INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

# *Sous-graphes avec contraintes sur le degré : difficulté et approximation*

Omid Amini — David Peleg — Stéphane Pérennes — Ignasi Sau — Saket Saurabh

**N° 6690**

September 2008

Thème COM


ISSN 0249-6399 ISRN INRIA/RR--6690--FR+ENG

# Sous-graphes avec contraintes sur le degré : difficulté et approximation *

Omid Amini† , David Peleg‡ , Stéphane Pérennes§ , Ignasi Sau¶‖ , Saket Saurabh**

Thème COM — Systèmes communicants
Projet MASCOTTE

Rapport de recherche n° 6690 — September 2008 — 29 pages

**Résumé :** A general instance of a Degree-Constrained Subgraph problem consists of an edge-weighted or vertex-weighted graph $G$ and the objective is to find an optimal weighted subgraph, subject to certain degree constraints on the vertices of the subgraph. This class of combinatorial problems has been extensively studied due to its numerous applications in network design. If the input graph is bipartite, these problems are equivalent to classical transportation and assignment problems in operations research. This paper considers three natural Degree-Constrained Subgraph problems and studies their behavior in terms of approximation algorithms. These problems take as input an undirected graph $G = (V, E)$, with $|V| = n$ and $|E| = m$. Our results, together with the definition of the three problems, are listed below.

* This work has been partially supported by European project IST FET AEOLUS, PACA region of France, Ministerio de Educación y Ciencia of Spain, European Regional Development Fund under project TEC2005-03575, Catalan Research Council under project 2005SGR00256 and COST action 293 GRAAL, and has been done in the context of the CRC Corso with France Telecom.

† Max-Planck-Institut für Informatik, Saarbrücken, GERMANY. amini@mpi-inf.mpg.de

‡ Weizmann Institute of Science, Rehovot, ISRAEL. david.peleg@weizmann.ac.il

§ Mascotte joint project - INRIA/CNRS-I3S/UNSA, Sophia-Antipolis, FRANCE. Stephane.Perennes@sophia.inria.fr

¶ Mascotte joint project - INRIA/CNRS-I3S/UNSA, Sophia-Antipolis, FRANCE. Ignasi.Sau@sophia.inria.fr

‖ Graph Theory and Combinatorics group at Applied Mathematics IV Department of UPC, Barcelona, SPAIN.

** Department of Informatics, University of Bergen, Bergen, NORWAY. saket.saurabh@ii.uib.no

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : +33 4 92 38 77 77 — Télécopie : +33 4 92 38 77 65

- The Maximum Degree-Bounded Connected Subgraph ($\mathrm{MDBCS}_d$) problem takes as input a weight function $\omega : E \to \mathbb{R}^+$ and an integer $d \geq 2$, and asks for a subset $E' \subseteq E$ such that the subgraph $G' = (V, E')$ is connected, has maximum degree at most $d$, and $\sum_{e \in E'} \omega(e)$ is maximized. This problem is one of the classical NP-hard problems listed by Garey and Johnson in *(Computers and Intractability, W.H. Freeman, 1979)*, but there were no results in the literature except for $d = 2$. We prove that $\mathrm{MDBCS}_d$ is not in Apx for any $d \geq 2$ (this was known only for $d = 2$) and we provide a $(\min\{m/\log n,\ nd/(2\log n)\})$-approximation algorithm for unweighted graphs, and a $(\min\{n/2,\ m/d\})$-approximation algorithm for weighted graphs. We also prove that when $G$ accepts a low-degree spanning tree, in terms of $d$, then $\mathrm{MDBCS}_d$ can be approximated within a small constant factor in unweighted graphs.
- The Minimum Subgraph of Minimum Degree$_{\geq d}$ ($\mathrm{MSMD}_d$) problem consists in finding a smallest subgraph of $G$ (in terms of number of vertices) with minimum degree at least $d$. We prove that $\mathrm{MSMD}_d$ is not in Apx for any $d \geq 3$ and we provide an $\mathcal{O}(n/\log n)$-approximation algorithm for the classes of graphs excluding a fixed graph as a minor, using dynamic programming techniques and a known structural result on graph minors. In particular, this approximation algorithm applies to planar graphs and graphs of bounded genus.
- The Dual Degree-Dense $k$-Subgraph ($\mathrm{DDD}k\mathrm{S}$) problem consists in finding a subgraph $H$ of $G$ such that $|V(H)| \leq k$ and $\delta_H$ is maximized, where $\delta_H$ is the minimum degree in $H$. We present a deterministic $\mathcal{O}(n^\delta)$-approximation algorithm in general graphs, for some universal constant $\delta < 1/3$.

**Mots-clés :** Approximation Algorithms, Degree-Constrained Subgraphs, Hardness of Approximation, Apx, PTAS, Dense Subgraphs, Graph Minors, Excluded Minor.

# Degree-Constrained Subgraph Problems : Hardness and Approximation Results ††

**Abstract:** A general instance of a Degree-Constrained Subgraph problem consists of an edge-weighted or vertex-weighted graph $G$ and the objective is to find an optimal weighted subgraph, subject to certain degree constraints on the vertices of the subgraph. This class of combinatorial problems has been extensively studied due to its numerous applications in network design. If the input graph is bipartite, these problems are equivalent to classical transportation and assignment problems in operations research. This paper considers three natural Degree-Constrained Subgraph problems and studies their behavior in terms of approximation algorithms. These problems take as input an undirected graph $G = (V, E)$, with $|V| = n$ and $|E| = m$. Our results, together with the definition of the three problems, are listed below.

- The Maximum Degree-Bounded Connected Subgraph (MDBCS$_d$) problem takes as input a weight function $\omega : E \to \mathbb{R}^+$ and an integer $d \geq 2$, and asks for a subset $E' \subseteq E$ such that the subgraph $G' = (V, E')$ is connected, has maximum degree at most $d$, and $\sum_{e \in E'} \omega(e)$ is maximized. This problem is one of the classical NP-hard problems listed by Garey and Johnson in *(Computers and Intractability, W.H. Freeman, 1979)*, but there were no results in the literature except for $d = 2$. We prove that MDBCS$_d$ is not in Apx for any $d \geq 2$ (this was known only for $d = 2$) and we provide a $(\min\{m/\log n,\ nd/(2\log n)\})$-approximation algorithm for unweighted graphs, and a $(\min\{n/2,\ m/d\})$-approximation algorithm for weighted graphs. We also prove that when $G$ accepts a low-degree spanning tree, in terms of $d$, then MDBCS$_d$ can be approximated within a small constant factor in unweighted graphs.
- The Minimum Subgraph of Minimum Degree$_{\geq d}$ (MSMD$_d$) problem consists in finding a smallest subgraph of $G$ (in terms of number of vertices) with minimum degree at least $d$. We prove that MSMD$_d$ is not in Apx for any $d \geq 3$ and we provide an $\mathcal{O}(n/\log n)$-approximation algorithm for the classes of graphs excluding a fixed graph as a minor, using dynamic programming techniques and a known structural result on graph minors. In particular, this approximation algorithm applies to planar graphs and graphs of bounded genus.
- The Dual Degree-Dense $k$-Subgraph (DDD$k$S) problem consists in finding a subgraph $H$ of $G$ such that $|V(H)| \leq k$ and $\delta_H$ is maximized, where $\delta_H$ is the minimum degree in $H$. We present a deterministic $\mathcal{O}(n^\delta)$-approximation algorithm in general graphs, for some universal constant $\delta < 1/3$.

**Key-words:** Approximation Algorithms, Degree-Constrained Subgraphs, Hardness of Approximation, Apx, PTAS, Dense Subgraphs, Graph Minors, Excluded Minor.

# 1 Introduction

In this paper we consider three natural Degree-Constrained Subgraph problems and study them in terms of approximation algorithms. A general instance of a Degree-Constrained Subgraph problem [1,5,24] consists of an edge-weighted or vertex-weighted graph $G$ and the objective is to find an optimal weighted subgraph, subject to certain degree constraints on the vertices of the subgraph. These problems have attracted a lot of attention in the last decades and have resulted in a large body of literature [1,5,10,12–14,16,19,22,24]. The most well-studied ones are probably the Minimum-Degree Spanning Tree [12] and the Minimum-Degree Steiner Tree [13] problems.

Beyond the esthetic and theoretical appeal of Degree-Constrained Subgraph problems, the reasons for such intensive study are rooted in their wide applicability in the areas of interconnection networks and routing algorithms, among others. For instance, given an interconnection network modeled by an undirected graph, one may be interested in finding a small subset of nodes having high degree of connectivity for each node. This translates to finding a small subgraph with a lower bound on the degree of its vertices, i.e. to the MSMD$_d$ problem. Note that if the input graph is bipartite, these problems are equivalent to classical transportation and assignment problems in operation research.
The first problem studied in the paper is a classical NP-hard problem listed in [15] (cf. Problem [GT26] for the unweighted version) :

> Maximum Degree-Bounded Connected Subgraph (MDBCS$_d$)
> **Input** : A graph $G = (V, E)$, a weight function $\omega : E \to \mathbb{R}^+$ and an integer $d \geq 2$.
> **Output** : A subset $E' \subseteq E$ such that the subgraph $G' = (V, E')$ is connected, has maximum degree at most $d$, and $\sum_{e \in E'} \omega(E)$ is maximized.

For $d = 2$, the unweighted MDBCS$_d$ problem corresponds to the Longest Path problem. Indeed, given the input graph $G$ (which can be assumed to be connected), let $P$ and $G'$ be optimal solutions of Longest Path and MDBCS$_2$ in $G$, respectively. Then observe that $|E(G')| = |E(P)|$ unless $G$ is Hamiltonian, in which case $|E(G')| = |E(P)| + 1$. One could also ask the question : what happens when $G'$ is not required to be connected in the definition of MDBCS$_d$? It turns out that without the connectivity constraint, both the edge version and the vertex version (where the goal is to maximize the total weight of the vertices of a subgraph respecting the degree constraints) of the MDBCS$_d$ problem are known to be solvable in polynomial time using matching techniques [7,15,18]. In fact, without connectivity constraints, even a more general version where the input contains an interval of allowed degrees for each node is known to be solvable in polynomial time.

The most general version of Degree-Constrained Subgraph problems is to find a subgraph under constraints given by lower and upper bounds on the degree of each vertex, the objective being to minimize or maximize some parameter (usually the size of the subgraph). A common variant ignores the lower bound on the degree and just requires the vertices of the subgraphs to have a given maximum degree [22], in which case the typical optimization criterion is to maximize the size of a subgraph satisfying the degree constraints. The resulting

problem is also called an Upper Degree-Constrained Subgraph problem in [14]. In contrast, we are unaware of existing results considering just a lower bound on the degrees of the vertices of the subgraph, except for combinatorial conditions on the existence of such a subgraph [10]. In an attempt to fill this void in the literature, the last two problems considered in this paper aim at minimizing the size of a subgraph and maximizing the lower bound on the minimum degree, respectively. For a graph $H$, let $\delta_H$ denote the minimum degree of the vertices in $H$.

Minimum Subgraph of Minimum Degree$_{\geq d}$ (MSMD$_d$)
**Input** : An undirected graph $G = (V, E)$ and an integer $d \geq 2$.
**Output** : A subset $S \subseteq V$ such that for $H = G[S]$, $\delta_H \geq d$ and $|S|$ is minimized.

