Laboratoire Bordelais de Recherche en Informatique
UMR 5800 - Université Bordeaux I, 351, cours de la Libération,
33405 Talence CEDEX, France

Research Report `RR-1326-04`

Small Diameter Bag Tree-Decompositions

by Yon Dourisbourre and Cyril Gavoille

May, 2004

Small Diameter Bag Tree-Decompositions

Yon Dourisboure* Cyril Gavoille*

27th May 2004

Abstract

This paper deals with the *length* of a Robertson-Seymour's tree-decomposition. The *tree-length* of a graph denotes the largest distance between two vertices of a bag of a tree-decomposition, minimized over all tree-decompositions of the graph. The study of this invariant may be interesting in its own right because the class of bounded tree-length graphs includes (but is not reduced to) bounded chordality graphs (like interval graphs, permutation graphs, AT-free graphs, etc.). For instance, we show that the tree-length of any outerplanar graph is $\lceil k/3 \rceil$, where k is the chordality of the graph, and we compute the tree-length of meshes.

More fundamentally we show that any algorithm computing a tree-decomposition approximating the tree-width (or the tree-length) of an n-vertex graph by a factor α or less does not give an α-approximation of the tree-length (resp. the tree-width) unless if $\alpha = \Omega(n^{1/5})$. We complete these results presenting several polynomial time constant approximate algorithms for the tree-length.

The introduction of this parameter is motivated by the design of compact distance labeling, compact routing tables with near-optimal route length, and by the construction of sparse additive spanners.

Keywords: tree-decomposition, tree-width, tree-length, chordality, small separator

1 Introduction

The notions of tree-decomposition and of tree-width are very rich concepts with many algorithmic implications. Many NP-complete problems are polynomial for graphs of bounded tree-width [Arn85, Bod96]. In addition, the notion of tree-width is in the kernel of the graph minor theory [Die00]. Intuitively tree-width can be seen as a measure of "how

*University of Bordeaux, LaBRI, {*Yon.Dourisboure,gavoille*}*@labri.fr*. Both authors are supported by the Actions Spécifiques CNRS "Dynamo" and "Algorithmique des grands graphes", by the project "PairA-Pair" of the ACI Masses de Données, and by the European Research Training Network COMBSTRU-HPRN-CT-2002-00278.

far" a graph is from a tree. The smaller the tree-width, the more tree-like the graph is. Determining the tree-width of a graph is NP-hard [ACP87]. However, Bodlaender [Bod96] gave a linear time algorithm for recognizing graphs of bounded tree-width. There are also $O(\log n)$-approximation algorithms for computing the tree-width of an arbitrary graph [BGHK95, Ree92], and even $O(\log k)$-approximation algorithms where k is the tree-width [Ami01, BKMT01].

However many hard problems have still polynomial solutions on large class of graphs even when the size of the separator is large (or large tree-width), like interval graphs. For instance, MINIMUM DOMINATING SET falls into this class [BJ82, BG88, Bra87, GR88]. Actually, the "shape" of the separator plays an important role. Introducing *well-separated* class of graphs Katz, Katz and Peleg [KKP00] showed that the graphs of this class enjoy distance labeling with short labels, namely the property that all the vertices of the graph can be labeled with $O(\log^2 n)$ bit binary strings so that the distance between two vertices can be retrieved from the two vertex labels, without any other source of information [Pel99]. They showed that interval and permutation graphs fall into that class (actually $O(\log n)$ bit labels suffice for interval graphs and more generally for circular-arc graphs [GP03]). The same phenomenon occurs for bounded clique-width graphs, for which Courcelle and Vanicat [CV03] showed that $O(\log^2 n)$ bit distance labeling scheme exists, whereas the tree-width of these graphs is unbounded in general. Another result in this field demonstrates that k-chordal graphs (i.e., the graphs with no induced cycle of length greater than k) support an approximate distance labeling scheme with an additive one-sided error of $\lfloor k/2 \rfloor$ and with $O(\log^2 n)$ bit labels [GKK$^+$01]. The scheme mainly based on the construction of a Robertson-Seymour's tree-decomposition of k-chordal graphs with small diameter bags (a bag is a subset of vertices induced by the vertex of a tree-decomposition, cf. Definition 1). Namely, in this tree-decomposition any two vertices of a bag are at distance at most $k/2$.

An application different from distance computation which is also related to global structure of the graph is the field of compact routing. Informally speaking, the routing problem can be presented as requiring us to assign two kinds of labels to every vertex of a graph. The first is the *address* of the vertex, whereas the second label is a data structure called the *local routing table*. The labels are assigned in such a way that at every source vertex v and given the address of any destination vertex u, one can decide the output port of an outgoing edge of v that leads to u. The decision must be taken locally in v, based solely on the two labels of v and with the address label of u. Many references can be found in [GP01, Gav01].

Tree-width k graphs have shortest-path routing scheme with addresses and routing tables of $O(k \log^2 n)$ bits per vertices [Pel00b]. However, when the diameter of the separator is small better results exist. Chordal graphs (a class including interval graphs, and whose separators are all cliques) support a routing scheme using addresses of $2 \log n$ bits and routing tables of $O(k \log n)$ bits, where k is the size of the maximum clique [Dou02]. This scheme guarantees an additive one-sided error of 2 on the route length. With the same error but with larger addresses, [DG02] proposed a scheme independent of separator size. Addresses and routing tables are of $O(\log^3 n / \log \log n)$ bits per vertex. Decreasing the error or reducing the memory requirements is still an open problem even for this class of graphs. In a forthcoming paper the last routing scheme is extended in case where separators are not necessarily cliques. The result is that even if the tree-width of a graph is unbounded, when the diameter of the separators is bounded by a constant, say δ, then the graph enjoy a routing scheme which guarantees an additive one-sided error of 2δ on the route length with addresses of size $O(\log^3 n/\log\log n)$ bits per vertex.

The same phenomenon occurs when one is interested in construction of sparse spanners. Given a graph, a spanner of it, is a subgraph with the same set of vertices and a subset of the set of edges. The main objective is to construct a spanner with few edges and in which the distance between every pair of vertices is close to their distance in the original graph: at most s times their distance plus r. In case where $s = 1$, the spanner is also called an additive r-spanner, and in case where $r = 0$ it is called a multiplicative s-spanner. There are many applications of spanners, for example, the complexity of a lot of distributed algorithms depends on the number of messages, itself depending on the number of edges [PU89a, Pel00a]. Sparse spanners occur also in the efficiency, of compact routing schemes [PU89b]. Unfortunately, given an arbitrary graph G and three integers s, r and m, it is NP-complete [PS89] to determine whether G admits a spanner with m or fewer edges, and in which the multiplicative error is s and the additive error is r (see also [BDLL04, BH98, CC95, DK99, FK01] for complexity issue).

However, when a graph G admits a tree-decomposition with small diameter separators, then it admits also good sparse spanners. For instance any chordal graph contains a multiplicative 3-spanner with $O(n \log n)$ edges [PS89]. Here again, the result can be generalised in case where the diameter of every separator is bounded by a constant δ. In this class of graph, [DDGY04] showed an additive 2δ-spanner with $O(n \log n)$ edges and also an additive 4δ-spanner with $O(n)$ edges.

Finally, we observe that in the field of graph visualization, one could taken advantage to represent a graph as a tree – there are efficient and convenient tree drawings [DBETT99] – in which the nodes of the tree can represent a set of vertices in the graph that are close from each other.

1.1 Our results

Let us define the *tree-length* of a graph as the largest distance between two vertices of a bag of a tree-decomposition, minimized over all Robertson-Seymour's tree-decompositions of the graph. In Section 2 we calculate the tree-length of outerplanar graphs, and of meshes. We also show in this section that any algorithm computing a tree-decomposition approximating the tree-width (or the tree-length) of a graph by a factor α or less does not give an α-approximation of the tree-length (resp. the tree-width) unless if $\alpha = \Omega(n^{1/5})$. More precisely we construct an n-vertex graph of tree-length δ and of tree-width k for which all its tree-decompositions are either of width $\Omega(kn^{1/5})$ or of length $\Omega(\delta n^{1/5})$. Finally, we consider in Section 2 the length of some specific tree-decompositions, namely trees of bounded depth, and paths.

Although in this paper we did not determine the computational complexity of the tree-length of an arbitrary graph, we present in Section 3 constant approximate algorithms. An algorithm computes in $O(nm)$ time a tree-decomposition of length $\lfloor k/2 \rfloor$ of a graph with n vertices, m edges, and chordality k. The second algorithm computes in time $O(n+m)$ a tree-decomposition of length $\lfloor k/2 \rfloor + 3$. We prove that these two algorithms are 3-approximations, and we present a polynomial time heuristic for which we conjecture that it is a 2-approximation.

In Section 4 we conclude by a list of open problems.

1.2 Definition

We need the notion of *tree-decomposition* used by Robertson and Seymour in their work on graph minors [RS86].

Definition 1 *A tree-decomposition of a graph G is a tree T whose vertices, called* bags*, are subsets of $V(G)$ such that:*

1. $\bigcup_{X \in V(T)} X = V(G)$;
2. *for all $\{u,v\} \in E(G)$, there exists $X \in V(T)$ such that $u, v \in X$; and*
3. *for all $X, Y, Z \in V(T)$, if Y is on the path from X to Z in T then $X \cap Z \subseteq Y$.*

It is not difficult to see that Rule 3 of Definition 1 is equivalent to say that for every vertex $x \in V(G)$, the set of bags containing x induces a subtree of T. A tree-decomposition is *reduced* if no bags is contained in another one. A leaf of such decomposition contains necessarily a vertex contained in none other bags. Thus by induction the number of bags of a reduced tree-decomposition does not exceed $n-1$ for an n-vertex connected graph (cf. [Bod98]).

For every induced subgraph H of G, the *diameter of H in G* is $\mathrm{diam}_G(H) = \max_{x,y \in V(H)} d_G(x,y)$, where $d_G(x,y)$ denotes the distance in G between x and y. $\mathrm{diam}(G)$ denotes $\mathrm{diam}_G(G)$. Observe that $\mathrm{diam}_G(H)$ can be constant even if H is not connected. By extension, for every subset $X \subseteq V(G)$, $\mathrm{diam}_G(X) = \mathrm{diam}_G(G[X])$ where $G[X]$ denotes the subgraph of G induced by X.

Let T be a tree-decomposition of a graph G. The *width of T* is $\mathrm{width}(T) = \max_{X \in V(T)} |X| - 1$. The *length of T* is $\mathrm{length}(T) = \max_{X \in V(T)} \mathrm{diam}_G(X)$. The *tree-width* and the *tree-length of G*, denoted by $\mathrm{tw}(G)$ and by $\mathrm{tl}(G)$, are respectively $\min_T \mathrm{width}(T)$ and $\min_T \mathrm{length}(T)$, where the minimum is taken over all tree-decompositions of G.

A graph is k-chordal if the length of the longest induced cycle is at most k. The *chordality of G* is the smallest integer k such that G is k-chordal. Trees are, by convention, of chordality 2. Chordal graphs are 3-chordal graphs. Graphs of tree-length 1 are exactly chordal graphs, since a graph is chordal if and only if it has a tree-decomposition whose bags are cliques.

The recognition problem for k-chordalness (that is the problem to decide whether a graph is not k-chordal) is coNP-complete for $k = \Theta(n^\epsilon)$ for any constant $\epsilon > 0$. (It comes from a reduction to Hamiltonian cycle problem in cubic graphs, cf. [Ueh99].) Relationship between tree-width, chordality, and degeneracy has been study in [BT97], and the tree-width expressed as a function of the diameter has been considered in [Epp00].

2 Optimal Length Tree-Decomposition

2.1 Preliminary Results

A subgraph H of G is *isometric* if $d_H(x,y) = d_G(x,y)$, for all $x,y \in V(H)$. It is a natural generalization of induced subgraph (any isometric subgraph is clearly an induced subgraph). In particular, in the case H is an isometric cycle of length k of G, then k must be at least the girth of G and at most the chordality of G. Note that every subgraph of a distance-hereditary graph is by definition an isometric subgraph. Isometric subgraph is a tool for tree-length as minor is for tree-width.

