Necessary edges in k-chordalizations of graphs

Hans L. Bodlaender*

Abstract

In this note, we look at which edges must always be added to a given graph $G = (V, E)$, when we want to make it a chordal graph with maximum clique size at most k by adding edges. lThis problem has a strong relation to the (algorithmic) theory of the treewidth of graphs. If $\{x, y\}$ is an edge in every chordal supergraph of G with maximum clique size at most k, we call the pair necessary for treewidth k. Some sufficient, or necessary and sufficient conditions are given for pairs of vertices to be necessary for treewidth k. For a fixed k, the set of all edges that always must be added when making the graph chordal with maximum clique size $\leq k$, can be found in linear time. If k is given as part of the input, then this problem is coNP-hard. A few similar results are given when interval graphs (and hence pathwidth) are used instead of chordal graphs and treewidth.

1 Introduction

Graphs with bounded treewidth play an important role in several recent investigations in algorithmic graph theory. In this note, we look at a specific problem, dealing with this type of graphs. We ask ourselves the question: which edges may we add to a given graph $G = (V, E)$, such that every tree decomposition of G of width at most k, still is a tree decomposition of the graph obtained by adding these edges to G? An equivalent way to state this problem is the following: which edges must be added always, when we want to make G a chordal graph with maximum clique size at most $k+1$ by adding edges?

This notion generalizes two algorithmic ideas, used in earlier papers on graphs with bounded treewidth. In [3] a subset of the set of all necessary edges is identified, and then added to G, and then as an intermediate step

*Department of Computer Science, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, the Netherlands. Email: hansb@cs.uu.nl

in a linear time algorithm to recognize graphs with treewidth at most k, for fixed k. The first step of the algorithm in [7], (for determining whether a given 3-colored graph is contained in a properly colored chordal graph) is to find all necessary edges for treewidth 2.

The original motivation for this research was for the design of algorithms for the problems to determine whether a given colored graph is contained in a properly colored chordal or interval graph, for small number of colors, larger than three. While the complexity of these problems is now more or less well understood (see e.g., [11, 6, 5]), the result of this paper may be perhaps useful as a step for solving special cases.

2 Definitions

The graphs in this paper are considered to be simple and undirected. We say that a set S separates vertices x and y in a graph G, if every path from x to y in G uses a vertex in S. The subgraph of $G = (V, E)$ induced by vertex set $W \subseteq V$ is denoted by $G[W] = (W, \{\{v, w\} \in E \mid v, w \in W\})$. For a graph $G = (V, E)$ and a set $S \subseteq V$, we denote $G - S = G[V - S]$, and $G + clique(S)$ as the graph, obtained from G by making all vertices in S adjacent.

The notion of treewidth has been introduced by Robertson and Seymour [12].

Definition 1 *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 collection of subsets of V, and $T = (I, F)$ a tree, such that*

- $\bigcup_{i \in I} X_i = V$
- *for all edges $\{v, w\} \in E$ there is an $i \in I$ with $v, w \in X_i$*
- *for all $i, j, k \in I$: if j is on the path from i to k in T, then $X_i \cap X_k \subseteq X_j$.*

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

A tree decomposition $(\{X_i \mid i \in I\}, T = (I, F))$ is called a path decomposition, if T is a path. The pathwidth of a graph G is the minimum width over all path decompositions of G.

The problems, whether a given graph G has treewidth or pathwidth at most a given integer k are NP-complete [1].

A graph $G = (V, E)$ is a chordal graph, if it does not contain an induced cycle of length at least four. We say a graph H is a *chordalization* of graph G, if H contains G as a subgraph, and H is chordal. H is said to be a k-chordalization of G, if H is a chordalization of G, and the maximum clique size in H is at most k. The following results are well known. (See e.g., [4].)