Dual Degree-Dense $k$-Subgraph (DDD$k$S)
**Input** : An undirected graph $G = (V, E)$ and a positive integer $k$.
**Output** : An induced subgraph $H$ of size $|V(H)| \leq k$, such that $\delta_H$ is maximized.

MSMD$_d$ is closely related to MDBCS$_d$. Indeed, MSMD$_d$ corresponds exactly to the dual (unweighted) node-minimization version of MDBCS$_d$. MSMS$_d$ is also a generalization of the Girth problem (finding a shortest cycle), which corresponds exactly to the case $d = 2$. In Amini *et al.* [4], the MSMD$_d$ problem was introduced and studied in the realm of the parameterized complexity. It was shown that MSMD$_d$ is W[1]-hard for $d \geq 3$ and explicit FPT algorithms were given for the class of graphs excluding a fixed graph as a minor and graphs of bounded local-treewidth. Besides the above discussion, our main motivation for studying MSMD$_d$ is its close relation to the well studied Dense $k$-Subgraph (D$k$S) [11,17] and Traffic Grooming [3] problems. Indeed, if good approximate solutions could be found for the MSMS$_d$ problem, then one could also find good approximate solutions (up to a constant factor) for the D$k$S and Traffic Grooming problems. Roughly, the idea is that a small subgraph with minimum degree at least $d$ has density at least $\frac{d}{2}$, and this provides an approximation for the densest subgraph (in fact, Traffic Grooming can be reduced, essentially, to finding dense subgraphs). See [3, 4] for further details.

The above discussion illustrates that the study of the above mentioned problems is very natural and that the results obtained for them can reverberate in several other important optimization problems, coming from both theoretical and practical domains.

**Our Results :** In this paper we obtain both approximation algorithms and results on hardness of approximation. All the hardness results are based on the hypothesis P $\neq$ NP. More precisely, our results are the following :

- We prove that the MDBCS$_d$ problem is not in Apx for any $d \geq 2$. On the other hand, we give an approximation algorithm for general unweighted graphs with ratio $\min\{m/\log n,\ nd/(2\log n)\}$, and an approximation algorithm for general weighted graphs with ratio $\min\{n/2,\ m/d\}$. The first algorithm uses an algorithm introduced in [2], that is based on the *color-coding* method. We also present a constant-factor approximation in Appendix D when the input graph accepts a low-degree spanning tree, in terms of the integer $d$.

- We prove that the MSMD$_d$ problem is not in APX for all $d \geq 3$. The proof is obtained by the following two steps. First, by a reduction from VERTEX COVER, we prove that MSMD$_d$ does not admit a PTAS. In particular, this implies that MSMD$_d$ is NP-hard for any $d \geq 3$. Then, we use the *error amplification* technique to prove that MSMD$_d$ is not in APX for any $d \geq 3$. On the positive side, we give an $\mathcal{O}(n/\log n)$-approximation algorithm for the class of graphs excluding a fixed graph $H$ as a minor, using a known structural result on graph minors and dynamic programming over graphs of bounded treewidth. In particular, this gives an $\mathcal{O}(n/\log n)$-approximation algorithm for planar graphs and graphs of bounded genus.
- We give a deterministic $\mathcal{O}(n^\delta)$-approximation algorithm for the DDD$k$S problem in general graphs, for some universal constant $\delta < 1/3$. We also provide a randomized $\mathcal{O}(\sqrt{n}\log n)$-approximation algorithm in Appendix I, which is completely different in nature. Although the approximation ratio is significantly worse, the idea of the proof is quite simple and nice.

**Organization of the paper :** In Section 2 we establish that MDBCS$_d$ is not in APX for any $d \geq 2$, and in Section 3 we present two approximation algorithms for unweighted and weighted general graphs, respectively. The constant-factor approximation for MDBCS$_d$ when the input graph accepts a low-degree spanning tree is provided in Appendix D for unweighted graphs. In Section 4 we prove that MSMD$_d$ is not in APX for any $d \geq 3$, and in Section 5 we give an $\mathcal{O}(n/\log n)$-approximation algorithm for the class of graphs excluding a fixed graph $H$ as a minor. In Section 6 we give two approximation algorithms for the DDD$k$S problem. Finally, we conclude with some remarks and open problems in Section 7. The omitted proofs and some basic definitions can be found in the appendices.

## 2 Hardness of Approximating MDBCS$_d$

As mentioned in Section 1, MDBCS$_2$ is exactly the LONGEST PATH problem, which is known to not accept any constant-factor approximation [16], unless P = NP. In this section we extend this result and prove that, under the assumption P$\neq$NP, MDBCS$_d$ is not in APX for any $d \geq 2$, proving first that MDBCS$_d$ is not in PTAS for any $d \geq 2$. We refer to Appendix A.1 for the definitions of the complexity classes APX, PTAS and for the notion of *gap-preserving reduction*, which will be used freely throughout the paper.

**Theorem 2.1** MDBCS$_d$ *does not admit a* PTAS *for any* $d \geq 2$, *unless* P = NP.

***Proof:*** We prove the result for the *case* when $d \geq 3$. The result for the case $d = 2$ follows from [16]. We give our reduction from TSP$(1,2)$, which does not have PTAS unless P = NP [21]. An instance of TSP$(1,2)$ consists of a complete graph $G = (V, E)$ on $n$ vertices and a weight function $f : E \to \{1, 2\}$ on its edges, and the objective is to find a traveling salesman tour of minimum edge weight in $G$.

We show that if we have a PTAS for MDBCS$_d$, $d \geq 3$, then we can construct a PTAS for TSP$(1,2)$. Towards this, we transform the graph $G$ into a new graph $G'$ with a modified weight function $g$ on its edges. For every vertex $v \in V$ we add $d-2$ new vertices

$\{v_1, \cdots, v_{d-2}\}$ and we add an edge from $v$ to every vertex $v_i$, $1 \leq i \leq d-2$. This concludes the description of $G'$. Let $V' = \{\{v_1, \cdots, v_{d-2}\} \mid v \in V\}$ be the set of new vertices, and let $E' = \{(v_i, v) \mid 1 \leq i \leq d-2, v \in V\}$ be the set of new edges. We define the weight function $g$ of $G'$ as follows : $g(e) = 3 - f(e)$ if $e \in E$ (weights of original edges get flipped), and $g(e) = 3$ if $e \in E'$.

Next we prove a claim showing the structure of the *maximal* solutions of MDBCS$_d$ in $G'$. Essentially, we show that given any solution $G_1$ of MDBCS$_d$ in $G'$ with value $W$, we can transform it into another solution $G_2$ of MDBCS$_d$ in $G'$ with value at least $W$, such that $G_2$ contains all the newly added edges and induces a hamiltonian cycle in $G$. The proof has been moved to Appendix B due to lack of space.

**Claim 1** *Given a solution $G_1 = (V \cup V', E_1)$ to* MDBCS$_d$ *in $G'$, we can transform it in polynomial time into a solution $G_2 = (V \cup V', E_2)$ of* MDBCS$_d$ *in $G'$ such that (a) $G_3 = (V, E \cap E_2)$ is a hamiltonian cycle in $G$ and; (b) $\sum_{e \in E_2} g(e) \geq \sum_{e' \in E_1} g(e')$.*

Suppose that there exists a PTAS for MDBCS$_d$ realized by an approximation scheme $\mathcal{A}_\delta$. This family of algorithms takes as input a graph $G''$ and a parameter $\delta > 0$, and returns a solution of MDBCS$_d$ of weight at least $(1-\delta)OPT_{G''}$, where $OPT_{G''}$ is the value of an optimum solution of MDBCS$_d$ in $G''$. Now we proceed to construct a PTAS for TSP$(1,2)$. Given a graph $G$, an instance of TSP$(1,2)$, and $\varepsilon > 0$, we do as follows :

1. Fix $\delta = h(\varepsilon, d)$ (to be specified later) and run $\mathcal{A}_\delta$ on $G'$ (the graph obtained from $G$ with the transformation described above).
2. Apply the polynomial time transformation described in Claim 1 on the solution obtained by $\mathcal{A}_\delta$ on $G'$. Let the new solution be $G^* = (V \cup V_1, E^*)$.
3. Return $E^* \cap E$ as the solution of TSP$(1,2)$.

Now we prove that the solution returned by our algorithm is of desired kind, that is $\sum_{e \in E^* \cap E} f(e) \leq (1+\varepsilon)O_T$, where $O_T$ is the weight of an optimum tour in $G$. Let such an optimum tour contain $a$ edges of weight 1 and $b$ edges of weight 2. Then $O_T = a + 2b$ and $a + b = n$. Equivalently $a = 2n - O_T$ and $b = O_T - n$. Let $O_D$ be the value of an optimum solution of MDBCS$_d$ in $G'$.

Then by Claim 1 and the flipping nature of the function $g$, we have that

$$O_D = (d-2)3n + 2a + b. \tag{1}$$

Let $3(d-2)n + O_D^*$ be the value of the solution returned by $\mathcal{A}_\delta$, where $O_D^*$ is the sum of the weights of the edges of the hamiltonian cycle in $G$, that is $O_D^* = \sum_{e \in E^* \cap E} g(e)$. Since $\mathcal{A}_\delta$ is a PTAS,

$$3(d-2)n + O_D^* \geq (1-\delta)O_D. \tag{2}$$

Combining Equation (1) and Inequality (2) gives

$$O_D^* \geq (1-\delta)O_D - 3(d-2)n = 3n - O_T + \delta O_T - n(3d-3)\delta. \tag{3}$$

On the other hand, the value of the solution returned by our algorithm for TSP$(1,2)$ is $O_T^* = 3n - O_D^*$ (since if $O_D^* = 2x + y$, $x$ being the number of edges of weight 2 and $y$ being the number of edges of weight 1, with $x+y = n$, then the value of the solution for TSP$(1,2)$ is $x + 2y$). Substituting $O_D^* = 3n - O_T^*$ in Inequality (3) and using that $O_T \geq n$ yields

$$O_T^* \leq O_T - \delta O_T + n(3d-3)\delta \leq O_T - \delta n + n(3d-3)\delta. \quad (4)$$

To show that $O_T^* \leq (1+\varepsilon)O_T$, by (4) it is enough to bound $-\delta n + n(3d-3)\delta \leq \varepsilon \cdot O_T$. Rather we will show that $-\delta n + n(3d-3)\delta \leq \varepsilon n$, which will automatically imply the required bound. This can be done by setting $\delta = h(\varepsilon, d) = \frac{\varepsilon}{3d-4}$, yielding a PTAS for TSP$(1,2)$. Since TSP$(1,2)$ does not admit a PTAS [21], the last assertion also rules out the existence of a PTAS for MDBCS$_d$ for any $d \geq 3$, unless P = NP. □

We are now ready to state the main result of this section. The proof consists in using the innaproximability constant given by Theorem 2.1 and applying the error amplification technique to rule out the existence of a constant-factor approximation. The whole proof of Theorem 2.2 has been moved to Appendix C due to lack of space.

**Theorem 2.2** MDBCS$_d$, $d \geq 2$, *does not admit any constant-factor approximation, unless* P = NP.

# 3 Approximating MDBCS$_d$

In this section we focus on approximating MDBCS$_d$. As seen in Section 2, MDBCS$_d$ does not admit any constant-factor approximation in general graphs. In Appendix D we show that when the input graph has a low-degree spanning tree (in terms of $d$), the problem becomes easy to approximate. Specifically, Proposition D.1 provides a constant-factor approximation for such graphs.

