



The Branch-width of Circular-arc Graphs

F. Mazoit

LIF, Université de Provence
13453 Marseille Cedex 13, France

Abstract. We prove that the branch-width of circular-arc graphs can be computed in polynomial time.

Key words: Branch-width, circular-arc graphs, algorithm.

1 Introduction

The notions of tree-width and tree-decomposition of a graph have been introduced by Robertson and Seymour [1] for their graph minor project. These notions have been intensively investigated for algorithmic purposes and it is well known that many intractable problems can be solved in polynomial (and very often in linear) time when the input is restricted to graphs with bounded tree-width (see [2] for a comprehensive survey). While working on their graph minor project, Robertson and Seymour defined, in connection with tree-width, the notion of branch-width [3]. They proved that for any graph G , $\text{bw}(G) \leq \text{tw}(G) + 1 \leq 1.5\text{bw}(G)$. Both bounds are tight and achievable on trees and complete graphs. Branch-width appeared to be an even more appropriate tool than tree-width for the graph minor theory. Since both parameters are so close, one can expect the algorithmic behaviour of these problems to be quite similar. However, this is not true. For example, on planar graphs branch-width can be computed in polynomial time [4] while computing the tree-width of a planar graph in polynomial time is a long standing open problem. An even more striking example was observed by Kloks et al. [5]: deciding the branch-width of a split-graph is NP-hard while deciding the tree-width of a split-graph can be done in linear time.

In [6], the author studied the relation between both tree-decompositions and branch-decompositions and, in particular, how they can be associated to triangulations in a similar way. Using his techniques, Fomin et al. [7] describe the analogue of minimal triangulations and potential maximal cliques for branch-width: *efficient triangulations* and *blocks*. They also note that, using a large enough family of blocks together with their *block branch-width*, it is possible to compute the branch-width of any graph in exponential time. The algorithm is essentially the same as the one Fomin et al. use [8] to compute the tree-width.

In this article, we use the same framework to show that it is possible to compute the branch-width of circular-arc graphs in polynomial time. Section 3 is devoted to a proof of the *efficient triangulation* theorem which is simpler than the original one [6]. Section 4 presents some results of [7] and Sect. 5 shows how to use these tools to compute the branch-width of circular-arc graphs.

2 Preliminaries

Throughout this paper, G is a graph with vertex set V and edge set E , $n = |V|$ and $m = |E|$. The *neighbourhood* $N(x)$ of a vertex x is the set of vertices adjacent to x . The *neighbourhood* of a set of vertices C is the set of vertices not in C that are adjacent to at least one vertex of C .

A set of vertices S is a *separator* if $G \setminus S$ has at least two connected components, an *a, b -separator* if a and b are in different connected components of $G \setminus S$, an *a, b -minimal separator* if no proper subset of S is an a, b -separator. The connected component of a in $G \setminus S$ is $C_a(S)$. The component $C_a(S)$ is a *full connected component* if S is the neighbourhood of $C_a(S)$. For an a, b -minimal separator S , both $C_a(S)$ and $C_b(S)$ are full. A set S is a *minimal separator* if there exist a and b such that S is an a, b -minimal separator or, which is equivalent, if $G \setminus S$ has at least two full connected components.

A *clique* of a graph G is a set of vertices of G that are pairwise adjacent in G . The maximum clique size of G is denoted by $\omega(G)$. A graph is *chordal* (or *triangulated*) if every cycle of length at least four has a chord, that is an edge between two non-consecutive vertices of the cycle.

Theorem 1 ([9]). *A graph is chordal if and only if it is the intersection graph of a family of sub-trees of a tree.*

If H is the intersection graph of a family $\{T_x \mid x \in V(H)\}$ of sub-trees of a tree T , every maximal clique Ω of H can be associated to a vertex v_Ω of T such that the set of vertices whose sub-tree contains v_Ω is exactly Ω . The minimal separators of H can be associated to edges of T in a similar way.

A graph H is a *super-graph* of a graph G if H and G have the same vertices and every edge of G is an edge of H . A *triangulation* of a graph G is a chordal super-graph of G . A triangulation H of G is *minimal* if no strict sub-graph of H is a triangulation of G .

Definition 1 (Efficient triangulation). *A triangulation H of G is efficient if*

1. *each minimal separator of H is also a minimal separator of G ;*
2. *for each minimal separator S of H , the connected components of $H \setminus S$ are exactly the connected components of $G \setminus S$.*

Note that according to a result of Parra and Scheffler [10], minimal triangulations are efficient.

