Generation of Graphs with Bounded Branchwidth[⋆]

Christophe Paul[1], Andrzej Proskurowski[2], and Jan Arne Telle[3]

[1] CNRS - LIRMM, Montpellier, France
`paul@lirmm.fr`
[2] Department of Computer and Information Science, University of Oregon, USA
`andrzej@cs.uoregon.edu`
[3] Department of Informatics, University of Bergen, Norway (Research conducted while on sabbatical at LIRMM)
`telle@ii.uib.no`

Abstract. Branchwidth is a connectivity parameter of graphs closely related to treewidth. Graphs of treewidth at most k can be generated algorithmically as the subgraphs of k-trees. n this paper, we investigate the family of edge-maximal graphs of branchwidth k, that we call k-branches. The k-branches are, just as the k-trees, a subclass of the chordal graphs where all minimal separators have size k. However, a striking difference arises when considering subgraph-minimal members of the family. Whereas K_{k+1} is the only subgraph-minimal k-tree, we show that for any $k \geq 7$ a minimal k-branch having q maximal cliques exists for any value of $q \notin \{3, 5\}$, except for $k = 8, q = 2$. We characterize subgraph-minimal k-branches for all values of k. Our investigation leads to a generation algorithm, that adds one or two new maximal cliques in each step, producing exactly the k-branches.

1 Introduction

Branchwidth and treewidth are mutually related connectivity parameters of graphs: whenever one of these parameters is bounded by some fixed constant for a graph, then so is the other [17]. Since many graph problems that are NP-hard in general can be solved in linear time when restricted to such classes of graphs both treewidth and branchwidth have played a large role in many investigations in algorithmic graph theory. Tree-decompositions have traditionally been the choice when solving NP-hard graph problems by dynamic programming to give FPT algorithms when parameterized by treewidth, see e.g. [2,16] for overviews. Recently it is the branchwidth parameter that has been in the focus of several algorithmic research results. For example, several papers [7,5,8,9,6] show that for graphs of bounded genus the base of the exponent in the running time of these FPT algorithms could be improved by the dynamic programming following instead a branch-decomposition of optimal branchwidth. Also, a strong

---

[⋆] This extended abstract does not contain all proofs. For a full version, please refer to [14] or contact one of the authors.

F.V. Fomin (Ed.): WG 2006, LNCS 4271, pp. 205–216, 2006.

heuristic algorithm for the travelling salesman problem [4] has been developed based on branch-decompositions and an exact (exponential-time) algorithm has been given to compute branchwidth [10]. Given these recent developments in favor of branchwidth one may wonder why treewidth has historically been preferred over branchwidth? Mainly, this is because of the equivalent definition of '*G has treewidth* $\leq k$*' by 'G is a partial k-tree*'. This alternative definition is intuitively appealing since the k-trees are the graphs generated by the following very simple algorithm: '*Start with* K_{k+1}*; Repeatedly choose a k-clique C and add a new vertex adjacent to vertices in C*'. Can we define branchwidth in an analogous algorithmic way? This is the question that has inspired our research and in this paper we give an affirmative answer.

We start by investigating the family of edge-maximal graphs of branchwidth k, that we call k-branches. The k-branches are chordal, as can be easily deduced from earlier work on branchwidth [11,10]. In Section 2 we report on related work [13] where we have given a characterization of k-branches. In Section 3 we consider subgraph-minimal k-branches. They form the starting graphs of our algorithm generating k-branches, just as the minimal k-tree K_{k+1} is the starting graph of the generation algorithm for k-trees. K_n has branchwidth $\lceil 2n/3 \rceil$ for any $n \geq 3$ and $K_{\lfloor 3(k-1)/2 \rfloor + 1}$ is one of the minimal k-branches. However, for $k \geq 7$ we find that there is a minimal k-branch on q maximal cliques for any $q \notin \{3, 5\}$, except for the pathological case $k = 8, q = 2$. We show that the minimal k-branches have clique trees that are caterpillars and give a characterization of the family of minimal k-branches for all values of k. Our investigation culminates in Section 4 with a non-deterministic generation algorithm, that adds one or two new maximal cliques in each step, yielding as output exactly the graphs that are k-branches, and whose spanning subgraphs (i.e., partial graphs) are exactly the graphs of branchwidth at most k. Our results lead to a better understanding of the branchwidth parameter by defining graphs of branchwidth k through the algorithmic concept of partial k-branches. The algorithm will generate a random graph of branchwidth k together its branch-decomposition and can be used to provide test instances for optimization codes based on branch-decomposition.

2 Definitions and Earlier Results

A *branch-decomposition* (T, μ) of a graph G is a tree T with nodes of degree one and three only, together with a bijection μ from the edge-set of G to the set of degree-one nodes (leaves) of T. For an edge e of T let T_1 and T_2 be the two subtrees resulting from $T \setminus \{e\}$, let G_1 and G_2 be the graphs induced by the edges of G mapped by μ to leaves of T_1 and T_2 respectively, and let $mid(e) = V(G_1) \cap V(G_2)$. The width of (T, μ) is the size of the largest $mid(e)$ thus defined. For a graph G its *branchwidth* $bw(G)$ is the smallest width of any branch-decomposition of G. [1]

---

[1] The connected graphs of branchwidth 1 are the stars, and constitute a somewhat pathological case. To simplify certain statements we therefore restrict attention to graphs having branchwidth $k \geq 2$.

