Minimum Size Tree-Decompositions*

Bi Li, Fatima Zahra Moataz, and Nicolas Nisse

[1] Inria, France
[2] Univ. Nice Sophia Antipolis, CNRS, I3S, UMR 7271, Sophia Antipolis, France

Abstract. *Tree-Decompositions* are the corner-stone of many dynamic programming algorithms for solving graph problems. Since the complexity of such algorithms generally depends exponentially on the *width* (size of the *bags*) of the decomposition, much work has been devoted to compute tree-decompositions with small width. However, practical algorithms computing tree-decompositions only exist for graphs with *treewidth* less than 4. In such graphs, the time-complexity of dynamic programming algorithms based on tree-decompositions is dominated by the *size* (number of bags) of the tree-decompositions. It is then interesting to try to minimize the size of the tree-decompositions.
In this extended abstract, we consider the problem of computing a tree-decomposition of a graph with width at most k and minimum size. More precisely, we focus on the following problem: given a fixed $k \geq 1$, what is the complexity of computing a tree-decomposition of width at most k with minimum size in the class of graphs with treewidth at most k? We prove that the problem is NP-complete for any fixed $k \geq 4$ and polynomial for $k \leq 2$. On going work also suggests it is polynomial for $k = 3$.

1 Introduction

A *tree-decomposition* of a graph G [11] is a way to represent G by a family of subsets of its vertex-set organized in a tree-like manner and satisfying some connectivity property. The *treewidth* of G measures the proximity of G with a tree. More formally, a tree decomposition of $G = (V, E)$ is a pair $(T, \mathcal{X})$ where $\mathcal{X} = \{X_t | t \in V(T)\}$ is a family of subsets, called *bags*, of V, and T is a tree, such that:

- $\bigcup_{t \in V(T)} X_t = V$;
- for any edge $uv \in E$, there is a bag X_t (for some node $t \in V(T)$) containing both u and v;
- for any vertex $v \in V$, the set $\{t \in V(T) | v \in X_t\}$ induces a subtree of T.

The *width* of a tree-decomposition $(T, \mathcal{X})$ is $max_{t \in V(T)} |X_t| - 1$ and its *size* is order $|V(T)|$ of T. The treewidth of G, denoted by $tw(G)$, is the minimum width over all possible tree-decompositions of G.

If T is constrained to be a path, $(T, \mathcal{X})$ is called a *path-decomposition* of G. The pathwidth of G, denoted by $pw(G)$, is the minimum width over all possible path-decompositions of G.

Tree-Decompositions are the corner-stone of many dynamic programming algorithms for solving graph problems. As an example, the famous Courcelle's Theorem states that any problem expressible in MSOL can be solved in linear-time in the class of bounded treewidth graphs [5]. Another framework based on graph decompositions is the *bi-dimensionality theory* that allowed the design of sub-exponential-time algorithms for many problems in the class of graphs excluding some fixed graph as a minor (e.g., [6]). Given a tree-decomposition with width w and size n, the time-complexity of most of such dynamic programming algorithms can be expressed as $O(2^w n)$ (or $O(2^{w \log w} n)$ in the case of *global* problems). Therefore, the problem of computing tree-decompositions with small width has drawn much attention in the last decades. It has been extensively studied and investigated from different angles: parametrized complexity, exact or approximation algorithms.

The above mentioned algorithms have mainly a theoretical interest because, on the one hand, their time-complexity exponential depends on the treewidth of graphs and, on the other hand, as far as we know, no practical algorithm exists that computes a "good" tree-decomposition for graphs with treewidth at least 5. However, in case of small (≤ 4) treewidth graphs, efficient (i.e., practical) algorithms exist to compute

*

tree-decompositions with optimal width. Moreover, in such case, the time-complexity of above-mentioned dynamic programming algorithms becomes dominated by the size of the tree-decompositions and, therefore, it becomes interesting to minimize it.