If X is a set of edges, $V(X)$ (the vertices of X) denotes the set of vertices incident to X . The *border* of X , $\delta(X)$, is the set of vertices $V(X) \cap V(E \setminus X)$. A *pack* of a set of vertices S is either an edge whose ends belong to S or the set of edges incident to a connected component of $G \setminus S$. Note that if X is a set of edges, a pack of $\delta(X)$ is either a subset of X or disjoint from X . We can thus define the *packs* of a set of edges X as being the packs of $\delta(X)$ that are subsets of X .

The notion of branch-width is due to Robertson and Seymour [3]. A *branch-decomposition* T of a graph G is a pair (T, τ) with T a ternary

tree and τ a bijective mapping from the leaves of T to the edges of G . The vertices of T are its *nodes*. For any edge e of \mathcal{T} , the two connected components $T_1^*(e)$ and $T_2^*(e)$ of $T \setminus e$ are the *e-branches* of \mathcal{T} . A *branch* of \mathcal{T} is a *e-branch* for some edge e . The set of edges of G mapped on the leaves of a branch T^* is its *ground*. By using the ground, we can define the packs, the border and the set $V(T^*)$ of a branch T^* . We also extend the definition of packs and border to the edges of a branch-decomposition. A branch-decomposition \mathcal{T} is *compatible* with a set of vertices S if S is a subset of at least one border of \mathcal{T} . The maximum size of the border of an edge of \mathcal{T} denoted by $\text{bw}(\mathcal{T})$ is called the *width* of the branch-decomposition. The *branch-width* ($\text{bw}(G)$) of a graph G is the minimum width of one of its branch-decompositions. Note that the definitions of branch-decomposition and branch-width also apply to hyper-graphs. The notion of branch-width is closely related to the well-known notion of tree-width. The *tree-width* of a graph G ($\text{tw}(G)$) is the minimum of $\omega(H)$ over the triangulations H of G . In particular, Robertson and Seymour [3] showed that $\text{bw}(G) \leq \text{tw}(G) + 1 \leq \lfloor 2 \text{bw}(G) / 3 \rfloor$. The branch-decompositions of a graph can also be associated to triangulations. Indeed, given a branch-decomposition \mathcal{T} of a graph G , we can associate to each vertex x of G the subtree T_x of T covering all the leaves of T containing edges incident to x . The border of a branch e of \mathcal{T} is exactly the set of vertices x of G such that e belongs to T_x . The intersection graph of the subtrees T_x is the triangulation $H_{\mathcal{T}}$ of G associated with \mathcal{T} .

Proposition 1. *Let G be a graph, \mathcal{T} be any branch-decomposition of G of optimal width and $H_{\mathcal{T}}$ be the triangulation of G associated with \mathcal{T} .*

$$\text{bw}(G) = \text{bw}(H_{\mathcal{T}}).$$

Proof. By induction on the number p of edges in $H_{\mathcal{T}} \setminus G$. If G and $H_{\mathcal{T}}$ are equal, the result is obvious.

Otherwise, let (x, y) be an edge of $H_{\mathcal{T}} \setminus G$ and G' be the graph $G \cup \{(x, y)\}$. Since (x, y) is an edge of $H_{\mathcal{T}}$, T_x and T_y have a non empty intersection and since (x, y) does not belong to G , this intersection contains an edge e of \mathcal{T} . Add to T a new vertex in the middle of e and a new leaf u attached to that vertex and map the edge (x, y) to u . The branch-decomposition \mathcal{T}' obtained is a branch-decomposition of G' such that $\text{bw}(\mathcal{T}) = \text{bw}(\mathcal{T}')$. Since \mathcal{T} is optimal for G , we have $\text{bw}(G) \geq \text{bw}(G')$ and since G is a sub-graph of G' , $\text{bw}(G) = \text{bw}(G')$. By construction, the triangulation $H_{\mathcal{T}'}$ of G' associated to \mathcal{T}' is equal to $H_{\mathcal{T}}$. By induction, $\text{bw}(H_{\mathcal{T}'}) = \text{bw}(G')$ which finishes the proof. \square

3 Tight branch-decompositions

Proposition 1 implies that the branch-width of G is the minimal branch-width of a triangulation of G . This latter result is also true for tree-width. It is also known that while dealing with tree-width, we can only consider minimal triangulations but the same restriction is not possible for branch-width for there are examples of chordal graphs H such that H is

never the triangulation associated to an optimal branch-decomposition of H . However we show that we can restrict ourselves to efficient triangulations. To prove this, we order the branch-decompositions of a given graph and then show that the triangulations arising from the minimal branch-decompositions are efficient.