A tree-decomposition $(T, \mathcal{X})$ of a graph G is an arrangement of the vertex subsets $\mathcal{X}$ of G, called bags, as nodes of the tree T such that for any two adjacent vertices in G there is some bag containing them both, and for each vertex of G the bags containing it induce a connected subtree. For a subtree T' of T the induced tree-decomposition $(T', \mathcal{X}')$ is the result of removing from $(T, \mathcal{X})$ all nodes of $V(T) \setminus V(T')$ and their corresponding bags.

Definition 1. *A k-*troika *(A, B, C) of a set X are 3 subsets of X such that $|A| \leq k$, $|B| \leq k$, $|C| \leq k$, and $A \cup B = A \cup C = C \cup B = X$. (A, B, C)* respects *$S_1, S_2, ..., S_q$ if any $S_i, 1 \leq i \leq q$ is contained in at least one of A, B or C.*

A necessary condition for a graph to be a k-branch is that it is a chordal graph where all minimal separators have size k, with the property that every maximal clique has a k-troika respecting the minimal separators contained in it [13]. This motivates the following definition.

Definition 2. *Let G be a chordal graph with C_G its set of maximal cliques and S_G its set of minimal separators. A tree-decomposition $(T, \mathcal{X})$ of G is called k-*full *if the following conditions hold: 1) The set of bags $\mathcal{X}$ is in 1-1 correspondence with $C_G \cup S_G$ (we call the nodes with bags in C_G the* maxclique nodes *and the nodes with bags in S_G the* minsep nodes*.) 2) The bags of the minsep nodes all have cardinality k. 3) There is an edge ij in the tree T iff $X_i \in S_G, X_j \in C_G$ and $X_i \subseteq X_j$. 4) Every maxclique bag X_j has a k-troika respecting its neighbor minsep bags.*

Note that if G has a k-full tree-decomposition then it is unique. We need additional constraints on k-full tree-decompositions to characterize exactly the k-branches.

Definition 3. *A* mergeable subtree *of a k-full tree-decomposition $(T, \mathcal{X})$ of a graph G is a subtree T' of T that: contains at least one edge, has leaves that are maxclique nodes, and satisfies:*

1. *$|\{v : v \in X$ where X a node in $T'\}| \leq \lfloor 3k/2 \rfloor$*
2. *Either the subtree T' has at most one node that in T has a neighbor in $V(T) \setminus V(T')$ or else T' is a path X, B, Y with X, B, Y and all their neighbors in T inducing a path A, X, B, Y, C satisfying $B \setminus (A \cup C) = \emptyset$.*

Lemma 1. *Let $A - X - B - Y - C$ be a path in T for some k-full tree-decomposition $(T, \mathcal{X})$ with X and Y maxclique nodes. $X \cup Y$ has a k-troika respecting A, C if and only if $|X \cup Y| \leq \lfloor 3k/2 \rfloor$ and $B \setminus (A \cup C) = \emptyset$.*

Proof. If $|X \cup Y| > \lfloor 3k/2 \rfloor$ then $X \cup Y$ does not have a k-troika. Let $P = B \setminus (A \cup C)$. Note that we have $A \cap C \subseteq B$ and since $|A| = |C| = k$ we have $|A \cap C| = 2k - |(X \cup Y) \setminus P|$. But then $|X \cup Y| + |A \cap C| = 2k + |P|$ and this means that by Theorem 2 of [15] (also by results of [11]) $X \cup Y$ has a k-troika respecting A, C if and only if $P = \emptyset$.

Lemma 1 is implicit in [13], and implies that for mergeable subtree T' we can add edges to G to make a clique of $\{v : v \in X$ where X a node in $T'\}$ without increasing branchwidth of G.

Definition 4. *A k-full tree-decomposition $(T, \mathcal{X})$ of a graph G is a* k-skeleton *of G if G has at least $\lfloor 3(k-1)/2 \rfloor + 1$ vertices and T does not have a mergeable subtree.*

Theorem 1. *[13] G is a k-branch $\Leftrightarrow$ G has a k-skeleton*

3 Minimal *k*-Branches

We characterize the subgraph-minimal k-branches on q maximal cliques, by describing the structure of the minimal k-skeletons, as defined below. We divide the caracterization into two Theorems, one for the cases when $k \leq 6$ or $q \leq 5$ and the other for the cases $k \geq 7, q \geq 6$.

