On the Treewidth and Pathwidth of Permutation Graphs

T. Kloks, H. Bodlaender

RUU-CS-92-13
March 1992

On the Treewidth and Pathwidth of Permutation Graphs

T. Kloks, H. Bodlaender

Technical Report RUU-CS-92-13
March 1992

Department of Computer Science
Utrecht University
P.O.Box 80.089
3508 TB Utrecht
The Netherlands

ISSN: 0924–3275

On the Treewidth and Pathwidth of Permutation Graphs

T. Kloks [*] H. Bodlaender [†]
Department of Computer Science, Utrecht University
P.O.Box 80.089, 3508 TB Utrecht, The Netherlands

Abstract

In this paper we discuss the problem of finding the treewidth and pathwidth of permutation graphs. If $G[\pi]$ is a permutation graph with treewidth k, then we show that the pathwidth of $G[\pi]$ is at most $2k$, and we give an algorithm which constructs a path-decomposition with width at most $2k$ in time $O(nk)$. We assume that the permutation π is given. For permutation graphs of which the treewidth is bounded by some constant, this result, together with previous results ([9], [15]), shows that the treewidth and pathwidth can be computed in linear time.

1 Introduction

In many recent investigations in computer science, the notions of treewidth and pathwidth play an increasingly important role. One reason for this is that many problems, including many well studied NP-complete graph problems, become solvable in polynomial and usually even linear time, when restricted to the class of graphs with bounded tree- or pathwidth. Of crucial importance for these algorithms is, that a tree-decomposition or path-decomposition of the graph is given in advance. Much research has been done in finding a tree-decomposition with a reasonable small treewidth. Recent results ([19]) show that an $O(n \log n)$ algorithm exists to find a suitable tree-decomposition for a graph with bounded treewidth. However, the constant hidden in the 'big oh', is exponential in the treewidth, limiting the practicality of this algorithm. However, for many special classes of graphs, it

[*]This author is supported by the foundation for Computer Science (S.I.O.N) of the Netherlands Organization for Scientific Research (N.W.O.), Email: ton@cs.ruu.nl.

[†]This author is partially supported by the ESPRIT II Basic Research Actions Program of the EC under contract no. 3075 (project ALCOM), Email:hansb@cs.ruu.nl.

has been shown that the treewidth can be computed efficiently. In this paper we discuss the problem of finding an approximate path-decomposition for a permutation graph.

Permutation graphs have a large number of applications in scheduling problems. See for example [12], where permutation graphs are used to describe a problem concerning the memory requirements of a number of programs at a certain time (see also [14]). Permutation graphs also arise in a natural way, in the problem of sorting a permutation, using queues in parallel. In [14] it is shown that this problem is closely related with the coloring problem of permutation graphs.

We show that the pathwidth of a permutation graph is at most two times the treewidth of that graph, and we give an algorithm which produces a path-decomposition which is at most two times off the optimal one. If the treewidth of the permutation graph is bounded by a constant, this result, together with earlier results show that an optimal tree- and path-decomposition can be computed in linear time.

2 Preliminaries

In this section we start with some definitions and easy lemmas. In the next section we give an algorithm which approximates the treewidth and pathwidth of a permutation graph and we show that it is correct. If the treewidth is bounded by a constant, these results together with earlier results ([9], [15]) show that the exact treewidth and pathwidth can be computed in linear time.

We think of a permutation π of the numbers $1, \ldots, n$, as the sequence $\pi = [\pi_1, \ldots, \pi_n]$. We use the notation π_i^{-1} for the position of the number i in this sequence.

Definition 2.1 *If π is a permutation of the numbers $1, \ldots, n$, we can construct an undirected graph $G[\pi] = (V, E)$ with vertex set $V = \{1, \ldots, n\}$, and edge set E:*

$$(i,j) \in E \Leftrightarrow (i-j)(\pi_i^{-1} - \pi_j^{-1}) < 0$$

An undirected graph is called a permutation graph *if there exists a permutation π such that $G \cong G[\pi]$.*

Notice that we can obtain the complement of $G[\pi]$, by reversing the sequence π. Hence the complement of a permutation graph is also a permutation graph. It is also easy to see that a permutation graph is a comparability graph (i.e there is a transitive orientation of the edges). The following characterization of permutation graphs appears in [18].

