On the Enumeration of Tree Decompositions

Nofar Carmeli, Batya Kenig, and Benny Kimelfeld

Technion – Israel Institute of Technology

1 Introduction

Many intractable computational problems on graphs admit tractable algorithms when applied to trees or forests. *Tree decomposition* extracts a tree structure from a graph by grouping nodes into *bags*, where each bag corresponds to a single node in of the tree. The corresponding operation on hypergraphs is that of a *generalized hypertree decomposition* [10], which entails a tree decomposition of the *primal* graph (which has the same set of nodes, and an edge between every two nodes that co-occur in a hyperedge) and an assignment of a hyperedge cover to each bag [11]. Tree decomposition and generalized hypertree decomposition have a plethora of applications, including join optimization in databases [7, 10, 21], constraint-satisfaction problems [17], computation of Nash equilibria in games [10], analysis of probabilistic graphical models [18], and weighted model counting [16, 19].

Past research has focused on obtaining a "good" tree decomposition for the given graph, where goodness is typically measured by means of the *width*—the maximal cardinality of a bag. Nevertheless, finding a tree decomposition of a minimal width is NP-hard [2]. Moreover, in various applications the measure of goodness is different from (though related to) the width [11,16]. Abseher et al. [1] empirically showed that the execution cost of dynamic programming algorithms over a tree decomposition is highly sensitive to features of the tree decomposition other than mere width; in particular, tree decompositions of the same width may entail highly diverging running times on the same problem instance.

In this paper, we describe our ongoing effort on the task of *enumerating* all (or a subset of) the tree decompositions of a graph. Such algorithms have been proposed in the past for small graphs (representing database queries), without complexity guarantees [15, 21]. Our main result so far is an enumeration algorithm that runs in incremental polynomial time, and our current efforts are on a practical and effective implementation.

2 Preliminaries

In this section we give some basic terminology.

Graphs. The graphs in this work are undirected. For a graph g, the set of nodes is denoted by $\mathsf{V}(g)$, and the set of edges (where an edge is a set $\{u, v\}$ of distinct nodes) is denoted by $\mathsf{E}(g)$.

Tree Decomposition. A *tree decomposition* d of a graph g is a pair (t, β), where t is a tree and $\beta : \mathsf{V}(t) \to 2^{\mathsf{V}(g)}$ is a function that maps every node of t into a set of nodes of g, so that *(a)* $\cup_{v \in \mathsf{V}(t)} \beta(v) = \mathsf{V}(g)$, *(b)* for every edge $e \in \mathsf{E}(g)$ there is a node $v \in \mathsf{V}(t)$ such that $e \subseteq \beta(v)$, *and (c)* for all $u, v, w \in \mathsf{V}(t)$, if v is on the path between u and w, then $\beta(v)$ contains $\beta(u) \cap \beta(w)$. For a tree decomposition $d = (t, \beta)$ and a node v of t, the set $\beta(v)$ is called a *bag* of d, and we denote by $\mathrm{bags}(d)$ the set $\{\beta(v) \mid v \in \mathsf{V}(t)\}$. Two tree decompositions d_1 and d_2 are *bag equivalent* if $\mathrm{bags}(d_1) = \mathrm{bags}(d_2)$.

When enumerating tree decompositions, we wish to avoid the generation of decompositions that are clearly useless. As an extreme example, if the input graph is already a tree, then usual applications are not interested in any of the tree decompositions (e.g., putting all nodes in a single bag) besides the original tree itself. In a common algorithm over a tree decomposition, the bags are the parts where an expensive (e.g., exponential-time) algorithm is applied. In such cases, it will be beneficial to split a bag, or remove it altogether, if possible. This leads to the notion of a *proper* tree decomposition. Formally, a tree decomposition d of a graph g is said to be proper if there does not exist any tree decomposition d' of g such that *(a)* every bag of d' is contained in some bag of d, *and (b)* $\mathrm{bags}(d) \not\subseteq \mathrm{bags}(d')$. In particular, a proper tree decomposition cannot be improved by removing or splitting a bag. For illustration, a chordal graph (e.g., a tree) may have exponentially many tree decompositions, but only a single tree decomposition up to bag equivalence.