Definition 5. *A k-branch G is a* minimal k-branch *if no strict subgraph of G is a k-branch. Let the set of* minimal k-skeletons *be $MS(k) = \{(T, \mathcal{X}) : (T, \mathcal{X})$ is a k-skeleton but for no proper subtree T' of T is the induced tree-decomposition $(T', \mathcal{X}')$ a k-skeleton$\}$. Let $MS(k, q)$ be the set of minimal k-skeletons on q maxclique nodes.*

If G is a minimal k-branch then for its k-skeleton $(T_G, \mathcal{X})$ we have $(T_G, \mathcal{X}) \in MS(k)$. However, the graph represented by a minimal k-skeleton may have some cliques that are too big for it to be a minimal k-branch. For example, if $(T, \mathcal{X})$ is the tree T having a single maxclique node on 6 vertices then we have $(T, \mathcal{X}) \in MS(4)$ since it is a minimal 4-skeleton but the graph K_6 that it represents is not a minimal 4-branch since it contains the 4-branch K_5 as a subgraph. Since our algorithm in Section 4 builds k-skeletons, rather than graphs, we focus in the following on the minimal k-skeletons.

Lemma 2. *In a minimal k-skeleton $(T, \mathcal{X})$, the tree T does not contain a maxclique leaf X with path $X - A - Y$ and both A and Y having degree 2.*

Lemma 3. *In a minimal k-skeleton $(T, \mathcal{X})$, any minsep node S of degree larger than 2 must have degree 3 with exactly one of its neighbors being a maxclique leaf and the other two having degree 2.*

Lemma 4. *In a minimal k-skeleton $(T, \mathcal{X})$, any maxclique node X of degree 3 has all 3 minsep neighbors A_1, A_2, A_3 of degree 2 and at least one of them has a maxclique leaf as neighbor.*

See Figure 1 for an illustration of the following Theorem, which characterizes the minimal k-skeletons on q maximal cliques when $k \leq 6$ or $q \leq 5$.

Theorem 2. *1. For $k \geqslant 2$, $MS(k, 1)$ contains $K_{\lfloor 3(k-1)/2 \rfloor + 1}$ and if k even then also $K_{\lfloor 3(k-1)/2 \rfloor + 2}$.*

2. *For* $k \leqslant 6$ *and* $k = 8$, $MS(k,2) = \emptyset$. *For* $k = 7$ *and* $k \geq 9$, $MS(k,2)$ *is nonempty and consists of the trees with two maxclique nodes of size* $x+k$ *and* $y+k$ *(with* k *common vertices) for any* $x \leq y$ *satisfying*

$$3 - (k \bmod 2) \leq x \leq y \leq \lceil k/2 \rceil - 2 \qquad \textit{and} \qquad x + y \geq \lfloor k/2 \rfloor + 1 \quad (1)$$

3. *For any* k, $MS(k,3) = \emptyset$.
4. *For* $k \leqslant 4$ *and* $k = 6$, $MS(k,4) = \emptyset$. *For* $k = 5$ *and* $k \geq 7$ $MS(k,4)$ *is nonempty, and consists of the* k*-full tree-decompositions* $(T,\mathcal{X})$ *on* $q = 4$ *maxclique nodes* X_1, X_2, X_3, Y *with* Y *a node of degree 3 in* T *such that*
   (a) $|Y|, |X_i| \leqslant \lfloor \frac{3(k-1)}{2} \rfloor$ *for any* $i \in [1,3]$*;*
   (b) *for any* $i \in [1,3]$, $|X_i \cup Y| \leqslant \lfloor \frac{3k}{2} \rfloor$*;*
   (c) $\lfloor \frac{3k}{2} \rfloor + 1 \leqslant |X_i \cup Y \cup X_j|$ *with* $1 \leq i < j \leq 3$
5. *For any* k, $MS(k,5) = \emptyset$
6. *For any* $k \leqslant 6$ *and* $q \geqslant 6$, $MS(k,q) = \emptyset$.