Lemma 1 *(i) For every chordal graph $G = (V, E)$, there exists a tree decomposition $(\{X_i \mid i \in I\}, T = (I, F))$ of G, such that every set X_i forms a clique in G, and for every maximal clique $W \subseteq V$, there exists an $i \in I$ with $W = X_i$.*
(ii) For every interval graph $G = (V, E)$, there exists a path decomposition $(\{X_i \mid i \in I\}, T = (I, F))$ of G, such that every set X_i forms a clique in G, and for every maximal clique $W \subseteq V$, there exists an $i \in I$ with $W = X_i$.
(iii) Let $(\{X_i \mid i \in I\}, T = (I, F))$ be a tree decomposition of G of width at most k. The graph $H = (V, E \cup E')$, with $E' = \{\{v, w\} \mid \exists i \in I : v, w \in X_i\}$, obtained by making every set X_i a clique, is chordal, and has maximum clique size at most $k + 1$.
(iv) Let $(\{X_i \mid i \in I\}, T = (I, F))$ be a path decomposition of G of width at most k. The graph $H = (V, E \cup E')$, with $E' = \{\{v, w\} \mid \exists i \in I : v, w \in X_i\}$, obtained by making every set X_i a clique, is an interval graph, and has maximum clique size at most $k + 1$.
(v) Let $(\{X_i \mid i \in I\}, T = (I, F))$ be a tree decomposition of G, and let $W \subseteq V$ form a clique in G. Then there exists an $i \in I$ with $W \subseteq X_i$.
(vi) [9] Let $(\{X_i \mid i \in I\}, T = (I, F))$ be a tree decomposition of G. Suppose W_1, W_2 induce a complete bipartite subgraph in G, i.e. for all $v \in W_1$, $w \in W_2$: $\{v, w\} \in E$. Then there exists an $i \in I$ with ($W_1 \subseteq X_i$ and $W_2 \cap X_i \neq \emptyset$) or ($W_2 \subseteq X_i$ and $W_1 \cap X_i \neq \emptyset$).

Definition 2 *Let $G = (V, E)$ be a graph.*
(i) We say a pair $\{x, y\}$ is a necessary edge for treewidth (pathwidth) k *of G, if for every tree decomposition (path decomposition) $(\{X_i \mid i \in I\}, T = (I, F))$ of width at most k, there is an $i \in I$ with $x, y \in X_i$. (ii) We say a pair $\{x, y\}$ is a* necessary edge for a k-chordalization of G*, if for every k-chordalization $H = (V, E')$ of G, $\{x, y\} \in E'$.*

For (vertex disjoint) graphs $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$, we define the product of G_1 and G_2 as $G_1 \times G_2 = (V_1 \cup V_2, E_1 \cup E_2 \cup \{\{v, w\} \mid v \in V_1,\ w \in V_2\})$.

Lemma 2 *[9] Let $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$ be graphs. (i) treewidth$(G_1 \times G_2) = \min($treewidth$(G_1) + |V_2|$, treewidth$(G_2) + |V_1|)$.*

(ii) $pathwidth(G_1 \times G_2) = \min(pathwidth(G_1) + |V_2|, pathwidth(G_2) + |V_1|)$.
(iii) If $treewidth(G_1) + |V_2|) = k < treewidth(G_2) + |V_1|$*, then for all* $v, w \in V_2 : \{v, w\}$ *is necessary for treewidth* k *of* $G_1 \times G_2$.
(iv) If $treewidth(G_1) + |V_2|) = k \leq treewidth(G_2) + |V_1|$*, then for all* $v, w \in V_1 : \{v, w\}$ *is necessary for treewidth* k *of* $G_1 \times G_2$*, if and only if* $\{v, w\}$ *is necessary for treewidth* $treewidth(G_1)$ *of* G_1.
(v) If $pathwidth(G_1) + |V_2|) = k < pathwidth(G_2) + |V_1|$*, then for all* $v, w \in V_2 : \{v, w\}$ *is necessary for pathwidth* k *of* $G_1 \times G_2$.
(vi) If $pathwidth(G_1) + |V_2|) = k \leq pathwidth(G_2) + |V_1|$*, then for all* $v, w \in V_1 : \{v, w\}$ *is necessary for pathwidth* k *of* $G_1 \times G_2$*, if and only if* $\{v, w\}$ *is necessary for pathwidth* $pathwidth(G_1)$ *of* G_1.

Proof: This result follows directly from Lemma 3.4 from [9] and its proof.

□

3 Graph-theoretic results

The following two lemmas can be easily derived.