Definition 2. A branch-decomposition \mathcal{T} of a graph G is tighter than \mathcal{T}' if $\text{bw}(\mathcal{T})$ is at most $\text{bw}(\mathcal{T}')$ and $H_{\mathcal{T}}$ is a subgraph of $H_{\mathcal{T}'}$.

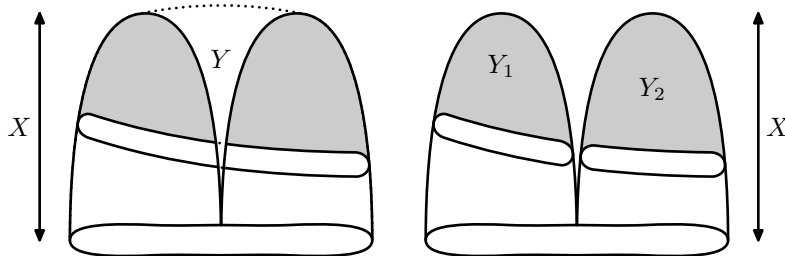
The following theorem defines a “cleaning” process that does not increase the tightness and that we can use to perform local optimisations to a branch-decomposition. Figure 1 gives an idea of how the proof works.

Theorem 2. Let \mathcal{T} be a branch-decomposition of a graph G and T^* a branch of \mathcal{T} .

There exists a branch-decomposition \mathcal{T}' obtained from \mathcal{T} by replacing T^* with another branch such that \mathcal{T}' is tighter than \mathcal{T} and such that every pack of T'^* is the ground of a sub-branch of T'^* .

Moreover, if a border S of T^* is neither a subset of $\delta(T^*)$, nor a subset of $V(X)$ with X a pack of T^* , then \mathcal{T}' is strictly tighter than \mathcal{T} .

Proof. For every pack X_i of T^* , consider the rooted sub-tree of T^* covering the leaves containing an edge of X_i . By contracting the edges, we can “remove” the nodes of degree 2 and thus obtain a branch $T_i'^*$ of ground X_i . By linking these branches by their root, we can obtain a branch T'^* of ground X .



On the left, we consider a branch T_X^* of a decomposition \mathcal{T} with a ground X that has two packs and a sub-branch T_Y^* of ground Y that “crosses” the two packs.

If we prune T_X^* to each pack and rearrange the new branches, the sub-branch T_Y^* and $\delta(Y)$ will be split in two.

The new decomposition is strictly tighter than \mathcal{T} .

Fig. 1. A trimmed branch.

We claim that the branch-decomposition \mathcal{T}' obtained by replacing T^* by T'^* in \mathcal{T} is tighter than \mathcal{T} . Indeed, by construction, every sub-branch B'_i of T'^* is a pruned sub-branch B of T^* . Since X_i is a pack of T^* , $\delta(B'_i)$ is

a subset of $\delta(B)$. Finally, since the border of all the edges added to link the branches $T_i'^*$ is a subset of $\delta(T^*)$, every border of T' is a subset of a border of T . This proves that T' is tighter than T .

Now suppose that S is a border of T^* which is not a subset of $\delta(T^*)$ and not a subset of any $V(X_i)$. Let T_S^* be a sub-branch of T^* of border S . Let u be a vertex of $S \setminus \delta(T^*)$ (such a vertex exists by hypothesis) and X_u be the pack of T^* such that $V(X_u)$ contains u . Since S is not a subset of $V(X_u)$, there exists a vertex v of S in $S \setminus V(X_u)$. By construction, T_u' and T_v' do not meet and T' is strictly tighter than T . \square

The branch-decomposition built in Th. 2 is *trimmed* along T^* . We can now easily prove Th. 3.

Theorem 3. *The triangulation associated to a tightest branch-decomposition of a graph G is an efficient triangulation of G .*

Proof. Let \mathcal{T} be a tightest branch-decomposition of G and $H_{\mathcal{T}}$ the triangulation associated to \mathcal{T} . Let S be a minimal separator of $H_{\mathcal{T}}$ and Ω_1 and Ω_2 be two maximal cliques of $H_{\mathcal{T}}$ containing S .

