i - h, & (u_i, u_h) \in E, \\ \ell_v = \left\{ \begin{array}{rl} \max \pi_v^v, & \\ \phi \geq \pi_v^v - \pi_u^v, & u \in N_1^L(v), \\ \pi_u^v \leq \ell_u, & u \in N_1^L(v), \\ \pi^v \in \Pi_{|N_1^L(v) \cup \{v\}|}, & \end{array} \right\}, & v \in F, \\ f_{v_i} = n - i + 1, & i = 1, \ldots, q, \\ \phi \geq i - h, & (v_i, v_h) \in E, \\ f_v = \left\{ \begin{array}{rl} \min \rho_v^v, & \\ \phi \geq \rho_u^v - \rho_v^v, & u \in N_1^R(v), \\ \rho_u^v \geq f_u, & u \in N_1^R(v), \\ \rho^v \in \Pi_{|N_1^R(v) \cup \{v\}|}, & \end{array} \right\}, & v \in F, \end{array} \tag{29}$$

where $N_1^R(v) := \{w \in N_1(v) : t_w = t_v - 1\}$ and $t_v := \min_{u \in R} d(u, v)$ for $v \in V$.

**Proposition 15** *ILP relaxation (29) can be solved in* $O(m \log n + n \log^2 n)$ *time.*

**Proof.** For a fixed $\phi$, the solution of the lower level ILPs can be carried out as for (25), in $O(m)$ time, whereas, once the $\ell$ and $f$ vectors have been computed, feasibility can be checked as for (23). □

Apparently, the parametric solution of (25) and (29) when the partial layout is extended has the same worst case complexity as the solution from scratch (see Propositions 14 and 15).

### 3.4 Enumerative schemes

The enumerative scheme used in [7] fixes the value of $\phi$ and looks for a layout of bandwidth $\phi$, if any exists. The layout is constructed from left to right and the backtracking condition is that the optimal value of relaxation (17) is larger than $\phi$.

Here, we present two enumerative schemes, one based on relaxations (18) and (25) and the other on (23) and (29). Since the parametric solution of (18) and (23) within an enumerative scheme is considerably faster than (25) and (29), we first solve the weaker relaxation and resort to the stronger one only if the subproblem is not fathomed.

Instead of performing binary search on $\phi$ every time the relaxation is solved, we decided to fix externally the value of $\phi$ and then check internally if the optimal solution value of the relaxations is larger than $\phi$ (in which case we backtrack) or not. This allows us to derive strong lower bounds on $\phi(G)$ within shorter time, as discussed in the next section. Since the scheme associated with relaxations (23) and (29), called LAYOUT_BOTH_WAYS, is essentially a generalization of the one based on (18) and (25), called LAYOUT_LEFT_TO_RIGHT, we illustrate the former and then show how the latter can be obtained as a special case.

In the enumeration, we keep track of the labels $f_v, \ell_v$ for each node $v$ not laid out so far. Letting $k+1$ and $n-q$ denote the leftmost and rightmost positions not assigned to any node in the layout, we decide whether to extend the layout occupying position $k+1$ or $n-q$ depending on the number of nodes that may be assigned to one of these two positions. The corresponding implementation into recursive procedure LAYOUT_BOTH_WAYS is given in Figure 4. The global variables are initialized as

$$F := V;\ \ell_v := n, f_v := 1\ (v \in V);\ k := 0;\ q := 0.$$

Procedures SAVE and RESTORE simply save and restore the vectors $\ell$ and $f$ before and after fixing the next node in the layout. Procedure UPDATE determines in $O(n)$ time the new vectors $\ell$ and $f$ associated with ILP relaxation (23) after the fixing, and is given in Figure 5. If these values are enough to prove infeasibility, i.e. FEASIBLE_POSITIONS returns FALSE, we backtrack. Otherwise, as described in the previous section, we compute tighter vectors $\ell$ and $f$ by solving the lower level ILP problems in (29), by procedure TIGHTER_POSITIONS (whose pseudo code is not reported), and then apply FEASIBLE_POSITIONS with these tighter values. If the outcome is TRUE again, we branch by calling LAYOUT_BOTH_WAYS recursively.

**procedure** LAYOUT_BOTH_WAYS();
**begin**
  **if** $F = \emptyset$ **then** SUCCESS;
  $A := \{v \in F : f_v = k + 1\}$; **comment:** nodes that may occupy the first available position;
  $B := \{v \in F : \ell_v = n - q\}$; **comment:** nodes that may occupy the last available position;
  **if** $|A| \leq |B|$ **then**
    **comment:** expand layout on the left;
    **for each** $v \in A$ **do**
      $F := F \setminus \{v\}$; $k := k + 1$; $\tau_k := v$; SAVE$(\ell, f)$;
      UPDATE($v$,LEFT);
      **if not** FEASIBLE_POSITIONS() **return**;
      TIGHTER_POSITIONS();
      **if not** FEASIBLE_POSITIONS() **return**;
      LAYOUT_BOTH_WAYS();
      $F := F \cup \{v\}$; $k := k - 1$; RESTORE$(\ell, f)$;
    **end for each**;
  **else**
    **comment:** expand layout on the right;
    **for each** $v \in B$ **do**
      $F := F \setminus \{v\}$; $q := q + 1$; $\tau_{n-q+1} := v$; SAVE$(\ell, f)$;
      UPDATE($v$,RIGHT);
      **if not** FEASIBLE_POSITIONS() **return**;
      TIGHTER_POSITIONS();
      **if not** FEASIBLE_POSITIONS() **return**;
      LAYOUT_BOTH_WAYS();
      $F := F \cup \{v\}$; $q := q - 1$; RESTORE$(\ell, f)$;
    **end for each**;
  **end if**;
**end**.

Figure 4: Enumerative procedure LAYOUT_BOTH_WAYS to find (if any) a feasible layout of bandwidth $\phi$.

**procedure** UPDATE($v$,DIR);
**begin**
  **for** $h := 1$ **to** $d(v, F)$ **do**
    **for each** $w \in N_h(v)$ **do**
      **if** DIR = LEFT **and** $\ell_w > k + h \cdot \phi$ **then** $\ell_w := k + h \cdot \phi$;
      **if** DIR = RIGHT **and** $f_w < (n - q + 1) - h \cdot \phi$ **then** $f_w := (n - q + 1) - h \cdot \phi$;
**end**.

Figure 5: Procedure to update $\ell$ and $f$ after fixing a node in the layout.

Overall, for each subproblem generated in the enumerative process (each recursive call to LAYOUT_BOTH_WAYS), the execution time is $O(n^2 \log n + nm)$, since we may call $\Omega(n)$ times FEASIBLE_POSITIONS (that has complexity $O(n \log n)$) and TIGHTER_POSITIONS (that has complexity $O(m)$).

Procedure LAYOUT_LEFT_TO_RIGHT is obtained from LAYOUT_BOTH_WAYS by always expanding the layout on the left ($R = \emptyset$ throughout the enumeration). The worst case execution time for each subproblem is $O(nm)$.

# 4 Computational Results

Our algorithms were implemented in ANSI C and tested on a PC AMD 1333 Mhz running Windows 98.