Lemma 3 *Let* $G = (V, E)$ *be a graph,* $x, y \in V$*. The following two statements are equivalent:*

1. $\{x, y\}$ *is a necessary edge for treewidth* k *of* G.
2. *Every* k*-chordalization of* G *contains the edge* $\{x, y\}$.

Lemma 4 *Let* $G = (V, E)$ *be a graph,* $x, y \in V$*. The following two statements are equivalent:*

1. $\{x, y\}$ *is a necessary edge for pathwidth* k *of* G.
2. *For every interval graph* $H = (V, F)$ *that contains* G *as a subgraph and has maximum clique size* $k + 1$, $\{x, y\} \in F$.

In [3], it has been shown, that every pair $\{x, y\}$ such that x and y have $k + 1$ common neighbors is necessary for treewidth k. In [7], it has been shown that every pair $\{x, y\}$, such that x and y have three vertex disjoint paths between them, is necessary for treewidth 2. The following lemma generalizes this result.

Lemma 5 *Let* $G = (V, E)$ *be a graph,* $x, y \in V$*. Suppose that there are* $k + 1$ *vertex disjoint paths from* x *to* y *in* G*. Then the edge* $\{x, y\}$ *is necessary for treewidth* k *of* G.

Proof: If $\{x,y\} \in E$, then clearly, $\{x,y\}$ is necessary. Suppose $\{x,y\} \notin E$. Let $(X,T) = (\{X_i \mid i \in I\}, T = (I,F))$ be a tree decomposition of G of width at most k. For each of the $k+1$ vertex disjoint paths, identify all inner vertices of that path, both in G, and in the tree decomposition (X,T). The resulting tree decomposition (X',T') is a tree decomposition of the resulting graph G' of width at most k. Note G' contains a 2 by $k+1$ complete bipartite subgraph: each of the vertices representing the inner vertices of one of the paths between x and y is adjacent to x and y. Let W be the set of these vertices, representing the paths. If there exists an $i \in I$ with $\{x,y\} \subseteq X_i$, then we are done. Otherwise, by Lemma 1(vi), there exists an $i \in I$, with $W \subseteq X_i$, and $\{x,y\} \cap X_i \neq \emptyset$. But now $|X_i| \geq k+2$, contradiction. □

It is also possible to state a necessary and sufficient condition that states that an edge is necessary.

Lemma 6 *Let $G=(V,E)$ be a graph of treewidth at most k, and let $x,y \in V$ be non-adjacent vertices. Then $\{x,y\}$ is a necessary edge, if and only if there does not exist a set $S \subseteq V$ with $|S| \leq k+1$, $\{x,y\} \cap S = \emptyset$, S separates x and y in G, and for every connected component W of $G-S$, the graph $G[W \cup S] + clique(S)$ has treewidth at most k.*

Proof: First, suppose we have a set S fulfilling the conditions of the lemma. We now can build a tree decomposition of G in the following way: build for each connected component W of $G-S$ a tree decomposition of width at most k of $G[W \cup S] + clique(S)$. Then, take the disjoint union of these tree decompositions, add one additional node i_0 with $X_{i_0} = S$, and for each component W, take a node i_W from its tree decomposition with $S \subseteq X_{i_W}$ (such a node must exist by Lemma 1(v)) and make it adjacent to i_0. We now have a tree decomposition of G of width at most k, and there is no set i with $x,y \in X_i$, i.e., $\{x,y\}$ is not necessary. (This construction is the same as the one, used in the algorithm of Arnborg, Corneil, and Proskurowski to recognize graphs of treewidth k [1].)

Now, suppose $\{x,y\}$ is not necessary. Take a tree decomposition $(\{X_i \mid i \in I\}, T = (I,F)$ of G of width at most k, with no set i with $x,y \in X_i$. Suppose $x \in X_{i_0}$, $y \in X_{i_1}$. Let i_2 be the last node on the path from i_0 to i_1 in T with $x \in X_{i_2}$, and let i_3 be the next node on this path. Note that $x \notin X_{i_3}$, and $y \notin X_{i_2}$. Let $S = X_{i_2} \cap X_{i_3}$. We have $\{x,y\} \cap S = \emptyset$. Moreover, S separates x and y, by the properties of tree decompositions.