The triangulation $H_{\mathcal{T}}$ is the intersection graph of the sub-trees T_x , thus Ω_1 and Ω_2 correspond to vertices v_{Ω_1} and v_{Ω_2} of T and there is an edge e_S on the path from v_{Ω_1} to v_{Ω_2} that corresponds to S . Let T_1^* and T_2^* be the two e_S -branches with v_{Ω_1} in T_1^* and v_{Ω_2} in T_2^* .

Suppose now that S is not a minimal separator of G , it has at most one full connected component. We can thus suppose that S is the border of no pack of T_1^* . By Th. 2, we can trim \mathcal{T} along T_1^* and suppose that all the packs of S in G are the grounds of some sub-branches of the two e -branches. In the resulting decomposition \mathcal{T}' , no border of a sub-branch of $T_1'^*$ contains S which proves that Ω_1 is not a maximal clique of $H_{\mathcal{T}'}$. Thus \mathcal{T}' is strictly tighter than \mathcal{T} which is absurd. The minimal separator S of $H_{\mathcal{T}}$ is also a minimal separator of G .

Using the same techniques, we can also deduce the fact that the connected components of $G \setminus S$ and $H_{\mathcal{T}} \setminus S$ are the same. \square

4 Block branch-width

Suppose that $\text{bw}(G) = \text{bw}(H_{\mathcal{T}})$. We can see the maximal cliques of $H_{\mathcal{T}}$ as pieces of a puzzle that match along minimal separators. Since we can suppose that $H_{\mathcal{T}}$ is efficient, we can characterize its maximal cliques as *blocks*.

Definition 3 (Block). *A set of vertices B of G is called a block if, for each connected component C_i of $G \setminus B$,*

- *its neighbourhood $S_i = N(C_i)$ is a minimal separator;*
- *$B \setminus S_i$ is non empty and contained in a connected component of $G \setminus S_i$.*

We say that the minimal separators S_i border the block B and we denote by $s(B)$ the number of these separators.

Let Ω be a maximal clique of $H_{\mathcal{T}}$. The branch-decomposition \mathcal{T} induces a branch-decomposition \mathcal{T}_{Ω} of the complete graph $K(\Omega)$ on Ω . This branch-decomposition *respects* $K(\Omega)$ in that \mathcal{T}_{Ω} is compatible with the minimal separators of $H_{\mathcal{T}}$ included in Ω . The branch-width of $H_{\mathcal{T}}$ is the maximum width of the branch-decompositions \mathcal{T}_{Ω} .

Definition 4 (Block branch-width). *The block branch-width of a block Ω ($\text{bbw}(\Omega)$) is the minimal width of a branch-decomposition of $K(\Omega)$ respecting Ω .*

Conversely, if we have optimal respectful branch-decompositions of the maximal cliques of $H_{\mathcal{T}}$, we can construct an optimal branch-decomposition of $H_{\mathcal{T}}$ which leads to the following theorem.

Theorem 4 ([7]).

$$\text{bw}(G) = \min_{H \text{ efficient triangulation of } G} \max\{\text{bbw}(\Omega) \mid \Omega \text{ maximal clique of } H\}.$$

If we have “enough” blocks of a graph G and if we know their block branch-width, computing the branch-width of G is indeed a large puzzle in which we try to match blocks of low block branch-width along minimal separators bordering them to construct a chordal graph. This puzzle can be solved in linear time in the number of blocks.

Theorem 5 ([7]). *Given a graph G and a complete list \mathcal{B}_G of blocks together with their block branch-widths, the branch-width of G can be computed in $\mathcal{O}(nm|\mathcal{B}_G|)$ time.*

The last tool we need to be able to compute the branch-width of a graph is to be able to compute the block branch-width of a block. Unfortunately, deciding the block branch-width of a block is strictly equivalent to deciding the branch-width of a split-graph which is NP-complete [5] as already stated. Fortunately, if a block Ω is bordered by “few” minimal separators, we can still compute $\text{bbw}(\Omega)$.

