Available online at www.sciencedirect.com
SCIENCE DIRECT®
Discrete Applied Mathematics 145 (2005) 143–154
DISCRETE APPLIED MATHEMATICS
www.elsevier.com/locate/dam

Tree decompositions with small cost☆

Hans L. Bodlaender[a], Fedor V. Fomin[b]

[a]*Institute of Information and Computing Sciences, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, The Netherlands*
[b]*Department of Informatics, University of Bergen, Norway*

Received 2 February 2002; received in revised form 16 August 2002; accepted 16 January 2004
Available online 25 September 2004

Abstract

The *f*-cost of a tree decomposition $(\{X_i \mid i \in I\},\ T = (I, F))$ for a function $f : \mathbf{N} \to \mathbf{R}^+$ is defined as $\sum_{i \in I} f(|X_i|)$. This measure associates with the running time or memory use of some algorithms that use the tree decomposition. In this paper, we investigate the problem to find tree decompositions of minimum *f*-cost. A function $f : \mathbf{N} \to \mathbf{R}^+$ is fast, if for every $i \in \mathbf{N}$: $f(i+1) \geqslant 2\, f(i)$. We show that for fast functions *f*, every graph *G* has a tree decomposition of minimum *f*-cost that corresponds to a minimal triangulation of *G*; if *f* is not fast, this does not hold. We give polynomial time algorithms for the problem, assuming *f* is a fast function, for graphs that have a polynomial number of minimal separators, for graphs of treewidth at most two, and for cographs, and show that the problem is NP-hard for bipartite graphs and for cobipartite graphs. We also discuss results for a weighted variant of the problem derived of an application from probabilistic networks.

Keywords: Treecost; Treewidth; Minimal separator; Triangulation; Probabilistic network

1. Introduction

It is well known that many problems that are intractable on general graphs become linear or polynomial time solvable on graphs of bounded treewidth. These algorithms often have the following form: first a tree decomposition of small treewidth is made, and then a dynamic programming algorithm is used, computing a table for each node of the tree. The time to process one node of the tree is exponential in the size of the associated set of vertices of the graph; thus, when the maximum size of such a set is bounded by a constant (i.e., the width of the tree decomposition is bounded by a constant), then the algorithm runs in linear time. However, two different tree decompositions of the same graph with the same width may still give different running times, e.g., when one has many large vertex sets associated to nodes, while the other has only few large vertex sets associated to nodes.

In several applications, the same tree decomposition will be used for several successive runs of an algorithm, e.g., with different data. An important example of such an application is the PROBABILISTIC INFERENCE problem on probabilistic networks. (This application will be briefly discussed in Section 8.) Hence, in many cases it makes sense to do more work on finding a good tree decomposition, and to use a more refined measure on what is a 'good' tree decomposition. Apart from extensive studies on the problem on the notion of treewidth and the notion of 'fill-in', more precise measures have been studied mainly in the context of probabilistic networks (see [22].)

E-mail addresses: hansb@cs.uu.nl (H.L. Bodlaender), fomin@ii.uib.no (F.V. Fomin).
☆ This research was partially supported by EC Contract IST-1999-14186: Project ALCOM-FT (Algorithms and Complexity - Future Technologies).

0166-218X/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2004.01.008

In this paper, we study a notion that more closely reflects the time needed when using the tree decomposition. Suppose processing a node of the tree decomposition whose associated set has size k costs $f(k)$ of some resource (e.g., time or space). Then, processing a tree decomposition of the form $(\{X_i \mid i \in I\}, T = (I, F))$ costs $\sum_{i \in I} f(|X_i|)$. (For precise definitions, see Section 2.) We call this measure the *f-cost* of the tree decomposition; the treecost of a graph G with respect to f is the minimum f-cost of a tree decomposition of G. In this paper, we investigate the problem of finding tree decompositions of minimum f-cost. In Section 10 we discuss in more detail how far this notion comes close to precisely measuring the resources needed by the algorithm.