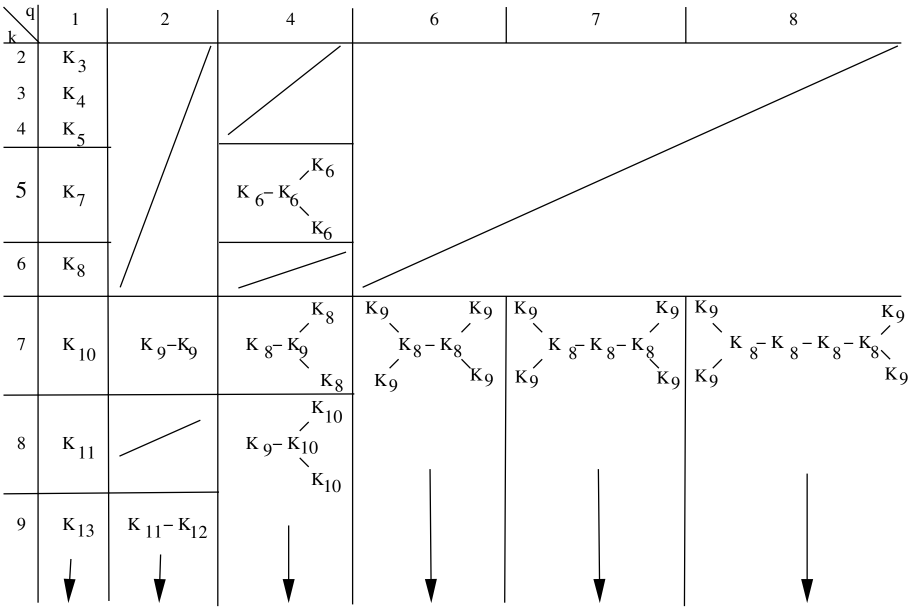

Fig. 1. Examples of minimal k-skeletons $(T,\mathcal{X})$ on q maxclique nodes, for $k \leq 9, q \leq 8$. Downward arrows indicate that minimal k-sketelons, with trees isomorphic to those depicted, exist also for larger k. Only the maxclique nodes are drawn. The minsep nodes have size k and appear on each edge of the trees. Only for the case $q = 8$ does the intersection of minsep nodes matter. For $q = 8$, if $A - X - B - Y - C$ the path with X and Y the maxclique nodes of degree two then minsep nodes A, B, C must satisfy $B \setminus (A \cup C) \neq \emptyset$.

From this characterization of minimal k-skeletons we can deduce the characterization of minimal k-branches. For lack of space we only sketch how to do this for the case $q = 2$. Note that if two distinct pairs $x \leq y$ and $x' \leq y'$ both satisfy Equations (1) then the graph associated with the first pair is a subgraph of the graph associated with the second pair if and only if $x \leq x'$ and $y \leq y'$. Thus, the minimal k-branches on $q = 2$ maximal cliques correspond with such smallest pairs $x \leq y$.

To describe the minimal k-skeletons for $k \geq 7$ having $q \geq 6$ the following definition of the adjacencies in a special caterpillar T will be useful (see also Figure 2).

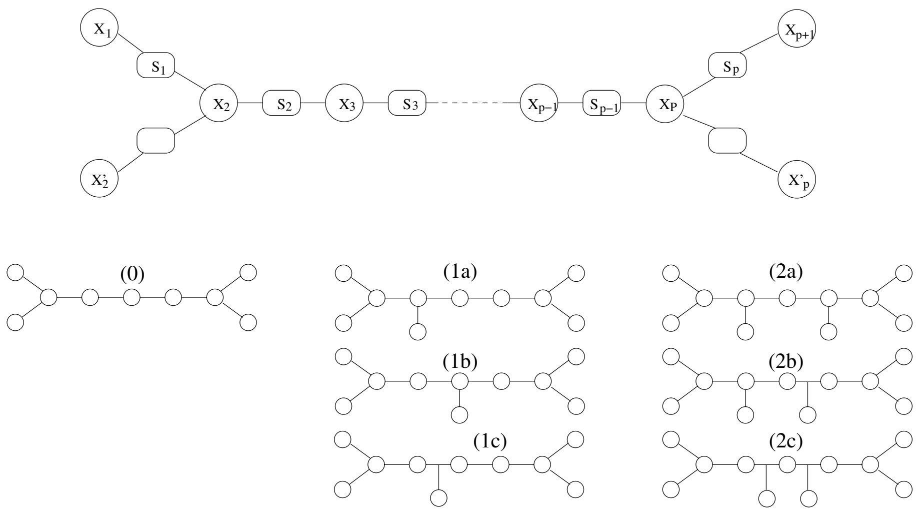

Fig. 2. On top is a special caterpillar with names of nodes as in Definition 6, in Theorem 3 and in Algorithm Stage 1. Below are the 7 non-isomorphic special caterpillars with $p = 6$, with maxclique nodes drawn as circles and minsep nodes not drawn explicitly but present on any edge between two adjacent maxclique nodes. Thus, 1c, 2b, 2c have minsep nodes of degree 3.

Definition 6. *A tree T is a* special caterpillar *if T consists of a body which is a path $X_1, S_1, X_2, S_2, \ldots X_p, S_p, X_{p+1}$ alternating between maxclique and minsep nodes for some $p \geq 3$ with added hairs of length one or two (a hair of length one being a new maxclique node added as neighbor of a minsep node of the body, and a hair of length two being two new adjacent maxclique-minsep nodes with the minsep node added as neighbor of a maxclique node of the body) satisfying the following conditions:*

1. *at most one hair for each node of the body*
2. *no hair on any of $X_1, S_1, S_2, S_{p-1}, S_p, X_{p+1}$*
3. *hair X_2' on X_2 and hair X_p' on X_p, but no hair on minseps $X_2' \cap X_2, X_p' \cap X_p$*
4. *if hair on S_i then no hair on X_i and no hair on X_{i+1}*
5. *if hair on X_i then not hairs on both of X_{i-1} and X_{i+1}*

Theorem 3. *$(T, \mathcal{X})$ is a minimal k-skeleton for some $k \geq 7$ on at least $q \geq 6$ maxclique nodes $\Leftrightarrow$ $(T, \mathcal{X})$ is a k-full tree-decomposition with T a special caterpillar whose bags satisfy (bag names as in Definition 6 and Figure 2):*