In this extended abstract, we deal with the problem of computing tree-decompositions with minimum size. Obviously, if the width is not constrained, then the problem is trivial since there always exists a tree-decomposition of a graph with one bag (the full vertex-set). Hence, given a graph G and an integer $k \geq tw(G)$, we consider the problem of minimizing the size of a tree-decomposition of G with width at most k.

Our results. Let k be any positive integer and G be any graph with treewidth at most k. Let $s_k(G)$ denote the smallest size of a tree-decomposition of G with width at most k. We first prove that, for any (fixed) $k \geq 4$, the problem of computing s_k is NP-hard in the class of graphs with treewidth at most k. Then, we prove that computing s_2 can be solved in polynomial-time in the class of graphs with treewidth at most 2.

Related Work. The problem of computing "good" tree-decompositions has been extensively studied. Computing optimal tree-decomposition - i.e., with width $tw(G)$ - is NP-complete in the class of general graphs G [1]. For any fixed $k \geq 1$, Bodlaender designed an algorithm that computes, in time $O(k^{k^3}n)$, a tree-decomposition of width k of any n-node graph with treewidth at most k [3]. Very recently, a single-exponential (in k) algorithm has been proposed that computes a tree-decomposition with width at most $5k$ in the class of graphs with treewidth at most k [4]. As far as we know, the only practical algorithms for computing optimal tree-decompositions hold for graphs with treewidth at most 1 (trivial since $tw(G) = 1$ if and only if G is a tree), 2 (graphs excluding K_4 as a minor) [13], 3 [2, 9, 10] and 4 [12].

We are not aware of any work dealing with the computation of tree-decompositions with minimum size. In [7], Dereniowski *et al.* consider the problem of size-constrained path-decompositions. Given any positive integer k and any graph G with pathwidth at most k. Let $l_k(G)$ denote the smallest size (length) of a path-decomposition of G with width at most k. For any fixed $k \geq 4$, computing l_k is NP-complete in the class of general graphs and it is NP-complete, for any fixed $k \geq 5$, in the class of connected graphs [7]. Moreover, computing l_k can be solved in polynomial-time in the class of graphs with pathwidth at most k for any $k \leq 3$. Finally, the "dual" problem is also hard: for any fixed $s \geq 2$, it is NP-complete in general graphs to compute the minimum width of a tree-decomposition with size s [7][3].

2 NP-hardness in the class of graphs with treewidth at least 4

In this section, we prove that:

Theorem 1. *For any fixed integer $k \geq 4$ (resp., $k \geq 5$), the problem of computing s_k is NP-complete in the class of graphs (resp., of connected graphs) with treewidth at most k.*

Note that the corresponding decision problem is clearly in NP, hence, we only need to prove it is NP-hard.

Our proof mainly follows the one of [7] for size-constrained path-decompositions. Hence, we recall here the two steps of the proof in [7]. First, it is proved that, if computing l_k is NP-hard for some $k \geq 4$ in general graphs, then the computation of l_{k+1} is NP-hard in the class of connected graphs. Then, it is shown that computing l_4 is NP-hard in general graphs with pathwidth at least 4, resp., computing l_5 is NP-hard in the class of connected graphs with pathwidth at least 5. The second step consists of a reduction from the 3-PARTITION problem [] to the one of computing l_4 (resp., of l_5 in connected graphs). Precisely, for any instance $\mathcal{I}$ of 3-PARTITION, a graph $G_{\mathcal{I}}$ is built such that $\mathcal{I}$ is a YES instance if and only if $l_4(G_{\mathcal{I}})$ equals some defined value $\ell_{\mathcal{I}}$.