Lemma 1 *The tree-length of any isometric subgraph of G is no more than the tree-length of G.*

Proof. Let δ be the tree-length of G, and let T be a tree-decomposition of G of length δ, and let H be an isometric subgraph of G. Remove from T all the bags with empty intersection with H, and remove from the remaining bags all the vertices from $G \setminus H$. It forms a forest T'. Since H is connected, T' is connected (a subtree of T), and thus a tree-decomposition of H. For every $B' \in V(T')$ and for all $x,y \in B'$, $d_H(x,y) = d_G(x,y)$, and $d_G(x,y) \leqslant \delta$ since B' is a subset of a bag $B \in V(T)$. It follows that $\mathrm{length}(T') \leqslant \delta$, completing the proof. $\square$

A *separating set* of a connected graph G is subset of vertices S such that $G \setminus S$ is disconnected, i.e., $G \setminus S$ is composed of two or more connected components. A *clique-separator decomposition* is a tree-decomposition such that the intersection of every two adjacent bags induced a clique, and such that no bag has a clique separating. E.g., a decomposition into biconnected components yields a clique-separator decomposition (the separators are cutvertices, cliques of size one). A clique-separator decomposition of a graph G can be computed in $O(nm)$ time [Tar85, Whi81], where $n = |V(G)|$ and $m = |E(G)|$.

Lemma 2 *For every clique-separator decomposition T of G,*

$$\mathrm{tw}(G) = \max_{X \in V(T)} \mathrm{tw}(G[X]) \quad \textit{and} \quad \mathrm{tl}(G) = \max_{X \in V(T)} \mathrm{tl}(G[X]) \ .$$

Proof. We have $\text{tw}(G) \geqslant \max_{X \in V(T)} \text{tw}(G[X])$ because $G[X]$ is a minor of G. Moreover $G[X]$ is an isometric subgraph of G (no shortest path between $u, v \in X$ can use a vertex $z \notin X$). Thus by Lemma 1, $\text{tl}(G) \geqslant \text{tl}(G[X])$ for every $X \in V(T)$, and so $\text{tl}(G) \geqslant \max_{X \in V(T)} \text{tl}(G[X])$.

We now prove the upper bound for the tree-length (for the tree-width, the proof is similar). Let T_X be a tree-decomposition of $G[X]$ of length $\text{tl}(G[X])$. Note that since $G[X]$ is an isometric subgraph of G, $G[X]$ is connected so T_X is correctly defined. Let Y_i be the ith neighbor of the bag X in T. Since $X \cap Y_i$ induces a clique, there must exist a bag of T_X, say Z_i, containing $X \cap Y_i$. We now construct a tree T' from T as follows: for every bag X of T, we first remove X from T, and then we connect T_X to $T \setminus \{X\}$ by adding an edge between Y_i and Z_i for every i. We check that T' is a tree-decomposition of G, and that its length is $\text{length}(T') = \max_{X \in V(T)} \text{length}(T_X)$. Since $\text{length}(T_X) = \text{tl}(G[X])$, we have $\text{length}(T') = \max_{X \in V(T)} \text{tl}(G[X])$. Since $\text{tl}(G) \leqslant \text{length}(T')$, we have proved that $\text{tl}(G) = \max_{X \in V(T)} \text{tl}(G[X])$ and similarly that $\text{tw}(G) = \max_{X \in V(T)} \text{tw}(G[X])$. □

So, whenever we are looking for an optimal tree-decomposition of G (w.r.t. its tree-width or its tree-length), one can restrict our attention to maximal subgraph of G without clique separating.

2.2 Outerplanar Graphs

We shall prove now that the tree-length of a cycle of length k is $\lceil k/3 \rceil$, and thus that every graph of girth g is of tree-length at least $g/3$. More generally, we have:

Theorem 1 *Every outerplanar graph has tree-length $\lceil k/3 \rceil$, where k is the chordality of the graph. Moreover, a tree-decomposition of width 2 and of length $\lceil k/3 \rceil$ can be constructed in linear time.*

To prove that, we first show that:

Lemma 3 *The cycle of length k has a tree-decomposition of length $\lceil k/3 \rceil$, and of width 2, computable in $O(k)$ time.*

Proof. Let C_k be a cycle of length k. C_k can be split into three paths P_{xy}, P_{yz}, P_{zx} of length at most $\lceil k/3 \rceil$ (cf. Fig. 1(a)).

Then it is easy to construct a tree-decomposition of C_k of length and width $\lceil k/3 \rceil$: a star whose the center is a bag composed of vertices x, y, z, and whose the leaves correspond to the three paths (cf. Fig. 1(a)).

A path can be decomposed into a tree-decomposition of width 2 such that the two extremities belongs to a same bag as shown on Fig. 1(b). So, in the previous tree-decomposition, we can extend the three leaves into three paths in order to obtain a

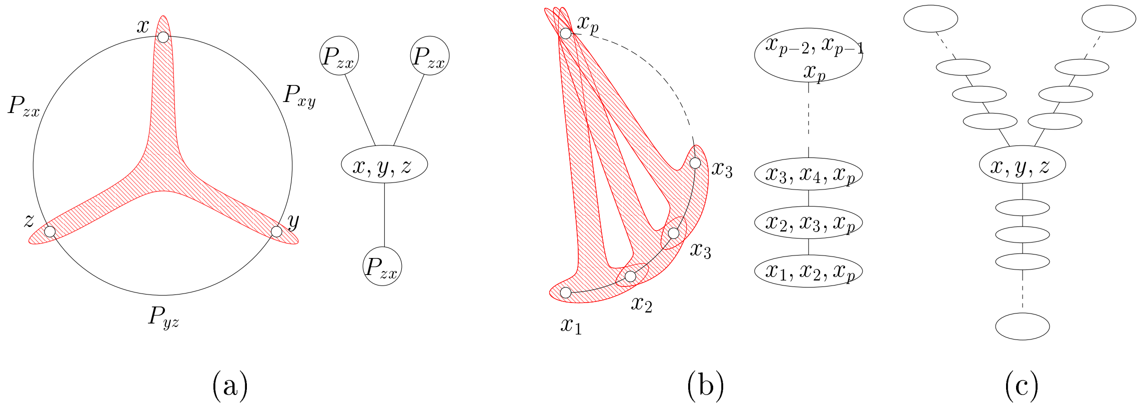


Figure 1: How to construct a tree-decomposition for C_k of length $\lceil k/3 \rceil$ and of width 2.

tree-decomposition of length $\lceil k/3 \rceil$ and of width 2 for C_k in $O(k)$ time. □

Lemma 4 *The tree-length of the cycle of length k is $\lceil k/3 \rceil$.*

Proof. From Lemma 3, it suffices to show that $\mathrm{tl}(C_k) \geqslant k/3$. If $k \leqslant 3$ the lemma is trivialy true. So assume that $k \geqslant 4$.

Let T be a tree-decomposition of C_k and assume that $\mathrm{length}(T) < k/3$. For every two vertices $u, v \in V(C_k)$ such that $d_{C_k}(u, v) < \lfloor k/2 \rfloor$, let P_{uv} denote the unique shortest path from u to v in C_k. An important observation is that for every bag B of T and for all $u, v \in B$ such that $d_{C_k}(u, v) = \mathrm{diam}_{C_k}(B)$, we have $B \subseteq P_{uv}$. Indeed, for $k \geqslant 4$, and for every vertex $w \in B$ such that $w \notin P_{uv}$ it yields $d_{C_k}(w, u) > d_{C_k}(u, v)$ or $d_{C_k}(w, v) > d_{C_k}(u, v)$, that is impossible since $d_{C_k}(u, v) = \mathrm{diam}_{C_k}(B)$.

Let L be a leaf of T and $x, y \in L$ such that $d_{C_k}(x, y) = \mathrm{diam}_{C_k}(L)$. We have $L \subseteq P_{xy}$. Let $x', y' \in C_k \setminus P_{xy}$ such that $\{x, x'\}$ and $\{y, y'\}$ are edges. By Rule 2 of Definition 1, there exist $X, Y \in V(T)$ such that $x, x' \in X$ and $y, y' \in Y$. Since $L \subseteq P_{xy}$ and $x', y' \notin P_{xy}$, $x', y' \notin L$. It follows that $X \neq L$ and $Y \neq L$. We have also that L has a unique neighbor in T, say Z. By Rule 3 of Definition 1, x belongs to all the bags of the path from L to X in T, and y to all the bags from L to Y. In particular $x, y \in Z$. Let $z, t \in Z$ such that $d_{C_k}(z, t) = \mathrm{diam}_{C_k}(Z)$. We have $Z \subseteq P_{zt}$. Since $x, y \in Z$ and $d_{C_k}(x, y) \leqslant d_{C_k}(z, t) < \lfloor k/2 \rfloor$, $P_{xy} \subseteq P_{zt}$. So $L \subseteq P_{zt}$. Therefore $L \cup Z \subseteq P_{zt}$, and so $\mathrm{diam}_{C_k}(L \cup Z) < k/3$.

The tree obtained from T by updating Z to the bag $L \cup Z$, and by removing L, is a tree-decomposition of C_k of length $< k/3$. Step by step we can merge all the leaves of T into a single bag A containing all the vertices of C_k. By induction $\mathrm{length}(A) < k/3$: a contradiction because $A = V(C_k)$, so $\mathrm{diam}_{C_k}(A) = \lfloor k/2 \rfloor \geqslant k/3$ for $k \geqslant 4$. □

We are now ready to prove Theorem 1:

Proof. Let G be an outerplanar graph of chordality k. G can be embedded in the plane such that each vertex lies on the outer face in linear time [CNAO85]. In such representation, inner faces are induced cycles. Observe also, that two inner faces share either nothing, either a vertex, or an edge. So we can obtain, in linear time, a clique-decomposition of G.

By Lemma 2 the tree-length of G is the maximum tree-length of every inner faces of G. We check that the longest induced cycle is an inner face, and we complete the proof applying Lemma 4. □

Moreover Lemma 4, combined with Lemma 1 proves that graphs having an isometric cycle of length k are of tree-length at least $k/3$. In general, the gap between the chordality and the tree-length is large. For instance, some bounded diameter graphs may have unbounded chordality (e.g., consider a wheel). However, we have:

Theorem 2 **[GKK⁺01]** *Every k-chordal graph has tree-length at most $k/2$.*

An $O(nm)$ time algorithm providing such a decomposition is shown in Section 3.1, and an $O(m+n)$ time algorithm providing a tree-decomposition of length $\lfloor k/2 \rfloor + 3$ is described in Section 3.2.

2.3 Tree-Length of Meshes

In this section we will prove the following theorem:

Theorem 3 *The tree-length of the mesh with p rows and q columns is* $\min\{p, q\}$ *if $p \neq q$ or p is even, and is $p - 1$ otherwise.*

It is very easy to show that the tree-length of the mesh is upper bounded by $\min\{p, q\}$. For instance, if $p \leqslant q$, it suffices to form a path composed of all two consecutive columns of the mesh, providing a length which is the distance between the first and the last row plus 1, that is p. However, as for the computation of the tree-width of the mesh (cf. [Die00]), the computation of the optimal lower bound for the tree-length is far from immediate.

So to prove Theorem 3, we need some preliminaries. Consider any tree-decomposition T of a graph G. For every $x \in V(G)$, we denote by T_x the subtree induced by the set of bags of T containing x. A subset $A \subseteq V(G)$ is said *connected* if the subgraph of G induced by A is connected. We say that $S \subseteq V(G)$ *separates* the subsets A and B if S is disjoint of A and B, and if every path connecting $a \in A$ to $b \in B$ intersects S. It is not difficult to check that for every edge $\{X, Y\}$ of T, $S = X \cap Y$ separates the connected components of $G \setminus S$ (cf. [Die00]).

Lemma 5 *Let A, B be two connected subsets of vertices of G. Either there exists a bag $X \in V(T)$ such that X intersects A and B, or there exists an edge $\{X, Y\} \in E(T)$ such that $X \cap Y$ separates A and B.*

Proof. Let $T_A = \bigcup_{a \in A} T_a$ and $T_B = \bigcup_{b \in B} T_b$. If $T_A \cap T_B \neq \varnothing$, then every bag X of $T_A \cap T_B$ intersects A and B, and the result holds. So assume that $T_A \cap T_B = \varnothing$.

For every edge $\{x, y\}$ of G, $T_x \cap T_y \neq \varnothing$, thus $T_x \cup T_y$ is connected and is a subtree of T. Because A is connected, it follows that T_A is a subtree (i.e., is not a forest). Similarly, T_B is a subtree.