Theorem 2.1 *A graph G is a permutation graph if and only if G and $\overline{G}$ are comparability graphs.*

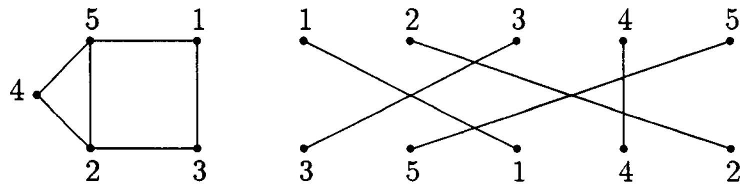

Figure 1: permutation graph and matching diagram

Permutation graphs are *perfect*, (i.e. for every induced subgraph the chromatic number is equal to the maximum cliquesize), and can be recognized in time $O(n^2)$, using the characterization of theorem 2.1 (see [22]). There exist fast algorithms for many NP-complete problems like **Clique**, **Independent set**, **Feedback vertex set** and **Dominating set** when restricted to permutation graphs (see [14], [13], [11] and [10]). For further information and some applications the reader is referred to [14].

In this paper we assume that the permutation π is given, and we show some results on the pathwidth and treewidth of $G[\pi]$, which is also called the *inversion graph* of π. If the permutation π is *not* given, transitive orientations of G and $\overline{G}$ can be computed in $O(n^2)$ time ([22]). Given these orientations, a permutation can be computed in $O(n^2)$ time (see e.g. [14]).

A permutation graph $G[\pi]$ is an intersection graph, which is illustrated by the *matching diagram* of π ([14]).

Definition 2.2 *Let π be a permutation of $1, \ldots, n$. The* matching diagram *of π can be obtained as follows. Write the number $1, \ldots, n$ horizontally from left to right. Underneath, write the numbers $\pi_1, \ldots, \pi_n$, also horizontally from left to right. Draw n straight line segments joining the two 1's, the two 2's, etc.*

The following result follows immediately from the definitions:

Lemma 2.1 *Let π be a permutation of $1, \ldots, n$. Two vertices i and j of $G[\pi]$ are adjacent if and only if the corresponding line segments in the matching diagram of π intersect.*

In the following section we show how to use these matching diagrams in computing an approximate path-decomposition for $G[\pi]$. Before we continue we give an example of these concepts in figure 1.

Definition 2.3 *A* tree-decomposition *of a graph $G = (V, E)$ is a pair $D = (S, T)$ with $T = (I, F)$ a tree and $S = \{X_i \mid i \in I\}$ a collection of subsets of vertices of G, one subset for each node in T, such that the following three conditions are satisfied:*

1. $\bigcup_{i \in I} X_i = V$.

2. For all edges $(v, w) \in E$ *there is a subset* $X_i \in S$*, such that both* v *and* w *are contained in* X_i.

3. For each vertex x*, the set* $\{i \in I \mid x \in X_i\}$ *forms a connected subtree of* T.

A path-decomposition *of a graph* G *is a tree-decomposition* (S, T) *such that* T *is a path. The* width *of a tree-decomposition* (S, T)*, with* $S = \{X_i \mid i \in I\}$*, is* $\max_{i \in I}(|X_i| - 1)$.

Definition 2.4 *The* treewidth *of* G *is the minimum width over all tree-decompositions of* G*. The* pathwidth *of* G *is the minimum width over all path-decompositions of* G.

An alternative way to define the class of graphs with treewidth at most k is by means of *partial k-trees.*

Definition 2.5 *A k-tree is defined recursively as follows: A clique with $k+1$ vertices is a k-tree. Given a k-tree T_n with n vertices, a k-tree with $n + 1$ vertices is constructed by making a new vertex x_{n+1} adjacent to a k-clique of T_n and nonadjacent to the $n - k$ other vertices of T_n. A* partial k-tree *is a subgraph of a k-tree.*

Notice that k-trees are triangulated, and have maximum clique size $k + 1$. It can be seen that if G is a partial k-tree with at least $k+1$ vertices, then there exists a k-tree *H with the same vertex set*, such that G is a subgraph of H. It can be shown that the class of graphs with treewidth at most k is exactly the class of partial k-trees (see e.g. [21], [16]). There exist linear time algorithms for many NP-complete problems, when restricted to the class of partial k-trees for some *constant* k and when a tree-decomposition with bounded width is given (see e.g. [1], [5], [7], [3] and [21]). Determining whether the treewidth or pathwidth of a given graph is at most a given integer k is NP-complete ([2]). In view of this the results of Robertson and Seymour on minor closed classes of graph are of great interest.