1. *either $|X_1 \cup X_2 \cup X_3| \leq 3k/2$ or $|X'_2 \cup X_2 \cup X_3| \leq 3k/2$ and also either $|X_{p+1} \cup X_p \cup X_{p-1}| \leq 3k/2$ or $|X'_p \cup X_p \cup X_{p-1}| \leq 3k/2$*
2. *$|X_1 \cup X_2 \cup X'_2| > 3k/2$ and $|X_{p+1} \cup X_p \cup X'_p| > 3k/2$*
3. *For maxcliques X, Y with a common neighbor, $|X| \leq \lfloor 3(k-1)/2 \rfloor$ and $|X \cup Y| \leq 3k/2$*
4. *If S_i has a hair then $S_i \setminus (S_{i-1} \cup S_{i+1}) = \emptyset$*
5. *If X_i has a hair then either i) no hair on X_{i-1} and $S_{i-1} \setminus (S_{i-2} \cup S_i) = \emptyset$ or ii) no hair on X_{i+1} and $S_i \setminus (S_{i-1} \cup S_{i+1}) = \emptyset$*
6. *If no hair on neither of X_i, S_i, X_{i+1} then $S_i \setminus (S_{i-1} \cup S_{i+1}) \neq \emptyset$*

Proof. $\Leftarrow$: We first show that the k-full tree-decomposition $(T, \mathcal{X})$ is a k-skeleton, by showing that T does not have a mergeable subtree as in Definition 3. Any subtree T' having at most one node that in T has a neighbor in $V(T) \setminus V(T')$ is by condition 2 not mergeable since we would have $|\{v : v \in X \text{ and } X \text{ a maxclique node in } T'\}| > \lfloor 3k/2 \rfloor$. Any subtree T' which is a path X, B, Y with X, B, Y and all their neighbors in T inducing a path A, X, B, Y, C will by condition 6 satisfy $B \setminus (A \cup C) \neq \emptyset$ and is thus not mergeable. Thus $(T, \mathcal{X})$ is a k-skeleton and it remains to show that it is a minimal k-skeleton. We prove by contradiction, that for any proper subtree T' of T the induced tree-decomposition $(T', \mathcal{X}')$ is not a k-skeleton. Unless the graph G' that $(T', \mathcal{X}')$ represents has at least $\lfloor 3(k-1)/2 \rfloor + 1$ vertices, $(T', \mathcal{X}')$ is not a k-skeleton. By condition 3 this means that T' must contain at least 2 maxclique nodes. We show that in any such T' there is a mergeable subtree T''. There are 5 special cases of subtrees T' to consider:

1. Suppose maxclique bags of T' are X_1, X_2, X'_2, X_3 or $X_{p-1}, X_p, X'_p, X_{p+1}$. In both cases the 3 maxclique bags satisfying the size constraint in condition 1 form the mergeable subtree T".
2. T' contains a leaf X having a minsep neighbor S of degree 2 that itself has neighbor Y. By condition 3, X, S, Y makes up the mergeable subtree T''.
3. T' contains two maxclique leaves X, Y with a common minsep neighbor S. Again by condition 3 X, S, Y is the mergeable subtree.
4. Suppose in T there was a hair on minsep S_i and that T' does not contain this hair but does contain $S_{i-1}, X_i, S_i, X_{i+1}, S_{i+1}$. In this case the mergeable subtree T'' is X_i, S_i, X_{i+1} by condition 4 and Lemma 1.
5. Suppose T has a hair on maxclique X_i and that T' does not contain this hair but that T' does contain $X_{i-1}, S_{i-1}, X_i, S_i, X_{i+1}$. Since T is a special caterpillar, neither S_{i-1} nor S_i has a hair. Thus, condition 5 and Lemma 1 guarantee that either the subtree X_{i-1}, S_{i-1}, X_i or the subtree X_i, S_i, X_{i+1} is mergeable.

$\Rightarrow$: Using Lemmas 2, 3 and 4. we can describe all trees $T \in MS(k)$. There are two trees containing respectively 1 and 2 maxclique nodes except for $k = 8$ (see

Theorem 2). For the remaining trees we note that Lemmas 2, 3 and 4 together imply that for any maxclique leaf X in T with parent A we have either: A of degree 3 with the other two neighbors of A having degree 2 and not being leaves (call these leaves of type i); or A of degree 2 with parent Y of degree 3 having 3 neighbors of degree 2 with 1, 2 or 3, respectively, of these being neighbors of a leaf (leaves of type ii.1, ii.2, ii.3 respectively.) Moreover, all nodes of degree 3 in T (which is the maximum) have at least one neighbor that is a leaf or neighbor of a leaf. Thus we can use the 4 types (i, ii.1, ii.2, ii.3) as building-blocks for any tree $T \in MS(k)$. If we use a building-block of type ii.3) then there is only a unique tree possible, with 4 maxclique nodes, covered already in Theorem 2. Building blocks of type i) and ii.1) contain one leaf and two nodes needing new neighbors, while type ii.2) contains two leaves and one node needing a new neighbor. Thus, when using building-blocks of types i), ii.1) or ii.2) we must always have exactly two building-blocks of type ii.2), that will correspond to two ends of the body of a caterpillar having hairs of length 1 (type ii.1) or 2 (type i). A minsep node of degree 3 cannot be adjacent to a maxclique node of degree 3, because the maxclique hair of this minsep could then have been dropped and we would still have an induced k-skeleton. Likewise, no three consecutive maxclique nodes of the body all have a hair since then the middle hair could have been dropped and we would still have an induced k-skeleton. Thus, T is a special caterpillar.