In this section we deal with general graphs. Concerning the Longest Path problem (which corresponds to the case $d = 2$ of MDBCS$_d$ as discussed in the introduction) the best approximation algorithm [6] has approximation ratio $\mathcal{O}(n(\log\log n/\log n)^2)$, which improved the ratio $\mathcal{O}(n/\log n)$ of [2]. Using the results of [2], we provide in Theorem 3.2 an approximation algorithm for MDBCS$_d$ in general unweighted graphs for any $d \geq 2$. Then we turn to weighted graphs, providing a completely new approximation algorithm for general weighted graphs in Theorem 3.3. Finally we compare both algorithms for unweighted graphs. To the best of our knowledge, these are the first approximation algorithms for MDBCS$_d$ in general graphs.

We need a preliminary lemma, that uses the following result :

**Proposition 3.1 ( [20])** *Any unordered tree on $n$ nodes can be represented using $2n + o(n)$ bits with adjacency being supported in $O(n)$ time.*

Let $\mathcal{T}_{n,d}$ be the set of non-isomorphic unlabeled trees on $n$ nodes with maximum degree at most $d$.

**Lemma 3.1** *The set $\mathcal{T}_{\log n,d}$ can be generated in polynomial time on $n$.*

***Proof:*** It is well known [23] that $|\mathcal{T}_{n,n-1}| \sim C\alpha^n n^{-5/2}$ as $n \to \infty$, where $C$ and $\alpha$ are positive constants. Hence, the set $\mathcal{T}_{\log n,\log n-1}$ has a number of elements polynomial on $n$. In addition, one can efficiently generate all the elements of $\mathcal{T}_{\log n,\log n-1}$, since by Proposition 3.1 any unlabeled tree on $\log n$ nodes can be represented using $2\log n + o(\log n)$ bits with adjacency being supported in $O(\log n)$ time. Finally, the set $\mathcal{T}_{\log n,d}$ is obtained from $\mathcal{T}_{\log n,\log n-1}$ by removing all the elements $T$ with $\Delta(T) > d$, where $\Delta(T)$ is the maximum degree of the tree $T$. □

The main ingredient of the first algorithm is a powerful result of [2], which uses the *color-coding* method.

**Theorem 3.1 ( [2])** *If a graph $G = (V, E)$ contains a subgraph isomorphic to a graph $H = (V_H, E_H)$ whose treewidth is at most $t$, then such a subgraph can be found in $2^{\mathcal{O}(|V_H|)} \cdot |V|^{t+1} \cdot \log |V|$ time.*

In particular, trees on $\log |V|$ vertices can be found in time $|V|^{\mathcal{O}(1)} \cdot \log |V|$. We are ready to describe our algorithm for unweighted graphs.

Algorithm $\mathcal{A}$ :

**(1)** Generate all the elements of $\mathcal{T}_{\log n,d}$. Define the set $\mathcal{F} := \{\}$.

**(2)** For each $T \in \mathcal{T}_{\log n,d}$, test if $G$ contains a subgraph isomorphic to $T$. If such a subgraph is found, add it to $\mathcal{F}$.

**(3)** If $\mathcal{F} = \emptyset$ OR $d > \log n$, output an arbitrary connected subgraph of $G$ with $d$ edges. Otherwise, output any element in $\mathcal{F}$.

**Theorem 3.2** *For all $d \geq 2$, algorithm $\mathcal{A}$ provides a $\rho$-approximation algorithm for* MDBCS$_d$ *in unweighted graphs, with $\rho = \frac{\min\{m, nd/2\}}{\log n}$.*

***Proof:*** Let us first see that the running time of algorithm $\mathcal{A}$ is polynomial on $n$. Indeed, steps **(1)** and **(2)** can be executed in polynomial time by Lemma 3.1 and Theorem 3.1, respectively. Step **(3)** takes constant time. Algorithm $\mathcal{A}$ is clearly correct, since by definition of the set $\mathcal{T}_{\log n,d}$ the output graph is a solution of MDBCS$_d$ in $G$.

Finally, let us consider the approximation ratio of algorithm $\mathcal{A}$. Let $OPT$ be the number of edges of an optimal solution of MDBCS$_d$ in $G$, and let $ALG$ be the number of edges of the solution found by algorithm $\mathcal{A}$. We distinguish two cases :

- If $OPT \geq \frac{d \cdot \log n}{2}$, then any optimal solution $\hat{H}$ has at least $\log n$ vertices. In particular, $\hat{H}$ contains a tree on $\log n$ vertices, and so does $G$. Hence, this tree will be found in step **(2)**, and therefore $ALG \geq \log n - 1$. (We can assume that $ALG = \log n$ by replacing everywhere $\mathcal{T}_{\log n,d}$ with $\mathcal{T}_{\log n+1,d}$.) On the other hand, we know that $OPT \leq \min\{m, nd/2\}$.
- Otherwise, if $OPT < \frac{d \cdot \log n}{2}$, then $ALG \geq d$. Note that such a connected subgraph with $d$ edges can be greedily found starting from any node of $G$.

In both cases, $\frac{OPT}{ALG} \leq \max\left\{\frac{\min\left\{m,\frac{nd}{2}\right\}}{\log n}, \frac{\log n}{2}\right\} = \frac{\min\{m,nd/2\}}{\log n}$ (since $\log n = \mathcal{O}(\sqrt{n})$), as claimed. □

In particular, if $d = 2$, algorithm $\mathcal{A}$ reduces to the LONGEST PATH algorithm of [2].

**Theorem 3.3** *The* MDBCS$_d$ *problem admits a* $\rho$*-approximation algorithm in weighted graphs, with* $\rho = \min\{n/2, m/d\}$.

***Proof:*** Let us describe the algorithm. Let $F$ be the set of $d$ heaviest edges in the input graph $G$, and let $W$ be the set of endpoints of those edges. We distinguish two cases according to the connectivity of the subgraph $H = (W, F)$. Let $\omega(F)$ denote the total weight of the edges in $F$.

If $H$ is connected, the algorithm returns $H$. We claim that this yields a $\rho$-approximation. Indeed, if an optimal solution consists of $m^*$ edges of total weight $\omega^*$, then $ALG = \omega(F) \geq \frac{\omega^*}{m^*} \cdot d$, since by the choice of $F$ the average weight of the edges in $F$ can not be smaller than the average weight of the edges of an optimal solution. As $m^* \leq m$ and $m^* \leq dn/2$, we get that $ALG \geq \frac{\omega^*}{m} \cdot d$ and $ALG \geq \frac{\omega^*}{dn/2} \cdot d = \frac{\omega^*}{n/2}$.

Now suppose $H = (W, F)$ consists of a collection $\mathcal{F}$ of $k$ connected components. Then we *glue* these components together in $k-1$ phases. In each phase, we pick two components $C, C' \in \mathcal{F}$, and combine them into a new connected component $\hat{C}$ by adding a connecting path, without touching any other connected component of $\mathcal{F}$. We then set $\mathcal{F} \leftarrow \mathcal{F} \setminus \{C, C'\} \cup \{\hat{C}\}$.

Each phase operates as follows. For every two components $C, C' \in \mathcal{F}$, compute their distance, defined as $d(C, C') = \min\{dist(u, u', G) \mid u \in C, u' \in C'\}$. Take a pair $C, C' \in \mathcal{F}$ attaining the smallest distance $d(C, C')$. Let $u \in C$ and $u' \in C'$ be two vertices realizing this distance, i.e. such that $dist(u, u', G) = d(C, C')$. Let $p(u, u')$ be a shortest path between $u$ and $u'$ in $G$. Let $\hat{C}$ be the connected component obtained by merging $C$, $C'$ and the path $p(u, u')$.

For the correctness proof, we need the following two observations :

First, observe that in every phase, the path $p(u, u')$ used to merge the components $C$ and $C'$ does not go through any other cluster $C''$, since otherwise, $d(C, C'')$ would be strictly smaller than $d(C, C')$, contradicting the choice of the pair $(C, C')$. Moreover, $p(u, u')$ does not go through any other vertex $v$ in the cluster $C$ except for its endpoint $u$, since otherwise, $dist(v, u', G) < dist(u, u', G)$, contradicting the choice of the pair $u, u'$. Similarly, $p(u, u')$ does not go through any other vertex $v'$ in $C'$.

We now claim that after $i$ phases, the maximum degree of $H$ satisfies $\Delta_H \leq d - k + i + 1$.

This is proved by induction on $i$. For $i = 0$, i.e. for the initial graph $H = (W, F)$, we observe that as $F$ consists of $d$ edges arranged in $k$ separate components, the largest component will have no more than $d - k + 1$ edges, hence $\Delta_H \leq d - k + 1$, as required. Now suppose the claim holds after $i - 1$ phases, and consider phase $i$. All nodes other than those of the path $p(u, u')$ maintain their degree from the previous phase. The nodes $u$ and $u'$ increase their degree by 1, so by the inductive hypothesis, their new degree is at most

$(d-k+(i-1)+1)+1 = d-k+i+1$, as required. Finally, the intermediate nodes of $p(u,u')$ have degree $2 \leq d-k+i+1$ (since $i \geq 1$ and $k \leq d$).

It follows that by the end of phase $k-1$, $\Delta_H \leq d-k+k-1+1 = d$. Also, at that point $H$ is connected. Hence $H$ is a valid solution.

Finally, the approximation ratio of the algorithm is still at most $\rho = \min\{n/2, m/d\}$, since this ratio was guaranteed for the originally selected $F$, and the final subgraph contains the set $F$. □

Let us now compare the algorithm of Theorem 3.2 (algorithm $\mathcal{A}$) and the algorithm of Theorem 3.3 (namely, algorithm $\mathcal{B}$) for unweighted graphs. Comparing both approximation ratios, we conclude that algorithm $\mathcal{A}$ performs better when $d < 2\log n$, while algorithm $\mathcal{B}$ is better when $d \geq 2\log n$. Running both algorithms and selecting the best solution we get the following

**Corollary 3.1** *The* MDBCS$_d$ *problem admits a* $\rho$*-approximation algorithm in unweighted graphs, with* $\rho = \min\{n/2,\ nd/(2\log n),\ m/d,\ m/\log n\}$.

# 4 Hardness of Approximating MSMD$_d$

The main theorem of this section, Theorem 4.2, shows that MSMD$_d$ does not admit a constant-factor approximation on general graphs, for $d \geq 3$. We first prove in Section 4.1 that MSMD$_d$ does not admit a PTAS and then, using the error amplification technique, we prove the main result. Our reduction is obtained from the well known Vertex Cover (VC) problem (see Appendix A.1).

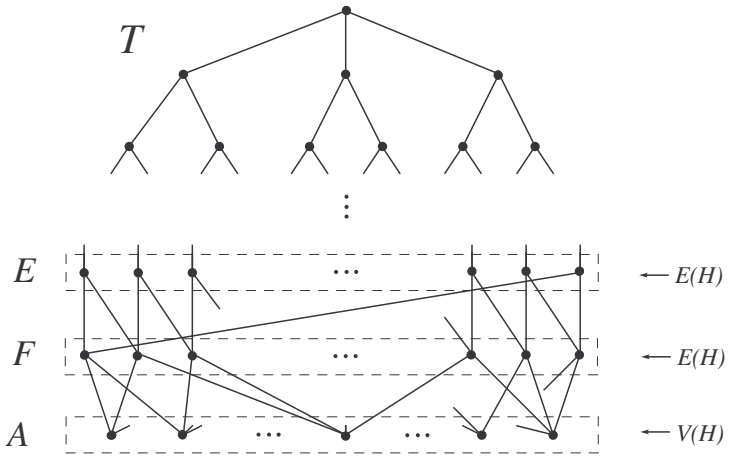
## 4.1 MSMD$_d$ does not admit a PTAS for any $d \geq 3$

We prove Theorem 4.1 for $d = 3$, moving the proof for $d \geq 4$ to Appendix E due to lack of space.