We can make a new tree decomposition of G, by subdividing the edge in T between i_2 and i_3. Let i_4 be the new node, and take $X_{i_4} = S$. This is a tree decomposition of $G + clique(S)$ of width at most k. Every graph

$G[W \cup S] + clique(S)$ is a subgraph of $G + clique(S)$, hence has treewidth at most k. □

4 On finding necessary edges

In this section, we consider the problem, given a graph $G = (V, E)$ and an integer k, to find the set of necessary edges. First, we show, that if k is variable, then this problem is coNP-hard.

Theorem 7 *(i) The following problem is coNP-complete: Given an integer k, a graph $G = (V, E)$ of treewidth at most k, and two non-adjacent vertices $x, y \in V$, is $\{x, y\}$ a necessary edge for treewidth k of G?*
(ii) The following problem is coNP-complete: Given an integer k, a graph $G = (V, E)$ of pathwidth at most k, and two non-adjacent vertices $x, y \in V$, is $\{x, y\}$ a necessary edge for pathwidth k of G?

Proof: (i) It is easy to see that this problem is in coNP: try to guess a k-chordalization of G which does not use edge $\{x, y\}$ (the maximum clique size of a chordal graph can be computed in polynomial time.) To prove coNP-hardness, we transform from TREEWIDTH. Let a graph $G = (V, E)$ be given, and an integer $k \leq |V|$. Write $n = |V|$. Let $G' = (\{v_1, \ldots, v_{n-k}\}, \emptyset)$ be a graph with $n - k$ vertices and no edges, vertex disjoint from G. Let $H = G' \times G$. One easily observes that the treewidth (and the pathwidth) of H is at most n.

Now we claim that the edge $\{v_1, v_2\}$ is necessary for treewidth n of H, if and only if the treewidth of G is larger than k. If the treewidth of G is at most k, then the edge $\{v_1, v_2\}$ is not necessary for treewidth n: note that the treewidth of G' is 0 and $\{v_1, v_2\}$ is not necessary for treewidth 0 of G', so by Lemma 2(iv), $\{v_1, v_2\}$ is not necessary for treewidth n of H. If the treewidth of G is larger than k, then by Lemma 2(iii), $\{v_1, v_2\}$ is necessary for treewidth n of H.

So we have a polynomial time transformation from the complement of treewidth to the problem, stated in the theorem. Hence, the latter is coNP-hard.

(ii) Similar to (i). Use the NP-completeness of the PATHWIDTH problem. □

For fixed k, one can find the set of all necessary edges for treewidth k or pathwidth k in linear time. There are two different ways to show the result in the case of treewidth. One is to exploit the fact that the condition from

Lemma 6 can be expressed in Monadic Second Order Logic, and use general results on the solvability of such problems on graphs of bounded treewidth (see e.g., [2, 10]). We do not give the details of this approach, but instead discuss how linear time algorithms for building path or tree decompositions of bounded width can be modified for this problem. The presentation of this result is here not self-contained. Instead, we rely on results and techniques from the (lengthy) paper [8].

The first step of the algorithm is to make a tree decomposition $(\{X_i \mid i \in I\}, T = (I, F))$ of the input graph $G = (V, E)$ of treewidth $\leq k$. This can be done in linear time [3]. (If the treewidth of G is larger than k, then we can stop directly.) We may assume that this tree decomposition is *nice*, in the sense of [8].

Clearly, all edges in E are necessary. A necessary condition for a pair $\{x, y\}$ to be necessary for treewidth k of G is that there exists an $i \in I$, with $x, y \in X_i$.

The algorithm from [8] uses dynamic programming to compute for every node a table - this table represents a finite set of characterizations of possible tree decompositions of width at most k of a subgraph associated with the node. The table of the root represents such characterizations of tree decompositions of width at most k of G. Now, it is straightforward to see from the algorithm from [8], that when we have the table of the root node r of T, then we can check directly (in $O(1)$ time) for every pair $x, y \in X_r$ if $\{x, y\}$ is necessary.

Thus, checking which pairs are necessary can be done quickly when we would have for every node $i \in I$ the table that that node would have, when it would have been the root of T. Call this set the *full root set* of i.