Definition 2.6 *An* elementary contraction *of a graph G is the operation which replaces two adjacent vertices u and v by a new vertex, and makes this new vertex adjacent to all vertices which were adjacent to u or to v. A graph is called a* minor *of G if there is a subgraph of G, which can be transformed into H by a series of elementary contractions. A class of graphs is* minor closed *if for every graph G in the class every minor of G is also in the class.*

The following well known lemma is easy to check (see e.g [16]).

Lemma 2.2 *For every fixed k, the classes of graphs with treewidth at most k and with pathwidth at most k are minor closed.*

Robertson and Seymour proved the following result ([20]).

Theorem 2.2 *Every class of minor closed graphs, is recognizable in $O(n^3)$ time.*

It follows that for every constant k there is a polynomial algorithm that recognizes graphs with treewidth at most k. In fact, for these classes faster algorithms exist. We list some of the results.

1. For $k = 2, 3$ there is a linear time algorithm for the treewidth problem using rewrite rules (see e.g. [4] and [17]).

2. For fixed $k \geq 4$ an $O(n \log n)$ algorithm exists which constructs a tree-decomposition with width k ([19], [9] and [15]).

3. If an (approximate) tree-decomposition with bounded width is given, the exact treewidth, and a corresponding tree-decomposition, can be computed in linear time ([9]). In this case, when also the pathwidth is bounded, also the pathwidth and an optimal path-decomposition can be computed in linear time.

For an introductory overview of recent results dealing with treewidth and pathwidth, the reader is referred to [6].

In the next section we show that, if a permutation graph G has treewidth at most k then the pathwidth is at most $2k$, and there exists an $O(kn)$ algorithm to find a path-decomposition for G with width at most $2k$. For *constant* k, this shows the existance of linear time algorithms to compute the treewidth and pathwidth of G.

3 An approximate path-decomposition

In this section, let $G[\pi]$ be a permutation graph with n vertices and with treewidth k. We show there exists a path-decomposition of width at most $2k$, and we give a linear time algorithm to compute this. The algorithm outputs a set X_i for $1 \leq i \leq n$. A vertex j is put in all sets X_k with $\pi_j^{-1} \leq k < j$ or $j \leq k < \pi_j^{-1}$. The precise algorithm is given in figure 2. For example, for the graph of figure 1, the computed sets are: $X_1 = \{1,3\}$, $X_2 = \{1,2,3,5\}$, $X_3 = \{2,5\}$, $X_4 = \{4,2,5\}$ and $X_5 = \emptyset$. Notice that this path-decomposition is not optimal since the pathwidth of this graph is 2 and the computed path-decomposition has width 3.

The next lemma shows that the constructed sets form indeed a path-decomposition.

Lemma 3.1 *Let $S = \{X_i \mid 1 \leq i \leq n\}$ be the subsets of vertices constructed by the algorithm. Let $P = (1, \ldots, n)$ be the path with n vertices. Then (S, P) is a path-decomposition for the permutation graph $G[\pi]$.*

Proof. We first show that each vertex is in at least one subset of S. Consider a vertex i. If $\pi_i^{-1} \geq i$ then i is in the subset X_i. If $\pi_i^{-1} < i$ then i is in the subset X_{i-1}. Next notice that the subsets containing i are consecutive. The only thing left

```
Procedure Pathdec (input $\pi$; output $X$)
for $i \leftarrow 1$ to $n$ do
    $X_i \leftarrow \emptyset$
for $j \leftarrow 1$ to $n$ do
    if $\pi_j^{-1} = j$ then $X_j \leftarrow X_j \cup \{j\}$
    if $\pi_j^{-1} > j$ then
      for $k \leftarrow j$ to $\pi_j^{-1} - 1$ do $X_k \leftarrow X_k \cup \{j\}$
    if $\pi_j^{-1} < j$ then
      for $k \leftarrow \pi_j^{-1}$ to $j - 1$ do $X_k \leftarrow X_k \cup \{j\}$
```

Figure 2: An algorithm to compute a path-decomposition of $G[\pi]$

to show is that every edge is in at least one subset. Consider again a vertex i and let j be a neighbour of i. Assume without loss of generality that $i < j$. In the matching diagram, the linesegment corresponding with j must intersect the linesegment of i. Since $i < j$, this implies that $\pi_j^{-1} < \pi_i^{-1}$. We consider the different orderings of i, j, π_i^{-1} and π_j^{-1}.