**Theorem 4.1** MSMD$_d$, $d \geq 3$, *is not in* PTAS, *unless* P = NP.

***Proof:*** We give a gap-preserving reduction from Vertex Cover. Let $H$ be an instance of Vertex Cover on $n$ vertices. We construct an instance $G = f(H)$ of MSMD$_3$. Without loss of generality, we can suppose that $H$ contains $3 \cdot 2^m$ edges for some integer $m$, and also that every vertex of $H$ has degree at least three.

Let $T$ be the complete ternary rooted tree with root $r$ and height $m+1$. The number of leaves of $T$ is $3 \cdot 2^m$, and $T$ contains $3 \cdot 2^{m+1} - 2$ vertices. Let us identify the leaves of $T$ with edges of $H$, and call this set $E$ (note that $E \subseteq V(T)$). We add another copy of $E$, called $F$, and a Hamiltonian cycle on $E \cup F$ inducing a bipartite graph with partition classes $E$ and $F$ as shown in Figure **??**. Let us also identify the vertices of $F$ with edges in $H$. Now we add $n$ new vertices $A$ identified with vertices of $H$, and join them to the leaves of $T$ according to the adjacency relations between the edges and vertices in $H$, i.e. an element $\ell \in T$ is




connected to $v \in A$ if the edge corresponding to $\ell$ in $H$ is adjacent to the vertex $v$ of $V(H)$. The graph $G$ built in this way is depicted in Figure **??**.

We claim that minimum subgraphs of $G$ of minimum degree at least three correspond to minimum vertex covers of $H$ and vice versa. To see this, first note that if such a subgraph $U$ of $G$ contains a vertex of $T \cup F$, then it should contain all the vertices of $T \cup F$, because of the degree constraints. Obviously $U$ cannot consist just of vertices of $A$, hence $U$ must contain all the vertices of $T \cup F$. Note that all the vertices of $F$ have degree two in $G[T \cup F]$. Therefore, the problem reduces to finding the smallest subset of vertices in $A$ covering all the vertices in $F$. This is exactly the VERTEX COVER problem for $H$. Thus, we have that

$$OPT_{\mathrm{MSMD}_3}(G) = OPT_{\mathrm{VC}}(H) + |V(T)| + |V(F)| = OPT_{\mathrm{VC}}(H) + 9 \cdot 2^m - 2 \ .$$

Using this formula, it is straightforward to check that $f$ is a gap-preserving reduction [25]. To complete the proof, note that VERTEX COVER is APX-hard, even restricted to graphs $H$ of size linear in $OPT_{\mathrm{VC}}(H)$. The existence of a PTAS for $\mathrm{MSMD}_3$ provides a PTAS for VERTEX COVER, which is a contradiction (under assumption APX $\neq$ PTAS). □

## 4.2 $\mathrm{MSMD}_d$ is not in APX for any $d \geq 3$

We are now ready to prove the following theorem :

**Theorem 4.2** $\mathrm{MSMD}_d$, $d \geq 3$, *does not admit any constant-factor approximation, unless* $\mathrm{P} = \mathrm{NP}$.

***Proof:*** We give again the details for $d = 3$, and prove the result for the case $d \geq 4$ in Appendix F. The proof is by appropriately applying the standard error amplification

technique. Let $\mathcal{G}_1 = \{G\}$ be the family of graphs we constructed above (Figure **??**) from the instances $H$ of vertex cover, $G$ being a typical member of this family, and let $\alpha > 1$ be the factor of inapproximability of MSMD$_3$, that exists by Theorem 4.1.

We construct a sequence of families of graphs $\mathcal{G}_k$, such that MSMD$_3$ is hard to approximate within a factor $\theta(\alpha^k)$ in the family $\mathcal{G}_k$. This proves that MSMD$_3$ does not have any constant-factor approximation. In the following $G_k$ will denote a typical element of $\mathcal{G}_k$ constructed using the element $G$ of $\mathcal{G}_1$. We describe the construction of $G_2$, and obtain the result by repeating the same construction inductively to obtain $G_k$. For every vertex $v$ in $G$ (denoting its degree by $d_v$), we construct a graph $G_v$ as follows. First, take a copy of $G$, and choose $d_v$ other arbitrary vertices $x_1, \ldots, x_{d_v}$ of degree three in $T \subset G$. Then, we replace each of these vertices $x_i$ with a cycle of length four, and join three of the vertices of the cycle to the three neighbors of $x_i$, $i = 1, \ldots, d_v$. Let $G_v$ be the graph obtained in this way. Note that it contains exactly $d_v$ vertices of degree two in $G_v$.

Now we take a copy of $G$, and replace each vertex $v$ with $G_v$. Then, we join the $d_v$ edges incident to $v$ to the $d_v$ vertices of degree two in $G_v$. This completes the construction of the graph $G_2$, illustrated in Figure G of Appendix G.

We have that $|V(G_2)| = |V(G)|^2 + o(|V(G)|^2)$, because each vertex of $G$ is replaced with a copy of $G$ where we had replaced some of the vertices with a cycle of length four.

To find a solution of MSMD$_3$ in $G_2$, note that for any $v \in V(G)$, once a vertex in $G_v$ is chosen, we have to look for MSMD$_3$ in $G$, which is hard up to a constant factor $\alpha$. But approximating the number of $v$'s for which we should touch $G_v$ is also MSMD$_3$ in $G$, which is hard up to the same factor $\alpha$. This proves that approximating MSMD$_3$ in $G_2$ is hard up to a factor $\alpha^2$. The proof of the theorem is completed by repeating this procedure, applying the same construction to obtain $G_3$, and inductively $G_k$. □

# 5 Approximating MSMD$_d$

In this section, it is shown that for fixed $d$, MSMD$_d$ is in P for graphs whose treewidth is $\mathcal{O}(\log n)$. This is done by giving a polynomial time algorithm based on dynamic programming. We refer to Appendix A.2 for the definitions of tree-decomposition and treewidth.

This dynamic programming algorithm is then used in Section 5.2 to provide an $\mathcal{O}(n/\log n)$-approximation algorithm of MSMD$_d$ for all classes of graphs excluding a fixed graph as a minor. This algorithm relies on a partitioning result for minor-excluded class of graphs, proved by Demaine et al. in [8].

## 5.1 MSMD$_d$ is in P for Graphs with Small Treewidth

In order to prove our results we need the following lemma, which gives the time complexity of finding a smallest induced subgraph of degree at least $d$ in graphs of bounded treewidth. The proof is based on standard dynamic programming techniques, and can be found in Appendix H.

**Lemma 5.1** *Let $G$ be a graph on $n$ vertices with a tree-decomposition of width at most $t$, and let $d$ be a positive integer. Then in time $\mathcal{O}((d+1)^t(t+1)^{d^2}n)$ we can either find a smallest induced subgraph of minimum degree at least $d$ in $G$, or identify that no such subgraph exists.*

A graph $G$ is *$q$-degenerated* if every induced subgraph of $G$ has a vertex of degree at most $q$. It is well known that there is a constant $c$ such that for every $h$, every graph with no $K_h$ minor is $ch\sqrt{\log h}$-degenerated [9]. This implies that $M$-minor-free graphs with $|M| = h$ are $ch\sqrt{\log h}$-degenerated and hence the largest value of $d$ for which $\mathrm{MSMD}_d$ is non-empty is $ch\sqrt{\log h}$, a constant. The above discussion, combined with the time complexity analysis mentioned in Lemma 5.1, imply the following

**Corollary 5.1** *Let $G$ be an $n$-vertex graph excluding a fixed graph $M$ as minor, with a tree-decomposition of width $\mathcal{O}(\log n)$, and let $d$ be a positive integer (a constant). Then in polynomial time one can either find a smallest induced subgraph of minimum degree at least $d$ in $G$, or conclude that no such subgraph exists.*

## 5.2 Approximation Algorithm for $M$-minor-Free Graphs

The following result of Demaine et al. [8] provides a way for partitioning the vertices of a graph excluding a fixed graph as a minor into subsets with small treewidth.

**Theorem 5.1 ( [8])** *For a fixed graph $M$, there is a constant $c_M$ such that for any integer $k \geq 1$ and for every $M$-minor-free graph $G$, the vertices of $G$ (or the edges of $G$) can be partitioned into $k+1$ sets such that any $k$ of the sets induce a graph of treewidth at most $c_M k$. Furthermore, such a partition can be found in polynomial time.*

One may assume without loss of generality that the minimum degree of the minor-free input graph $G = (V, E)$ is at least $d$ (by removing all the vertices of lower degree), and also that $|V(G)| = n = 2^p$ for some integer $p \geq 0$ (otherwise, replace $\log n$ with $\lceil \log n \rceil$ in the description of the algorithm).
Description of the algorithm :

- **(1)** Relying on Theorem 5.1, partition $V(G)$ in polynomial time into $\log n + 1$ sets $V_0, \ldots, V_{\log n}$ such that any $\log n$ of the sets induce a graph of treewidth at most $c_M \log n$, where $c_M$ is a constant depending only on the excluded graph $M$.
- **(2)** Run the dynamic programming algorithm of Section 5.1 on all the subgraphs $G_i = G[V \setminus V_i]$ of $\log n$ sets, $i = 0, \ldots, \log n$.
- **(3)** This procedure finds all the solutions of size at most $\log n$. If no solution is found, output the whole graph $G$.

This algorithm clearly provides an $\mathcal{O}(n/\log n)$-approximation for $\mathrm{MSMD}_d$ in minor-free graphs, for all $d \geq 3$. The running time of the algorithm is polynomial in $n$, since in step **(2)**, for each $G_i$, the dynamic programming algorithm runs in $\mathcal{O}((d+1)^{t_i}(t_i+1)^{d^2}n)$ time, where $t_i$ is the treewidth of $G_i$, which is at most $c_M \log n$.

# 6 Approximating DDD$k$S

We provide a deterministic approximation algorithm for the DDD$k$S problem in Theorem 6.1 (strongly based on the algorithm for D$k$S of [11]), and a randomized approximation algorithm in Appendix I. Even if the performance of the randomized algorithm is worse, we include it because the idea behind the algorithm is quite simple.

**Theorem 6.1** *The* DDD$k$S *problem admits a deterministic $\mathcal{O}(n^\delta)$-approximation algorithm, for some universal constant $\delta < 1/3$.*

***Proof:*** Given an input graph $G$, let $\rho_k^{OPT}$ be the optimal average degree of a subgraph of $G$ on exactly $k$ vertices (i.e. the optimum of D$k$S), and let $\delta_k^{OPT}$ be the optimal minimum degree of a subgraph of $G$ with at most $k$ vertices (i.e. the optimum of DDD$k$S). Let $C$ be the approximation ratio of the algorithm for D$k$S of [11], i.e. $C = O(n^\delta)$ for some universal constant $\delta < 1/3$. Given a graph $H$, let $\rho(H)$ denote the average degree of $H$, and let $\delta(H)$ denote the minimum degree of $H$.