Let us define $X \in T_A$ and $Z \in T_B$ as the two mutually closest bags, closest in the metric of T. Since T_A and T_B are subtrees of T, X and Z are unique. Let Y be the adjacent bag of X located on the path from X to Z in T. Observe that $Y \notin T_A$. The set $S = X \cap Y$ does not contain any vertex of A (otherwise Y would belong to T_A), and S does not contain any vertex of B (otherwise X would contain a vertex of B, and T_A would not be disjoint of T_B). So, S is disjoint of A and of B. It follows that S separates the set of vertices in T_A from the set of vertices in T_B, so in particular A and B. □

Lemma 6 *Let $\{x, x'\}, \{y, y'\}$ be two edges of G. If $T_x \cap T_y$ is not empty, then $T_x \cap T_y$ contains a bag intersecting all the paths between x' and y'.*

Proof. Let X (resp. Y) be a bag containing the edge $\{x, x'\}$ (resp. $\{y, y'\}$), and let P be the path from X to Y in T. Assume $T_x \cap T_y \neq \varnothing$. We have that $T_x \cup T_y$ and $T_x \cap T_y$ are trees. So P intersects $T_x \cap T_y$ (otherwise $P \cup T_x \cup T_y$ would contain a cycle). Let U be any bag of $P \cap T_x \cap T_y$. Since $U \in P$, U intersects all the paths from any vertex of X to any vertex of Y. In particular U intersects all the paths between x' and $'y$. □

Proof of Theorem 3. Let G denote the $p \times q$ mesh, and T denote a tree-decomposition such that $\mathrm{length}(T) = \mathrm{tl}(G)$. We assume $p \geqslant 2$, because for $p = 1$ (i.e, G is a path), the result clearly holds Consider the set A composed of the vertices of the first column of G, and the set B composed of the vertices of the last column. From Lemma 5, T contains a bag:

1. either intersecting A and B (and so $\mathrm{length}(T) \geqslant q - 1$); or
2. intersecting all the paths from A to B, in particular intersecting the first row and the last row (and so $\mathrm{length}(T) \geqslant p - 1$).

Assume that $p \leqslant q$. From above, $\mathrm{length}(T) \geqslant p - 1$. On the other hand, using a standard tree-decomposition of G (for instance, form a path composed of all two consecutive columns of G), we have $\mathrm{length}(T) \leqslant p$. So, $\mathrm{length}(T) = p - 1$ or p.

Assume that $\mathrm{length}(T) = p - 1$. Case 1 does not occur if $q > p$, otherwise $\mathrm{length}(T) \geqslant q - 1 \geqslant p$. If $p = q$ and Case 1 occurs, then exchanging rows and columns we have that T contains a bag intersecting the first row and the last row. So we are left with Case 2 where a bag X intersects the first row, say in x, and the last row, say in y. Since we have assumed $\mathrm{length}(T) = p - 1$, x and y belongs to the same column.

W.l.o.g. assume that this column is at distance at least $\lceil (q-1)/2 \rceil$ from A. Let P be the path from x to y lying on the perimeter of G and including all the vertices of A. Let x' be the neighbor of x in P and y' be the neighbor of y in P. If $p \geqslant 2$, then x' belongs to the first row and y' belongs to the last row. From Lemma 6, $T_x \cap T_y$ contains a bag which intersects all the paths from x' to y'. In particular T contains a bag, say Z, containing x, y and a vertex z of P (z distinct from x and y). Let $d = \min_z \max\{d_G(x,z), d_G(y,z)\}$. We have $\mathrm{diam}_G(Z) \geqslant d$ and thus $\mathrm{length}(T) \geqslant d$. Since z must belong to A (since otherwise we would have $d \geqslant p$), we check that a vertex $z \in A$ minimizing d must be located on the $\lceil (p-1)/2 \rceil$-th row, and thus $d = \lceil (p-1)/2 \rceil + \lceil (q-1)/2 \rceil$. If $p \neq q$ (i.e., $q \geqslant p+1$), then $d \geqslant \lceil (p-1)/2 \rceil + \lceil p/2 \rceil = p$: a contradiction. If p is even, then $d \geqslant 2\lceil (p-1)/2 \rceil = p$: a contradiction.

Therefore, $\mathrm{length}(T) = p-1$ implies $p = q$ and p odd.

It remains to check that $p = q$ odd implies $\mathrm{length}(T) = p-1$. For that we construct a tree-decomposition of G of length $p-1$ composed of one bag (chosen as the root of the tree-decomposition) and of four leaves as follows (cf. Fig. 2): the root bag forms a "cross" and is composed of all the vertices located on the $(p-1)/2$-th column or on the $(p-1)/2$-th row (recall that $p = q$ is odd). And, each leaf is composed of the a quadrant of the mesh delimited by the cross (including a suitable part the cross). The length of this decomposition is $2\lceil (p-1)/2 \rceil = p-1$, completing the proof. $\square$

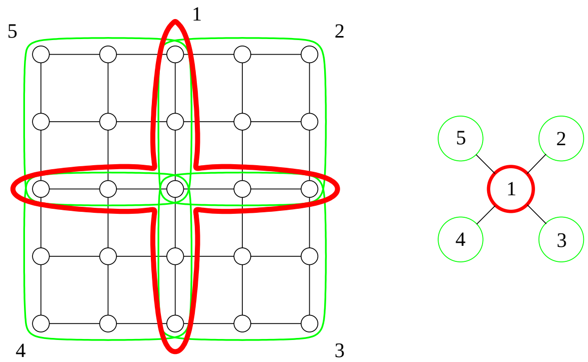


Figure 2: The case $p = q$ odd.

Proposition 1 *The mesh $M_{p,q}$ has a tree-decomposition T such that:*

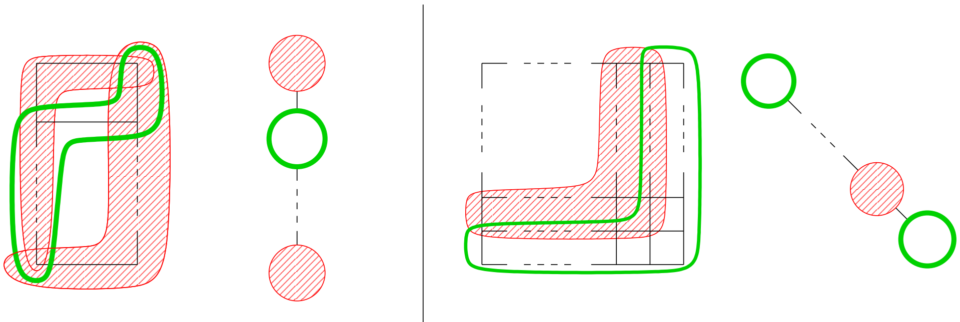
- *if $p \neq q$ or p is even:* $\mathrm{length}(T) = \mathrm{tl}(M_{p,q})$ *and* $\mathrm{width}(T) = \mathrm{tw}(M_{p,q})$
- *if $p = q$ and p is odd:*
  - $\mathrm{length}(T) = \mathrm{tl}(M_{p,q})$ *and* $\mathrm{width}(T) \leqslant 2\mathrm{tw}(M_{p,q})$
  - *or* $\mathrm{length}(T) = \mathrm{tl}(M_{p,q}) + 1$ *and* $\mathrm{width}(T) = \mathrm{tw}(M_{p,q})$

Proof. Recall that the tree-width of $M_{p,q}$ is $\min\{p, q\}$.

Suppose that if $p \neq q$ or p is even. In this case, each bag of the tree-decompostion presented in Fig. 2 can be split in a path of length $p-1$ (see Fig. 3(a)). The length and the width of this tree-decomposition is p.

Suppose now that $p = q$ and p is odd. The tree-decomposition proposed in the previous case clearly satisfies $\mathrm{length}(T) = \mathrm{tl}(M_{p,q}) + 1$ and $\mathrm{width}(T) = \mathrm{tw}(M_{p,q})$.

To prove the last result of Proposition 1, observe that, in the tree-decomposition presented in Fig. 2, the root of the "cross" is of diameter p and contains $2p-1$ vertices. Moreover the four leaves can be split in a path of length $(p-1)/2$ (see Fig. 3(b)). In this way we obtain a tree-decomposition of length $p-1$ and of width $2p-2 \leqslant 2\mathrm{tw}(M_{p,q})$.



(a) (b)

Figure 3: Optimizations of both length and width for meshes.

□

2.4 Width-Length Trade-off

There is no relation between the tree-length and the tree-width of a graph. For instance, n-vertex cliques have tree-width $n-1$ and tree-length 1. Conversely, we have seen that cycles have tree-length $\lceil n/3 \rceil$ whereas their tree-width is two. However, cycles support a tree-decomposition optimizing length and width (cf. Theorem 1).

So, let us define an α-*optimal* tree-decomposition for a graph G as a tree-decomposition T of G such that $\mathrm{length}(T) \leqslant \alpha\,\mathrm{tl}(T)$ and $\mathrm{length}(T) \leqslant \alpha\,\mathrm{tw}(T)$. Theorem 1 implies that every outerplanar graph has an 1-optimal tree-decomposition. Proposition 1 shows for instance that any mesh M admits a $(1 + 1/\mathrm{tl}(M))$-optimal tree-decomposition. Actually, many graphs support 1-optimal tree-decomposition: trees, cycles, chordal graphs, and uniform sub-divisions of chordal graphs[1].

[1]One set of bags consists of the bags of the tree-decomposition of the underlying chordal graph, and each path corresponding to an sub-divised edge is composed of a set of bags of width two as depicted in

At this step, an interesting question is to know whether every n-vertex graph has an α_0-optimal tree-decomposition? Observe that if α_0 is bounded then, although not sufficient, there approximation algorithms for tree-width are good candidates for approximation algorithms for tree-length. Conversely, if α_0 is not bounded then, no algorithms approximating one parameter can approximate the other one.

Unfortunately, the second case occurs proving that tree-length and tree-width computation are fundamentaly two different problems.

Theorem 4 *There exists a graph with at most n vertices on which any α-optimal tree-decomposition requires $\alpha \geqslant \frac{1}{2}n^{1/5} - \frac{1}{2}$.*

Before proving Theorem 4, let us denote by $M(p,q,r)$ the graph composed of a $p \times q$ mesh (p rows, q columns), $p \leqslant q$, in which every edge of a column is replaced by a path of length r. The number of vertices of $M(p,q,r)$ is $pqr - q(r-1)$.

Lemma 7 *Let T be any tree-decomposition of $M(p,q,r)$. Then, if* $\mathrm{length}(T) < (p-1)r$, *then* $\mathrm{width}(T) \geqslant q$.

Proof. For short we denote by M the graph $M(p,q,r)$. Let T be any tree-decomposition of M.

A pair $\{A,B\}$ is *k-connected* if A,B are some connected subsets of $V(M)$ and if every subset that separates A and B is of cardinality at least k. We denote by $d_M(A,B) = \min\{d_M(a,b) \mid a \in A, b \in B\}$.

Claim 1 *Let $\{A,B\}$ be a k-connected pair of M. Then, if* $\mathrm{length}(T) < d_M(A,B)$, *then* $\mathrm{width}(T) \geqslant k$.

Indeed, if $\mathrm{length}(T) < d_M(A,B)$, then there are no bags in T containing a vertex of A and a vertex of B. From Lemma 5, there exists an edge $\{X_i,X_j\}$ of T such that $S = X_i \cap X_j$ separates A from B. Since $\{A,B\}$ is k-connected, $|S| \geqslant k$. Since making reduced a tree-decomposition does not affect its width and its length, we can assume that T is reduced. So $|X_i| \geqslant k+1$ because $S \subset X_i$ is not possible. It follows that $\mathrm{width}(T) \geqslant |X_i| - 1 \geqslant k$ as claimed.

Let A (resp. B) be the set of vertices consisting of the first row (resp. last row). We have $d_M(A,B) = (p-1)r$ and the pair $\{A,B\}$ is q-connected. By Claim 1, if $\mathrm{length}(T) < (p-1)r$, then $\mathrm{width}(T) \geqslant q$ completing the proof of Lemma 7. □

Fig. 1(b). The length of this tree-decomposition is no more than the length t of the edge-path, which is optimal from the lower bound derived from Theorem 1, as the longest isometric cycle is of length $3t$ vertices.

