Discrete Applied Mathematics 43 (1993) 97–101
North-Holland

Note

Tree-width, path-width, and cutwidth*

Ephraim Korach** and Nir Solel***

Department of Computer Science, Technion - Israel Institute of Technology, Haifa 32000, Israel

Received 24 September 1990
Revised 25 September 1992

Abstract

Korach, E. and N. Solel, Tree-width, path-width, and cutwidth, Discrete Applied Mathematics 43 (1993) 97–101.

Let $\mathrm{tw}(G)$, $\mathrm{pw}(G)$, $c(G)$, $\Delta(G)$ denote, respectively, the tree-width, path-width, cutwidth and the maximum degree of a graph G on n vertices. It is known that $c(G) \geq \mathrm{tw}(G)$. We prove that $c(G)=O(\mathrm{tw}(G)\cdot\Delta(G)\cdot\log n)$, and if $(\{X_i : i\in I\}, T=(I,A))$ is a tree decomposition of G with tree-width $\leq k$ then $c(G)\leq(k+1)\cdot\Delta(G)\cdot c(T)$. In case that a tree decomposition is given, or that the tree-width is bounded by a constant, efficient algorithms for finding a numbering with cutwidth within the upper bounds are implicit in the proofs. We obtain the above results by showing that $\mathrm{pw}(G)=O(\log n\cdot\mathrm{tw}(G))$, and $\mathrm{pw}(G)\leq(k+1)\cdot c(T)$.

1. Introduction

Given a graph $G=(V,E)$, $|V|=n$, a *numbering* of G is a one-to-one mapping $L_G: V\to\{1,\ldots,n\}$. The *cutwidth* of a numbering L_G is $\max_{1\leq p<n}|\{(u,v)\in E: L_G(u)\leq p<L_G(v)\}|$.

The cutwidth $c(G)$ of G is the minimum cutwidth over all the numberings. The *cut* of a graph at p, $1\leq p<n$, under a numbering L_G is the set of edges $\{(u,v)\in E: L_G(u)\leq p<L_G(v)\}$. The problem of finding the cutwidth of a graph is also known as the *min-cut linear arrangement* problem. The *degree* of a graph $G=(V,E)$, denoted

Correspondence to: Dr. E. Korach, Ben-Gurion University of the Nev, Beer-Sheva 84105, Israel

* This research was partially supported by Technion V.P.R. Fund – Coleman Cohen Research Fund.

** First author's current address: Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel.
*** Second author's current address: National Semiconducter Design Center, Herzelia, Israel.

by $\Delta(G)$ is the maximum degree of a vertex in V. Finding the cutwidth of a graph is NP-hard (the corresponding decision problem is NP-complete [7]). The problem remains NP-complete on series-parallel graphs, i.e., graphs having tree-width ≤ 2 [11], and thus we do not expect to find a polynomial algorithm that finds the cutwidth of an arbitrary graph with bounded tree-width. The problem is solvable in polynomial time on trees: First, in [8] an approximation algorithm for arbitrary trees is given, and also an exact algorithm for complete k-ary trees. For graphs with $\Delta(G) \geq 3$ an $O(n \log^{\Delta(G)-2} n)$ algorithm is given in [5], and in [15] the problem is solved in $O(n \log n)$ time on an arbitrary tree. In [9], for fixed k, an $O(n^{k-1})$ algorithm is given, which determines whether for an arbitrary graph G, $c(G) \leq k$, and also characterizations for graphs with cutwidth 2 and 3 are given.

The *tree-width* of a graph G, denoted by $\mathrm{tw}(G)$, is defined as follows [13]: A *tree decomposition* of a graph $G=(V,E)$ is a pair (X,T) where $T=(I,A)$ is a tree, and $X=\{X_i: i \in I\}$ is a family of subsets of V, such that:

(1.1) $\bigcup_{i \in I} X_i = V$.

(1.2) Every edge of G has both its ends in some X_i $(i \in I)$.

(1.3) For all $i,j,k \in I$, if j lies on the path from i to k in T, then $X_i \cap X_k \subseteq X_j$.

The tree-width of a tree decomposition is $\max_{i \in I} |X_i| - 1$. The tree-width of G is the minimum tree-width taken over all possible tree decompositions of G. The *path-width* of a graph G, denoted by $\mathrm{pw}(G)$, is defined in a similar way [12], but the tree in the above definition is a path.

In [6] it is shown that $c(G) \leq \mathrm{pw}(G) \cdot \Delta(G)$, and in [2] it is proved that for any graph G, $c(G) \geq \mathrm{tw}(G)$. An important application of the min-cut linear arrangement problem is in VLSI layout design. The problem is also related to the search number, the bandwidth and the topological bandwidth of a graph (see e.g. [6, 9, 15]).