Since our enumeration schemes check the existence of a layout of bandwidth $\phi$, we have to decide the sequence of input values of $\phi$. This choice has to be performed with some care since the difficulty of the search may strongly depend on $\phi$, as one would expect. Somewhat surprisingly, this difficulty does not increase monotonically as the distance between the tentative $\phi$ and the correct value $\phi(G)$ decreases. To be more precise, given two values $\phi_1, \phi_2$ with $\phi(G) \leq \phi_1 < \phi_2$ enumeration often takes much longer with input value $\phi_2$ than with $\phi_1$. This may simply be bad luck (in both cases enumeration stops as soon as a feasible layout equal to the input $\phi$ is found), but may also be motivated by the fact that the search space with value $\phi_1$ is much smaller, and enumeration is quickly driven to a solution. On the other hand, for $\phi_1 < \phi_2 < \phi(G)$, enumeration with input value $\phi_1$ always takes shorter since there is no solution and all the search space must be examined. This implies that classical binary search on the value of $\phi(G)$ is widely outperformed by a method that starts from $\phi$ equal to a lower bound on $\phi(G)$ and increases $\phi$ by one as long as no feasible layout of value $\phi$ exists.

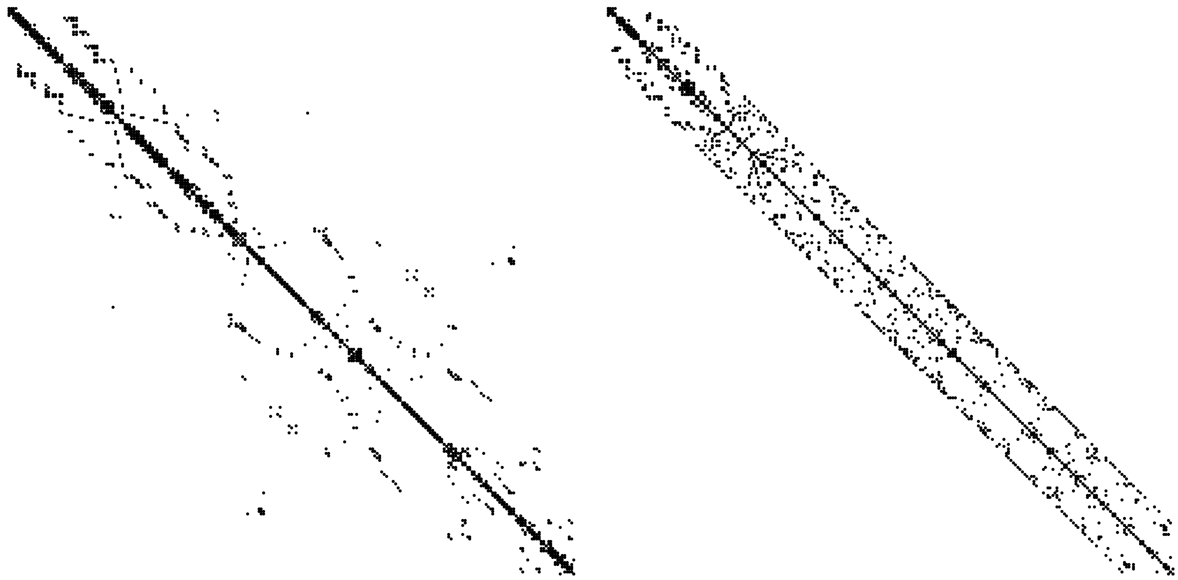
According to the discussion above, our procedure works as follows. We let $\underline{\phi} := \max\{\alpha(G), \gamma(G)\}$ and $\overline{\phi} := \psi$ be initial lower and upper bounds on $\phi(G)$, where $\psi$ denotes the bandwidth of the layout in which nodes appear in the same order as they appear on input. We impose a global time limit $T$ on the overall procedure, as well as a (smaller) time limit $t$ for the enumeration for each specific value of $\phi$. Starting with $\phi := \underline{\phi}$ and increasing $\phi$ by one at each iteration, we apply enumeration for each value $\phi \in \{\underline{\phi}, \ldots, \overline{\phi} - 1\}$. If the enumeration ends before the time limit $t$, we update $\overline{\phi}$ in case of success and $\underline{\phi}$ in case of failure (i.e. if no layout of bandwidth $\phi$ exists). If after trying all values of $\phi$, we have $\underline{\phi} < \overline{\phi}$ (i.e. the time limit $t$ was exceeded in some enumeration) and the global time limit $T$ has not been reached, we repeat enumeration for all $\phi \in \{\underline{\phi}, \ldots, \overline{\phi} - 1\}$, each time with a new time limit equal to the maximum between $2t$ and the remaining time divided by $\overline{\phi} - \underline{\phi}$. This method typically gives good solutions also for the instances that we cannot solve to optimality, since we also try higher values of $\phi$ when we cannot terminate enumeration for a specific value. In our experiments, $T$ was set to one hour, $t$ to one minute, and the time limit is checked every 10000 subproblems.

A heuristic that comes for free in our enumeration schemes, but tends to give good results only if the input $\phi$ is close to $\phi(G)$, is the following. The solution of relaxations (18), (23), (25), and (29) provides a heuristic layout $\tau$ of $G$: computing the value of the layout and possibly updating the best layout so far takes $O(m)$ time (with a very small multiplicative constant), without increasing significantly the running time of the enumeration.

We report results for real-world instances from the library *Matrix Market* [2]. In particular, we considered the 39 instances in the library with $n \leq 250$. Table 3 provides the characteristics of these instances, that are reported in the same order as they appear in the library, distinguishing between matrices for which only the zero-nonzero pattern is given (instances *bcspwr01–dwt___87*) and those for which the actual values of the entries are available (instances *bcsstk01–tub100*). The table gives for each instance the values of $n$, $m$ and the associated density *dens*:= $2m/(n(n-1))$; the values of the lower bounds $\delta(G)$, $\varepsilon(G)$, $\alpha(G)$, $\gamma(G)$ (all computable within very short time) and $\beta(G)$, along with the corresponding computing time *time* and the number *subpr* of subproblems explored (i.e., max-clique problems solved) in the enumeration using the method in Section 2.2; the value $\psi$ of the bandwidth for the "input" layout (see above); and the value of $\phi(G)$, giving the range of possible values if this value is unknown. Note that instances *bcsstk02* and *tub100* are trivial and reported just for the sake of completeness. The table shows that the time to compute $\beta(G)$ is typically very small, with few exceptions (for instance *can__187* we could not prove within 5 hours the optimality of value 8 since within that time we could only complete enumeration for $d \leq 12$). On the other hand, the quality of lower bounds $\alpha(G)$, $\gamma(G)$ and $\beta(G)$ is on average comparable, and typically much better than $\varepsilon(G)$ and $\delta(G)$. More precisely, $\alpha(G)$ is in most cases close to $\beta(G)$, with the irrelevant exception of *bcsstk02* (complete graph). In a few cases, either $\beta(G)$ is notably better than $\gamma(G)$ or viceversa (the former situation being more frequent). Overall, $\max\{\alpha(G), \gamma(G)\}$ appears to be a good lower bound on $\phi(G)$.

Table 4 reports the results of LAYOUT_LEFT_TO_RIGHT with and without using ILP relaxation (25) (i.e. with and without calling TIGHTER_POSITIONS in the fathoming test), and LAYOUT_BOTH_WAYS (using ILP relaxation (29)). We report the values of the best lower bound $\underline{\phi}$ and upper bound $\overline{\phi}$ on $\phi(G)$ (that coincide when the instance was solved to optimality) along with the overall number *subpr* of subproblems explored in the enumeration and the corresponding running time *time* and the time *h-time* needed to find the best layout. Comparison of the first two columns shows the clear advantage of using also a more complex relaxation in the bounding step, which leads in many cases to a significant reduction in the running time, allowing us to solve 4 more instances to proven optimality and leading to consistently smaller gaps for the instances that can be solved. The same conclusions can be drawn by comparing LAYOUT_BOTH_WAYS with and without using ILP relaxation (29), so we do not report detailed results for the latter. Comparison of the second and third column shows that the performance of our two enumerative schemes varies significantly depending on the instance. Since we do not know how to distinguish a priori whether one enumerative scheme will be better than the other for a given instance, a hybrid method obtained by combining the two, called HYBRID, turns out to be a good compromise.