We know, by [11], that we can find a subgraph $H_k$ of $G$ on $k$ vertices such that $\rho(H_k) \geq \rho_k^{OPT}/C$. Removing recursively the vertices of $H_k$ with degree strictly smaller that $\rho(H_k)/2$ we obtain a subgraph $H'_k$ of $H_k$ on at most $k$ vertices such that $\delta(H'_k) \geq \rho(H_k)/2 \geq \rho_k^{OPT}/(2C)$.

The next step consists in proving that there exists an integer $k_0$, $1 \leq k_0 \leq k$, such that $\rho_{k_0}^{OPT} \geq \delta_k^{OPT}$, so we can run the D$k$S algorithm for each $k' \leq k$, remove low-degree vertices each time, and take the best solution of DDD$k$S among $H'_2, H'_3, \ldots, H'_{k-1}, H'_k$.

Finally, let us prove that $k_0$ exists. Let $H$ be the optimal solution of DDD$k$S, $\delta(H) = \delta_k^{OPT}$. Let $k_0 = |V(H)|$ ($k_0 \leq k$). This is the $k_0$ we are looking for, because $\rho_{k_0}^{OPT} \geq \rho(H) \geq \delta(H) = \delta_k^{OPT}$.

The above procedure clearly constitutes a $(2C)$-approximation for DDD$k$S. □

# 7 Conclusions

This paper considered three Degree-Constrained Subgraph problems and studied their behavior in terms of approximation algorithms and hardness of approximation. Our main results and several interesting questions that remain open are discussed below.

We proved that the MDBCS$_d$ problem is not in Apx for any $d \geq 2$, and we provided a deterministic approximation algorithm with ratio $\min\{m/\log n,\ nd/(2\log n)\}$ (resp. $\min\{n/2,\ m/d\}$) for general unweighted (resp. weighted) graphs. Finally, we gave a constant-factor approximation when the input graph accepts a low-degree spanning tree. Closing the huge gap between the hardness bound and the approximation ratio of our algorithm looks like a promising research direction. It was proved in [16] that if any polynomial time algorithm can approximate the Longest Path problem to a ratio of $2^{\mathcal{O}(\log^{1-\varepsilon} n)}$, for any $\varepsilon > 0$, then NP has a quasi-polynomial deterministic time simulation. Nevertheless, this result does

not apply directly to the MDBCS$_d$ problem for *all* $d \geq 2$, so a different strategy should be devised.

We proved that the MSMD$_d$ problem is not in APX for any $d \geq 3$. It would be interesting to strengthen this hardness result using the power of the PCP theorem. On the positive side, we gave an $\mathcal{O}(n/\log n)$-approximation algorithm for the class of graphs excluding a fixed graph $H$ as a minor. Finally, finding an approximation algorithm for MSMD$_d$ in general graphs seems to be a challenging open problem. It seems that MSMD$_d$ remains hard even for proper minor-closed classes of graphs.

We provided a $\mathcal{O}(n^\delta)$-approximation algorithm for the DDD$k$S problem, for some universal constant $\delta < 1/3$. It would be interesting to provide hardness results complementing this approximation algorithm. Another avenue for further research could be to consider a mixed version between DDD$k$S and MSMD$_d$, that would result in a two-criteria optimization problem. Namely, given a graph $G$, the goal would be to maximize the minimum degree while minimizing the size of the subgraph, both parameters being subject to a lower and an upper bound, respectively.

## Références

[1] L. Addario-Berry, K. Dalal, and B. Reed. Degree constrained subgraphs. *Discrete Appl. Math.*, 156(7) :1168–1174, 2008.

[2] N. Alon, R. Yuster, and U. Zwick. Color-coding : a new method for finding simple paths, cycles and other small subgraphs within large graphs. In *Proceedings of the 26th annual ACM symposium on Theory of Computing*, pages 326–335, New York, USA, 1994.

[3] O. Amini, S. Pérennes, and I. Sau. Hardness and Approximation of Traffic Grooming. In *Proceedings of the 18th International Symposium on Algorithms and Computation*, volume 4835 of *LNCS*, pages 561–573, volume 4835 of LNCS series, 2007.

[4] O. Amini, I. Sau, and S. Saurabh. Parameterized Complexity of the Smallest Degree-Constrained Subgraph Problem. In *Proceedings of the International Workshop on Parameterized and Exact Computation*, volume 5008 of *LNCS*, pages 13–29, volume 5008 of LNCS series, 2008.

[5] R. P. Anstee. Minimum vertex weighted deficiency of $(g, f)$-factors : a greedy algorithm. *Discrete Appl. Math.*, 44(1-3) :247–260, 1993.

[6] A. Björklund and T. Husfeldt. Finding a Path of Superlogarithmic Length. *SIAM J. Comput.*, 32(6) :1395–1402, 2003.

[7] W. Cook, W. Cunningham, W. Pulleyblank, and A. Schrijver. *Combinatorial Optimization*. John Wiley and Sons, New York, 1998.

[8] E. Demaine, M. Hajiaghayi, and K. C. Kawarabayashi. Algorithmic Graph Minor Theory : Decomposition, Approximation and Coloring. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 637–646, October 2005.

[9] R. Diestel. *Graph Theory*. Springer-Verlag, 2005.

[10] P. Erdős, R. Faudree, C. C. Rousseau, and R. H. Schelp. Subgraphs of minimal degree $k$. *Discrete Math.*, 85(1) :53–58, 1990.

[11] U. Feige, D. Peleg, and G. Kortsarz. The Dense $k$-Subgraph Problem. *Algorithmica*, 29(3) :410–421, 2001.

[12] M. Fürer and B. Raghavachari. Approximating the minimum-degree spanning tree to within one from the optimal degree. In *Proceedings of the 3rd annual ACM-SIAM Symposium on Discrete Algorithms*, pages 317–324, USA, 1992.

[13] M. Fürer and B. Raghavachari. Approximating the minimum-degree steiner tree to within one of optimal. In *SODA : selected papers from the third annual ACM-SIAM symposium on Discrete algorithms*, pages 409–423, USA, 1994.

[14] H. Gabow. An efficient reduction technique for degree-constrained subgraph and bidirected network flow problems. In *Proceedings of the 15th annual ACM symposium on Theory of Computing*, pages 448–456, USA, 1983. ACM Press.

[15] M. Garey and D. Johnson. *Computers and Intractability*. W.H. Freeman, San Francisco, 1979.

[16] D. Karger, R. Motwani, and G. Ramkumar. On approximating the longest path in a graph. *Algorithmica*, 18(1) :82–98, 1997.

[17] S. Khot. Ruling out PTAS for graph min-bisection, densest subgraph and bipartite clique. In *45th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 136–145, 2004.

[18] L. Lovász and M. Plummer. *Matching Theory*. Annals of Discrete Math. 29, North-Holland, 1986.

[19] C. Lund and M. Yannakakis. The Approximation of Maximum Subgraph Problems. *Automata, Languages and Programming : 20th International Colloquium, ICALP 93, Lund, Sweden, July 5-9, 1993 : Proceedings*, 1993.

[20] J. Munro and V. Raman. Succinct Representation of Balanced Parentheses and Static Trees. *SIAM Journal on Computing*, 31(3) :762–776, 2001.

[21] C. Papadimitriou and M. Yannakakis. The traveling salesman problem with distances one and two. *Mathematics of Operations Research*, 18(1) :1–11, 1993.

[22] R. Ravi, M. Marathe, S. Ravi, D. Rosenkrantz, and H. H. III. Approximation algorithms for degree-constrained minimum-cost network-design problems. *Algorithmica*, 31(1) :58–78, 2001.

[23] O. Richard. The Number of Trees. *Annals of Mathematics*, Second Series 49(3) :583–599, 1948.

[24] Y. Shiloach. Another look at the degree constrained subgraph problem. *Inf. Process. Lett.*, 12(2) :89–92, 1981.

[25] V. Vazirani. *Approximation Algorithms*. Springer-Verlag, 2003.

[26] S. Win. On a Connection Between the Existence of $k$-Trees and the Toughness of a Graph. *Graphs and Combinatorics*, 5(1) :201–205, 1989.

# A Basic Definitions

## A.1 Approximation Algorithms and Gap-preserving Reductions

Given an NP-hard minimization (resp. maximization) problem $\Pi$ and a polynomial time algorithm $\mathcal{A}$, let $OPT_\Pi(I)$ be the optimal value of the problem $\Pi$ for the instance $I$, and let $ALG(I)$ be the value given by algorithm $\mathcal{A}$ for the instance $I$. We say that $\mathcal{A}$ is an $\alpha$-approximation algorithm for $\Pi$ if for any instance $I$ of $\Pi$, $OPT_\Pi(I)/ALG(I) \geq \alpha$ (resp. $OPT_\Pi(I)/ALG(I) \leq \alpha$).

The class Apx consists of all NP-hard optimization problems that can be approximated within a constant factor. The subclass PTAS (Polynomial Time Approximation Scheme) contains the problems that can be approximated in polynomial time within a ratio $1+\varepsilon$ for *any* constant $\varepsilon > 0$. Assuming P $\neq$ NP, there is a strict inclusion of PTAS in Apx (for instance, Vertex Cover is in Apx $\setminus$ PTAS), hence an Apx-hardness result for a problem implies the non-existence of a PTAS.

For our inapproximability results, we make use of the following reductions (cf. [25]).

**Definition A.1 (Gap-preserving reduction)** *For two minimization problems $\Pi_1$ and $\Pi_2$, a* gap-preserving reduction *from $\Pi_1$ to $\Pi_2$, parameterized by ($f_1$,$\alpha$) and ($f_2$,$\beta$), is a procedure that given an instance $x$ of $\Pi_1$, computes in polynomial time an instance $y$ of $\Pi_2$ such that :*

- *if $OPT(x) \leq f_1(x)$, then $OPT(y) \leq f_2(x)$.*
- *if $OPT(x) > \alpha(|x|)f_1(x)$, then $OPT(y) > \beta(|x|)f_2(x)$.*

The usefulness of gap-preserving reductions stems from the following known fact :

**Lemma A.1** *If there is a gap-preserving reduction from $\Pi_1$ to $\Pi_2$ and it is* NP*-hard to approximate $\Pi_1$ within a factor strictly less than $\alpha$, then it is also* NP*-hard to approximate $\Pi_2$ within a factor strictly less than $\beta$.*

Finally, let us recall the definition of the vertex cover problem, from which we obtain the hardness reduction of Section 4.

Vertex Cover (VC)
**Input** : An undirected graph $G = (V, E)$.
**Output** : A subset $V' \subseteq V$ of the minimum size such that for every edge $e = (u, v)$, either $u \in V'$ or $v \in V'$.

## A.2 Tree-decomposition and Treewidth

**Definition A.2 (Tree-decomposition, treewidth)** *A* tree-decomposition *of a graph $G = (V, E)$ is a pair $(T, \mathcal{X})$, where $T = (I, F)$ is a tree, and $\mathcal{X} = \{X_i\}$, $i \in I$ is a family of subsets of $V(G)$, called* bags *and indexed by the nodes of $T$, such that*

1. *each vertex $v \in V$ appears in at least one bag, i.e. $\bigcup_{i \in I} X_i = V$ ;*
2. *for each $v \in V$ the set of nodes indexed by $\{i \mid i \in I,\ v \in X_i\}$ forms a subtree of $T$ ;*
3. *For each edge $e = (x, y) \in E$, there is an $i \in I$ such that $x, y \in X_i$.*

*The width of a tree-decomposition is defined as* $\max_{i\in I}\{|X_i|-1\}$. *The treewidth of* $G$, *denoted by* $tw(G)$, *is the minimum width of a tree-decomposition of* $G$.