Chordality and Triangulation. Let g be a graph. For a cycle c in g, a *chord* of c is an edge $e \in \mathsf{E}(g)$ that connects two nodes that are non-adjacent in c. We say that g is *chordal* if every cycle of g of length greater than three has a chord. A *triangulation* of a graph g is a graph h such that $\mathsf{V}(g) = \mathsf{V}(h)$, $\mathsf{E}(g) \subseteq \mathsf{E}(h)$, and h is chordal. A *minimal triangulation* of g is a triangulation h of g with the following property: for every graph h' with $\mathsf{V}(g) = \mathsf{V}(h')$, if $\mathsf{E}(g) \subseteq \mathsf{E}(h') \subsetneq \mathsf{E}(h)$, then h' is non-chordal (i.e., h' is not a triangulation of g). In particular, if g is already chordal then g is the only minimal triangulation of itself.

Enumeration Algorithms. Our goal is to devise efficient algorithms for enumerating (proper) tree decompositions. *Polynomial running time* is an inadequate yardstick of efficiency for this problem, since it may be the case that the number of tree decompositions is exponential. Johnson et al. [13] introduced several different notions of efficiency for enumeration algorithms, and we recall these now. *Polynomial total time* means that the total execution time is polynomial in the combined size of the input and the output. *Incremental polynomial time* means that the *delay* after the Nth answer (i.e., the time until the $(N+1)$st answer) is polynomial in $N + n$, where n is the size of the input. Finally, *polynomial delay* means that the every delay is polynomial only in n. Observe that polynomial delay is stronger than incremental polynomial time, which in turn is stronger than polynomial total time.

3 Results

We now describe some of the results we established so far in our ongoing research. In search of an algorithm for enumerating tree decompositions, we started by looking at the problem of enumerating minimal triangulations. Later, we will discuss the connection between the two problems. The first result states that we can enumerate the minimal triangulations in incremental polynomial time.

Theorem 1. *There is an algorithm that, given an input graph g, enumerates the minimal triangulations of g in incremental polynomial time.*

The proof of Theorem 1 builds on the algorithm of Berry et al. [3] for enumerating all the *minimal separators* of a graph, a characterisation of minimal triangulations by means of minimal separators, due to Parra and Scheffler [20], and an algorithm of Cohen et al. [5,6] for enumerating maximal node sets under hereditary graph properties.

The next result states the relationship between proper tree decompositions and minimal triangulations. This result is obtained by combining known results by Heggernes [12] and Jordan [14].

Proposition 1. *Let g be a graph. There is a bijection M between the minimal triangulations of g and the bag-equivalence classes of the proper tree decompositions of g. The function M maps a minimal triangulation h of g to the proper tree decompositions of g that have the maximal cliques of h as bags.*

Note that a *maximal clique* is a clique that is not properly contained in any other clique. Let g be a graph, and let h be a triangulation of g. Let G be the graph that has the maximal cliques of h as its node set, and an edge between every two nodes, weighted by the size of the intersection of its incidents. The tree decompositions that h maps to in Proposition 1 correspond to the maximum-weight spanning trees of G [14]. It is known that the maximum-weight spanning trees of a graph can be enumerated with polynomial delay [22]. Gavril [8] showed that in chordal graphs the number of maximal cliques of h is at most the number of nodes of h. Combining these results with Theorem 1 and Proposition 1, we get the following corollary.

Corollary 1. *There is an algorithm that, given an input graph g, enumerates the proper tree decompositions of g in incremental polynomial time.*

4 Outlook

So far, we have established an algorithm with complexity guarantees for enumerating the minimal triangulations (and the proper tree decompositions) of a graph. In the next steps, we plan to investigate the implementation of our algorithm, in two aspects. The first is that of *efficiency*: we plan to find techniques for optimizing and parallelising the computation (e.g., in the spirit of

previous algorithms of a similar nature [9]). The second aspect is that of a *partial* enumeration: we plan to study the problem of enumerating a subset of the minimal triangulations, and in particular explore the ability of utilizing previous tree-decomposition algorithms [4] for improving the overall quality of that subset. In terms of theoretical directions, it remains open whether the minimal triangulations can be enumerated with polynomial delay, and we plan to further investigate this problem.