Method HYBRID is simply obtained by running first LAYOUT_BOTH_WAYS for $\phi \in \{\underline{\phi}, \ldots, \overline{\phi} - 1\}$ with time limit $t$ for each run, then LAYOUT_LEFT_TO_RIGHT for $\phi \in \{\underline{\phi}, \ldots, \overline{\phi} - 1\}$, again with time limit $t$. Then LAYOUT_BOTH_WAYS is run again for $\phi \in \{\underline{\phi}, \ldots, \overline{\phi} - 1\}$ with time limit equal to the maximum between $2t$ and the remaining time divided



(a) Initial layout: $\psi = 115$ (b) Optimal layout: $\phi(G) = 21$

Figure 6: Initial layout and minimum-bandwidth layout for Matrix *dwt__245* with $n = 245$ from the MatrixMarket collection.

by $\overline{\phi} - \underline{\phi}$ (of course, the values $\underline{\phi}$ and $\overline{\phi}$ typically change during the execution).

Table 5 compares the method of [7], implicitly based on ILP relaxation (17), for which a very efficient C-language implementation was kindly sent us by the authors, and HYBRID, showing that our method is much better on these (sparse) instances, in that it can solve overall 25 out of the 39 instances, whereas [7] terminates for only 10 instances. For the instances that we cannot solve to optimality, our final lower bound is notably better than the one provided by [7]. Note that we can solve to proven optimality 24 out of 30 instances with $n \leq 200$. The largest instance solved is *dwt__245*, with $n = 245$, whose initial and optimal layout are displayed in Figure 6.

We also compared the algorithms on instances generated according to the random model in [19] and considered in the computational results of [7]. These instances are generated as follows. Given integers $n$ and $\varphi < n$ and a probability $p$, we generate a random graph $G$ with $n$ nodes $\{1, \ldots, n\}$ in which no edge $(u, v)$ for $|u - v| > \varphi$ is present, and each edge $(u, v)$ with $|u - v| \leq \varphi$ is present with probability $p$ (which is independent of the presence of other edges). Of course $\phi(G) \leq \varphi$, and equality holds with a probability increasing with $p$. In any case, the actual value of $\phi(G)$ tends to be close to $\varphi$ even for relatively small values of $p$. In order to "hyde" the layout of bandwidth $\varphi$, the nodes are randomly permuted before giving them on input. This makes it nontrivial even to find such a layout. Considering the discussion at the end of Section 3.2, the results of our methods for the cases in which $\varphi > n/2$ (and therefore $\phi(G)$ is not significantly smaller than $n/2$) are not particularly meaningful, since much simpler relaxations than those we use (e.g. the relaxation that is implicitly used by [7]) yield the same lower bounds and are orders of magnitude faster to solve than ours. Therefore, we mainly concentrate on the sparse cases in which $\varphi \leq n/2$.

Tables 6 and 7 compare the method in [7] with HYBRID for $p = 0.5$ and $p = 0.3$, and various combinations of $n$ and $\varphi$. For each combination, we report average results out of 10 instances with a time limit of $T = 5$ minutes for each instance and $t = 5$ seconds, providing the number *succ* of instances solved to proven optimality, the average computing time *time* out of these *succ* instances, and the average lower bound $\underline{\phi}$ and upper bound $\overline{\phi}$ on the optimal bandwidth for the remaining (unsolved) instances. For the method of [7] we only give $\overline{\phi}$ since the method does not compute upper bounds. The tables show that our method widely outperforms that of [7] for these instances. The instances with $p = 0.5$ are easier, the optimal value being often $\varphi$. In a few cases, our method returns $\underline{\phi} = \varphi(= \phi(G))$ but cannot find an optimal layout. This shows on the one hand the effectiveness of our method in finding lower bounds, and on the other the difficulty (at least for our method, but probably intrinsic) in finding an optimal layout (as opposed to proving optimality). The gap between $\underline{\phi}$ and $\overline{\phi}$ is always small (also the lower bound provided by [7] is of good quality), whereas the computing time for the instances solved greatly differs from instance to instance. In any case, we can solve to proven optimality the majority of the instances for $p = 0.5$, but only a few of the instances for $p = 0.3$ and $n$, $\varphi$ large.

Finally, Tables 8 and 9 report the same information as Tables 6 and 7 for instances with large $n$ and $\varphi$. The time limit was set to $T = 500$ seconds in this case, with $t = 10$ seconds and time limit checked every 1000 subproblems. These tables show that we can solve to proven optimality some of the instances with $n = 1000$, the average gap being around 1% for the unsolved cases.

# 5 Conclusions

Solving to proven optimality the bandwidth problem is certainly a challenging task, on which classical exact methods in combinatorial optimization seem to fail miserably. In this paper, we have presented an approach based on simple ILP relaxations of the problem, in which the integrality constraints are essential to achieve useful lower bounds. The method is suited for cases in which the bandwidth value is smaller than half the number of nodes, since otherwise simpler lower bounding methods should be used, and is successful on most instances from the literature associated with graphs with up to 200 nodes. On the other hand, there are instances with less than 150 nodes for which the gap between the best known upper and lower bound is almost 20%. This gives wide space for improvement, possibly by using other nonstandard methods: we hope that the present paper will stimulate further research in this direction.

## Acknowledgments

The work of the first author was partially supported by MIUR and CNR, Italy, and the work of the second author was partially supported by "Ministerio de Ciencia y Tecnología", Spain. We are grateful to Federico Malucelli for helpful discussions and to Gianna Del Corso and Givanni Manzini for providing us with an implementation of their exact algorithm.

## References


[1] A. Blum, G. Konjevod, R. Ravi and S. Vempala, "Semi-Definite Relaxations for Minimum Bandwidth and Other Vertex-Ordering Problems", *Theoretical Computer Science* 235 (2000) 25–42.

[2] R. Boisvert, R. Pozo, K. Remington, R. Barrett, J. Dongarra, "Matrix Market: a Web Resource for Test Matrix Collections", in R. Boisvert (ed.), *The Quality of Numerical Software: Assessment and Enhancement*, Chapman and Hall, London, 1997, `http://math.nist.gov/MatrixMarket/`.

[3] P. Chinn, J Chvátalova, A. Dewdney and N. Gibbs, "The Bandwidth Problem for Graph and Matrices - a Survey", *Journal of Graph Theory* 6 (1982) 223–254.

[4] F.R.K. Chung and P.D. Seymour,"Graphs with Small Bandwidth and Cutwidth", *Discrete Mathematics* 75 (1989) 113–119.

[5] V. Chvátal, "A Remark on a Problem of Harary", *Czechoslovak Mathematics Journal* 20 (1970) 95.

[6] E. Cuthill and J. McKee, "Reducing the Bandwidth of Sparse Symmetric Matrices", *ACM National Conference Proceedings* 24 (1969) 157–172.

[7] G.M. Del Corso and G. Manzini, "Finding Exact Solutions to the Bandwidth Minimization Problem", *Computing* 62 (1999) 189–203.

[8] G.M. Del Corso and F. Romani "Heuristic Spectral Techniques for the Reduction of Bandwidth and Work-Bound of Sparse Matrices", *Numerical Algorithms* 28 (2001) 117–136.

[9] G.W. Dueck and J. Jeffs, "A Heuristic Bandwidth Reduction Algorithm", *Journal of Combinatorial Mathematics and Combinatorial Computing* 18 (1995) 97–108.

[10] A. Esposito, M.S.F. Catalano, F. Malucelli and L. Tarricone, "A New Matrix Bandwidth Reduction Algorithm", *Operations Research Letters* 23 (1999) 99–107.