# B Proof of Claim 1

We prove the claim in a series of observations improving the solution, and apply them in order of their appearance. For a given edge set $F$, let $X(F)$ be the set of vertices containing the end-points of the edges in $F$.

(a) Suppose $E_1 \cap E' = \emptyset$. Then $H = (X(E_1), E_1)$ is connected and every vertex $v \in X(E_1)$ has degree at most $d$ in $H$. This implies that $H$ contains a cycle, so removing any edge from this cycle will not break connectivity. So we can remove any edge $(u, v)$ from this cycle and add the edges $(u_1, u)$ and $(v_1, v)$, obtaining a solution of larger weight. Therefore, we assume henceforth that $E_1 \cap E' \neq \emptyset$.

(b) Suppose $V \setminus X(E_1) \neq \emptyset$, that is there is a vertex $v \in V$ which is not contained in $X(E_1)$. In this case, by Observation (a), there exists a vertex $u \in X(E_1)$ such that one of the edges $(u_i, u)$, $1 \leq i \leq d-2$, is in $E_1$. We then set $E_1 \leftarrow E_1 - \{(u_i, u)\} + \{(u, v), (v, v_i) \mid 1 \leq i \leq d-2\}$. Clearly we maintain connectivity (as removing edges from $E'$ does not break connectivity) and the weight of solution increases by at least 1. We repeat this procedure until the current solution contains all the vertices of $G$.

(c) Suppose $H' = (V, E \cap E_1)$ is neither a spanning tree nor a hamiltonian cycle. Notice that $H'$ is connected, as removing degree 1 vertices of $V'$ does not break connectivity. This implies that there is a cycle $C$ in $H'$ such that there is a vertex $v$ on it whose degree is at least 3 in $H'$ (otherwise, $H'$ would be disconnected). This implies that there exists an edge $e = (v, v_i)$ such that $e \notin E_1$. Let $(u, v)$ be an edge on $C$. We then set $E_1 \leftarrow E_1 - \{(u, v)\} + \{(v, v_i)\}$. Clearly we maintain connectivity (as removing an edge from a cycle does not break connectivity) and the weight of the solution increases by at least 1.

(d) Suppose $H' = (V, E \cap E_1)$ is a spanning tree. If $H'$ is a path then the end-points of this path, say $u$ and $v$, have degre 1 in $H'$, hence we can add the edge $(u, v)$ and obtain a solution of higher weight. So let us suppose that $H'$ is not a path, hence there exists a vertex $v$ of degree at least 3 in $H'$. This implies that there exists an edge $e = (v, v_i)$ such that $e \notin E_1$. Let $(u, v)$ be an edge incident to $v$ in the spanning tree $H'$. Consider the spanning forest $H' - \{(u, v)\}$, consisting of two sub-tress $H'_u$ and $H'_v$ containing $u$ and $v$ respectively. We select a leaf $w_1 \in H'_u$ and a leaf $w_2 \in H'_v$ ($w_2 \neq v$), and we set $E_1 \leftarrow E_1 - \{(u, v)\} + \{(v, v_i), (w_1, w_2)\}$. Clearly the resultant graph is connected and has higher weight.

We can apply the above rules in polynomial time to obatin a graph $G_3$ which is a solution of MDBCS$_d$ in $G'$ and satisfies the conditions described in the statement of the claim.

# C Proof of Theorem 2.2

We first state the following technical lemma.

**Lemma C.1** *For all $d \geq 2$, MDBCS$_d$ restricted to the class of graphs for which any optimal solution contains at least 2 vertices of degree at most $d-1$ is* NP*-complete.*

***Proof:*** We know that MDBCS$_d$ is NP-complete in general graphs for all $d \geq 2$ [15], even when all the weights of the edges are equal to 1. Let $G = (V, E)$ be a general input graph with all weights equal to 1. For each (unordered) pair of vertices $u, v \in V$, $u \neq v$, we construct a graph $G_{u,v}$ in the following way : $G_{u,v}$ is obtained from $G$ by adding two new vertices $u'$ and $v'$, plus the edges $\{u', u\}$ and $\{v', v\}$ with weight $W \geq 2|E(G)|$. It is clear that, for each pair $\{u, v\}$, any optimal solution for $G_{u,v}$ contains the edges $\{u', u\}$ and $\{v', v\}$, hence any optimal solution contains the 2 vertices $u'$ and $v'$ of degree one, $1 \leq d-1$. Let us see that if we could solve MDBCS$_d$ in $G_{u,v}$ in polynomial time for each pair $u, v$, then we could also find an optimal solution for $G$ in polynomial time, which would be a contradiction. Indeed, let $OPT_{u,v}$ be the weight of an optimal solution of MDBCS$_d$ in $G_{u,v}$. Let

$$OPT^G_{u,v} = \begin{cases} OPT_{u,v} - 2W + 1, & \text{if } \{u, v\} \in E(G) \text{ and it is not in the optimal solution in } G_{u,v}, \\ OPT_{u,v} - 2W, & \text{otherwise.} \end{cases}$$

Then the number of edges of an optimal solution of MDBCS$_d$ in $G$ is exactly $\max_{u,v} OPT^G_{u,v}$, that can be computed in polynomial time. The lemma follows. □

Again, we prove the result for $d \geq 3$, the result for $d = 2$ following from [16]. We will use the error amplification technique. Let $\alpha > 1$ be the inapproximability constant given by Theorem 2.1. Given a family of graphs $\mathcal{G}$ with a typical element $G = (V, E)$ with $|V(G)| = n$ and $|E(G)| = m$, such that MDBCS$_d$ is hard to approximate in this family within a factor $\alpha > 1$, we will build a sequence of families of graphs $\mathcal{G} = \mathcal{G}^1, \mathcal{G}^2, \ldots$, such that MDBCS$_d$ is hard to approximate in $\mathcal{G}^k$ within a factor $\alpha^k$. This implies that MDBCS$_d$ is not in APX. In the following $G^i$ will be a typical element of $\mathcal{G}^i$. Let us suppose that there exists an algorithm $\mathcal{C}$ for approximating the optimal value of MDBCS$_d$ on any graph within a constant factor of $\rho > 1$, and derive a contradiction.

Assume without loss of generality that all the weights of the edges of $G$ are equal to 1 (this can be assumed by replacing the edges of weight $W$ by a ternary tree of total size $W - 1$, for $d > 2$, and adding two edges of weight 1 to $u_i$ and $v_{i+1}$). Combinining Lemma C.1 and the proof of Theorem 2.1, it is easy to see that MDBCS$_d$ does not admit PTAS restricted to graphs for which any optimal solution contains at least two vertices of degree $d-1$. (Indeed, note that in the family of graphs contructed in Theorem 2.1, any optimal solution of MDBCS$_d$ contains $n(d-2)$ vertices of degree 1.) So can also assume without loss of generality that any optimal solution in $G$ (and inductively also in $G^k$) contains at least 2 vertices of degree $d-1$.

Let $OPT_k$ and $H_k$ be the weight of an optimal solution and an optimal connected subgraph in $G^k$, respectively. We proceed to illustrate in detail the construction of $G^2$. For each

pair of vertices $\{u, v\} \in V^2$, $u \neq v$, we build the graph $G^2_{u,v}$ in the following way : we take the graph $G$ and we replace each edge $e_i = (x, y) \in E(G)$, $i = 1, \ldots, m$, with a copy $G_i$ of $G$ (again, the copy of the vertex $u \in V(G)$ in $G_i$ is labeled $u_i$), and we add the edges $(x, u_i)$ and $(y, v_i)$ with weight $\varepsilon_2$, $0 < \varepsilon_2 << 1$ for $i = 1, \ldots, m$. We define $G^2$ as the graph $G^2_{u,v}$ for which algorithm $\mathcal{C}$ gives the best solution.

**Claim 2** $OPT_2 = OPT_1^2 + 2\varepsilon_2 \cdot OPT_1 \approx OPT_1^2$.

Since any optimal solution in $G$ contains at least 2 vertices with degree at most $d-1$, the best solution in $G^2$ contains $OPT_1$ copies of $H_1$, one for each edge of $H_1$, plus 2 edges with weight $\varepsilon_2$ for each copy of $H_1$.

**Claim 3** *Given any solution $S_2$ in $G^2$ with weight $x$, it is possible to find a solution $S_1$ in $G$ with weight at least $\sqrt{x}$.*

To prove the claim, we distinguish two cases :

- **Case a :** $S_2$ intersects at least $\sqrt{x}$ copies of $G$.
  Let $S_1$ be the subgraph of $G$ induced by the edges corresponding to these copies of $G$ in $G^2$.
- **Case b :** $S_2$ intersects strictly fewer than $\sqrt{x}$ copies of $G$.
  Let $S_1$ be $S_2 \cap G_i$, with $G_i$ being the copy of $G$ in $G^2$ such that $|E(S_2 \cap G_i)|$ is maximized.

In both cases $S_1$ is connected, has maximum degree at most $d$, and has at least $\sqrt{x}$ edges.

$\rho$-approximation in $G^2$, then it is possible to find a solution for $G$ with weight at least $\sqrt{\frac{OPT_2}{\rho}} \geq \frac{OPT_1}{\sqrt{\rho}}$. That is, there exists a $\sqrt{\rho}$-approximation in $G$.

Inductively, to build $G^k$ we take a sequence of weights for the edges connecting the copies of $G^{k-1}$ such that

$$0 < \varepsilon_k << \varepsilon_{k-1} << \ldots << \varepsilon_3 << \varepsilon_2 << 1.$$

Claim 2 becomes $OPT_k \approx OPT_{k-1}^2 + 2\varepsilon_k \cdot OPT_{k-1} \geq OPT_{k-1}^2 \geq OPT_{k-2}^4 \geq \ldots \geq OPT_1^{2^k}$, and the same arguments apply. As the size of $G^k$ is a polynomial function on the size of $G$, this means that given a $\rho$-approximation algorithm for MDBCS$_d$ for $G^k$ in $\mathcal{G}^k$ with running time polynomial in the size of $G^k$, one can obtain a $\rho^{1/2^k}$-approximation algorithm for MDBCS$_d$ for $G$ in $\mathcal{G}$ and with running time polynomial in $|G|$. But there exists an integer $k$ such that $\rho^{2^{-k}} < \alpha$, contradicting Theorem 2.1. The theorem follows.

# D Approximating MDBCS$_d$ in Graphs with Low-degree Spanning Trees

We first state a simple lemma about the optimal solutions of the polynomial problem MDBS$_d$ (the definition is the same as the MDBCS$_d$ problem, except that the connectivity of the output subgraph is not required).