It appears that it is important whether the function f satisfies a certain condition which we call *fast*: a function $f : \mathbf{N} \to \mathbf{R}^+$ is fast, if for every k, $f(k+1) \geqslant 2 f(k)$. Most applications of treewidth in our framework will have functions that are fast (in particular, many of the classical algorithms using tree decompositions for well-known graph problems have fast cost functions.) To a tree decomposition we can associate a triangulation (chordal supergraph) of input graph G in a natural way. Now, every graph has a tree decomposition of minimum f-cost that can be associated with a *minimal* triangulation, if and only if f is fast. This will be shown in Section 3. This result will be used in later sections to show that the problem of finding minimum f-cost tree decompositions can be solved in polynomial time for graphs that have a polynomial number of separators (Section 4), and in linear time for cographs (Section 5), and for graphs of treewidth at most two (Section 6); assuming in each case that f is fast and polynomial time computable. In Section 7, we discuss a conjecture on the relation between triangulations of minimum f-cost and minimum treewidth, and show that for a fixed k, one can find a triangulation of minimum f-cost among those of treewidth at most k in polynomial time. A variant of the problems for weighted graphs with an application to probabilistic networks is discussed in Section 8. In Section 9, we show the unsurprising but unfortunate result that for each fast f, the TREECOST$_f$ problem is NP-hard for cobipartite graphs and for bipartite graphs. Also, in these cases there is no constant factor approximation algorithm, unless $P = \text{NP}$. Some final remarks are made in Section 10.

2. Preliminaries

We use the following notations: $G = (V, E)$ is an undirected and finite graph with vertex set V and the edge set E, assumed to be without self-loops or parallel edges. Unless otherwise specified, n denotes the number of vertices and m the number of edges of G. The *(open) neighborhood* of a vertex v in a graph G is $N_G(v) = \{u \in V : \{u, v\} \in E\}$ and the *closed neighborhood* of v is $N_G[v] = N_G(v) \cup \{v\}$. For a vertex set $S \subseteq V$ we denote $N_G[S] = \bigcup_{v \in S} N[v]$ and $N(S) = N[S] \backslash S$. If G is clear from the context, we write $N(v)$, $N[v]$, etc. $d_G(v) := |N_G(v)|$ is the degree of v in G. $G - v$ is the graph, obtained by removing v and its incident edges from G.

For a set $S \subseteq V$ of vertices of a graph $G = (V, E)$ we denote by $G[S]$ the subgraph of G induced by S. A set $W \subseteq V$ of vertices is a *clique* in graph $G = (V, E)$ if $G[W]$ is a complete graph, i.e. every pair of vertices from W induces an edge of G. A set $W \subseteq V$ of vertices is a *maximal clique* in $G = (V, E)$, if W is a clique in G and W is not a proper subset of another clique in G.

A *chord* of a cycle C is an edge not in C that has both endpoints in C. A *chordless cycle* in G is a cycle of length more than three that has no chord. A graph G is *chordal* if it does not contain a chordless cycle.

A *triangulation* of a graph G is a graph H on the same vertex set as G that contains all edges of G and is chordal. A *minimal* triangulation of G is a triangulation H such that no proper subgraph of H is a triangulation of G.

Definition. A *tree decomposition* of a graph $G = (V, E)$ is a pair $(\{X_i \mid i \in I\}, T = (I, F))$, with $\{X_i \mid i \in I\}$ a family of subsets of V and T a tree, such that

- $\bigcup_{i \in I} X_i = V$.
- For all $\{v, w\} \in E$, there is an $i \in I$ with $v, w \in X_i$.
- For all $i_0, i_1, i_2 \in I$: if i_1 is on the path from i_0 to i_2 in T, then $X_{i_0} \cap X_{i_2} \subseteq X_{i_1}$.

The *width* of tree decomposition $(\{X_i \mid i \in I\}, T = (I, F))$ is $\max_{i \in I} |X_i| - 1$. The treewidth of a graph G is the minimum width of a tree decomposition of G.

The following well-known result is due to Gavril [12].

Theorem 1 (*Gavril [12]*). *Graph G is chordal if and only there is a* clique tree *of G, i.e. tree decomposition $(\{X_i \mid i \in I\}, T = (I, F))$ of G such that for every node i of T there is a maximal clique W of G such that $X_i = W$.*

A vertex $v \in V$ is *simplicial* in graph $G = (V, E)$, if $N_G(v)$ is a clique. Every chordal graph on at least two vertices contains at least two simplicial vertices [11].