Deciding the tree-width of a graph is NP-hard [1], however for a given constant k it was shown in [1] that deciding whether a graph G has tree-width $\leq k$ and constructing an appropriate tree decomposition can be done in $O(n^{k+2})$ time, and in [4] an $O(n^2)$ nonconstructive algorithm for the problem is given. Moreover, for $k=2$ and $k=3$ the decision and construction problems can be solved in $O(n)$ time (see [4]). Some known families of graphs with bounded tree-width are: trees and forests $(\mathrm{tw}(G) \leq 1)$, series-parallel graphs $(\mathrm{tw}(G) \leq 2)$, grid $d * n$ $(\mathrm{tw}(G) \leq d)$, d-outerplanar graphs $(\mathrm{tw}(G) \leq 3d-1)$ and many others (see e.g. [2]).

The aim of this note is to prove upper bounds on the cutwidth as a function of the tree-width. These bounds are stated in Corollaries 7–9. We achieve these bounds using results of [6], and by proving some connections between tree-width, path-width and cutwidth. The proofs yield efficient algorithms for numbering G with cutwidth within the bounds. In some of them the algorithm from [15] is used as a subroutine.

2. Results

Before presenting our results, we quote three lemmas that we use.

Lemma 1 [6]. $c(G) \leq \mathrm{pw}(G) \cdot \Delta(G)$.

Lemma 2 [5]. *The number of vertices in the smallest degree-3 tree with cutwidth c is* $3^{c-1}+1$.

Lemma 3 [2, Theorem 5.4]. *For every graph* G, $c(G) \geq \mathrm{pw}(G)$.

Lemma 4. *Let* $G=(V,E)$ *be a graph, and let* $(\{X_i: i \in I\}, T=(I,A))$ *be a tree decomposition of* G *with tree-width* $= k$. *Then* $\mathrm{pw}(G) \leq (k+1) \cdot c(T)$.

Proof. We follow the method used in the proof of [2, Theorem 5.4]. Let L_T be a numbering of T with cutwidth $= c(T)$. Let the subsets in the path decomposition (Y,P) of G be defined as follows: $Y_p = \bigcup \{X_i: L_T(i) \leq p,\ L_T(j) > p$ and $(i,j) \in A$ (i.e., the edge (i,j) of T is in the cut at $p)\} \cup \{X_{L_T^{-1}(p)}\}$, for each $1 \leq p \leq |I|$.

The above definition of subsets defines a legal path decomposition: Conditions (1.1) and (1.2) in the definition of a path decomposition are met since every vertex and every edge in G are contained in at least one subset X_i. Condition (1.3) is met since for all $v \in V$, all the subsets X_i that contain v induce a connected subtree in T, and therefore v will be in a connected subpath in the path decomposition. Note that for a connected graph G, if no edge incident to $L_G(w)=p$ is included in the cut at p then the size of the cut at $p-1$ is larger than the size of the cut at p. Therefore each set Y_p is the union of at most $c(T)$ subsets, and thus its size is at most $(k+1) \cdot c(T)$. □

Theorem 5. *For any forest* F *on* n *vertices,* $\mathrm{pw}(F) = O(\log n)$.

We present two different algorithmic proofs for the above theorem. The first proof is direct, but the constants achieved in the second proof are better. There are graphs for which the algorithm in the first proof may achieve a better path decomposition, and other graphs for which the second algorithm is better.

Proofs of Theorem 5. *Proof* I: Find a minimum separating set of vertices in F (a singleton, or the empty set) such that its removal separates F into two parts, each one with no more than $2/3n$ vertices. This separating set will be added to all subsets X_i in the path decomposition. Continue this process recursively with each one of the two forests, i.e., in each forest find such a separating set, remove it from the forest, and add it to all subsets in the path decomposition in the corresponding forest.

Proof II: Let (X,T) be a tree decomposition of F. Transform T to a new tree, T' with maximum vertex degree of three, and no more than $2n$ vertices (for existence, and a simple linear algorithm, see [10]). From Lemma 2 it follows that a degree-three tree with at most $2n$ vertices, has cutwidth of at most $\log n/\log 3 + 1/\log 3 + 1$. Finally, define the subsets Y_i of the path decomposition as in the proof

of Lemma 4. As shown there, the resulting path decomposition has path-width which is no more than $2 \cdot (\log n/\log 3 + 1/\log 3 + 1)$. □

Remarks. (1) The time complexity of the algorithms in Proof I and II is $O(n \log n)$ (in Proof II finding an optimal numbering for T' takes $O(n \log n)$ [15], and all the other parts of the algorithm take linear time). We do not know whether the problem of finding the path-width of a tree is polynomial, neither if it is NP-complete.

(2) The path-width of the path decomposition achieved by the algorithm in Proof I, in the worst case, is bounded above by $\log n/\log(3/2)$, where in the algorithm of Proof II the upper bound on the path-width is $2 \cdot (\log n/\log 3 + 1/\log 3 + 1)$, i.e., in the worst case, the bound on the path-width achieved by the second algorithm is better.

(3) For a star graph, the algorithm in I achieves an optimal path decomposition (with path-width = 1), while the algorithm in II might result in a worse path decomposition. On the other hand, for the path graph, the algorithm in I always achieves a path decomposition with path-width logarithmic in n, while the algorithm in II, for the "natural" tree decomposition of the path graph, produces an optimal path decomposition (with path-width = 1).