1. If $i < j \leq \pi_j^{-1} < \pi_i^{-1}$, then both i and i are contained in the subset X_j.
2. If $i \leq \pi_j^{-1} \leq j \leq \pi_i^{-1}$ then both are contained in $X_{\pi_j^{-1}}$.
3. If $i \leq \pi_j^{-1} < \pi_i^{-1} \leq j$, then both are contained in $X_{\pi_j^{-1}}$.
4. If $\pi_j^{-1} \leq i \leq \pi_i^{-1} \leq j$, then both are contained in X_i.
5. If $\pi_j^{-1} < \pi_i^{-1} \leq i < j$, then both i and j must be in $X_{\pi_i^{-1}}$.

□

We now show that the width of this path-decomposition is at most $2k$. The following lemma will be useful (see also [8]).

Lemma 3.2 *For the complete bipartite graph $G = K(m,n)$, the treewidth is* $\min(m,n)$.

Proof. Assume $m \leq n$. Let V be the set of vertices of G. Let V_1 be the independent set with n vertices and let $V_2 = V \setminus V_1$. If we add all edges between vertices in V_2 (making a clique of V_2), then we obtain a m-tree. Thus the treewidth of G is at most m. Since K_{m+1} is a minor of G, the treewidth of G is at least m. □

Lemma 3.3 *Each subset produced by the algorithm has at most $2k + 1$ elements.*

Proof. Consider a subset X_i. Notice that $X_i \subset S_1 \cup S_2 \cup \{i\}$ where S_1 and S_2 are defined by:

$$\begin{aligned} S_1 &= \{j \mid j \leq i < \pi_j^{-1}\} \\ S_2 &= \{j \mid \pi_j^{-1} \leq i < j\} \end{aligned}$$

Note that, as π is a permutation, there must be as many lines in the matching diagram with their upper point left of i and their lower point right of i, as lines with their upper point right of i and their lower point left of i. Hence $|S_1| = |S_2|$. Every vertex in S_1 is adjacent to every vertex in S_2, hence the subgraph induced by $S_1 \cup S_2$ contains a complete bipartite subgraph $K(m, m)$, with $m = |S_1|$. By lemma 3.2, this implies that $k \geq m$. Hence $X_i| \leq |S_1| + |S_2| + 1 \leq 2k + 1$. □

Notice that, by lemma 3.3 the algorithm can be implemented to run in $O(nk)$ time, since at each step one new element is put into a subset. Hence we have proved the following theorem:

Theorem 3.1 *If $G[\pi]$ is a permutation graph with treewidth at most k, then the pathwidth of $G[\pi]$ is at most $2k$, and the $O(nk)$ time algorithm of figure 2 produces a path-decomposition with width at most $2k$.*

In [9], an algorithm is decribed which, given a tree-decomposition of a graph G, with width bounded by some constant, produces a tree-decomposition with width equal to the treewidth of G, in linear time. Also, for every constant c, a linear time algorithm is given, which produces a path-decomposition with width equal to the pathwidth of G, for graphs G with pathwidth at most c. Using these results we obtain the following theorem:

Theorem 3.2 *Let k be some constant. If $G[\pi]$ is a permutation graph with treewidth at most k, then there exists a linear time algorithm which produces a tree-decomposition with width equal to the treewidth of $G[\pi]$. Also, a linear time algorithm exists which produces a path-decomposition with width equal to the pathwidth of $G[\pi]$.*

4 Conclusions

In this paper we described a very simple and efficient algorithm which produces an approximate path-decomposition for permutation graphs which is at most a factor two off the optimal one. There are classes of perfect graphs for which the treewidth and pathwidth can be computed efficiently. For example cographs [8] and interval graphs. The treewidth can also be computed efficiently for chordal graphs and circular arc graphs [23]. It would be of interest to know if there is a fast algorithm which computes the treewidth of permutation graphs. Another interesting conclusion from this paper is that the pathwidth and treewidth of a permutation graph differ at most by a factor two. It would be of interest to know for which other classes of graphs the pathwidth and treewidth differ by a constant factor. For cographs and interval graphs the treewidth and pathwidth are equal ([8]).

5 Acknowledgements

We like to thank D. Seese and B. Reed for valuable discussions.