To end the proof, it suffices to note that conditions 1-6 of the Theorem hold, since otherwise T would either have had a mergeable subtree or it would have been a k-skeleton but not minimal. For example, if condition 6 did not hold for some i then by Lemma 1 $X_i - S_i - X_{i+1}$ would have been a mergeable subtree. For space reasons we do not give the details of all cases.

4 An Algorithm That Generates *k*-Branches

In this section we give an algorithm generating each possible k-skeleton, which by Theorem 1 will correspond to generation of the k-branches.

Definition 7. *We get an* extended *k-skeleton by taking a k-skeleton $(T, \mathcal{X})$ and adding zero or more minsep leaves with bag-size k as neighbors of maxclique nodes of T while ensuring that each maxclique node still has a k-troika respecting its minsep neighbors. When starting with a minimal k-skeleton $(T, \mathcal{X})$ the result is an* extended minimal *k-skeleton. We define $ES(k)$ to be the set of extended k-skeletons and $EMS(k)$ to be the set of extended minimal k-skeletons.*

Recall that $MS(k)$ are the minimal k-skeletons, and note that by definition $MS(k) \subseteq EMS(k) \subseteq ES(k)$. The algorithm is organised in 3 stages with the outputs of the previous stage forming the inputs to the next stage. STAGE 1 generates $MS(k)$, STAGE 2 generates $EMS(k)$ and STAGE 3 generates $ES(k)$. Note that the extended k-skeletons $ES(k)$ have the dual property that we get a k-skeleton both if we remove all minsep leaves and also if we add a new legal maxclique leaf to each minsep leaf. For our generation algorithm this implies that generating k-branches is equivalent to generating extended k-skeletons where all leaves are maxclique nodes. The reason we generate extended k-skeletons, and not only the k-skeletons, is to be able to enforce that all eventual minsep neighbors of a maxclique node are added as soon as the maxclique node is added. This to easily satisfy the constraint that a maxclique node have a k-troika respecting its minsep neighbors.

Description of STAGE 1*:* Generation of the minimal k-skeletons $MS(k)$.
See Algorithm 1. The minimal k-skeletons on 1, 2, 4, or 6 maxclique nodes are generated by the special rules `1clique, 2clique, 4clique` or `6clique` respectively. The special caterpillar T in a minimal k-skeleton $(T, \mathcal{X})$ on 6 maxclique nodes is unique, $p = 6$ in Definition 6

For the minimal k-skeletons $(T, \mathcal{X})$ on more than 6 maxcliques we enter a Repeat-loop that will generate the special caterpillar T from left to right by adding in each iteration one or two new maxclique nodes to the current right end of its body. The Repeat-loop is prefixed and postfixed by special operations `Start` and `End` that add the building-blocks that in the proof of Theorem 3 are called type ii.2). Note that throughout the code the names of parameters denoting maxclique and minsep nodes are in accordance with Definition 6 and Figure 2, with certain exceptions. In particular, in the prefix operation `Start`$(X_1, X_2, X_2', X_3, Hair, S_3)$ the parameter $Hair$ is a maxclique leaf hair of length 2 added to maxclique X_3.

In the repeat-loop we maintain the loop invariant that S_i will be the minimal separator at the current right end of the body at which construction of the caterpillar will continue. Throughout STAGE 1 we make the assumption that when adding a new maxclique node X adjacent to some minsep node S then for any other neighbor Y of S the pair X, Y must satisfy condition 3 of Theorem 3. To not clutter the code we do not explicitly state these conditions. When adding new maxclique nodes, both here and in *Stage 3*, the syntax for the operation is `ADD`$(oldsep, newclique, newsep1, newsep2)$, where the two latter parameters may be missing. The *newclique* node is added as a neighbor of *oldsep* and the *newsep* nodes are added as neighbors of *newclique*. Thus, to extend the rightmost end of the body by a path S_i, X_{i+1}, S_{i+1} we use in `rule I` and `rule II` the operation `ADD`(S_i, X_{i+1}, S_{i+1}). In `rule III` we are additionally adding a hair of length two consisting of minsep B and maxclique $Hair$ to the new maxclique node X_{i+1} and express this by the two operations `ADD`$(S_i, X_{i+1}, B, S_{i+1})$ and `ADD`$(B, Hair)$. In `rule IV` we are additionally adding a hair W of length one to minsep node S_i and express this by the additional operation `ADD`(S_i, W). The boolean values `HasHair` and `NeedsPair` govern which of `rule I` to `rule IV` can be applied while ensuring that the conditions for minimal k-branches are fullfilled. `HasHair` is True iff the rightmost maxclique node X_i of the current body has a hair H attached to it. `NeedsPair` is True iff `HasHair` is True and the next-to-last maxclique node X_{i-1} would not be mergeable with X_i even if we had removed the hair H (in which case the next maxclique node X_{i+1} must satisfy that X_{i+1} and X_i would be mergeable if we had removed H.)