Theorem 6 ([7]). *The block branch-width of any block B can be computed in $\mathcal{O}(3^{s(B)})$ time.*

We can sketch the proof as follows. In a decomposition \mathcal{T} of Ω , there is a node v_{Ω} corresponding to Ω and three branches T_i^* attached to v_{Ω} . If \mathcal{T} respects Ω , then the minimal separators bordering Ω are partitioned in three according to the branch in which they appear as a border. If we choose a 3-partition of the minimal separators, we can compute the optimal width of a decomposition leading to this 3-partition in constant time. To compute the block branch-width, we only have to try all the possible 3-partitions.

5 Circular-arc graphs

In this section, we apply Th. 5 to circular-arc graphs to prove that their branch-width can be computed in polynomial time.

A *circular-arc* graph is the intersection graph of the arcs of a circle. The tree-width of circular-arc graphs can be computed in polynomial time as it is shown in [11]. To prove this, the authors use a circular interpretation of the graph (which can be obtained in linear time [12]) and give a geometrical interpretation of maximal potential cliques which allows them to prove that a tree-decomposition corresponds to a planar triangulation of some polygon. We will follow exactly the same path to prove that the branch-width of circular-arc graphs can be computed in polynomial time.

From now on, G is a circular-arc graph. By shifting them a little, we can suppose that ends of two distinct arcs of an intersection model \mathcal{I} of G are also distinct. We will only consider such representations. Between two such ends, we put a *scan-point*. A *scan-line* is a chord of the disk Σ between two distinct scan-points; these chords are different from the chords of a chordal graph. It is easy to see that there are $2n$ scan-points and $n(2n - 1)$ scan-lines. The arcs inside of which lie the ends of a scan-line are *cut* by the scan-line. A scan-line λ or a family of scan-lines A *realises* the set $V(\lambda)$ or $V(A)$ of vertices whose arcs they cut.

Let S be a minimal separator of G and C a full component of S . The union of the arcs of the connected component C is also an arc μ_C which is bordered by two scan-points. The scan-line defined by these scan-points is *close to* C . It is easy to see that the scan-line close to C realises S . Since blocks of a graph are characterised by the minimal separators that border them, we can realise blocks with scan-lines. More precisely, a block Ω is characterised by the connected components of $G \setminus \Omega$. The scan-lines close to these connected components define a *block-realiser* of Ω :

Definition 5 (Block-realiser). *A realiser of a block Ω is a family of scan-lines A such that:*

1. *A realises the minimal separators bordering Ω ;*
2. *no two scan-lines of A cross;*
3. *there is a connected component $\Sigma \setminus A$ which is incident to all the scan-lines of A (the domain of A);*
4. *every minimal separator bordering Ω is realised by at least one scan-line of A .*

Figure 2 shows two realisers. The block-realiser we have just described is the *loose* realiser. In this realiser, distinct minimal separators correspond to distinct scan-lines. It may be possible to use less scan-lines. For example, if S is a subset of S' , we only need to realise S' . By grouping some minimal separators under a common scan-line, we can hope to bound the number of scan-lines of a realiser. This would prove that there is a polynomial number of blocks.

More precisely, let \mathcal{T} be a branch-decomposition of G and Ω a block of G associated with \mathcal{T} . Let v_Ω be a vertex of T corresponding to Ω and T_1^* , T_2^* and T_3^* be the three connected components of $T \setminus \{v_\Omega\}$.

Definition 6 (Respectful realiser). A scan-line λ respects a branch T^* if $V(\lambda)$ is a subset of $V(T^*)$.

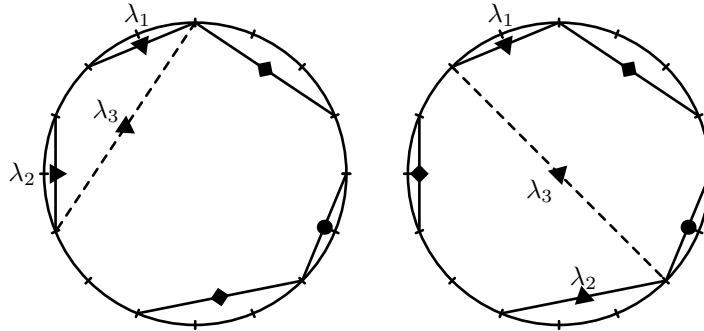
A realiser Λ of Ω respects the three branches T_i^* if it is a realiser and if all its scan-lines respect one of the branches.

By construction, the loose realiser is respectful.

Proposition 2. Let Ω be a block of a tightest branch-decomposition \mathcal{T} of G and v_Ω and T_i^* be defined as above.