[11] U. Feige, "Approximating the Bandwidth via Volume Respecting Embeddings", *Journal of Computer and System Sciences* 60 (2000) 510–539.

[12] M.R. Garey, R.L. Graham, D.S. Johnson and D.E. Knuth, "Complexity Results for Bandwidth Minimization", *SIAM Journal on Applied Mathematics* 34 (1978) 477–495.

[13] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman (1979).

[14] N.E. Gibbs, W.G. Poole and P.K. Stockmeier, "An Algorithm for Reducing the Bandwidth and Profile of Sparse Matrices", *SIAM Journal on Numerical Analysis* 13 (1976) 236–250.

[15] N.E. Gibbs, W.G. Poole and P.K. Stockmeyer, "A Comparison of Several Bandwidth and Profile Reduction Algorithms", *ACM Transactions on Mathematical Software* 2 (1976) 322-330.

[16] E.M. Gurari and I.H. Sudborough, "Improved Dynamic Programming Algorithms for Bandwidth Minimization and the Mincut Linear Arrangement Problem", *Journal of Algorithms* 5 (1984) 531–546.

[17] D.S. Johnson and M.A. Trick (eds.), *Cliques, Coloring and Satisfiability: Second DIMACS Implementation Challenge*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science 26, AMS Press (1996).

[18] R. Martí, M. Laguna, F. Glover and V. Campos, "Reducing the Bandwidth of a Sparse Matrix with Tabu Search", *European Journal of Operational Research*, 135 (2001) 450–459.

[19] J.S. Turner, "On the Probable Performance of Heuristics for Bandwidth Minimization", *SIAM Journal on Computing* 15 (1986) 561–580.

[20] W. Unger, "The Complexity of the Approximation of the Bandwidth Problem", *Proceedings of the 39th IEEE Symposium on the Foundations of Computer Science*, IEEE Computer Scoiety Press (1998) 82–91.

| name | $n$ | $m$ | dens | $\delta(G)$ | $\varepsilon(G)$ | $\alpha(G)$ | $\gamma(G)$ | $\beta(G)$ | time | subpr | $\psi$ | $\phi(G)$ |
|---:|---:|---:|---:|---:|---:|---:|---:|---:|---:|---:|---:|---:|
| bcspwr01 | 39 | 46 | 0.06 | 1 | 3 | 4 | 4 | 4 | 0.00 | 96 | 38 | 5 |
| bcspwr02 | 49 | 59 | 0.05 | 1 | 3 | 5 | 6 | 6 | 0.00 | 225 | 34 | 7 |
| bcspwr03 | 118 | 179 | 0.03 | 1 | 5 | 9 | 9 | 10 | 0.22 | 568 | 115 | 10-11 |
| can__144 | 144 | 576 | 0.06 | 5 | 7 | 7 | 12 | 11 | 2462.09 | 559 | 142 | 13 |
| can__161 | 161 | 608 | 0.05 | 5 | 8 | 12 | 16 | 18 | 7235.50 | 782 | 79 | 18 |
| can__187 | 187 | 652 | 0.04 | 4 | 5 | 8 | 12 | 8 | >5 hours | — | 63 | 13 |
| can__229 | 229 | 888 | 0.03 | 2 | 6 | 15 | 21 | 19 | 5630.36 | 2429 | 172 | 24-29 |
| can___24 | 24 | 68 | 0.25 | 3 | 4 | 5 | 5 | 5 | 0.00 | 62 | 21 | 5 |
| can___61 | 61 | 248 | 0.14 | 4 | 12 | 13 | 11 | 13 | 0.00 | 179 | 50 | 13 |
| can___62 | 62 | 78 | 0.04 | 1 | 3 | 5 | 5 | 5 | 0.00 | 160 | 48 | 6 |
| can___73 | 73 | 152 | 0.06 | 2 | 4 | 9 | 14 | 12 | 1.82 | 442 | 39 | 16 |
| can___96 | 96 | 336 | 0.07 | 5 | 4 | 7 | 13 | 12 | 10.33 | 380 | 31 | 13 |
| dwt__162 | 162 | 510 | 0.04 | 1 | 4 | 9 | 8 | 9 | 1.15 | 464 | 156 | 10-13 |
| dwt__193 | 193 | 1650 | 0.09 | 7 | 15 | 30 | 29 | 30 | 2.42 | 1026 | 62 | 31-38 |
| dwt__209 | 209 | 871 | 0.04 | 3 | 8 | 20 | 18 | 21 | 3.51 | 2116 | 184 | 21-26 |
| dwt__221 | 221 | 814 | 0.03 | 3 | 6 | 11 | 9 | 12 | 3.62 | 848 | 187 | 11-13 |
| dwt__245 | 245 | 730 | 0.02 | 1 | 6 | 21 | 16 | 21 | 4.12 | 2052 | 115 | 21 |
| dwt___59 | 59 | 104 | 0.06 | 1 | 3 | 5 | 5 | 5 | 0.00 | 121 | 25 | 6 |
| dwt___66 | 66 | 127 | 0.06 | 1 | 3 | 3 | 3 | 3 | 0.16 | 83 | 44 | 3 |
| dwt___72 | 72 | 75 | 0.03 | 1 | 2 | 4 | 4 | 4 | 0.00 | 123 | 12 | 5 |
| dwt___87 | 87 | 227 | 0.06 | 1 | 6 | 10 | 8 | 10 | 0.06 | 293 | 63 | 10 |
| bcsstk01 | 48 | 176 | 0.16 | 4 | 6 | 9 | 15 | 14 | 0.00 | 147 | 35 | 16 |
| bcsstk02 | 66 | 2145 | 1.00 | 65 | 33 | 33 | 65 | 65 | 0.05 | 133 | 65 | 65 |
| bcsstk04 | 132 | 1758 | 0.20 | 14 | 23 | 32 | 28 | 34 | 1.70 | 580 | 47 | 37 |
| bcsstk05 | 153 | 1135 | 0.10 | 3 | 12 | 16 | 14 | 18 | 0.99 | 569 | 28 | 20 |
| bwm200 | 200 | 298 | 0.01 | 2 | 2 | 2 | 2 | 2 | 144.45 | 111 | 100 | 2 |
| e05r0000 | 236 | 2928 | 0.11 | 7 | 31 | 44 | 47 | 47 | 188.18 | 1770 | 66 | 51-53 |
| fidap001 | 216 | 2187 | 0.09 | 7 | 25 | 40 | 38 | 40 | 10.39 | 1295 | 64 | 40-41 |
| fidap005 | 27 | 126 | 0.36 | 5 | 7 | 7 | 6 | 8 | 0.00 | 33 | 20 | 8 |
| fidapm05 | 42 | 239 | 0.28 | 5 | 10 | 10 | 9 | 10 | 0.00 | 81 | 35 | 11 |
| lund_a | 147 | 1151 | 0.11 | 4 | 10 | 17 | 14 | 17 | 0.94 | 731 | 23 | 19-23 |
| lund_b | 147 | 1147 | 0.11 | 4 | 10 | 17 | 14 | 17 | 0.93 | 731 | 23 | 19-23 |
| nos4 | 100 | 247 | 0.05 | 1 | 3 | 9 | 9 | 9 | 0.11 | 390 | 13 | 10 |
| pde225 | 225 | 532 | 0.02 | 2 | 2 | 10 | 10 | 10 | 92.55 | 1179 | 15 | 11-15 |
| rdb200 | 200 | 460 | 0.02 | 3 | 3 | 12 | 13 | 12 | 22.74 | 1086 | 20 | 15-16 |
| saylr1 | 238 | 564 | 0.02 | 2 | 2 | 10 | 10 | 10 | 84.26 | 1178 | 14 | 12-14 |
| steam1 | 240 | 1881 | 0.07 | 8 | 10 | 26 | 26 | 26 | 23.57 | 2181 | 146 | 32-45 |
| steam3 | 80 | 424 | 0.13 | 7 | 6 | 6 | 7 | 7 | 0.33 | 27 | 43 | 7 |
| tub100 | 100 | 148 | 0.03 | 2 | 2 | 2 | 2 | 2 | 7.64 | 61 | 2 | 2 |