Proof of Theorem 4. Let $M = M(p,q,r)$ for some suitable parameters p,q,r we will fixed later. The tree-length of M is $\delta \leqslant q-1+r$: the tree-decomposition we can consider is a path, each bag consists of two consecutive rows with the paths linking them. The tree-width of M is $k = \min\{p,q\} = p$, because the degree two vertices do not increase the tree-width of a graph.

Let T be any α-optimal tree-decomposition of M. By contradiction, assume that α is chosen such that:

$$\alpha \;<\; \min\left\{\frac{(p-1)r}{q-1+r}, \frac{q}{p}\right\} \;.$$

By Lemma 7, we have that $\mathrm{length}(T) < (p-1)r$ implies that $\mathrm{width}(T) \geqslant q$. Since T is α-optimal, and since $\delta \leqslant q-1+r$, $\mathrm{length}(T) \leqslant \alpha\delta \leqslant \alpha(q-1+r) < (p-1)r$ by definition of α. So $\mathrm{width}(T) \geqslant q > \alpha p = \alpha k$ which is a contradiction with the fact that T is an α-optimal tree-decomposition of M. It follows that $\alpha \geqslant \min\{(p-1)r/(q-1+r), q/p\}$.

Let us choose $p = n^{1/5}$ and $q = r = n^{2/5}$, so that $p \leqslant q$. The number of vertices of M is $pqr - q(r-1)$ that is at most n, and we get that $\alpha \geqslant \frac{1}{2}n^{1/5} - \frac{1}{2}$, which completes the proof. □

2.5 Specific Tree-Decompositions

A natural question about tree-decompositions is to know whether one can restrict tree-decompositions to some specific sub-class of trees, say paths or bounded depth trees, without sacrifying too much on the width or the length of the decomposition.

For instance Bodlaender et al. asked in [BH95] the question of reducing the depth of a tree-decomposition without increasing too much its width (to speed-up parallel algorithms). More precisely, they showed that every n-vertex graph of tree-width k has a tree-decomposition of depth $O(\sqrt{n})$ and width $t \leqslant 2k$, a tree-decomposition of depth $O(\log n)$ and width $t \leqslant 3k-1$, and that forcing a width $t < 3k-1$ might produce (in the worst-case) a depth of $\Omega(\sqrt{n})$, and forcing a width $t < 2k$ might produce a depth of $\Omega(n)$.

Unfortunately, this trade-off property between depth and width does not transfer to length. More precisely, as we will see, there is no constant c for which every n-vertex graph G has a tree-decomposition of length at most $c \cdot \mathrm{tl}(G)$ and depth $o(n)$.

Theorem 5 *Let T be any tree-decomposition of a graph G. Then,* $\mathrm{length}(T) \cdot (\mathrm{diam}(T)+1) \geqslant \mathrm{diam}(G)$. *In particular, for a path with n vertices (which is of tree-length 1), every tree-decomposition of depth h must be of length $\Omega(n/h)$.*

Proof. Let us fix two vertices $x,y \in V(G)$, and a shortest path P from x to y in G. Let X,Y be the bags of T such that $x \in X$, $y \in Y$ and such that $d_T(X,Y)$ is minimum (X and Y are unique since P is connected). Let $X = Q_0, Q_1, \ldots, Q_k = Y$ be the path from X to Y in T. We define q_i as the closest vertex of x that is in Q_i, for $i \in \{0,\ldots,k\}$, and we let $q_{k+1} = y$. So $q_0 = x$, and $q_{k+1} = y$.

Let us show that $q_{i+1} \in Q_i$, for every $i \in \{0, \dots, k\}$. Note that this is true for $i = k$, so assume $i < k$. If $q_{i+1} \notin Q_i$, then $q_{i+1} \in Q_{i+1} \setminus Q_i$. In this case, P intersects $Q_i \cap Q_{i+1}$ (recall that $S = Q_i \cap Q_{i+1}$ separates the connected components of $G \setminus S$). And, there is a vertex $w \in Q_i \cap Q_{i+1} \cap P$ with $d_G(x, w) < d_G(x, q_{i+1})$: a contradiction with the definition of q_{i+1}.

Since for every $i \in \{0, \dots, k\}$, $q_{i+1} \in Q_i$, it follows that q_i and q_{i+1} are both in Q_i, and thus $d_G(q_i, q_{i+1}) \leqslant \mathrm{length}(T)$. Therefore, $\sum_{i=0}^{k} d_G(q_i, q_{i+1}) \leqslant (k+1)\mathrm{length}(T)$. P is a shortest path thus $\sum_{i=0}^{k} d_G(q_i, q_{i+1}) = d_G(q_0, q_{k+1}) = d_G(x, y)$. Noting that $k \leqslant \mathrm{diam}(T)$, we have therefore showed that, for all $x, y \in V(G)$, $d_G(x, y) \leqslant (\mathrm{diam}(T) + 1)\mathrm{length}(T)$. We complete the proof choosing x, y such that $d_G(x, y) = \mathrm{diam}(G)$. □

Paths are ofently used to restrict the set of possible tree-decompositions. In this case we simply deal with *path-decompositions*. The *path-width* of a graph is defined as the minimum width of every path-decomposition of G. It is known that the path-width of every graph does not exceed $\log n$ times its tree-width. Unfortunately, this property does not transfer to tree-length.

Let Y_k be the tree composed of three paths of length k having an extremity in common.

Theorem 6 *Let T be any path-decomposition of a graph G. If G has Y_k as isometric subgraph, then* $\mathrm{length}(T) \geqslant k$. *In particular, there are n-vertex graphs of tree-length* 1 *for which every path-decomposition is of length at least $\Omega(n)$.*

Proof. Let T be any path-decomposition of G, and assume that the bags of T are $X_1, \dots, X_p$ ordered linearly along the path. Consider a copy isomorphic to Y_k in G, and denote by A, B, C the sets of vertices of each branch of Y_k, A, B, C, having a common intersections, say vertex r. Since A induces a connected subgraph in G, it follows that the set of bags containing a vertex of A induces a sub-path of T, say $X_a, X_{a+1}, \dots, X_{a'}$. Similarly, the sets of bags containing a vertex of B and of C induces sub-paths we denote respectively by $X_b, X_{b+1}, \dots, X_{b'}$ and $X_c, X_{c+1}, \dots, X_{c'}$. Because r is common to A, B, C, there is a bag X_i containing r such that $i \in [a, a'] \cap [b, b'] \cap [c, c']$. W.l.o.g. assume that $a = \min\{a, b, c\}$ and $c' = \max\{a', b', c'\}$ (possibly $c' = a'$). Since $i \in [a, a'] \cap [b, b'] \cap [c, c']$, we have $[b, b'] \subseteq [a, a'] \cup [c, c']$. If we let X_j be a bag containing the leaf ℓ of the branch B, so with $j \in [b, b']$, it turns our that $j \in [a, a']$ or $j \in [c, c']$. Therefore, X_j contains ℓ and a vertex either of A or of C. Thus $\mathrm{diam}_G(X_j) \geqslant k$. □

3 Approximation Algorithm

In this section we present three algorithms providing different performances and complexities we carefully analyze.

3.1 Algorithm LexM

This section concerns the proof of the following result.

Theorem 7 *Algorithm* LexM *proposed by Rose, Tarjan and Lueker in [RTL76] computes in $O(nm)$ time a tree-decomposition T of a graph G such that:*

- $\mathrm{length}(T) \leqslant k/2$ *where k is the chordality of G;*
- $\mathrm{length}(T) \leqslant 3 \cdot \mathrm{tl}(G) + 1$.

Moreover for every integer $i > 1$, there exists a graph of tree-length $2i$ such that LexM *computes a tree-decomposition of length $6i + 1$.*

A *triangulation* of G is a chordal graph H such that $V(H) = V(G)$, and $E(G) \subseteq E(H)$. The triangulation H is *minimal* if the graph obtained from H by deleting any edge is not a triangulation of G.

Proposition 2 *Algorithm* LexM *computes a tree-decomposition T of G of length at most $\lfloor k/2 \rfloor$, where k is the chordality of G.*

Proposition 2 can be proved using the three following lemmas:

Lemma 8 [RTL76] *Given a graph G on n vertices and m edges, in $O(nm)$ time, it is possible to compute a minimal triangulation of it.*

Lemma 9 [PS97] *Let G be a graph. Every minimal triangulation of G can be obtained by selecting a maximal set of pairwise parallel minimal separators of G, and by filling them in cliques.*

Lemma 10 [GKK$^+$00] *Let G be a graph of chordality bounded by k, and S be a minimal separator of G then* $\mathrm{diam}_G(S) \leqslant k/2$.

These three lemmas directly imply Proposition 2, noting that a tree-decomposition of chordal graphs in maximal cliques can be done in linear time [BP94]. However, the proofs are not trivial. Here we present a self-sufficient proof based on the study of the LexM Algorithm.

This algorithm uses a lexicographic ordering scheme which is a special type of breadth-first-search. During the search, the vertices are numbered from n to 1. In the following $\alpha(i)$ will denote the vertex numbered i and $\alpha^{-1}(u)$ will denote the number assigned to u. Each vertex u has also a *label*, denoted by $\mathrm{label}(u)$, consisting of a set of numbers selected from $\{1, \ldots, n\}$ ordered in *decreasing* order. Given two labels $L_1 = \{p_1, \ldots, p_k\}$

and $L_2 = \{q_1, \ldots, q_l\}$, we define $L_1 < L_2$ if, for some j, $p_i = q_i$ for $i = 1, \ldots, j-1$ and $p_j < q_j$, or if $p_i = q_i$ for $i = 1, \ldots, k$ and $k < l$. Given a graph $G = (V, E)$ on n vertices and m edges, the following algorithm computes in $O(nm)$ time an ordering on V and a supergraph $H = (V, E \cup E')$.

Algorithm LexM
Input: A graph $G = (V, E)$
Result: A supergraph of G: $H = (V, E' \cup E)$
begin

  Assign empty label to all vertices of G and empty set to E';
  for *i from n downto* 1 **do**
    Select:
      pick an unnumbered vertex u with largest label;
      Assign to u the number i: $\alpha(i) = u$;
    Update:
      for *each unnumbered vertex v such that there is a chain* $u = w_1, w_2, \ldots w_{p+1} = v$ *with w_j unnumbered and* $\text{label}(w_j) < \text{label}(v)$ *for all* $j \in \{2, \ldots, p\}$ **do**
        add i to label(v);
        add $\{u, v\}$ to E';
      end
  end
end

By induction one can see that for every $i \in \{1, \ldots, n-1\}$ the neighbours of $\alpha(i)$ in $H[\{\alpha(i), \ldots, \alpha(n)\}]$ induce a clique. The ordering α is a perfect elimination ordering of H, so H is chordal. Thus:

Lemma 11 *Given a graph $G = (V, E)$ on n vertices and m edges, in $O(nm)$ time, algorithm* LexM *compute a triangulation $H = (V, E \cup E')$.*

To prove Proposition 2, We will show that when statement Update adds an edge $\{u, v\}$ in E', then there exists a chordless cycle C passing via u, v. In this way the distance in G between two vertices adjacent in H is at most $k/2$.

From now u, v denote two fixed vertices such that statement Update adds the edges $\{u, v\}$, and P denotes the chain, supposed chordless, $u = w_1, w_2, \ldots w_{p+1} = v$ satisfying the condition of statement Update.

The chordless cycle C will be composed by three chordless paths $C = P \cup Q \cup R$ where Q is a path satisfying the following lemma, and R will be describe later (see Fig. 4 for a exemple).

Lemma 12 *Let x be the neighbor of v with the maximum number. There exists a vertex x' numbered before u but not before x, such that there exists a path Q from x' to v such that $P \cap Q$ is a chordless path using no vertex numbered before x.*

Proof. Since x is the neighbor of v with maximum number, before x is numbered, label$(v) = \varnothing$. Morevover it is the same for every vertex of P, otherwise one of them has a greater label than v: a contradiction this the definition of P.

If for all $i \in \{2, \ldots, p\}$, $\{x, w_i\} \notin E$, then the path v, x satisfy the Lemma.