Our contribution consists first in showing that the first step of [7] directly extends to the case of tree-decompositions. That is, it directly implies that, if computing s_k is NP-hard for some $k \geq 4$ in general graphs, then the computation of s_{k+1} is NP-hard in connected graphs. Our main contribution of this section is to show that, for the graphs $G_{\mathcal{I}}$ built in the reduction proposed in [7], any tree-decomposition of $G_{\mathcal{I}}$ with width at most 4 and minimum size is a path decomposition. Hence, in this class of graphs, $l_4 = s_4$ and, therefore, for any instance $\mathcal{I}$ of 3-PARTITION, $\mathcal{I}$ is a YES instance if and only if $s_4(G_{\mathcal{I}})$ equals $\ell_{\mathcal{I}}$. Theorem 1 follows.

[3] This result was proved in [7] in terms of path-decomposition but it is straightforward to extend it to tree-decomposition.

3 Polynomial cases

In this section, we give preliminary results on when minimum size tree-decompositions can be computed in polynomial-time. We first investigate the case of forests.

Theorem 2. *For any $k \in \{1, 2, 3\}$, a tree-decomposition with size $s_k(F)$ and width k can be computed in polynomial-time in the class of forests F.*

Note that the computation of s_1 is trivially polynomial since, in particular, $s_1(T) = n - 1$ for any n-node tree. For $k \in \{2, 3\}$, we design polynomial-time algorithms for computing s_k in the class of forests. These are recursive algorithms that proceed greedily. Intuitively, in any forest F, we can identify a subgraph S of size at most $k + 1$ such that S is a bag in a minimum-size tree-decomposition of F with width k.

Due to lack of space, we only give an example in the case $k = 3$. Let F be a forest and let $v \in V(F)$ adjacent to exactly one non-leaf node. Moreover, assume that v is adjacent to at least three leaves $a, b, c \in V(F)$. Our algorithm first computes (recursively) a minimum-size tree-decomposition $\mathcal{T}$ of $F \setminus \{a, b, c\}$ with width at most 3. Let B be any bag of $\mathcal{T}$ containing v. We prove that the tree-decomposition obtained from $\mathcal{T}$ by adding a bag $\{v, a, b, c\}$ adjacent to B is a minimum-size tree-decomposition of F with width 3.

The key point is that the number of cases to consider is relatively small. In particular, in the case of trees, we prove that there always exists a minimum-size tree-decomposition with width at most $k \leq 3$ where each bag induces a (connected) subtree. That is, in case of trees, the cases to consider are all trees with at most 4 nodes. It seems that our algorithms cannot be easily extended for $k \geq 4$ since, in particular, this connectivity property is not valid anymore for $k > 4$ (see conclusion).

Then, we focus on graphs with treewidth 2.

Theorem 3. *A tree-decomposition of size $s_2(G)$ and width 2 can be computed in polynomial-time in the class of graphs G with treewidth 2.*

The first step of the proof of the above theorem is to consider 2-connected graphs with treewidth 2. It is known that any 2-connected graph has treewidth 2 if and only if it has an *open nested ear decomposition* starting from a single edge [8]. In particular, this implies that any such a graph contains a node with degree 2. Given a 2-connected graph G with treewidth 2, let v be a node with degree 2 and u and w its neighbors. Let G' obtained from G by contracting the edge $\{u, v\}$ (or equivalently, removing v and adding an edge between u and w). Our algorithm first computes (recursively) a minimum-size tree-decomposition $\mathcal{T}$ of G' with width 2. One bag B of it contains $\{u, w\}$. We prove that the tree-decomposition obtained from $\mathcal{T}$ by adding a bag $\{v, u, w\}$ adjacent to B is a minimum-size tree-decomposition of G with width 2.

Then, we consider the case of general graphs with treewidth 2. Given such a graph G, let $G_1, \cdots, G_r$ be its *blocks*, i.e., its inclusion-maximal 2-connected components and let G' be the graph obtained by removing all edges of the G_is ($i \leq r$) and removing all nodes that are not cut-vertices. By definition G' is a forest. We prove that a minimum-size tree-decomposition of G with width 2 can easily be obtained by combining minimum-size tree-decompositions with width 2 of the subgraphs G_i (computable in polynomial-time by above paragraph) and of F (using Theorem 2). More precisely, we prove that there always exists a minimum-size tree-decomposition of G with width 2 that does not contain *mixed* bags, i.e., bags containing two nodes in some G_i and one node not in G_i.