(4) In [12] the tree Y_λ is defined and shown to have path-width $= \lceil (\lambda + 1)/2 \rceil$. Therefore, the upper bound in Theorem 5 is tight, since Y_λ contains $3 \cdot 2^\lambda - 2$ vertices.

Theorem 6. *For any graph G on n vertices,* $\mathrm{pw}(G) = O(\log n \cdot \mathrm{tw}(G))$.

Proof. Let $(\{X_i: i \in I\}, T = (I, A))$ be a tree decomposition of G. Find a path decomposition $(\{Y_j: j \in J\}, P = (J, B))$ of T, with path-width $= O(\log n)$. The path decomposition $(\{Z_j: j \in J\}, P = (J, B))$ of G is the following: For each $j \in J$, $Z_j = \bigcup \{X_i: i \in Y_j\}$. One can see that this defines a legal path decomposition and by Theorem 5 its path-width is $O(\log n \cdot \mathrm{tw}(G))$. □

Corollary 7. *For any graph G on n vertices,* $c(G) = O(\log n \cdot \mathrm{tw}(G) \cdot \Delta(G))$.

Proof. Follows from Theorem 6 and Lemma 1. □

From Corollary 7 we have the following:

Corollary 8. *Let G be a graph on n vertices, where* $\mathrm{tw}(G)$ *and* $\Delta(G)$ *are bounded by constants. Then* $c(G) = O(\log n)$.

Remark. From Lemma 2, for every n, there exists a degree-three tree (which has tree-width = 1) whose cutwidth $= \lfloor \log 3 \cdot \log(n-1) \rfloor + 1$ (see [5]) and thus in the worst case the cutwidth of graphs with bounded tree-width and bounded degree is $\Omega(\log n)$. Therefore the upper bound in Corollary 8 is tight.

Corollary 9. *Let (X, T) be a tree decomposition of a graph G with tree-width $= k$. Then $c(G) \leq c(T) \cdot (k+1) \cdot \Delta(G)$.*

Proof. Follows from Lemmas 1 and 4. □

Complexity aspects: All the proofs we have given are algorithmic. Achieving a numbering with cutwidth within the bound in Lemma 1 (if a path decomposition of the graph is given) can be done in linear time (see [6]). Therefore, for a graph G with constant tree-width, finding a numbering with cutwidth satisfying the bound given in Corollary 7 can be done in polynomial time (since finding an appropriate tree decomposition, in that case, takes polynomial time). For a graph G, given together with a tree decomposition (X, T) with tree-width $= k$, obtaining a numbering with cutwidth satisfying the bound in Corollary 9 takes $O(n \log n)$ time (since finding a numbering for T with minimum cutwidth takes $O(n \log n)$ time [15]).

References

[1] S. Arnborg, D. Corneil and A. Proskurovski, Complexity of finding embeddings in a K-tree, SIAM J. Algebraic Discrete Methods 8 (1987) 277–284.

[2] H.L. Bodlaender, Classes of graphs with bounded tree-width, Bulletin of EATCS (1988) 116–128.

[3] H.L. Bodlaender, Dynamic programming on graphs with bounded tree-width, in: T. Lepisto and A. Salomaa, eds., ICALP 1988, Lecture Notes in Computer Science 317 (Springer, Berlin, 1988).

[4] H.L. Bodlaender, Improved self-reduction algorithms for graphs with bounded tree-width, in: M. Nagl, ed., WG 1989, Lecture Notes in Computer Science 411 (Springer, Berlin, 1989).

[5] M.J. Chung, F. Makedon, I.H. Sudborough and T. Turner, Polynomial algorithms for the min-cut linear arrangement problem on degree restricted trees, SIAM J. Comput. 14 (1985) 158–177.

[6] F.R.K. Chung and P.D. Seymour, Graphs with small bandwidth and cutwidth, Discrete Math. 75 (1989) 113–119.

[7] M.R. Garey and D.S. Johnson, Computers and Intractability (Freeman, New York, 1979).

[8] T. Lengauer, Upper and lower bounds on the complexity of the min cut linear arrangement problem on trees, SIAM J. Algebraic Discrete Methods 3 (1982) 99–113.

[9] F. Makedon and I.H. Sudborough, On minimizing width in linear layouts, Discrete Appl. Math. 23 (1989) 243–265.

[10] J. Matousek and R. Thomas, On the complexity of finding iso- and other morphisms for partial k-trees, Manuscript (1988).

[11] B. Monien and I.H. Sudborough, Min-cut is NP-complete for edge weighted trees, in: ICALP 1986, Lecture Notes in Computer Science 226 (Springer, Berlin, 1986).

[12] N. Robertson and P.D. Seymour, Graph minors I. Excluding a forest, J. Combin. Theory Ser. B 35 (1983) 39–61.

[13] N. Robertson and P.D. Seymour, Graph minors II. Algorithmic aspects of tree width, J. Algorithms 7 (1986) 309–322.

[14] N. Robertson and P.D. Seymour, Graph minors IV. Tree-width and well-quasi-ordering, J. Combin. Theory Ser. B 48 (1990) 227–254.

[15] M. Yannakakis, A polynomial algorithm for the min cut linear arrangement of trees, J. ACM 32 (1985) 950–989.