Table 3: Characteristics of the real-world instances.

| | LAYOUT_LEFT_TO_RIGHT without (25) | | | | | LAYOUT_LEFT_TO_RIGHT with (25) | | | | | LAYOUT_BOTH_WAYS with (25) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| name | $\underline{\phi}$ | $\overline{\phi}$ | h-time | time | subpr | $\underline{\phi}$ | $\overline{\phi}$ | h-time | time | subpr | $\underline{\phi}$ | $\overline{\phi}$ | h-time | time | subpr |
| bcspwr01 | 5 | 5 | 0.11 | 0.11 | 1453 | 5 | 5 | 0.11 | 0.17 | 986 | 5 | 5 | 0.00 | 1.16 | 89955 |
| bcspwr02 | 7 | 7 | 0.00 | 0.00 | 148 | 7 | 7 | 0.06 | 0.06 | 146 | 7 | 7 | 0.93 | 0.93 | 77098 |
| bcspwr03 | 9 | 11 | 130.34 | t.l. | 4150000 | 9 | 11 | 125.45 | t.l. | 3780000 | 10 | 11 | 60.31 | t.l. | 67040000 |
| can__144 | 12 | 13 | 61.30 | t.l. | 2960000 | 13 | 13 | 0.05 | 0.05 | 146 | 13 | 13 | 0.00 | 0.00 | 146 |
| can__161 | 18 | 18 | 0.06 | 0.06 | 166 | 18 | 18 | 0.05 | 0.05 | 164 | 18 | 18 | 0.00 | 0.00 | 164 |
| can__187 | 13 | 14 | 64.87 | t.l. | 3520000 | 13 | 13 | 46.91 | 46.91 | 22394 | 13 | 13 | 0.00 | 0.00 | 205 |
| can__229 | 23 | 38 | 1232.15 | t.l. | 1020000 | 24 | 33 | 761.76 | t.l. | 1010000 | 24 | 29 | 338.56 | t.l. | 13070000 |
| can___24 | 5 | 5 | 0.00 | 0.00 | 25 | 5 | 5 | 0.00 | 0.00 | 25 | 5 | 5 | 0.00 | 0.00 | 25 |
| can___61 | 13 | 13 | 0.05 | 0.05 | 62 | 13 | 13 | 0.05 | 0.05 | 62 | 13 | 13 | 0.00 | 0.00 | 62 |
| can___62 | 6 | 6 | 0.06 | 0.06 | 182 | 6 | 6 | 0.05 | 0.05 | 166 | 6 | 6 | 0.00 | 0.00 | 549 |
| can___73 | 16 | 19 | 186.64 | t.l. | 14240000 | 16 | 16 | 0.11 | 0.11 | 302 | 16 | 16 | 0.60 | 0.60 | 5012 |
| can___96 | 13 | 13 | 0.00 | 0.00 | 97 | 13 | 13 | 0.00 | 0.00 | 97 | 13 | 13 | 0.00 | 0.00 | 97 |
| dwt__162 | 10 | 13 | 204.98 | t.l. | 1850000 | 11 | 13 | 208.39 | t.l. | 1450000 | 11 | 14 | 0.05 | t.l. | 28480000 |
| dwt__193 | 30 | 39 | 683.76 | t.l. | 1500000 | 30 | 38 | 586.88 | t.l. | 1590000 | 31 | 41 | 616.37 | t.l. | 13970000 |
| dwt__209 | 20 | 50 | 2084.48 | t.l. | 940210 | 20 | 50 | 2074.16 | t.l. | 1090210 | 21 | 28 | 427.76 | t.l. | 14810000 |
| dwt__221 | 11 | 15 | 326.92 | t.l. | 4840000 | 11 | 13 | 147.15 | t.l. | 1230000 | 11 | 14 | 186.30 | t.l. | 13510000 |
| dwt__245 | 21 | 40 | 1544.34 | t.l. | 530246 | 21 | 33 | 1033.42 | t.l. | 1990000 | 21 | 23 | 124.46 | t.l. | 10390000 |
| dwt___59 | 6 | 6 | 0.06 | 0.06 | 976 | 6 | 6 | 0.05 | 0.05 | 530 | 6 | 6 | 340.37 | 340.37 | 18012246 |
| dwt___66 | 3 | 3 | 0.00 | 0.00 | 67 | 3 | 3 | 0.00 | 0.00 | 67 | 3 | 3 | 0.00 | 0.00 | 67 |
| dwt___72 | 5 | 5 | 237.01 | 237.01 | 2637713 | 5 | 5 | 246.45 | 246.45 | 2607713 | 5 | 5 | 0.17 | 0.17 | 7146 |
| dwt___87 | 10 | 10 | 378.77 | 378.77 | 851532 | 10 | 10 | 202.13 | 202.13 | 449394 | 10 | 10 | 0.00 | 0.00 | 88 |
| bcsstk01 | 16 | 16 | 0.00 | 0.00 | 469 | 16 | 16 | 0.05 | 0.05 | 463 | 16 | 16 | 0.00 | 0.00 | 106 |
| bcsstk02 | 65 | 65 | 0.00 | 0.00 | 0 | 65 | 65 | 0.00 | 0.00 | 0 | 65 | 65 | 0.00 | 0.00 | 0 |
| bcsstk04 | 33 | 41 | 496.25 | t.l. | 11970000 | 36 | 38 | 133.74 | t.l. | 3850000 | 37 | 37 | 34.16 | 34.16 | 47776 |
| bcsstk05 | 18 | 21 | 284.95 | t.l. | 3490000 | 19 | 20 | 75.69 | t.l. | 1850000 | 20 | 20 | 0.05 | 0.05 | 169 |
| bwm200 | 2 | 2 | 0.11 | 0.11 | 201 | 2 | 2 | 0.06 | 0.06 | 201 | 2 | 2 | 0.00 | 0.00 | 201 |
| e05r0000 | 50 | 66 | 0.05 | t.l. | 1170000 | 50 | 53 | 263.59 | t.l. | 850000 | 50 | 53 | 305.71 | t.l. | 1770000 |
| fidap001 | 40 | 42 | 130.78 | t.l. | 1640000 | 40 | 41 | 78.88 | t.l. | 920000 | 40 | 43 | 189.11 | t.l. | 6880000 |
| fidap005 | 8 | 8 | 0.00 | 0.00 | 59 | 8 | 8 | 0.00 | 0.00 | 29 | 8 | 8 | 0.00 | 0.00 | 29 |
| fidapm05 | 11 | 11 | 60.42 | 2732.11 | 32484996 | 11 | 11 | 0.66 | 0.66 | 3956 | 11 | 11 | 0.00 | 130.56 | 3191485 |
| lund_a | 17 | 23 | 0.00 | t.l. | 1790000 | 19 | 23 | 0.05 | t.l. | 1830000 | 19 | 23 | 0.06 | t.l. | 15480000 |
| lund_b | 17 | 23 | 0.05 | t.l. | 2230000 | 19 | 23 | 4.50 | t.l. | 1830000 | 19 | 23 | 4.72 | t.l. | 14260000 |
| nos4 | 10 | 10 | 6.86 | 6.86 | 8874 | 10 | 10 | 6.32 | 6.32 | 7962 | 10 | 10 | 0.00 | 145.44 | 2471871 |
| pde225 | 11 | 15 | 0.06 | t.l. | 820000 | 11 | 15 | 0.05 | t.l. | 830000 | 11 | 15 | 0.05 | t.l. | 13870000 |
| rdb200 | 15 | 20 | 0.00 | t.l. | 1170000 | 15 | 20 | 0.00 | t.l. | 1070000 | 15 | 20 | 3.74 | t.l. | 15930000 |
| saylr1 | 11 | 14 | 0.06 | t.l. | 560000 | 12 | 14 | 0.11 | t.l. | 750000 | 11 | 14 | 0.00 | t.l. | 8070000 |
| steam1 | 29 | 64 | 2845.52 | t.l. | 740245 | 32 | 56 | 2168.79 | t.l. | 490241 | 32 | 45 | 827.23 | t.l. | 7990000 |
| steam3 | 7 | 7 | 0.00 | 0.00 | 81 | 7 | 7 | 0.00 | 0.00 | 81 | 7 | 7 | 0.00 | 0.00 | 81 |
| tub100 | 2 | 2 | 0.00 | 0.00 | 0 | 2 | 2 | 0.00 | 0.00 | 0 | 2 | 2 | 0.00 | 0.00 | 0 |