So assume that there exists $i \in \{2, \ldots, p\}$ such that $\{x, w_i\} \in E$. In this case after x is numbered label(w_i) = label$(v) = \alpha^{-1}(x)$. But, by assumption on P, at the time u is about to be numbered, label$(w_i) <$ label(v), thus there exists a vertex x' numbered after x but before u such that $\alpha^{-1}(x') \in$ label(v) and $\alpha^{-1}(x') \notin$ label(w_i). Let x' be the first such vertex and let $Q = x_1x_2, \ldots x_{q+1}$ be the chordless path between $x' = x_1$ and $v = x_{q+1}$ found by statement Update at the time x' is numbered and thanks to which label(v) receives $\alpha^{-1}(x')$ (see again Fig. 4 if you need). Assume that there exist a chord between Q and P, i.e., there exists $j \in \{2, \ldots, q+1\}$ and $k \in \{2, \ldots, p\}$ such that $\{x_j, w_k\} \in E$. Then the path $w_i, \ldots, w_k, x_j, \ldots x'$ implies that statement Update adds $\alpha^{-1}(x')$ in label(w_i), a contradiction. □

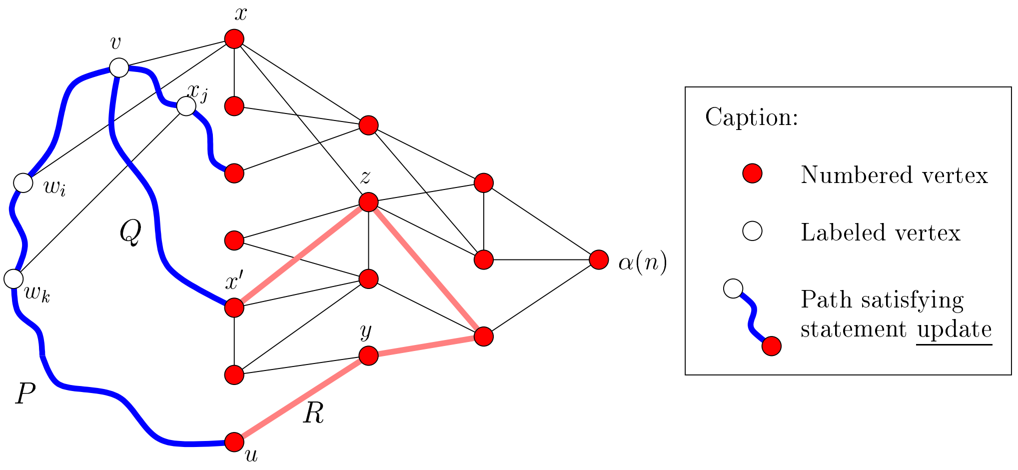


Figure 4: State of G when u is numbered.

Proof of Proposition 2. If $\{u, x\} \in E$ then $d_G(u, v) \leqslant 2$, and since $k > 3$, Proposition 2 is trivially true.

So assume that $\{u, x\} \notin E$. We have already found P and Q such that $P \cup Q$ is a chordless path from u to x' and containing v. Let us construct a last path R such that $P \cup Q \cup R$ is a chordless cycle.

If $\{x', u\}$ is an edge then the path $R = x', u$ satisfy the condition.

So assume that $\{x', u\} \notin E$. The path R will be a chordless path between x' and u using only vertices which are numbered before x. In this way we insure that there is no edge between R and $P \cup Q$, otherwise its extremity in $P \cup Q$ has a label greater than label(v): a contradiction with the definitions of P and of Q. Since $\{u, x\} \notin E$, u has a neighbor, say y, numbered before x, otherwise label(u) is less than label(v) and so u is be numbered before v: a contradiction. If $\{x', y\} \in E$ then we have found $R = x', y, u$. Else we can prove similarily that x' has a neighbor, say z numbered before y (and so before x). If $\{u, z\} \in E$ then we have found $R = x', z, u$. Otherwise, since LexM is a BFS of G, there exists a chordless path R' between y and z using only vertices numbered before them. R is simply x',R',u.

Conclusion $P \cup Q \cup R$ is a chordless cycle containing u and v. Thus $d_G(u, v) \leqslant k/2$ this complete the proof of Proposition 2. □

Remark 1 *The bound of Proposition 2 is thin because for the cycle of length k,* LexM *computes a tree-decomposition of length $\lfloor k/2 \rfloor$. Observe, however, that this is not optimal since the tree-length of the cycle is $\lceil k/3 \rceil$ (by Theorem 1).*

Proposition 3 *Let G be a graph, and T be the tree-decomposition of G computed by Algorithm* LexM*,* $\mathrm{length}(T) \leqslant 3 \cdot \mathrm{tl}(G) + 1$.

Proof. Let u, v be two vertices non adjacent in G, but adjacent in H, let us prove that $d_G(u, v) \leqslant 3 \cdot \mathrm{tl}(G) + 1$. Suppose w.l.o.g. that u is numbered before v. Let T_0 be a tree-decomposition of G of minimum length and rooted at a vertex containing $\alpha(n)$ (see Fig. 5).

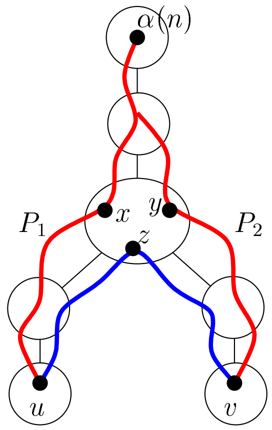


Figure 5: Vertices u, v such that $\{u, v\} \in E'$ in a tree-decomposition of optimal length.

Let P_1, P_2 be two shortest paths, P_1 from $\alpha(n)$ to u and P_2 from $\alpha(n)$ to v. Let U be a bag of T_0 containing u, and V one containing v. Both P_1 and P_2 must use at least one vertex of $X = \mathrm{nca}_{T_0}(U, V)$. Let $x \in P_1 \cap X$ and $y \in P_2 \cap X$. Since $\{u, v\} \in E'$, there exists a path from u to v using only vertices numbered after v. This path must use a vertex z of X.

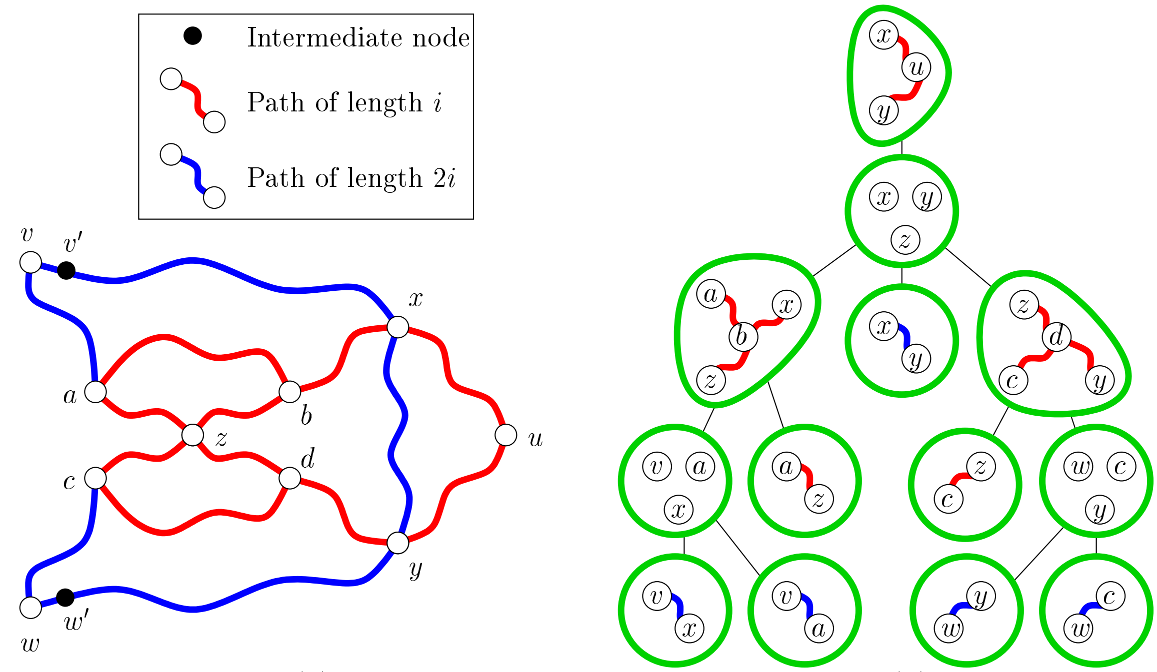
Morevover, Algorithm LexM is based on a BFS-ordering of G so if $d_G(x, u) \geqslant d_G(x, z)+1$ and $d_G(y, v) \geqslant d_G(y, z) + 1$ then z is numbered before v: a contradiction.

Thus $d_G(x, u) + d_G(y, v) \leqslant d_G(x, z) + d_G(y, z) + 1$ which is at most $2\delta + 1$. And since $d_G(u, v) \leqslant d_G(x, u) + d_G(x, y) + d_G(y, v)$, we have $d_G(u, v) \leqslant 3 \cdot \mathrm{tl}(G) + 1$ □

Corollary 1 LexM *cannot be used to approximate the tree-width of any graph (cf. Theorem 4).*

Proposition 4 *For all $i > 1$, there exists a graph G of tree-length $2i$ such that Algorithm* LexM *can returns a tree-decomposition of length $6i - 1$.*

Proof. Let G be the graph presented on Fig. 6.(a). G has $20i - 5$ vertices. One can check that an execution of Algorithm LexM can start at $u = \alpha(20i - 5)$ then assign $x = \alpha(20i - 5 - (2i - 1)) = \alpha(18i - 4)$, then $y = \alpha(18i - 3)$ and so on.



(a) (b)

Figure 6: A graph (a) and a tree-decompsition of it of length $2i$ (b), such that Algorithm LexM computes a tree-decomposition of length $6i - 1$.

Let v' be the neighbor of v in the shortest path from v to x, and let w' be the neighbor of w in the shortest path from w to y. Step by step, v' receives number $6i + 5$ then w' receives number $6i$. Moreover when w' is numbered, v is the only vertex of the path from v to w' passing by a, z, c, w which contains the number of v' in its label. So the label of each vertex of this path is less than the label of v. So this path satisfies statement $\overline{\text{Update}}$, so in the triangulation of G obtained by this execution, v and w' are neighbours, so the length of the tree-decomposition obtained is at least $d_G(v, w') = 6i - 1$.

Moreover G is a graph of tree-length $2i$. Indeed it has an isometric cycle of length $6i$, and there exists a tree-decomposition of it of length $2i$ given in Fig. 6(b). $\square$

3.2 Algorithm BFS-Layering

In this section we propose an algorithm (BFS-Layering), based on the notion of *Layering-tree* introduced in [CD00], such that:

Theorem 8 *Algorithm* BFS-Layering *constructs in $O(n + m)$ time a tree-decomposition T of any graph G such that:*

- length$(T) \leqslant k/2 + 3$, *where k is the chordality of G.*
- length$(T) \leqslant 3 \cdot \mathrm{tl}(G) + 1$*;*

Moreover, for all $i > 1$, there exists a graph of tree-length $2i$ for which the tree-decomposition returned by Algorithm BFS-Layering *is at least* $6i + 1$.

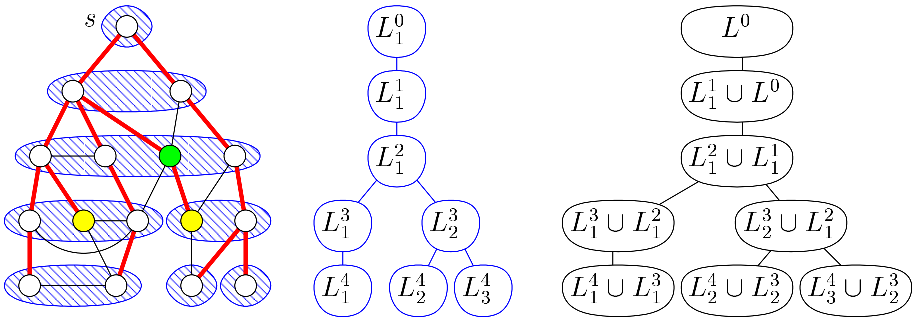
Let G be a graph with a distinguished vertex s. For every integer $i \geqslant 0$, we define $L^i := \{u \in V(G) \mid d_G(s, u) = i\}$. A *layering partition* of G is a partition of each set L^i into $L^i_1, \dots, L^i_{p_i}$ such that $u, v \in L^i_j$ if and only if there exists a path from u to v using only intermediate vertices w such that $d_G(s, w) \geqslant i$.