**Lemma D.1** *Given a graph $G$ and two integers $d, k$ , $1 < k \leq d$, such that $k$ divides $d$, let $OPT_d$ and $OPT_{d/k}$ be the optimal solutions of* MDBS$_d$ *and* MDBS$_{d/k}$ *in $G$, respectively. Then*

$$OPT_d \leq \frac{3k}{2} \cdot OPT_{d/k}$$

***Proof:*** Let $\hat{H}_d$ be the subgraph of $G$ attaining $OPT_d$. By the classical Vizing's theorem [9], there exists a coloring of the edges of $\hat{H}_d$ using at most $d+1$ colors. Then we order these chromatic classes according to non-increasing total edge-weight, and let $H_{d/k}$ be the subgraph of $G$ induced by the first $d/k$ classes. Then the maximum degree of $H_{d/k}$ does not exceed $d/k$, and the sum of the weights of its edges is at least $\frac{d \cdot OPT_d}{k \cdot (d+1)}$. Hence

$$OPT_d \leq \frac{d+1}{d} \cdot k \cdot OPT_{d/k}$$

For $d \geq 2$, the function $\frac{d+1}{d}$ is maximized when $d = 2$. □

For example, if $G = C_5$ and $d = k = 2$, then $OPT_2 = 5 \leq 3/2 \cdot 2 \cdot OPT_1 = 3 \cdot 2 = 6$.

**Definition D.1 ($k$-tree)** *A* $k$-tree *of a connected graph is a spanning tree with maximum degree at most $k$.*

We are now ready to describe our approximation algorithm.

**Proposition D.1** *Given two integers $d, \ell$ , $1 < \ell < d$, let $\mathcal{G}_{d,\ell}$ be the class of graphs that have a $(d/\ell - 1)$-tree. Then, for any $G \in \mathcal{G}_{d,\ell}$,* MDBCS$_d$ *can be approximated in $G$ within a constant factor $\frac{3}{2}\frac{\ell}{\ell-1}$.*

***Proof:*** Assume without loss of generality that $\ell$ divides $d$, otherwise replace $d/\ell$ with $\lceil d/\ell \rceil$. Since $G$ has a $(d/\ell - 1)$-tree, by [12] we can find in polynomial time a spanning tree $S$ of $G$ with maximum degree at most $d/\ell$. Let $k = \frac{\ell}{\ell-1}$, and let $H$ be the optimal solution of MDBS$_{d/k}$ in $G$ (recall that MDBS$_d$ is in P, but the output graph is not necessarily connected). Then it is clear that the graph $S \cup H$ is a solution of MDBCS$_d$ in $G$, since it is connected and has maximum degree at most $d$. By Lemma D.1 and using the fact that any solution for MDBCS$_d$ is also a solution for MDBS$_d$, $S \cup H$ provides a $\frac{3}{2}\frac{\ell}{\ell-1}$-approximation for MDBCS$_d$ in $G$. □

For example, if $G$ has a spanning tree of maximum degree at most $d/2 - 1$, then Proposition D.1 states that MDBCS$_d$ admits a 3-approximation in $G$.

## D.1 Relation of MDBCS$_d$ with the Toughness of a Graph

We need some preliminary definitions. Given a graph $G$, we note by $\kappa(G)$ the number of connected components of $G$.

**Definition D.2 (Toughness of a graph [26])** *A graph $G = (V, E)$ has* toughness $t(G)$ *if $t(G)$ is the largest number $t$ such that, for any subset $S \subseteq V$, $|S| \geq t \cdot \kappa(G[V \setminus S])$, provided that $\kappa(G[V \setminus S]) > 1$.*

Win proved in [26] that if $t(G) \geq \frac{1}{k-2}$, with $k \geq 3$, then $G$ has a $k$-tree :

**Theorem D.1 (Win [26])** *Let $G$ be a graph. If $t(G) \geq \frac{1}{k-2}$, with $k \geq 3$, then $G$ has a $k$-tree.*

Let us see the relation of the definitions above with the MDBCS$_d$ problem. If a graph $G$ does not satisfy the conditions of Proposition D.1, then $G$ does not have a $(d/2-1)$-tree. In this case one has some additional knowledge about the structure of $G$. Namely, Theorem D.1 states that, provided that $d \geq 8$, the *toughness* $t(G)$ of $G$ satisfies $t(G) < \frac{1}{d/2-3}$, and this means that there exists a subset $S \subseteq V(G)$ such that

$$\kappa(G[V \setminus S]) > |S| \cdot \left(\frac{d}{2} - 3\right).$$

It would be interesting to explore if this structural result permits to approximate MDBCS$_d$ in $G$ efficiently.

# E Proof of Theorem 4.1 for $d \geq 4$

Again, the proof consists in a *gap-preserving reduction* from Vertex Cover :

Let $H$ be an instance of Vertex Cover on $n$ vertices. We will construct an instance $G = f(H)$ of MSMD$_d$. Without loss of generality, we can suppose that $H$ contains $d \cdot (d-1)^m$ edges, for some $m$, and also that every vertex of $H$ has degree at least $d$.

Let $T$ be the complete $d$-ary rooted tree with root $r$ and height $m+1$. It is easy to see that the number of leaves of $T$ is $d \cdot (d-1)^m$, and that $T$ contains $1 + d \cdot \frac{(d-1)^{m+1}-1}{d-2}$ vertices. Let us identify the leaves of $T$ with edges of $H$, and call this set $E$ (note that $E \subseteq V(T)$). We add another copy of $E$, called $F$, and the following edges (suppose that $m$ is big enough) according to the parity of $d$ :

- if $d$ is even : $\frac{d-2}{2}$ Hamiltonian cycles on $E \cup F$, each one inducing a bipartite graph with partition classes $E$ and $F$, plus one perfect matching between $E$ and $F$.
- if $d$ is odd : $\frac{d-1}{2}$ Hamiltonian cycles on $E \cup F$, each one inducing a bipartite graph with partition classes $E$ and $F$.

Let us also identify the vertices of $F$ with edges in $H$. Now we add $n$ new vertices $A$ identified with vertices of $H$, and join them to the leaves of $T$ according to the adjacency relations between the edges and vertices in $H$, i.e. an element $\ell \in T$ is connected to $v \in A$ if the edge corresponding to $\ell$ in $H$ is adjacent to the vertex $v$ of $V(H)$. Note that the vertices of $E$ have regular degree $d$, and those of $F$ have regular degree $d+1$. Now the same argument of the proof of Theorem 4.1 applies to this case, proving the Apx-hardness of MSMD$_d$ for $d > 3$.

# F Proof of Theorem 4.2 for $d \geq 4$

Again, the proof consists in applying the error amplification technique. Let $G_1 := G$ be the graph we constructed in Appendix E, and let $\alpha > 1$ be the factor of inapproximability of MSMD$_d$, that exists by Theorem 4.1. We construct a sequence of graphs $G_k$, such that MSMD$_d$ is hard to approximate within a factor $\theta(\alpha^k)$ in $G_k$. This proves that MSMD$_d$ does not have any constant-factor approximation. Indeed, suppose that MSMD$_d$ admits a $C$-approximation for some constant $C > 0$. Then we can choose $k$ such that $\alpha^k > C$, and then MSMD$_d$ is hard to approximate in $G_k$ within a factor $\alpha^k > C$, a contradiction.

We describe the construction of $G_2$, and we obtain the result by repeating the same construction inductively to obtain $G_k$. For every vertex $v$ in $G$ (we note its degree by $d_v$), we construct a graph $G_v$ as follows : first we take a copy of $G$, and choose $d_v$ other arbitrary vertices $x_1, \ldots, x_{d_v}$ of degree $d$ in $T \subset G$. Then we replace each of these vertices $x_i$ with :

- if $d$ is odd : a graph on $d+1$ vertices with regular degree $d-1$.
- if $d$ is even : a graph on $d+2$ vertices having one vertex $v^*$ of degree $d+1$, and all the others degree $d-1$.

Then we join $d$ of the vertices of this new graph (different from $v^*$) to the $d$ neighbors of $x_i$, $i = 1, \ldots, d_v$. edges incident to $v$, to $d$ of these vertices. Let us call $v'$ the only remaining vertex of degree 2 on this cycle. Let $G_v$ be the graph obtained in this way. Note that we have exactly $d_v$ vertices of degree $d-1$ in $G_v$.

Now we take a copy of $G$, and replace each vertex $v$ by $G_v$. Then we join the $d_v$ edges incident to $v$ to the $d_v$ vertices of degree two in $G_v$. This completes the construction of the graph $G_2$.

We have that $|V(G_2)| = |V(G)|^2 + o(|V(G)|^2)$, because we replace each vertex of $G$ with a copy of $G$ where we had replaced some of the vertices with a graph of size $d+1$ or $d+2$. The same argument of the proof of Theorem 4.2 applies to this case, proving the APX-hardness of MSMD$_d$ for $d > 3$.

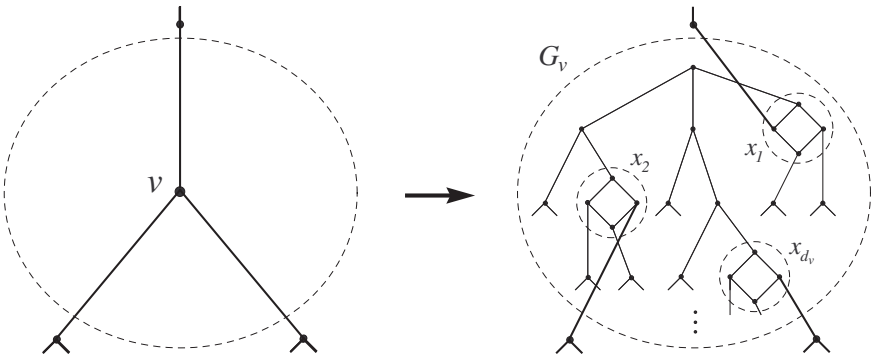
# G Error Amplification in the Proof of Theorem 4.2

See FIG. G.

# H Proof of Lemma 5.1

Let $(T, \mathcal{X})$ be the given tree-decomposition. We suppose that $T$ is a rooted tree, and that the decomposition is *nice*, which means :

- Any node has at most two children ;
- For any node $t$ with exactly two children $t_1$ and $t_2$, we have $X_t = X_{t_1} = X_{t_2}$ ;
- For any node $t$ with exactly one child $s$ we have $X_t \subset X_s$ and $|X_s| = |X_t| + 1$, or $X_s \subset X_t$ and $|X_t| = |X_s| + 1$.




Note that such a decomposition always exists and can be found in linear time, and in fact we can suppose that $|V(T)| = \mathcal{O}(n)$. As it is usual in algorithms based on tree decompositions, we propose a dynamic programming approach based on this decomposition, which at the end either produces a connected subgraph of $G$ of minimum degree at least $d$ and of size at most $k$, or decides that $G$ does not have any such subgraph.

Now that the tree decomposition is rooted, we can speak of the subgraph defined by the subtree rooted at node $i$. More precisely, for any node $i$ of $T$, let $Y_i$ be the set of all vertices that appear either in $X_i$ or in $X_j$ for some descendant $j$ of $i$. Denote by $G[Y_i]$ the graph induced by the nodes in $Y_i$.

Note that if $i$ is a node in the tree and $j_1$ and $j_2$ are two children, then $Y_{j_1}$ and $Y_{j_2}$ are disjoint except for vertices in $X_i$, i.e. $Y_{j_1} \cap Y_{j_2} = X_i$. A $\{0, 1, 2, 3, \ldots, d\}$-coloring of vertices in $X_i$ is a function $c_i : X_i \to \{0, 1, \ldots, d-1, d\}$. Let $supp(c) = \{v \in X_i |\ c(v) \neq 0\}$ be the *support* of $c$.