Finally, we consider the case $k = 3$ for graphs with treewidth 2. We prove that:

Theorem 4. *A tree-decomposition of size $s_3(G)$ and width at most 3 can be computed in polynomial-time in the class of 2-connected graphs G with treewidth 2.*

Unfortunately, in the case $k = 3$, minimum-size tree-decomposition with width 3 may always contain mixed bags. This makes the computation of s_3 in the case of connected graphs with treewidth 2 more tricky. We are currently investigated this case.

4 Conclusion

In this extended abstract, we gave preliminary results on the complexity of minimizing the size of tree-decompositions with given width. Table 1 summarizes our results as well as the remaining open questions. We currently investigate the case of s_3 in the class of graphs with treewidth 2 or 3 and we conjecture it is polynomially solvable. The problem of computing s_k, for $k \geq 4$, seems more intricate already in the case of trees. Indeed, our polynomial-time algorithms to compute s_k, $k \leq 3$, in trees mainly rely on the fact that, for any tree T, there exists a minimum-size tree-decomposition of T with width at most 3, where each bag induces a connected subtree. This is unfortunately not true anymore in the case of tree-decomposition with width 5. As an example, consider the tree (with 10 nodes) obtained from a star with three 3 leaves by subdividing twice each edge.

	s_1	s_2	s_3	$s_k, k \geq 4$
$tw = 1$	$P(trivial)$	P	P	?
$tw = 2$	-	P	?	?
$tw = 3$	-	-	?	?
$tw \geq 4$	-	-	-	NP-hard

Table 1. Summary of the complexity results.

References

1. Arnborg, S., Corneil, D. G., and Proskurowski, A. Complexity of finding embeddings in a k-tree. *SIAM J. Algebraic Discrete Methods 8*, 2 (Apr. 1987), 277–284.
2. Arnborg, S., and Proskurowski, A. Characterization and recognition of partial 3-trees. *SIAM J. Algebraic Discrete Methods 7*, 2 (Apr. 1986), 305–314.
3. Bodlaender, H. L. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput. 25*, 6 (1996), 1305–1317.
4. Bodlaender, H. L., Drange, P. G., Dregi, M. S., Fomin, F. V., Lokshtanov, D., and Pilipczuk, M. A $o(c^k n)$ 5-approximation algorithm for treewidth. *CoRR abs/1304.6321* (2013).
5. Courcelle, B. The monadic second-order logic of graphs. i. recognizable sets of finite graphs. *Information and Computation 85*, 1 (1990), 12 – 75.
6. Demaine, E. D., and Hajiaghayi, M. The bidimensionality theory and its algorithmic applications. *Comput. J. 51*, 3 (2008), 292–302.
7. Dereniowski, D., Kubiak, W., and Zwols, Y. Minimum length path decompositions. *CoRR abs/1302.2788* (2013).
8. Eppstein, D. Parallel recognition of series-parallel graphs. *Inf. Comput. 98*, 1 (May 1992), 41–55.
9. Kajitani, Y., Ishizuka, A., and Ueno, S. Characterization of partial 3-trees in terms of three structures. *Graphs and Combinatorics 2*, 1 (1986), 233–246.
10. Matousek, J., and Thomas, R. Algorithms finding tree-decompositions of graphs. *Journal of Algorithms 12*, 1 (1991), 1 – 22.
11. Robertson, N., and Seymour, P. D. Graph minors. ii. algorithmic aspects of tree-width. *J. Algorithms 7*, 3 (1986), 309–322.
12. Sanders, D. P. On linear recognition of tree-width at most four. *SIAM J. Discret. Math. 9*, 1 (1996), 101–117.
13. Wald, J. A., and Colbourn, C. J. Steiner trees, partial 2-trees, and minimum ifi networks. *Networks 13*, 2 (1983), 159–167.