Let H be the graph whose vertex set is the collection of all the parts L^i_j. In H, two vertices L^i_j and $L^{i'}_{j'}$ are adjacent if and only if there exists $u \in L^i_j$ and $v \in L^{i'}_{j'}$ such that u and v are adjacent in G (see Fig. 7 for an example). The vertex s is called the *source* of H.

Lemma 13 [CD00] *The graph H, called* layering-tree *of G, is a tree and is computable in linear time. for an exemple).*

Let us define the following algorithm:

Lemma 14 *Let T be the tree computed by* BFS-Layering*, T is a tree-decomposition of G.*



(a) (b) (c)

Figure 7: A 5-chordal graph G, a Layering-tree of G, and the tree-decomposition of G assciated with that layering.

Algorithm BFS-Layering
Input: A graph $G = (V, E)$
Result: A tree-decomposition T of G
begin
  Let H be a Layering-tree of G;
  Let T be a copy of H;
  for *every vertex U of T* **do**
    $U \leftarrow U \cup V$ where V is the parent of U in H;
  end
end

Proof. H is a partition of the set of vertices of G, so every vertex of G is contained in a node of H. By construction, it is the same for T. Moreover a vertex of G is in exactly one node of H. Thus, in the tree T, a vertex u of G belongs only to a node and to all of its childs: a subtree, so Rule 3 of the definition of a tree-decomposition is satisfied.

Let $\{u, v\}$ be an edge of G. If $d_G(s, u) = d_G(s, v)$, then by construction u and v belong to a same node of H and thus of T. Otherwise u and v let $L_j^i, L_{j'}^{i'}$ be the two node of H such that $u \in L_j^i$ and $v \in L_{j'}^{i'}$. By construction L_j^i and $L_{j'}^{i'}$ are adjacent in H. Therefore there exists a vertex of T containing both u and v. □

Lemma 15 *Let G be a graph of chordality k and let T be the tree-decomposition of G computed by* BFS-Layering, length$(T) \leqslant k/2 + 3$.

Proof. Let G be a graph of chordality k, let H be a Layering-tree of G and let u, v be two vertices of G which belong to a same node of H: L_j^i.

By definition of L^i_j, there exists a path from u to v using only vertices wich are at distance is at most i from s. Let P_1 be a such chordless path. Moreover u and v are connected by a path of which all verticies (except u and v) are at distance at most $i-1$ from s. Let P_2 be a such chordless path (see Fig. 8).

If there is no chord between P_1 and P_2 then $P_1 \cup P_2$ is an induced cycle containing both u and v, and thus $d_G(u,v) \leqslant k/2$.

Otherwise, as shown in Fig. 8, chords can exist only between the neighbor of u in P_2: u' (or the neighbor of v in P_2: v') and a vertex of P_1.

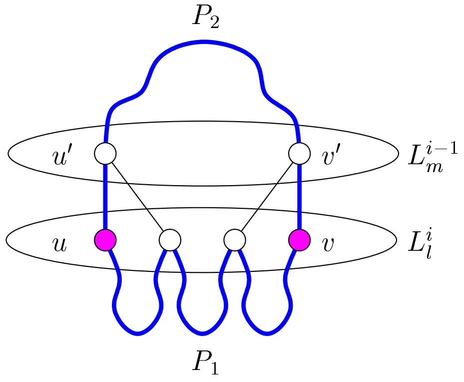


Figure 8: The distance between u and v is at most $k/2+2$.

In the worst case, there is a chord from both u' and v'. Nevertheless there is an induced cycle containing u' and v'. Thus $d_G(u',v') \leqslant k/2$ and $d_G(u,v) \leqslant k/2+2$.

Let T be the tree-decomposition computed by BFS-Layering and let X be an arbitrary bag of T. By construction, there exists i,j,j' such that $X = L^i_j \cup L^{i-1}_{j'}$. Let x,y be two vertices of X. If both x,y belong to L^i_j or to $L^{i-1}_{j'}$, then we have shown that $d_G(x,y) \leqslant k/2+2$. Otherwise assume w.l.o.g. that $x \in L^i_j$ and $y \in L^{i-1}_{j'}$, by construction x has a neighbor w in $L^{i-1}_{j'}$ and since $d_G(w,y) \leqslant k/2+2$, we can conclude that $d_G(x,y) \leqslant k/2+3$. □

Remark 2 *The bound of Proposition 15 is thin. Indeed for the graph of Fig. 9,* BFS-Layering *puts the vertices x and y in a same bag. Moreover the chordality of G is 4 and $d_G(x,y) = 5 = 4/2+3$.*

Lemma 16 *Let T be the tree-decomposition computed by* BFS-Layering, *$\mathrm{length}(T) \leqslant 3 \cdot \mathrm{tl}(G)+1$.*

Proof. Let G be a graph, and H be a Layering-tree of it. Let T_0 be a tree-decomposition of G of minimum length, rooted at a bag containing s. We will make a proof similar to the one of the Proposition 3.

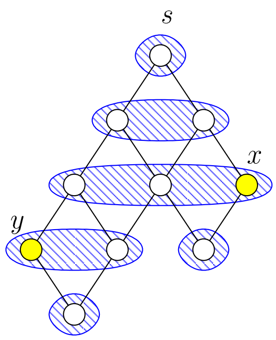


Figure 9: A graph of chordality k such that BFS-Layering compute a tree-decomposition of length $k/2+3$.

Let u, v be two vertices of G which belong to a same node of H: L^i_j. Let U be a bag of T_0 containing u, and V one containing v. Any shortest path from s to u (Resp. to v) has to use at least a vertex x (Resp. y) of $X = \mathrm{nca}_{T_0}(U, V)$. Since u, v are both in L^i_j, then there exists a path between u and v such that for every vertex w of this path, $d_G(s, w) \geqslant i$. This path has to use a vertex $z \in X$. Assume that $d_G(u, x) \geqslant \mathrm{tl}(G) + 1$ or $d_G(v, y) \geqslant \mathrm{tl}(G) + 1$, then $d_G(s, z) \leqslant i - 1$: a contradiction. And since $d_G(u, v) \leqslant d_G(u, x) + d_G(x, y) + d_G(y, u)$ we obtain $d_G(u, v) \leqslant 3 \cdot \mathrm{tl}(G)$.

It follows that for every u, v such that u and v belong to a same bag of T, $d_G(u, v) \leqslant 3 \cdot \mathrm{tl}(G) + 1$. □

Proposition 5 *For every $i > 1$, there exists a graph G of tree-length $2i$ such that Algorithm* BFS-Layering *can return a tree-decomposition T of length $6i + 1$.*

Proof. Let G be the graph defined in Fig. 6(a) in which one adds a new vertex v'' connected at v only.

If Algorithm BFS-Layering starts at u, then clearly L^{3i}_1 contains v, a, z, c, w. In the Layering-tree, one of its childs contains v''. It follows that one bag of the tree-decomposition contains v'', v, a, z, c, w, and $d_G(v'', w) = 6i + 1$. □

Theorem 8 follows from lemmas 14, 15, 16 and from Proposition 5.

3.3 An Heuristic: Disk-Tree

This section deals with the 2-approximation of the tree-length of any graph. The heuristic we present is based on the idea that, given a graph G, if there exists a tree-decomposition T of length k, then every bag of T is included in the disk of radius k centered at any of its vertices. So disks of radius k are good candidates to construct a tree-decomposition of length $2k$. But constructing for every vertex of G its disk of radius k is not a solution because it is not always possible to organize them in a tree. This explains why in our heuristic, the center of a disk is chosen is in border(T), and why Init and Reduce remove some vertices.

Before entering the details, we introduce some definitions and notations:

- The set of vertices of G covered by T is: covered$(T) = \cup_{X \in V(T)} X$. The set of vertices covered by T but with at least one neighbor not covered is: border$(T) = \{u \in \text{covered}(T) \mid N(u) \setminus \text{covered}(T) \neq \varnothing\}$.
- For every subset X of $V(G)$ and for every $u \in X$, out(u, X) denotes the subset of vertices of X connected to u by a path of which all vertices (except the two extremities) are not in X. In other terms, let $CC_1, CC_2, \cdots, CC_p$ be all the connected components of $G[V(G) \setminus X]$ such that $\forall i \in \{1, \cdots, p\}$, $u \in N(CC_i)$, then out$(u, X) = \cup_{i \in \{1, \cdots, p\}} N(CC_i)$.
- Let S be a subset of $V(G)$, and let u, v be two vertices of S. u and v are in *conflict* if $v \in \text{out}(u, \text{covered}(T) \cup S)$ and $d_G(u, v) > k$. When it occurs, we have to remove one of them, but we will do it careffuly.
- For every vertex u of G covered by T, B(u) denotes the bag of minimum depth in T containing u. The tree T is construct by depth, that is to say, a vertex c can be chosen to be the center of a new disk, if and only if $c \in \text{border}(T)$ and $\forall v \in \text{border}(T)$, depth(B$(c)$) $\geqslant$ depth(B(v)). The set of vertices which can be chosen is denoted C.

We also need the following lemma. It is separate from our heuristic, but will be usefull to prove that our heuristic builds a subtree of a tree-decomposition of G.

Lemma 17 *Let T be a* subtree of a tree-decomposition *of a graph G. Then:*

1. *T is a tree-decompositon of the subgraph of G, induced by the set of covered vertices.*
2. *Let $CC_1, CC_2, \ldots, CC_m$ be the connected components of the subgraph of G induced by the set of vertices not covered by T. For every CC_i there exists a bag X_i of T, such that $N(CC_i) \subseteq X_i$.*

Proof.

$\Rightarrow$ Let T be a subtree of a tree-decomposition T_0 of a graph G. Clearly T is a tree-decomposition of the subgraph induced by the covered vertices.

Let u be a vertex of G not covered by T, and let $X \in V(T)$ be the closest bag of B(u) in T_0. Since T_0 is a tree-decompsition of G, each path from u to any vertex covered by T has to use at least one vertex of X. Thus, this is also true for all the vertices reachable from u by a path containing only non covered vertices. So we conclude that the neighbourhood of the connected component of $G[V(G) \setminus \text{covered}(T)]$ containing u is incuding in X.

$\Leftarrow$ Let T be a tree satisfying the two properties of Lemma 17. For every $i \in \{1, \cdots, m\}$ let us define $CC_i^+ = CC_i \cup N(CC_i)$. Let T^+ be the tree containing a copy of T in which one adds each set CC_i^+ by connecting it to the bag X_i.

Let $\{x, y\}$ be an edge of G. Assume that there exists $i \in \{1, \cdots, m\}$ such that $x \in CC_i$, by assumption, either $y \in X_i \cap N(CC_i)$ or $y \in CC_i$. Thus, this edge is contained in CC_i^+: a bag of T^+. Assume now that both x and y are covered by T, then by assumption on T, there exists a bag of T (and thus of T^+) containing $\{x, y\}$.

Finally, let $i \in \{1, \cdots, m\}$ and $u \in CC_i^+$. If $u \in CC_i$ then u belongs exactly to one bag of T^+. Otherwise by assumption on T, the set of bags containing u induces a subtree of T, and since u belongs to the neighbor of CC_i^+, then it is still true in T^+.

We have proved that T^+ is a tree-decomposition of G.

$\square$

Now we can present our heuristic:

Algorithm Disk-Tree
Input: A graph $G = (V, E)$ and an integer k
Result: either FAIL or T: a tree-decomposition of G
begin
    Let u be an arbitrary vertex of G;
    $T \leftarrow (\{u\}, \varnothing)$ $C \leftarrow \{u\}$;
    while $\text{covered}(T) \neq V(G)$ **do**
        Init:
            chose $c \in C$, if c does not exist return(FAIL);
            $S \leftarrow$ connected component of c in $G[V(G) \setminus (\text{covered}(T) \setminus \{c\})]$;
            $S \leftarrow S \cap D^k(c)$, where $D^k(c)$ is the disk of radius k centered at c;
            $S \leftarrow S \cup \text{out}(c, \text{covered}(T))$;
        Reduce:
            while *there exists two vertices u, v in conflict in S* **do**
                if $(v \in \text{border}(T))$ *or*
                  $(u \notin \text{border}(T)$ *and* $d_G(c_{i+1}, u) \geqslant d_G(c_{i+1}, v))$ **then**
                    remove u from S;
                else remove v from S;
            end
        Update:
            if $S \not\subseteq \text{covered}(T)$ **then**
                $T \leftarrow (V(T) \cup \{S\}\ ,\ E(T) \cup \{\{\text{B}(c), S\}\})$;
                $C \leftarrow \{u \in \text{border}(T) \mid \forall v \in \text{border}(T), \text{depth}(\text{B}(u)) \geqslant \text{depth}(\text{B}(v))\}$;
            else remove c from C;
    end
    Return(T);
end

From now, for every $i \geqslant 1$, T_i denotes the tree built by Disk-Tree with i bags, and its bags are $\{S_1, S_2, \ldots, S_i\}$. Morevover c_j denotes the vertex chosen by Disk-Tree to build S_j.