Table 4: Comparison of enumeration schemes.

| | method of [7] | | | HYBRID | | | | |
|---|---|---|---|---|---|---|---|---|
| name | $\underline{\phi}$ | time | subpr | $\underline{\phi}$ | $\overline{\phi}$ | h-time | time | subpr |
| bcspwr01 | 5 | 41.91 | 14858401 | 5 | 5 | 0.05 | 1.75 | 89955 |
| bcspwr02 | 6 | t.l. | 1111508124 | 7 | 7 | 0.88 | 0.88 | 77098 |
| bcspwr03 | 9 | t.l. | 745964464 | 10 | 11 | 60.03 | t.l. | 66410000 |
| can__144 | 11 | t.l. | 720547084 | 13 | 13 | 0.00 | 0.00 | 146 |
| can__161 | 16 | t.l. | 995371667 | 18 | 18 | 0.00 | 0.00 | 164 |
| can__187 | 12 | t.l. | 888556396 | 13 | 13 | 0.00 | 0.00 | 205 |
| can__229 | 19 | t.l. | 523991741 | 24 | 29 | 331.91 | t.l. | 11760000 |
| can___24 | 5 | 0.00 | 145 | 5 | 5 | 0.00 | 0.00 | 25 |
| can___61 | 12 | t.l. | 981671303 | 13 | 13 | 0.00 | 0.00 | 62 |
| can___62 | 5 | t.l. | 1011543437 | 6 | 6 | 0.00 | 0.00 | 549 |
| can___73 | 12 | t.l. | 1242189524 | 16 | 16 | 0.60 | 0.60 | 5012 |
| can___96 | 13 | t.l. | 1579396857 | 13 | 13 | 0.00 | 0.00 | 97 |
| dwt__162 | 8 | t.l. | 591233876 | 11 | 13 | 316.37 | t.l. | 34120000 |
| dwt__193 | 28 | t.l. | 1047040262 | 31 | 38 | 1031.61 | t.l. | 14680000 |
| dwt__209 | 18 | t.l. | 368405422 | 21 | 28 | 421.39 | t.l. | 13170000 |
| dwt__221 | 9 | t.l. | 871272016 | 11 | 13 | 301.43 | t.l. | 9710000 |
| dwt__245 | 15 | t.l. | 365660902 | 21 | 21 | 2616.03 | 2616.03 | 10604348 |
| dwt___59 | 6 | 13.57 | 4024432 | 6 | 6 | 60.03 | 60.03 | 3122523 |
| dwt___66 | 3 | 0.00 | 66 | 3 | 3 | 0.00 | 0.00 | 67 |
| dwt___72 | 4 | t.l. | 997042066 | 5 | 5 | 0.16 | 0.16 | 7146 |
| dwt___87 | 8 | t.l. | 1026957908 | 10 | 10 | 0.00 | 0.00 | 88 |
| bcsstk01 | 15 | t.l. | 1371830047 | 16 | 16 | 0.00 | 0.00 | 106 |
| bcsstk02 | 65 | 0.00 | 2146 | 65 | 65 | 0.00 | 0.00 | 0 |
| bcsstk04 | 28 | t.l. | 1533739977 | 37 | 37 | 34.16 | 34.16 | 47776 |
| bcsstk05 | 15 | t.l. | 453246149 | 20 | 20 | 0.06 | 0.06 | 169 |
| bwm200 | 2 | 0.00 | 201 | 2 | 2 | 0.00 | 0.00 | 201 |
| e05r0000 | 47 | t.l. | 755374321 | 50 | 53 | 277.43 | t.l. | 2060000 |
| fidap001 | 36 | t.l. | 1000415412 | 40 | 41 | 245.08 | t.l. | 6040000 |
| fidap005 | 8 | 0.00 | 688 | 8 | 8 | 0.00 | 0.00 | 29 |
| fidapm05 | 11 | 126.39 | 60356522 | 11 | 11 | 0.00 | 60.75 | 1521913 |
| lund_a | 14 | t.l. | 1357107291 | 19 | 23 | 0.06 | t.l. | 14650000 |
| lund_b | 14 | t.l. | 1357257996 | 19 | 23 | 0.00 | t.l. | 13540000 |
| nos4 | 9 | t.l. | 582865781 | 10 | 10 | 0.00 | 66.29 | 1002861 |
| pde225 | 8 | t.l. | 315172726 | 11 | 15 | 0.00 | t.l. | 12480000 |
| rdb200 | 11 | t.l. | 476306794 | 15 | 20 | 4.34 | t.l. | 14520000 |
| saylr1 | 9 | t.l. | 273666777 | 11 | 14 | 0.00 | t.l. | 8640000 |
| steam1 | 24 | t.l. | 1703461772 | 32 | 45 | 788.35 | t.l. | 6050000 |
| steam3 | 7 | 0.00 | 159 | 7 | 7 | 0.00 | 0.00 | 81 |
| tub100 | 2 | 0.00 | 100 | 2 | 2 | 0.00 | 0.00 | 0 |

Table 5: Comparison of the method of [7] and HYBRID.