References

1. M. Abseher, F. Dusberger, N. Musliu, and S. Woltran. Improving the efficiency of dynamic programming on tree decompositions via machine learning. In *IJCAI*, pages 275–282. AAAI Press, 2015.
2. S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embeddings in ak-tree. *SIAM Journal on Algebraic Discrete Methods*, 8(2):277–284, 1987.
3. A. Berry, J. P. Bordat, and O. Cogis. Generating all the minimal separators of a graph. In P. Widmayer, G. Neyer, and S. Eidenbenz, editors, *WG*, volume 1665 of *Lecture Notes in Computer Science*, pages 167–172. Springer, 1999.
4. H. L. Bodlaender and A. M. C. A. Koster. Treewidth computations i. upper bounds. *Inf. Comput.*, 208(3):259–275, 2010.
5. S. Cohen, I. Fadida, Y. Kanza, B. Kimelfeld, and Y. Sagiv. Full disjunctions: Polynomial-delay iterators in action. In *VLDB*, pages 739–750. ACM, 2006.
6. S. Cohen, B. Kimelfeld, and Y. Sagiv. Generating all maximal induced subgraphs for hereditary and connected-hereditary graph properties. *J. Comput. Syst. Sci.*, 74(7):1147–1159, 2008.
7. J. Flum, M. Frick, and M. Grohe. Query evaluation via tree-decompositions. *J. ACM*, 49(6):716–752, Nov. 2002.
8. F. Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *J. Combinatorial Theory*, 16:47–56, 1974.
9. K. Golenberg, B. Kimelfeld, and Y. Sagiv. Optimizing and parallelizing ranked enumeration. *PVLDB*, 4(11):1028–1039, 2011.
10. G. Gottlob, G. Greco, and F. Scarcello. Pure nash equilibria: Hard and easy games. *J. Artif. Intell. Res. (JAIR)*, 24:357–406, 2005.
11. G. Gottlob, M. Grohe, N. Musliu, M. Samer, and F. Scarcello. Hypertree decompositions: Structure, algorithms, and applications. In *WG*, volume 3787 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2005.
12. P. Heggernes. Treewidth, partial k-trees, and chordal graphs. *unpublished*, 2006.
13. D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis. On generating all maximal independent sets. *Inf. Process. Lett.*, 27(3):119–123, 1988.
14. M. Jordan. *An Introduction to Probabilistic Graphical Models*, chapter 17. University of California, Berkeley, 2002.
15. O. Kalinsky, Y. Etsion, and B. Kimelfeld. Flexible caching in trie joins. *CoRR*, abs/1602.08721, 2016.
16. B. Kenig and A. Gal. On the impact of junction-tree topology on weighted model counting. In *SUM*, volume 9310 of *Lecture Notes in Computer Science*, pages 83–98. Springer, 2015.
17. P. G. Kolaitis and M. Y. Vardi. Conjunctive-query containment and constraint satisfaction. *J. Comput. Syst. Sci.*, 61(2):302–332, 2000.

18. S. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society*, B, 50(2):157–224, 1988.
19. W. Li, P. Poupart, and P. van Beek. Exploiting causal independence using weighted model counting. In *AAAI*, pages 337–343. AAAI Press, 2008.
20. A. Parra and P. Scheffler. Characterizations and algorithmic applications of chordal graph embeddings. *Discrete Applied Mathematics*, 79(1-3):171–188, 1997.
21. S. Tu and C. Ré. DunceCap: Query plans using generalized hypertree decompositions. In *SIGMOD*, pages 2077–2078. ACM, 2015.
22. T. Yamada, S. Kataoka, and K. Watanabe. Listing all the minimum spanning trees in an undirected graph. *Int. J. Comput. Math.*, 87(14):3175–3185, 2010.