For all $i \geqslant 1$ we define the following property:

$$P_i = \text{“}T_i \text{ is a subtree of a tree-decomposition of } G\text{”}.$$

Lemma 18 *If P_i is true, then* $\text{out}(c_{i+1}, \text{covered}(T_i)) \subseteq \text{B}(c_{i+1})$.

Proof. Assume that P_i is true, and let CC be a connected component of $G[G \setminus \text{covered}(T_i)]$ containing c_{i+1}.

Let $u \in CC$, since c_{i+1} is the vertex of $\text{border}(T_i)$ whose bag is of maximum depth in T_i, we obtained that $\mathrm{B}(u)$ is not a descendant of $\mathrm{B}(c_{i+1})$.

Moreover, lemma 17 implies that there exists a bag X of T_i such that $N(CC) \subseteq X$. Since T_i is a tree-decomposition of the covered vertices, X is a descendant of $B(v)$ for all $v \in N(CC)$. In particular X is a descendant of $B(u)$ and $B(c_{i+1})$. Since $\mathrm{B}(u)$ is not a descendant of $\mathrm{B}(c_{i+1})$, it is an ancestor of (or is equal to) $\mathrm{B}(c_{i+1})$. So $u \in B(c_{i+1})$. □

Lemma 19 *P_i implies P_{i+1}.*

Proof. Let G be a graph and i be an integer such that P_i is true. We will start by proving the that the second property of Lemma 17 is satisfied by T_{i+1}.

Since, P_i is true, the property is true for each connected component CC of $G[V(G) \setminus \text{covered}(T_i)]$ such that $N(CC) \not\subseteq B(c_{i+1})$.

So let CC be a connected component in $G[V(G) \setminus \text{covered}(T_i)]$ such that $N(CC) \subseteq B(c_{i+1})$. Let $CC^- = CC \setminus S_{i+1}$ and $u \in CC^-$. We have to show that either $N(CC^-) \subseteq B(c_{i+1})$ or $N(CC^-) \subseteq S_{i+1}$. So we have to prove that the situation presented Fig. 10 is impossible.

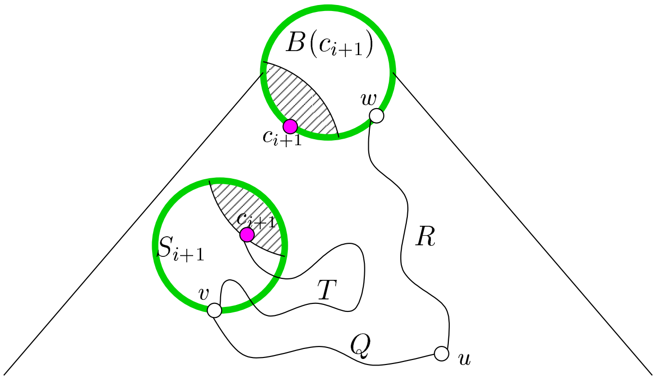


Figure 10: Impossible case, because $Q \Rightarrow \neg R$.

Assume that the situation presented in Fig. 10 occurs, i.e., assume that there exists $v \in S_{i+1} \setminus B(c_{i+1})$ and $w \in B(c_{i+1}) \setminus S_{i+1}$, such that $v, w \in N(CC)$. Since $v \in S_{i+1} \setminus B(c_{i+1})$ then there exists a path R from c_{i+1} to v which avoids $B(c_{i+1})$. So the thre paths T, Q, R show that $w \in \text{out}(c_{i+1}, \text{covered}(T_i))$, thus by construction $w \in S_{i+1}$: a contradiction.

So we have proved that T_{i+1} satisfies property 2 of Lemma 17. Now we will proved that T_{i+1} is a tree-decomposition of the vertices which it covers.

Since P_i is true, if there exists an edge between two vertices covered by T_{i+1} which is not ina bag, then this edge is between a vertex of $S_{i+1} \setminus \mathrm{B}(c_{i+1})$ and a vertex of $\mathrm{B}(c_{i+1}) \setminus S_{i+1}$.

It is the same situation that in the previous case but this time $P \cup Q$ is an edge. The result is similar: this edge cannot exist.

Finally, let us show that for any vertex u covered by T_{i+1}, the set of bags containing u is a subtree of T_{i+1}. This is true by assumption for all the vertices which are not in S_{i+1}. Let $u \in S_{i+1}$, if u is not covered by T_i then S_{i+1} is the only bag containing u, it is a subtree. The last case is when u is covered by T_i. In this case, by construction $u \in \text{out}(c_{i+1}, \text{covered}(T_i))$, thus Lemma 18 shows that $u \in \text{B}(c_{i+1})$. Moreover when the bag S_{i+1} is introduced in T_i, it is connected to $\text{B}(c_{i+1})$. Therefore the set of the bags of T_{i+1} containing u still induces a subtree.

By Lemma 17, T_{i+1} is thus a subtree of a tree-decomposition of G. □

Proposition 6 *The heuristic* Disk-Tree *finishes in polynomial time. Moreover when it finishes successfully, T is a tree-decomposition of G of length at most $2k$.*

Proof. First of all, observe that if Disk-Tree finishes successfully then all vertices of G are covered by T. Moreover P_1 is trivialy true, thus Lemma 19 implies that T is a tree-decomposition of G.

Now let us show that any bag S_i is included in $D^k(c_i)$ (the disk of radius k centrer at c_i). Let $u \in S_i$ then

- either u is not covered by T_{i-1}, then by construction $u \in D^k(c_i)$.
- or u is covered by T_{i-1}. Then by construction $u \in \text{out}(c_i, \text{covered}(T_{i-1}))$ and we have already shown in the proof of Lemma 18 that $u \in \text{B}(c_i)$. Therefore, let $j < i$ such that $\text{B}(c_i) = S_j$ i.e., c_i is covered by T_j but not by T_{j-1}, we have $u \in \text{out}(c_i, \text{covered}(T_{j-1} \cup S_j))$. Thus when S_j is added to T, if $d_G(c_i, u) > k$ then c_i and u are in conflict thus the phase Reduce removes either u, or c_i: a contradiction. Thus $d_G(c_i, u) \leqslant k$.

Finally, observe that Disk-Tree introduces the bag S_i into T_{i-1} if and only if S_i contains at least one vertex not covered by T_{i-1}. Thus at the end T contains at most n bags. Moreover the number of vertices in the set C is also at most n. So the total number of loops is at most n^2. Clearly one loop is done in polynomial time, so Disk-Tree finishes in polynomial time. □

Proposition 7 *If $k \geqslant 3 \cdot \text{tl}(G) - 2$, then after the phase Init, there is no vertices in conflict. Thus* Disk-Tree *finishes successfully.*

Proof. Observe that at each time, the phase Init puts in S_i at least one vertex not covered by T_{i-1}: the neighbor of c_i. Let u, v be two vertices of S_i such that $u \in \text{out}(v, \text{covered}(T_{i-1} \cup S_i))$ just after the phase Init.

As we did for the proofs of Proposition 3 and Lemma 16, we consider T_0, a tree-decomposition of G of length minimum rooted at a bag containing c_i.

Any shortest path from c_i to u (Resp. v) has to use at least one vertex x (Resp. y) of $Z = \mathrm{nca}_{T_0}(\mathrm{B}(u), \mathrm{B}(v))$. Since u and v are in conflict, there is exists a path between u and v of which all the intermediates vertices are not in $\mathrm{covered}(T_{i-1}) \cup S_i$, so in particular for every vertex w of this path, $d_G(c_i, w) > k$. This path must use a vertex $z \in Z$. Moreover if $d_G(x,u) \geqslant d_G(x,z)$ then $d_G(c_i,z) \leqslant d_G(c_i,u) \leqslant k$: a contradiction. So $d_G(x,u) < d_G(x,z) \leqslant \delta$. We obtain similarily that $d_G(y,v) < d_G(y,z) \leqslant \delta$. Finally we have that $d_G(u,v) \leqslant d_G(u,x) + d_G(x,y) + d_G(y,v) \leqslant 3\delta - 2$.

We can conclude: if $k \geqslant 3 \cdot \mathrm{tl}(G) - 2$ then u and v are not in conflict. □

For example if the graph G is chordal and if $k = 1$, the phase Reduce does not remove any vertex. Thus the heuristic returns a tree-decomposition of length 2.

Moreover you have probably noticed that in each proof of this section we never use neither the fact that we remove carefully one of two vertices in conflict, nor the ability to choose an other vertex in cases where the previous fails. They serve only to make the heuristic stronger in case where k is less than $3\mathrm{tl}(G) - 2$, and they justify the following conjecture:

Conjecture 1 Disk-Tree *finishes successfully for* $k \geqslant \mathrm{tl}(G)$, *i.e.,* Disk-Tree *is a* 2-*approximation algorithm.*

This conjecture is partly justified by the fact that the heuristic Disk-Tree computes a better tree-decomposition for the counter-example used for the two previous algorithms (see Fig. 11). Starting on the same vertex u, Disk-Tree finishes successfully with $k = 2i$. The next six points explain why the heuristic succeeds after six loops.

1. Initially the phase Init builds a disk which goes from u to the vertices x', b, d and y'. Some of these vertices are in conflict, the phase Reduce gradually removes the paths from x to x', from x to b, from y to d, and from y to y'.

2. At the following step we can choose between starting at x or at y. Since the graph is symmetric, the choice does not change anything. Let us suppose that one chooses x. The phase Init builds a bag containing the shortest paths from x to v, from x to z and from x to a. The vertex y is also put in the bag. There is no conflict.

3. Then one can choose either v or a or z. We cannot choose y because $\mathrm{B}(y)$ is of depth lower than $\mathrm{B}(v)$, $\mathrm{B}(a)$ and $\mathrm{B}(z)$. Let us suppose that the choice is made on a, the phase Init builds a bag containing the shortest paths from z to v. There is no conflict.

4. The only possible choice is now z. The phase Init builds a bag with the shortest paths from z to z', from z to y and from c to d. The vertices y and z' are in conflict.

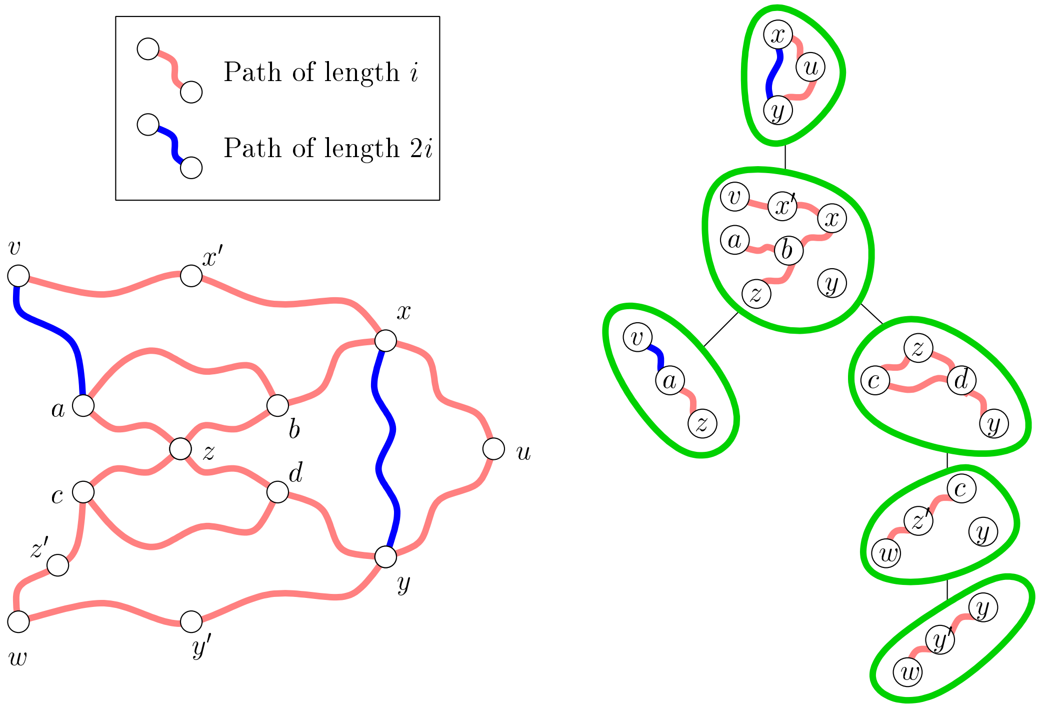


Figure 11: Tree-decomposition obtained by the heuristic Disk-Tree on the counter-example.