| $\varphi$ | $n$ | dens | $\alpha(G)$ | $\gamma(G)$ | $\psi$ | method of [7] succ | time | $\underline{\phi}$ | HYBRID succ | time | $\underline{\phi}$ | $\overline{\phi}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 10 | 50 | 0.111 | 8.3 | 7.8 | 45.7 | 4 | 64.76 | 8.5 | 9 | 49.99 | 8.0 | 9.0 |
| | 100 | 0.059 | 9.2 | 8.7 | 95.1 | 1 | 194.77 | 8.8 | 7 | 0.02 | 9.0 | 10.0 |
| | 150 | 0.040 | 9.6 | 8.7 | 142.9 | 0 | - | 8.8 | 8 | 3.42 | 9.5 | 10.5 |
| | 200 | 0.030 | 9.7 | 9.0 | 192.5 | 0 | - | 8.8 | 8 | 4.42 | 9.0 | 10.0 |
| | 250 | 0.024 | 9.7 | 9.0 | 241.1 | 0 | - | 9.0 | 8 | 13.50 | 9.5 | 11.0 |
| 20 | 50 | 0.198 | 12.7 | 15.4 | 46.6 | 0 | - | 17.1 | 8 | 9.25 | 18.5 | 19.5 |
| | 100 | 0.110 | 18.4 | 18.0 | 95.3 | 0 | - | 16.9 | 8 | 0.29 | 19.0 | 20.0 |
| | 150 | 0.076 | 19.4 | 18.4 | 146.8 | 0 | - | 17.2 | 5 | 11.04 | 19.2 | 20.4 |
| | 200 | 0.057 | 19.6 | 18.5 | 196.7 | 0 | - | 18.0 | 7 | 1.81 | 19.7 | 20.7 |
| | 250 | 0.046 | 19.7 | 18.7 | 243.4 | 0 | - | 17.9 | 7 | 5.08 | 19.0 | 20.7 |
| 30 | 100 | 0.156 | 24.5 | 25.9 | 96.8 | 0 | - | 26.2 | 5 | 7.77 | 28.8 | 29.8 |
| | 150 | 0.108 | 28.1 | 27.1 | 147.2 | 0 | - | 25.6 | 5 | 1.11 | 29.0 | 30.6 |
| | 200 | 0.083 | 29.0 | 28.0 | 196.1 | 0 | - | 26.1 | 2 | 12.55 | 29.0 | 30.3 |
| | 250 | 0.068 | 29.4 | 28.1 | 245.8 | 0 | - | 27.1 | 4 | 0.06 | 29.0 | 30.0 |
| 40 | 100 | 0.194 | 25.0 | 33.2 | 96.9 | 0 | - | 34.6 | 1 | 16.31 | 37.9 | 39.4 |
| | 150 | 0.138 | 36.5 | 37.1 | 147.4 | 0 | - | 35.9 | 1 | 0.06 | 38.7 | 40.1 |
| | 200 | 0.108 | 38.1 | 37.5 | 196.5 | 0 | - | 34.8 | 1 | 0.11 | 39.0 | 40.2 |
| | 250 | 0.088 | 39.1 | 37.8 | 245.6 | 0 | - | 36.0 | 2 | 145.63 | 39.0 | 40.3 |
| 50 | 100 | 0.227 | 25.0 | 41.2 | 97.3 | 0 | - | 43.7 | 4 | 45.09 | 47.0 | 49.2 |
| | 150 | 0.166 | 38.0 | 45.9 | 146.8 | 0 | - | 43.0 | 2 | 0.05 | 48.1 | 49.8 |
| | 200 | 0.131 | 47.3 | 47.1 | 196.9 | 0 | - | 42.9 | 5 | 2.01 | 49.0 | 51.0 |
| | 250 | 0.108 | 48.3 | 47.6 | 246.9 | 0 | - | 43.8 | 2 | 7.53 | 49.1 | 50.5 |
| 60 | 150 | 0.192 | 38.0 | 51.5 | 147.3 | 0 | - | 51.9 | 0 | - | 57.9 | 59.9 |
| | 200 | 0.153 | 50.0 | 56.0 | 197.1 | 0 | - | 51.6 | 0 | - | 58.6 | 60.5 |
| | 250 | 0.127 | 57.6 | 57.3 | 246.7 | 0 | - | 51.7 | 1 | 185.48 | 58.9 | 60.6 |
| 70 | 150 | 0.215 | 38.0 | 62.2 | 146.9 | 0 | - | 60.0 | 6 | 35.77 | 68.0 | 69.3 |
| | 200 | 0.174 | 50.0 | 65.8 | 196.5 | 0 | - | 62.0 | 0 | - | 68.0 | 69.9 |
| | 250 | 0.145 | 62.5 | 66.2 | 247.3 | 0 | - | 63.0 | 1 | 91.89 | 68.2 | 70.4 |
| 80 | 200 | 0.192 | 50.0 | 72.0 | 196.7 | 0 | - | 68.0 | 0 | - | 78.0 | 80.3 |
| | 250 | 0.162 | 63.0 | 76.1 | 246.8 | 0 | - | 66.8 | 1 | 2.85 | 78.3 | 80.2 |
| 90 | 200 | 0.210 | 50.0 | 81.6 | 197.6 | 0 | - | 74.2 | 1 | 8.62 | 87.4 | 90.4 |
| | 250 | 0.178 | 63.0 | 83.1 | 246.8 | 0 | - | 81.5 | 1 | 291.55 | 87.8 | 89.7 |
| 100 | 200 | 0.226 | 50.0 | 92.9 | 198.1 | 0 | - | 82.9 | 4 | 221.91 | 97.3 | 99.0 |
| | 250 | 0.193 | 63.0 | 92.5 | 247.3 | 0 | - | 83.6 | 1 | 278.14 | 97.8 | 99.8 |

Table 6: Results on randomly generated instances for $p = 0.3$ (10 trials for each line).

| | | | | | | method of [7] | | | HYBRID | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\varphi$ | $n$ | dens | $\alpha(G)$ | $\gamma(G)$ | $\psi$ | succ | time | $\underline{\phi}$ | succ | time | $\underline{\phi}$ | $\overline{\phi}$ |
| | 50 | 0.182 | 9.9 | 9.4 | 47.3 | 10 | 7.92 | - | 10 | 0.80 | - | - |
| | 100 | 0.096 | 10.0 | 9.5 | 95.9 | 10 | 30.02 | - | 10 | 2.16 | - | - |
| 10 | 150 | 0.065 | 10.0 | 9.8 | 144.0 | 9 | 4.92 | 10.0 | 10 | 0.55 | - | - |
| | 200 | 0.049 | 10.0 | 10.0 | 193.5 | 0 | - | 10.0 | 10 | 2.15 | - | - |
| | 250 | 0.039 | 10.0 | 10.0 | 241.0 | 8 | 7.42 | 10.0 | 10 | 4.72 | - | - |
| | 50 | 0.326 | 13.2 | 17.7 | 47.2 | 4 | 131.12 | 19.5 | 9 | 25.36 | 19.0 | 20.0 |
| | 100 | 0.181 | 19.9 | 19.3 | 96.8 | 1 | 253.75 | 19.6 | 9 | 0.03 | 20.0 | 21.0 |
| 20 | 150 | 0.125 | 20.0 | 19.8 | 146.8 | 2 | 77.58 | 19.4 | 10 | 1.14 | - | - |
| | 200 | 0.095 | 20.0 | 19.9 | 197.4 | 0 | - | 19.4 | 9 | 1.06 | 20.0 | 21.0 |
| | 250 | 0.076 | 20.0 | 19.9 | 244.5 | 0 | - | 20.0 | 10 | 2.08 | - | - |
| | 100 | 0.257 | 25.0 | 28.6 | 98.0 | 0 | - | 29.5 | 7 | 1.76 | 30.0 | 31.0 |
| 30 | 150 | 0.180 | 30.0 | 29.3 | 147.5 | 0 | - | 29.2 | 6 | 2.13 | 30.0 | 31.0 |
| | 200 | 0.138 | 30.0 | 29.2 | 197.3 | 0 | - | 29.1 | 6 | 17.00 | 30.0 | 31.3 |
| | 250 | 0.113 | 30.0 | 29.3 | 246.9 | 0 | - | 29.4 | 7 | 4.51 | 30.0 | 31.3 |
| | 100 | 0.321 | 25.5 | 38.3 | 97.9 | 0 | - | 39.0 | 7 | 27.74 | 39.7 | 40.7 |
| 40 | 150 | 0.230 | 38.0 | 39.2 | 147.9 | 0 | - | 39.1 | 7 | 2.17 | 40.0 | 41.3 |
| | 200 | 0.180 | 40.0 | 39.3 | 197.1 | 0 | - | 38.9 | 9 | 1.29 | 40.0 | 41.0 |
| | 250 | 0.147 | 40.0 | 39.4 | 247.8 | 0 | - | 38.9 | 3 | 0.09 | 40.0 | 41.0 |
| | 100 | 0.376 | 28.9 | 47.6 | 97.9 | 0 | - | 49.3 | 5 | 0.27 | 49.0 | 50.4 |
| 50 | 150 | 0.277 | 38.0 | 48.7 | 147.6 | 0 | - | 49.2 | 7 | 2.61 | 49.7 | 51.0 |
| | 200 | 0.219 | 49.3 | 49.1 | 197.3 | 0 | - | 49.0 | 7 | 0.07 | 49.7 | 51.0 |
| | 250 | 0.180 | 50.0 | 49.0 | 247.6 | 0 | - | 49.0 | 5 | 0.09 | 50.0 | 51.2 |
| | 150 | 0.320 | 38.1 | 57.8 | 147.7 | 0 | - | 59.1 | 5 | 0.08 | 59.6 | 60.8 |
| 60 | 200 | 0.255 | 50.0 | 58.8 | 197.6 | 0 | - | 58.8 | 8 | 4.09 | 60.0 | 61.0 |
| | 250 | 0.211 | 59.7 | 59.1 | 247.4 | 0 | - | 59.0 | 6 | 12.34 | 60.0 | 61.5 |
| | 150 | 0.358 | 40.6 | 67.7 | 147.8 | 0 | - | 68.8 | 7 | 7.40 | 70.0 | 71.3 |
| 70 | 200 | 0.289 | 50.0 | 67.9 | 197.7 | 0 | - | 68.8 | 9 | 21.62 | 70.0 | 71.0 |
| | 250 | 0.241 | 63.0 | 68.6 | 248.4 | 0 | - | 68.8 | 5 | 4.84 | 70.0 | 72.0 |
| 80 | 200 | 0.320 | 50.0 | 77.5 | 198.3 | 0 | - | 79.0 | 6 | 5.86 | 79.8 | 80.8 |
| | 250 | 0.269 | 63.0 | 78.8 | 247.9 | 0 | - | 78.8 | 6 | 22.06 | 80.0 | 81.0 |
| 90 | 200 | 0.349 | 51.2 | 87.4 | 198.2 | 0 | - | 88.5 | 8 | 3.21 | 90.0 | 92.0 |
| | 250 | 0.295 | 63.0 | 87.8 | 248.0 | 0 | - | 88.4 | 5 | 1.02 | 90.0 | 91.2 |
| 100 | 200 | 0.375 | 55.0 | 97.8 | 198.3 | 0 | - | 98.8 | 2 | 94.33 | 99.3 | 100.8 |
| | 250 | 0.320 | 63.0 | 97.5 | 247.9 | 0 | - | 98.4 | 3 | 3.92 | 99.7 | 101.4 |