References

[1] S. Arnborg, Efficient algorithms for combinatorial problems on graphs with bounded decomposability — A survey. *BIT*, **25**, 2 – 23, 1985.

[2] S. Arnborg, D.G. Corneil and A. Proskurowski, Complexity of finding embeddings in a k-tree, *SIAM J. Alg. Disc. Meth.*, **8**, 277 – 284, 1987.

[3] S. Arnborg, J. Lagergren and D. Seese, Easy problems for tree-decomposable graphs, *J. Algorithms*, **12**, 308 – 340, 1991.

[4] S. Arnborg and A. Proskurowski, Characterization and recognition of partial 3-trees, *SIAM J. Alg. Disc. Meth.*, **7**, 305 – 314, 1986.

[5] S. Arnborg and A. Proskurowski, Linear time algorithms for NP-hard problems restricted to partial k-trees. *Disc. Appl. Math.*, **23**, 11 – 24, 1989.

[6] H.L. Bodlaender, A tourist guide through treewidth, Technical report RUU-CS-92-12, Department of computer science, Utrecht University, Utrecht, The Netherlands, 1992. To appear in: *Proceedings 7th International Meeting of Young Computer Scientists*, Springer Verlag Lecture Notes in Computer Science.

[7] H.L. Bodlaender, Dynamic programming algorithms on graphs with bounded treewidth, *Proceedings of the 15th International colloquium on Automata, Languages and Programming*, 105 – 119, Springer Verlag, Lecture Notes in Computer Science, vol. 317, 1988.

[8] H. Bodlaender and R.H. Möhring, The pathwidth and treewidth of cographs, In *Proceedings 2nd Scandinavian Workshop on Algorithm Theory*, 301 – 309, Springer Verlag Lecture Notes in Computer Science vol. 447, 1990.

[9] H. Bodlaender and T. Kloks, Better algorithms for the pathwidth and treewidth of graphs, *Proceedings of the 18th International colloquium on Automata, Languages and Programming*, 544 – 555, Springer Verlag, Lecture Notes in Computer Science, vol. 510, 1991.

[10] A. Brandstädt and D. Kratsch, On the restriction of some NP-complete graph problems to permutation graphs, *Fundamentals of Computation Theory, proc. FCT* 1985, 53 – 62, Lecture Notes in Comp. Science vol. 199, Springer Verlag New York, 1985.

[11] A. Brandstädt and D. Kratsch, On domination problems for permutation and other perfect graphs, *Theor. Comput. Sci.* **54**, 181 – 198, 1987.

[12] S. Even, A. Pnueli and A. Lempel, Permutation graphs and transitive graphs, *J. Assoc. Comput. Mach.* **19**, 400 – 410, 1972.

[13] M. Farber and M. Keil, Domination in permutation graphs, *J. Algorithms* **6**, 309 – 321, 1985.

[14] M.C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.

[15] J. Lagergren and S. Arnborg, Finding minimal forbidden minors using a finite congruence, *Proceedings of the* 18*th International colloquium on Automata, Languages and Programming*, 532–543, Springer Verlag, Lecture Notes in Computer Science, vol. 510, 1991.

[16] J. van Leeuwen, Graph algorithms. In *Handbook of Theoretical Computer Science, A: Algorithms an Complexity Theory*, 527–631, Amsterdam, 1990. North Holland Publ. Comp.

[17] J. Matoušek and R. Thomas, Algorithms Finding Tree-Decompositions of Graphs, *Journal of Algorithms* **12**, 1 – 22, 1991.

[18] A. Pnueli, A. Lempel, and S. Even, Transitive orientation of graphs and identification of permutation graphs, *Canad. J. Math.* **23**, 160 – 175 1971.

[19] B. Reed, Finding approximate separators and computing treewidth quickly, To appear in: Proceedings STOC'92, 1992.

[20] N. Robertson and P.D. Seymour, Graph minors—A survey. In I. Anderson, editor, *Surveys in Combinatorics*, 153 – 171. Camebridge Univ. Press 1985.

[21] P. Scheffler, Linear time algorithms for NP-complete problems restricted to partial k-trees, Report R-MATH-03/87, Karl-Weierstrass-Institut Für Mathematik, Berlin, GDR 1987.

[22] J. Spinrad, On comparability and permutation graphs, *SIAM J. Comp.* **14**, No. 3, August 1985.

[23] R. Sundaram, K. Sher Singh and C. Pandu Rangan, Treewidth of circular arc graphs, Manuscript 1991.