We can compute the full root sets of all nodes $i \in I$ in linear time. This can be done as follows. First, we run the standard algorithm from [8]: every node i has its full set. (This set basically can be associated with the subtree with root i, when T is viewed as a rooted tree with root r.)

Then, for each edge $\{i, j\}$ in T, let $I_{i,j}$ be the set of nodes in T, whose path in T to i passes through j. (If i is the father of j, then $I_{i,j}$ is the set containing j and all descendants of j.) Write $V_{i,j} = \bigcup_{i' \in I_{i,j}} X_i$, and $G_{i,j} = G[V_{i,j}]$.

For each edge $\{i, j\}$ in T, with i the father of j, we now want to compute the characteristic rooted at i, when we make j the father of i. In other words, the graph to which this full set corresponds is the graph $G_{j,i}$; in the previous round of the algorithm, we already have computed the full set corresponding to $G_{i,j}$. Computing these full sets can be done with similar procedures as in [7], but now we process nodes in a top-down order. When we have all these full sets, then we can compute for each node $i \in I$, the full root set of i. (It can be computed, by using the full sets of for the (at most 3) edges, adjacent to i.)

To test whether an edge $\{x, y\}$ is necessary, look at one node i with $x, y \in X_i$. If there is a characteristic in the root full set of i, where x and y are non-adjacent in the trunk and do not have overlapping intervals in the interval models, then $\{x, y\}$ is not necessary; otherwise, $\{x, y\}$ is necessary for treewidth k of G. (See [8].) The total time of this computation is linear.

The same type of computation can be used to compute all necessary edges for pathwidth k.

Theorem 8 *(i) For all k, there exists a linear time algorithm, that, when given a graph $G = (V, E)$, finds all necessary edges for treewidth k of G.*
(ii) For all k, there exists a linear time algorithm, that, when given a graph $G = (V, E)$, finds all necessary edges for pathwidth k of G.

5 Conclusions

The original motivation for this research was in the problem in chordalizing or intervalizing colored graphs. In these problems, we have a colored graph, and ask whether it is a subgraph of a properly colored chordal graph (TRIANGULATING COLORED GRAPHS) or interval graph (INTERVALIZING COLORED GRAPHS). The treewidth of yes-instances of these problems is necessarily bounded by the number of colors used minus 1. Perhaps algorithms that find necessary edges for treewidth k or pathwidth k can be of use for algorithms that solve TRIANGULATING COLORED GRAPHS or INTERVALIZING COLORED GRAPHS faster in some special cases. Unfortunately, the linear time algorithms that are presented here have large constant factors, making them impractical in their stated form.

References

[1] 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.

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

[3] H. L. Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25:1305–1317, 1996.

[4] H. L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theor. Comp. Sc.*, 209:1–45, 1998.

[5] H. L. Bodlaender and B. de Fluiter. On intervalizing k-colored graphs for DNA physical mapping. *Disc. Appl. Math.*, 71:55–77, 1996.

[6] H. L. Bodlaender, M. R. Fellows, M. T. Hallett, H. T. Wareham, and T. J. Warnow. The hardness of perfect phylogeny, feasible register assignment and other problems on thin colored graphs. *Theor. Comp. Sc.*, 244:167–188, 2000.

[7] H. L. Bodlaender and T. Kloks. A simple linear time algorithm for triangulating three-colored graphs. *J. Algorithms*, 15:160–172, 1993.

[8] H. L. Bodlaender and T. Kloks. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *J. Algorithms*, 21:358–402, 1996.

[9] H. L. Bodlaender and R. H. Möhring. The pathwidth and treewidth of cographs. *SIAM J. Disc. Math.*, 6:181–188, 1993.

[10] B. Courcelle and M. Mosbah. Monadic second-order evaluations on tree-decomposable graphs. *Theor. Comp. Sc.*, 109:49–82, 1993.

[11] H. Kaplan, R. Shamir, and R. E. Tarjan. Tractability of parameterized completion problems on chordal and interval graphs. *Found. Comput. Ser.*, pages 780 – 791, 1994.

[12] N. Robertson and P. D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *J. Algorithms*, 7:309–322, 1986.