For any such $\{0, 1, \ldots, d\}$-coloring $c$ of vertices in $X_i$, let $a(i, c)$ be the minimum size of an induced subgraph $H(i, c)$ of $G[Y_i]$, which has degree $c(v)$ for every $v \in X_i$ with $c(v) \neq d$, and degree at least $d$ on its other vertices. Note that $H(i, c) \cap X_i = supp(c)$. If such a subgraph does not exist, we define $a(i, c) = +\infty$.

We develop recursive formulas for $a(i, c)$. In the base case, $i$ is a leaf of the tree decomposition. Hence $Y_i = X_i$. We would like to know the size of the minimum induced subgraph with prescribed degrees, but this is exactly $|supp(c)|$ if $G[supp(c)]$ satisfies the degree conditions, and is $+\infty$ if it does not.

In the recursive case, node $i$ has at least one child. We distinguish between three cases, depending on the size of the bag of $i$ and its number of children :

– Assume first that $i$ has only one child, say $j$, $X_i \subset X_j$, and so $|X_j| = |X_i| + 1$ and $X_i = X_j \setminus \{v\}$ for some vertex $v$. Also, $Y_i = Y_j$, since $X_i$ does not add any new vertices. Consider a coloring $c : X_i \to \{0, 1, \ldots, d\}$. Consider the two colorings $c_0 : X_j \to \{0, 1, \ldots, d\}$ and $c_1 : X_j \to \{0, 1, \ldots, d\}$ of $X_j$, defined as follows : $c_0 = c_1 = c$ on $X_i$, and $c_0(v) = 0$, $c_1(v) = d$. Then we let $a(i, c) = \min\{a(j, c_0), a(j, c_1)\}$.

- Now assume that $i$ has only one child, say again $j$, and that $|X_j| = |X_i| - 1$ and so $X_j \subset X_i$ and $X_j = X_i \setminus \{v\}$ for some vertex $v$. Also, $Y_j = Y_i \setminus \{v\}$. Let $c$ be a coloring of $X_i$. It is clear that the only neighbors of $v$ in $G[Y_i]$ are already in $X_i$.
  - If $c(v) \geq 1$, for any collection $\mathcal{A}$ of $c(v)$ edges in $G[X_i]$ connecting $v$ to vertices $v_1, \ldots, v_{c(v)}$, with $c(v_i) \geq 1$ (note that such a collection may not exist at all), we consider the coloring $c_{\mathcal{A}}$ of $X_j$ as follows : $c_{\mathcal{A}}(v_i) = c(v_i) - 1$ for any $1 \leq i \leq c(v)$, and $c_{\mathcal{A}}(w) = c(w)$ for any other vertex $w$. Then we define

$$a(i,c) = \min_{\mathcal{A}}\{a(j, c_{\mathcal{A}})\} + 1 \ .$$

  - If $c(v) = 0$, we simply define $a(i,c) = a(j,c)$.

  Note that we have at most $(t+1)^{d+1}$ choices for such a collection $\mathcal{A}$.
- In the last case, we can suppose that $i$ has two children $j_1$ and $j_2$, and so $X_i = X_{j_1} = X_{j_2}$. Let $c$ be a coloring of $X_i$, then $supp(c) \subset X_i$ is part of the subgraph we are looking for. For any vertex $v \in X_i$, we calculate the degree $d_{G[X_i]}(v)$. We suppose that $v$ has degree $d_1^v, d_2^v$ in $H \cap G[Y_{j_1}], H \cap G[Y_{j_2}]$ ($H$ is the subgraph we are looking for). These degree sequences should be in such a way to guarantee the degree condition on $v$ imposed by the coloring $c$. In other words, if $c(v) \leq d-1$ then we should have $d_1^v + d_2^v - d_{G[X_i]} = c(v)$, and if $c(v) = d$, then $d_1^v + d_2^v - d_{G[X_i]} \geq d$. Every such sequence $\mathcal{D} = \{d_1^v, d_2^v |\ v \in X_i\}$ on vertices of $X_i$ determines two colorings $c_1^{\mathcal{D}}$ and $c_2^{\mathcal{D}}$ of $X_{j_1}$ and $X_{j_2}$ respectively. For each such pair of colorings, let $H_1$ and $H_2$ be the minimum subgraphs with these degree constraints in $G[Y_{j_1}]$ and $G[Y_{j_2}]$ respectively. Then $H_1 \cup H_2$ satisfies the degree constraints imposed by $c$. We define

$$a(i,c) = \min_{\mathcal{D}}\{|H| \mid H = H_1 \cup H_2\}$$

  for all degree distributions as above. For every vertex we have at most $d^2$ possible degree choices for $d_1^v$ and $d_2^v$. We have also $|X_i| \leq t+1$. This implies that the minimum is taken over at most $(t+1)^{d^2}$ colorings.

As the size of our tree-decomposition is linear on $n$, we can determine all the values $a(i,c)$ for every $i \in V(T)$ and every coloring of $X_i$ in time linear in $n$. Now return the minimum value of $a(i,c)$ computed for all colorings $c$, for values in the set $\{0, d\}$ assigning at least one non-zero value. The time dependence on $t$ follows from the size of the bags and the choices we made using the colorings.

# I A Randomized Approximation Algorithm for DDD$k$S

**Theorem I.1** *The* DDD$k$S *problem admits a randomized $\mathcal{O}(\sqrt{n}\log n)$-approximation algorithm.*

***Proof:*** For every $1 \leq d \leq n$, let $H[d]$ be the maximum subgraph of $G$ with minimum degree $\delta_{H[d]} \geq d$, in the sense that $H[d]$ contains any other subgraph $H$ of $G$ of minimum degree at least $d$. Let also $n[d]$ be $|V(H[d])|$. The first stage of the algorithm computes $H[d]$ for every

$1 \leq d \leq n$. This is easily done by initializing $H[1] = G$ and then successively removing from $H[d]$ all the vertices of degree $d$ to obtain $H[d+1]$. Note that $n[d]$ can be zero, i.e. $H[d]$ can be the empty subgraph. The algorithm stops whenever it finds $n[d] = 0$.

Let $\tilde{d}$ be the index such that $n[\tilde{d}] > 0$ and $n[\tilde{d}+1] = 0$ (clearly $\tilde{d} \leq n-1$). If $k \geq n[\tilde{d}]$, then $H[\tilde{d}]$ is an exact solution to the problem, hence the output to the DDD$k$S problem is $\tilde{d}$. It remains to handle the case where $k < n[\tilde{d}]$. In this case, it is also clear that the solution $d^*$ we are looking for is bounded by $\tilde{d}$, i.e. $d^* < \tilde{d}$. Two cases may occur.

- **Case a** : $k \leq 16\sqrt{n}\log n$ OR $\tilde{d} \leq 16\sqrt{n}\log n$.
  In this case any connected subgraph of $G$ of size at most $k$ (for example a connected subtree of a spanning tree of $G$ of size $k$, or even just an edge) has minimum degree at least one, hence it provides a solution that is within a factor $1/(16\sqrt{n}\log n)$ of the optimal solution.
- **Case b** : Both $\tilde{d}, k > 16\sqrt{n}\log n$.
  Construct a subgraph $H$ of $H[\tilde{d}]$ in the following way : select each vertex of $H[\tilde{d}]$ with probability $1/\sqrt{n}$, and take $H$ to be the induced subgraph of $H[\tilde{d}]$ by the set of selected vertices. Let $n_0 = |V(H)|$.

**Claim 4** *The number of selected vertices satisfies $n_0 \leq 2n[\tilde{d}]/\sqrt{n}$ with probability at least $1 - 1/n^4$. In particular, $n_0 \leq k$ with probability at least $1 - 1/n^4$.*

***Proof:*** Observe that $n_0$ can be expressed as the sum of $n[\tilde{d}]$ independent Boolean random variables $B_1, \ldots, B_{n[\tilde{d}]}$. Since $\mathbb{E}[n_0] = n[\tilde{d}]/\sqrt{n}$, applying Chernoff's bound on the upper tail yields

$$\mathbb{P}\mathrm{rob}\left[B_1 + \cdots + B_{n[\tilde{d}]} > \frac{2n[\tilde{d}]}{\sqrt{n}}\right] \;<\; \exp\left(-\frac{n[\tilde{d}]}{4\sqrt{n}}\right).$$

Therefore, because $n[\tilde{d}] > k > 16\sqrt{n}\log n$, we have

$$\mathbb{P}\mathrm{rob}\left[n_0 > \frac{2n[\tilde{d}]}{\sqrt{n}}\right] \;<\; \exp(-4\log n) \;=\; \frac{1}{n^4}\,,$$

and since $n[\tilde{d}] \leq n$, with probability at least $1 - \frac{1}{n^4}$, $n_0 \leq 2n[\tilde{d}]/\sqrt{n} \leq 2\sqrt{n} < 16\sqrt{n}\log n < k$. □

**Claim 5** *For every vertex $v \in V(H)$, $deg_H(v) \geq \frac{\tilde{d}}{2\sqrt{n}}$ with probability at least $1 - 1/n^2$.*

***Proof:*** Observe first that $deg_H(v)$ is a sum of $deg_{H[\tilde{d}]}(v)$ independent Boolean random variables, and so the expected degree of $v$ in $H$ is $deg_{H[\tilde{d}]}(v)/\sqrt{n} \;\geq\; \tilde{d}/\sqrt{n}$. This is because every vertex of $H[\tilde{d}]$ has degree at least $\tilde{d}$. This implies

$$\mathbb{P}\mathrm{rob}\left[deg_H(v) < \frac{\tilde{d}}{2\sqrt{n}}\right] \;\leq\; \mathbb{P}\mathrm{rob}\left(deg_H(v) \;<\; \frac{deg_{H[\tilde{d}]}(v)}{2\sqrt{n}}\right).$$

Applying Chernoff's bound on the lower tail we have

$$\mathbb{P}\text{rob}\left[deg_H(v) < \frac{deg_{H[\tilde{d}]}(v)}{2\sqrt{n}}\right] \;<\; \exp\left(-\frac{deg_{H[\tilde{d}]}(v)}{8\sqrt{n}}\right) \;\leq\; \exp\left(-\frac{\tilde{d}}{8\sqrt{n}}\right),$$

which in turn implies (because $\tilde{d} > 16\sqrt{n}\log n$),

$$\mathbb{P}\text{rob}\left[deg_H(v) < \frac{\tilde{d}}{2\sqrt{n}}\right] \;\leq\; \exp\left(-\frac{16\sqrt{n}\log n}{8\sqrt{n}}\right) \;=\; \frac{1}{n^2}.$$

□

**Claim 6** *$\delta_H \geq \tilde{d}/(2\sqrt{n})$ with probability at least $1 - 1/n$.*

***Proof:*** By Claim 5, the probability that *any* node $v$ of $H$ has $deg_H(v) < \tilde{d}/(2\sqrt{n})$ is at most $\frac{1}{n^2} \cdot |H| \leq 1/n$. □

Claim 4 and Claim 6 together show that with probability at least $1 - \frac{1}{n} - \frac{1}{n^4} \geq 1 - \frac{2}{n}$, $H$ has at most $k$ vertices and has minimum degree at least $\tilde{d}/(2\sqrt{n})$. Therefore, with high probability, $H$ provides a solution of DDD$k$S which is within a factor $1/(2\sqrt{n})$ of the optimal solution. This concludes the proof of the theorem. □

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399