Algorithm 1. Stage 1: Generate any $(T, \mathcal{X}) \in MS(k)$ by choosing 1,2,3,4 or 5

```
1: $(T, \mathcal{X}) := 1clique(X)$ s.t. Theorem 2, case $q = 1$ holds ;
2: $(T, \mathcal{X}) := 2clique(X, Y)$ s.t. Theorem 2, case $q = 2$ holds ;
3: $(T, \mathcal{X}) := 4clique(X, Y, Z, W)$ s.t. Theorem 2, case $q = 4$ holds ;
4: $(T, \mathcal{X}) := 6clique(X_1, X_2, X_2', X_3, X_3', X_4)$ s.t. $T$ is the unique special
   caterpillar with $p = 3$ and $q = 6$ and conditions 1,2,3 in Theorem 3 hold ;
5: begin
     first choose a or b while ensuring that condition 1 and 2 of Theorem 3 hold;
     a: Start$(X_1, X_2, X_2', X_3, S_3)$, $i := 3$, HasHair:= 0, NeedsPair:= 0;
     b: Start$(X_1, X_2, X_2', X_3, Hair, S_3)$, $i := 3$, HasHair:= 1, NeedsPair:= 1;
     repeat
         if HasHair and NeedsPair then choose rule I;
         else if HasHair and not NeedsPair then choose rule I,II or III;
         else choose rule II, III or IV;
         rule I: ADD$(S_i, X_{i+1}, S_{i+1})$ s.t. $S_i \setminus (S_{i-1} \cup S_{i+1}) = \emptyset$;
         rule II: ADD$(S_i, X_{i+1}, S_{i+1})$ s.t. $S_i \setminus (S_{i-1} \cup S_{i+1}) \neq \emptyset$;
         rule III: ADD$(S_i, X_{i+1}, B, S_{i+1})$ and ADD$(B, Hair)$;
         rule IV: ADD$(S_i, X_{i+1}, S_{i+1})$ and ADD$(S_i, W)$ s.t. $S_i \setminus (S_{i-1} \cup S_{i+1}) = \emptyset$;
         if rule III was chosen then HasHair:= 1 and
         NeedsPair:= $(S_i \setminus (S_{i-1} \cup S_{i+1}) \neq \emptyset)$;
         else HasHair:= 0 and NeedsPair:= 0;
         $i := i + 1$;
     until body of caterpillar is finished and NeedsPair= 0 ;
     End$(S_i, X_i, X_i', X_{i+1})$ s.t. Thm 3 (cond. 1 and 2) holds with $i = p$;
   end
```

Description of STAGE 2: Generation of the set $EMS(k)$
For space reasons we do not show a separate algorithm in the text. The input to Stage 2 is a minimal k-skeleton $(T, \mathcal{X}) \in MS(k)$ as generated by Stage 1. Stage 2 is a repeat-loop that can be exited at any time and which in each iteration adds one new minsep leaf S as neighbor of some maxclique node X of $(T, \mathcal{X})$, according to Definition 7. We must ensure that X will still have a k-troika respecting its minsep neighbors. If X already had one neighbor A then condition `OK`(X, A, S) must hold, if it had two neighbors A, B then condition `OK`(X, A, B, S) must hold, while if it had three neighbors then a new neighbor cannot be added. These conditions are used also in Stage 3 and defined by: '`OK`(X, A, B) is True iff $|X| + |A \cap B| \leq 2k$' and '`OK`(X, A, B, C) is True iff $|A \cup B| = |A \cup C| = |B \cup C| = |X|$'.

Lemma 5. $EMS(k) = \{(T, \mathcal{X}) : \exists$ *sequence of choices in* Stage 1 *and in* Stage 2 *s.t.* Stage 2 *gives as output* $(T, \mathcal{X})\}$

Description of STAGE 3: Generate the set $ES(k)$.
See Algorithm 2. As in Stage 1 the rule adding a new maxclique node X adjacent to an existing minsep node A with new promise leaves B and C will have the syntax `ADD`(A, X, B, C). In case we have one or zero promise leaves the syntax is `ADD`(A, X, B) and `ADD`(A, X). The shorthand `ADD`$(A, X, ...)$ can be replaced by

any of the 3 rules. Similarly, the shorthand `OK`$(X, A, ...)$ appearing right after some `ADD`$(A, X, ...)$ has the intepretation that any third and fourth parameters B and C of the `ADD` also becomes a third and fourth parameter of the `OK`.