Since y is still covered, z' is removed by the phase <u>Reduce</u>. Step by step, all the path between c and z' is removed.

5. The only possible choice is now c. The bag obtained contains the shortest path between c and w, and the vertex y.

6. The last bag contains finally the shortest path between w and y. All vertices of the graph are then covered and thus the heuristic finishes successfully.

4 Conclusion and further works

In this paper we introduce a new graph parameter, the tree-length, useful to obtain good approximations for several problems (including distance labeling, compact routing and additive sparse spanners). We have computed the tree-length of meshes and outerplanar graphs.

The first question we left open is:

Question 1 *Is it true that every graph of chordality k has tree-length at most $\lceil k/3 \rceil$?*

We have anwsered positively if G is outerplanar for which the tree-length is exactly $\lceil k/3 \rceil$, and it is known that the tree-length of any graph can not be greater than its chordality divided by 2.

We have shown in this paper that tree-length is very different from the tree-width, another parameter related to tree-decompositions. Indeed some graphs have bounded tree-width but unbounded tree-length, and the reverse is also true. Moreover, we have proved that it is not always possible to minimize both width and length of a tree-decomposition. For example we have presented an n-vertex graph for which any tree-decomposition is far from the mininum on either its tree-length or its tree-width within a factor $\Omega(n^{1/5})$. Let us denote by $\alpha(G)$ the smallest real α such that G has an α-optimal tree-decomposition, then we ask:

Question 2 *Is it true that for every graph G of n vertices we have $\alpha(G) = O(\sqrt{n})$?*

Note that the answer will indicate for example how far the width of the tree-decomposition computed by LexM is from the tree-width of the graph.

An other difference between the tree-length and the tree-width is that, in case of tree-width it is possible to modify the depth of the tree-decomposition, or its degree, without increasing to much its width. We have proved that this is false in case of tree-length.

About complexity issue, we have also shown a difference between tree-length and tree-width. Indeed it is possible to approximate (in polynomial time) the tree-length of a graph within a constant factor, whereas the best known polynomial time algorithm for the tree-width is far from the minimum within a logarithmic factor. The main question left open is this paper is:

Question 3 *What is the time complexity of computing the tree-length of any graph?*

Note that determining whether a graph is or not of tree-length 1 can be done in linear time. So a first step to answer Question 3 would be to know the complexity of determining if a graph is of tree-length 2.

Finally, if it is true that the problem is NP-Complete in general, then this justifies the interest of Conjecture 1. We recall this conjecture:

Conjecture 1 *Algorithm* Disk-Tree *finishes successfully for $k \geqslant \mathrm{tl}(G)$, i.e.,* Disk-Tree *is a polynomial time 2-approximation algorithm of the tree-length of any graph.*

References

[ACP87] Stefan Arnborg, Derek G. Corneil, and Andrzej Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal Algebraic Discrete Math.*, 8:277–284, 1987.

[Ami01] Eyal Amir. Efficient approximation for triangulation of minimum treewidth. In 17^{th} *Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 7–15. Morgan Kaufmann, August 2001.

[Arn85] Stefan Arnborg. Efficient algorithms for combinatorial problems on graphs with bounded decomposability – a survey. *BIT*, 25:2–23, 1985.

[BDLL04] Andreas Brandstädt, Feodor F. Dragan, H.-O. Le, and Van Bang Le. Tree spanners on chordal graphs: Complexity and algorithms. *Theoretical Computer Science*, 310:329–354, 2004.

[BG88] Alan A. Bertossi and A. Gori. Total domination and irredundance in weighted interval graphs. *SIAM Journal on Discrete Mathematics*, 1:317–327, 1988.

[BGHK95] H. Bodlaender, J. Gilbert, H. Hafsteinsson, and T. Kloks. Approximating treewidth, pathwidth, frontsize, and shortest elimination tree. *Journal of Algorithms*, 18:238–255, 1995.

[BH95] Hans Leo Bodlaender and Torben Hagerup. Tree decompositions of small diameter. In 23^{rd} *International Symposium on Mathematical Foundations of Computer Sciences (MFCS)*, volume 1450 of Lecture Notes in Computer Science, pages 702–712. Springer-Verlag, 1995.

[BH98] Ulrik Brandes and Dagmar Handke. NP-completeness results for minimum planar spanners. *Discrete Mathematics & Theoretical Computer Science*, 3(1):1–10, 1998.

[BJ82] Kellogg S. Booth and J. Howard Johnson. Dominating set in chordal graphs. *SIAM Journal on Computing*, 11:191–199, 1982.

[BKMT01] Vincent Bouchitté, Dieter Kratsch, Haiko Müller, and Ioan Todinca. On treewidth approximations. In 1^{st} *Cologne-Twente Workshop on Graphs and Combinatorial Optimization (CTW)*, June 2001. Electronic Notes in Discrete Mathematics, vol. 8, May 2001.

[Bod96] Hans Leo Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25:1305–1317, 1996.

[Bod98] Hans Leo Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209(1-2):1–45, 1998.

[BP94] J.R.S. Blair and B.W. Peyton. On finding minimum-diameter clique trees. *Nordic Journal of Computing*, 1(2):173–201, 1994.

[Bra87] Andreas Brandstädt. The computational complexity of feedback vertex set, hamiltonian circuit, dominating set, steiner tree, and bandwidth on special perfect graphs. *J. Inf. Process. Cybern.*, 23:471–477, 1987.

[BT97] Hans Leo Bodlaender and Dimitrios M. Thilikos. Treewidth for graphs with small chordality. *Discrete Applied Mathematics*, 79:45–61, 1997.

[CC95] Leizhen Cai and Derek G. Corneil. Tree spanners. *SIAM Journal on Discrete Mathematics*, 8(3):359–387, 1995.

[CD00] Victor D. Chepoi and Feodor F. Dragan. Distance approximating trees in graphs. In Elsevier, editor, 6^{th} *International Conference on Graph Theory (ICGT)*. Electronical Notes in Discrete Mathematics, August 2000.

[CNAO85] Norishige Chiba, Takao Nishizeki, Shigenobu Abe, and Takao Ozawa. A linear algorithm for embedding planar graphs using pq-trees. *Journal of Computer and System Sciences*, 30(1):54–76, 1985.

[CV03] Bruno Courcelle and Rémi Vanicat. Query efficient implementation of graphs of bounded clique-width. *Discrete Applied Mathematics*, 131:129–150, 2003.

[DBETT99] Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. *Graph Drawing: Algorithms for the visualization of graphs.* Prentice Hall, 1999.

[DDGY04] Yon Dourisboure, Feoder F. Dragan, Cyril Gavoille, and Chenyu Yan. Spanners for bounded tree-length graphs, May 2004. Submitted.

[DG02] Yon Dourisboure and Cyril Gavoille. Improved compact routing scheme for chordal graphs. In 16^{th} *International Symposium on Distributed Computing (DISC)*, volume 2508 of Lecture Notes in Computer Science, pages 252–264. Springer, October 2002.

[Die00] Reinhard Diestel. *Graph Theory (second edition)*, volume 173 of Graduate Texts in Mathematics. Springer, February 2000.

[DK99] Yevgeniy Dodis and Sanjeev Khanna. Designing networks with bounded pairwise distance. In 30^{th} *Annual ACM Symposium on Theory of Computing (STOC)*, pages 750–759, 1999.

[Dou02] Yon Dourisboure. An additive stretched routing scheme for chordal graphs. In 28^{th} *International Workshop, Graph - Theoretic Concepts in Computer Science (WG)*, volume 2573 Lecture Notes in Computer Science, pages 150–163. Springer, June 2002.

[Epp00] David Eppstein. Diameter and treewidth in minor-closed graph families. *Algorithmica*, 27:275–291, 2000.

[FK01] Sándor P. Fekete and Jana Kremer. Tree spanners in planar graphs. *Discrete Applied Mathematics*, 108:85–103, 2001.

[Gav01] Cyril Gavoille. Routing in distributed networks: Overview and open problems. *ACM SIGACT News - Distributed Computing Column*, 32(1):36–52, March 2001.

[GKK+00] Cyril Gavoille, Michal Katz, Nir A. Katz, Christophe Paul, and David Peleg. Approximate distance labeling schemes. Research Report RR-1250-00, LaBRI, University of Bordeaux, 351, cours de la Libération, 33405 Talence Cedex, France, December 2000.

[GKK+01] Cyril Gavoille, Michal Katz, Nir A. Katz, Christophe Paul, and David Peleg. Approximate distance labeling schemes. In F. Meyer auf der Heide, editor, 9^{th} *Annual European Symposium on Algorithms (ESA)*, volume 2161 of Lecture Notes in Computer Science, pages 476–488. Springer, August 2001.

[GP01] Cyril Gavoille and David Peleg. Compact and localized distributed data structures. Research Report RR-1261-01, LaBRI, University of Bordeaux, 351, cours de la Libération, 33405 Talence Cedex, France, August 2001.

[GP03] Cyril Gavoille and Christophe Paul. Optimal distance labeling schemes for interval and circular-arc graphs. In G. Di Battista and U. Zwick, editors, 11^{th} *Annual European Symposium on Algorithms (ESA)*, volume 2832 of Lecture Notes in Computer Science, pages 254–265. Springer, September 2003.

[GR88] Ramalingam G. and P. Rangan. A unified approach to dominating problems on interval graphs. *Information Processing Letters*, 27:271–274, 1988.

[KKP00] Michal Katz, Nir A. Katz, and David Peleg. Distance labeling schemes for well-separated graph classes. In H. Reichel and S. Tison, editors, 17^{th} *Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 1770 of Lecture Notes in Computer Science, pages 516–528. Springer Verlag, February 2000.

[Pel99] David Peleg. Proximity-preserving labeling schemes and their applications. In Peter Widmayer, Gabriele Neyer, and Stephan Eidenbenz, editors, 25^{th} *International Workshop, Graph - Theoretic Concepts in Computer Science (WG)*, volume 1665 of Lecture Notes in Computer Science, pages 30–41. Springer, June 1999.

[Pel00a] David Peleg. *Distributed Computing: A Locality-Sensitive Approach.* SIAM Monographs on Discrete Mathematics and Applications, 2000.

[Pel00b] David Peleg. Proximity-preserving labeling schemes. *Journal of Graph Theory*, 33:167–176, 2000.

[PS89] David Peleg and Alejandro A. Schäffer. Graph spanners. *Journal of Graph Theory*, 13(1):99–116, 1989.

[PS97] Andreas Parra and Petra Scheffler. Characterizations and algorithmic applications of chordal graphs embeddings. *Discrete Applied Mathematics*, 79(1-3):171–188, 1997.

[PU89a] David Peleg and Jeffrey D. Ullman. An optimal synchornizer for the hypercube. *SIAM Journal on Computing*, 18:740–747, 1989.

[PU89b] David Peleg and Eli Upfal. A trade-off between space and efficiency for routing tables. *Journal of the ACM*, 36(3):510–530, July 1989.

[Ree92] Bruce Reed. Finding approximate separators and computing treewidth quickly. In 24^{th} *Annual ACM Symposium on Theory of Computing (STOC)*, pages 221–228, 1992.

[RS86] Neil Robertson and Paul D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *Journal of Algorithms*, 7:309–322, 1986.

[RTL76] Donald Rose, R. Endre Tarjan, and George S. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM Journal on Computing*, 5:266–283, 1976.

[Tar85] Robert Endre Tarjan. Decomposition by clique separators. *Discrete Mathematics*, 55:221–232, 1985.

[Ueh99] Ryuhei Uehara. Tractable and intractable problems on generalized chordal graphs. Technical Report COMP98-83, IEICE, Faculty of Natural Sciences, Komazawa University, Japan, March 1999.

[Whi81] Sue Hays Whitesides. An algorithm for finding clique cut-sets. *Information Processing Letters*, 12:31–32, 1981.