Table 7: Results on randomly generated instances for $p = 0.5$ (10 trials for each line).

| | | | | | | HYBRID | | | |
|---|---|---|---|---|---|---|---|---|---|
| $\varphi$ | $n$ | dens | $\alpha(G)$ | $\gamma(G)$ | $\psi$ | succ | time | $\underline{\phi}$ | $\overline{\phi}$ |
| 50 | 400 | 0.072 | 49.6 | 48.2 | 396.5 | 3 | 94.56 | 49.7 | 51.0 |
| | 600 | 0.049 | 50.0 | 48.4 | 595.2 | 1 | 15.99 | 50.0 | 51.4 |
| | 800 | 0.037 | 50.0 | 48.7 | 793.5 | 3 | 34.12 | 50.0 | 51.4 |
| | 1000 | 0.030 | 50.0 | 48.9 | 991.0 | 1 | 2.09 | 50.0 | 52.2 |
| 100 | 400 | 0.134 | 97.7 | 97.7 | 396.6 | 0 | - | 98.9 | 100.5 |
| | 600 | 0.094 | 99.1 | 98.0 | 597.2 | 1 | 16.15 | 99.1 | 101.1 |
| | 800 | 0.072 | 99.8 | 98.3 | 796.5 | 1 | 1.65 | 99.8 | 101.6 |
| | 1000 | 0.059 | 99.9 | 98.2 | 995.6 | 1 | 3.08 | 99.9 | 102.8 |
| 150 | 400 | 0.188 | 100.0 | 142.6 | 397.4 | 0 | - | 148.0 | 151.0 |
| | 600 | 0.135 | 147.5 | 146.8 | 597.8 | 0 | - | 149.1 | 151.6 |
| | 800 | 0.105 | 148.9 | 147.5 | 795.8 | 0 | - | 149.2 | 152.0 |
| | 1000 | 0.085 | 149.8 | 147.8 | 996.3 | 1 | 4.31 | 149.8 | 152.6 |
| 200 | 400 | 0.231 | 100.0 | 192.4 | 397.3 | 1 | 409.44 | 197.2 | 206.9 |
| | 600 | 0.171 | 150.0 | 196.2 | 596.6 | 0 | - | 198.3 | 200.6 |
| | 800 | 0.135 | 197.5 | 197.1 | 797.6 | 0 | - | 199.0 | 201.5 |
| | 1000 | 0.111 | 198.8 | 197.4 | 996.2 | 0 | - | 199.0 | 202.5 |

Table 8: Results on large randomly generated instances for $p = 0.3$ (10 trials for each line).

| | | | | | | HYBRID | | | |
|---|---|---|---|---|---|---|---|---|---|
| $\varphi$ | $n$ | dens | $\alpha(G)$ | $\gamma(G)$ | $\psi$ | succ | time | $\underline{\phi}$ | $\overline{\phi}$ |
| 50 | 400 | 0.119 | 50.0 | 49.3 | 397.4 | 2 | 6.10 | 50.0 | 51.5 |
| | 600 | 0.081 | 50.0 | 49.9 | 595.6 | 3 | 126.33 | 50.0 | 51.3 |
| | 800 | 0.061 | 50.0 | 49.9 | 795.5 | 2 | 39.10 | 50.0 | 51.3 |
| | 1000 | 0.049 | 50.0 | 49.9 | 993.7 | 0 | - | 50.0 | 51.7 |
| 100 | 400 | 0.222 | 99.6 | 99.0 | 397.4 | 3 | 4.34 | 100.0 | 101.1 |
| | 600 | 0.155 | 100.0 | 99.5 | 597.6 | 3 | 6.45 | 100.0 | 101.0 |
| | 800 | 0.119 | 100.0 | 99.6 | 797.0 | 3 | 18.79 | 100.0 | 101.7 |
| | 1000 | 0.096 | 100.0 | 99.7 | 997.0 | 0 | - | 100.0 | 101.5 |
| 150 | 400 | 0.309 | 100.0 | 147.7 | 397.8 | 4 | 3.85 | 149.8 | 151.5 |
| | 600 | 0.222 | 149.5 | 149.0 | 597.8 | 3 | 1.93 | 150.0 | 151.0 |
| | 800 | 0.172 | 150.0 | 149.1 | 796.8 | 2 | 18.37 | 150.0 | 151.3 |
| | 1000 | 0.140 | 150.0 | 149.3 | 997.0 | 0 | - | 150.0 | 151.8 |
| 200 | 400 | 0.380 | 110.5 | 197.8 | 397.9 | 3 | 20.79 | 199.0 | 200.9 |
| | 600 | 0.281 | 150.0 | 199.1 | 597.4 | 4 | 14.04 | 200.0 | 201.2 |
| | 800 | 0.221 | 199.6 | 199.2 | 797.6 | 3 | 17.07 | 200.0 | 201.3 |
| | 1000 | 0.182 | 200.0 | 199.2 | 996.9 | 0 | - | 200.0 | 202.0 |

Table 9: Results on large randomly generated instances for $p = 0.5$ (10 trials for each line).