Algorithm 2. STAGE 3: Takes as input some $(T, X) \in EMS(k)$ produced by STAGE 2 and builds on this to produce as output an extended k-skeleton in $ES(k)$

```
repeat
    Choose a minsep node $A$ of $T$;
    if $A$ a leaf with parent $W$ having a single other neighbor $S$ then
        choose 1, 2, 3, 4 or 5;
        1: ADD$(A, New)$ s.t. $|W \cup New| > \lfloor 3k/2 \rfloor$;
        2: ADD$(A, New, B)$ s.t. OK$(New, A, B)$ and $|W \cup New| + |B \cap S| > 2k$;
        3: ADD$(A, New1)$ and ADD$(A, New2)$ s.t. $|New1 \cup New2| > \lfloor 3k/2 \rfloor$;
        4: ADD$(A, New1, B, ...)$ and ADD$(A, New2, ...)$ s.t. OK$(New1, A, B, ...)$ and
           OK$(New2, A, ...)$;
        5: ADD$(A, New, B, C)$ s.t. OK$(New, A, B, C)$;
    else
        choose 6 or 7;
        6: ADD$(A, New, B, ...)$ s.t. OK;
        7: ADD$(A, New)$ s.t. $|Y \cup New| > \lfloor 3k/2 \rfloor$ for $\forall Y$ maxclique leaf with
           parent $A$;
until done ;
Output extended $k$-skeleton $(T, \mathcal{X})$, which represents a $k$-branch iff it has no
minsep leaves;
```

Theorem 4. *$ES(k) = \{(T, \mathcal{X}) : \exists$ sequence of choices of rules in the 3 stages s.t. output is $(T, \mathcal{X})\}$*

5 Concluding Remarks

The results of this paper lead to a new understanding of the branchwidth parameter by defining graphs of branchwidth k through the algorithmic concept of partial k-branches. The given algorithm will generate the k-skeleton of a random edge-maximal graph of branchwidth k. This algorithm can be used to provide test instances for optimization codes based on branch-decomposition.

References

1. H.L. Bodlaender, T. Kloks and D. Kratsch. Treewidth and pathwidth of permutation graphs. *SIAM J.Computing*, 25:1305-1317, 1996.
2. H.L. Bodlaender. Treewidth: Algorithmic techniques and results. In *22nd International Symposium on Mathematical Foundations of Computer Science (MFCS)*. Vol. 1295 of *Lecture Notes in Computer Science*, p. 19–36, 1997.
3. H.L. Bodlaender and D.M. Thilikos. Graphs with branchwidth at most three. *Journal of Algorithms*, 32:167–194, 1999.

4. W. Cook and P.D. Seymour. Tour merging via branch-decompositions. *Journal on Computing*, 15:233–248, 2003.
5. E. Demaine, F. Fomin, M. Hajiaghayi, and D.M. Thilikos. Fixed-parameter algorithms for (k,r)-center in planar graphs and map graphs. In *30th Int. Colloquium on Automata, Languages, and Programming (ICALP).* Vol. 2719 of *Lecture Notes in Computer Science*, p. 829–844, 2003.
6. F. Dorn, E. Penninkx, H.L. Bodlaender and F.V. Fomin. Efficient Exact Algorithms on Planar Graphs: Exploiting Sphere Cut Branch Decompositions. In *13th European Symposium on Algorithm (ESA).* Vol. 3669 of *Lecture Notes in Computer Science*, p. 95-106, 2005.
7. F. Fomin and D.M. Thilikos. Dominating sets in planar graphs: Branch-width and exponential speedup. In *14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, p. 168–177, 2003.
8. F. Fomin and D.M. Thilikos. A simple and fast approach for solving problems on planar graphs. In *22nd Annual Symposium on Theoretical Aspect of Computer Science (STACS)* Vol. 2996 of *Lecture Notes in Computer Science*, p. 56-67, 2004.
9. F. Fomin and D. Thilikos. Fast parameterized algorithms for graphs on surfaces: Linear kernel and exponential speedup. In *31st International Colloquium on Automata, Languages, and Programming (ICALP)*, Vol. 3142 of *Lecture Notes in Computer Science*, p. 581-592, 2004.
10. F. Fomin, F. Mazoit and I. Todinca. Computing branchwidth via efficient triangulation and blocks. In *31st Workshop on Graph Theoretic Concepts in Computer Science (WG)*, Vol. 3787 of *Lecture Notes in Computer Science*, p. 374-384, 2005.
11. T. Kloks, J. Kratochvil, and H. Müller. New branchwidth territories. *Discrete Applied Mathematics.* 145:266-275, 2005.
12. J. Kleinberg and E. Tardos. Algorithm design. *Addison-Wesley*, 2005.
13. C. Paul and J.A. Telle. Edge-maximal graphs of branchwidth k. In *International Conference on Graph Theory - ICGT.* Vol. 23 *Electronic Notes in Discrete Mathematics*, 363-368, 2005.
14. C. Paul and A. Proskurowski and J.A. Telle. Algorithm generation of graphs of branchwidht $\leqslant k$. LIRMM Technical report number RR-05047. 2005.
15. C. Paul and J.A. Telle. New tools and simpler algorithms for branchwidth. In *13th European Symposium on Algorithm (ESA).* Vol. 3669 of *Lecture Notes in Computer Science*, p. 379-390, 2005.
16. B. Reed. Treewidth and tangles, a new measure of connectivity and some applications. In *Surveys in Combinatorics.* Vol. 241 of *London Mathematical Society Lecture Note Series* Cambridge University Press, 1997.
17. N. Robertson and P.D. Seymour. Graph minors X: Obstructions to tree-decomposition. *Journal on Combinatorial Theory Series B*, 52:153–190, 1991.
18. D. Rose. On simple characterization of k-trees. *Discrete Mathematics*, 7:317–322, 1974.