A realiser Λ of Ω using as few scan-points as possible respecting T_i^* has at most three scan-lines.

Proof. Suppose for a contradiction that Λ has at least four scan-lines. At least two scan-lines λ_1 and λ_2 of Λ respect the same branch $T_{i_0}^*$. The ends of λ_1 and λ_2 define a chord λ_3 of the domain of Λ which respects $T_{i_0}^*$ (see Fig. 2). This chord partitions Λ in Λ_1 and Λ_2 . If $V(\Lambda_1)$ is a subset of $V(\lambda_3)$, then $\Lambda_2 \cup \{\lambda_3\}$ is a realiser of Ω respecting T_i^* which uses strictly less scan-points than Λ which is absurd. For the same reason, $V(\Lambda_2)$ is not a subset of $V(\lambda_3)$.



Two chords of a same branch have the same symbol on them.

In the first case, we can either reduce the size of the realiser or produce a strictly tighter decomposition than \mathcal{T} .

In the second case, we produce a strictly tighter decomposition than \mathcal{T} .

Fig. 2. Two realisers of a block and two scan-lines λ_3 .

Let \mathcal{T}' be the decomposition \mathcal{T} trimmed along T_1^* , T_2^* and T_3^* .

By rearranging the sub-branches of T_i^* corresponding the packs of T_i^* , we can build two branches $T_{i_1}^*$ and $T_{i_2}^*$ such that $V(T_{i_1}^*)$ contains $V(\Lambda_1)$ and $V(T_{i_2}^*)$ contains $V(\Lambda_2)$. By rearranging the three branches $T_{i_1}^*$ and the three branches $T_{i_2}^*$ in two branches T_1^* and T_2^* that we link, we can define a new branch-decomposition \mathcal{T}'' which is tighter than \mathcal{T} . Moreover, by construction $V(\lambda_1)$ and $V(\Lambda_2)$ are cliques of $H_{\mathcal{T}''}$ but Ω is not one because λ_3 separates two vertices of Ω . This implies that \mathcal{T}'' is strictly tighter than \mathcal{T} which is absurd and finishes the proof. \square

Since there are $O(n^2)$ scan-lines, Prop. 2 implies that there are at most $\mathcal{O}(n^6)$ blocks that can appear in a tightest branch-decomposition of G . Moreover, since the realiser of Ω gives the “good” three-partition of the minimal separators bordering Ω , we can compute $\text{bbw}(\Omega)$ in constant time using Th. 6. These last two results show that we can use Th. 5 to prove:

Theorem 7. *There is a polynomial time algorithm to compute the branch-width of a circular-arc graph.*

6 Conclusion and open problems

Theorem 5 can be used for any class of graphs. If we can bound the number of “interesting” blocks in a class of graphs \mathcal{C} and if we can compute the block branch-width of these blocks, it shows that we can compute the branch-width of the graphs in \mathcal{C} efficiently. This can easily be done with graphs of bounded asteroidal number with a polynomial number of minimal separators for which we can also compute the tree-width in polynomial time. The specific ideas used for the circular-arc graph rely on the existence of scan-lines that can realise minimal separators. A scan-line λ_3 using the ends points of two other scan-lines λ_1 and λ_2 must realise a subset of $V(\lambda_1) \cup V(\lambda_2)$. Such a notion exists for circular permutation graphs and more generally for d -trapezoid circular graphs.

The work we have conducted seems to show that the branch-width problem is more difficult than the tree-width problem. The only class we know for which this might not be the case is the class of planar graphs. Otherwise, if we can compute the branch-width of a class of graphs, then we can compute the tree-width for this same class and with a more efficient algorithm. We feel that this is because tree-decompositions cannot decompose cliques whereas branch-decompositions can. Indeed, in our Th. 5, we not only need to be able to compute the blocks but we need to compute their block branch-width. This second point has no equivalent in the tree-width version of the algorithm. We feel that there could be some theorem stating that if we only use minimal separators, triangulations and blocks it is more difficult to compute branch-width than tree-width.

References

1. Robertson, N., Seymour, P.: Graphs minors. III. Planar tree-width. *Journal of Combinatorial Theory Series B* **36** (1984) 49–64
2. Bodlaender, H.: A partial k -arboretum of graphs with bounded treewidth. *Theoretical computer science* **209** (1998) 1–45
3. Robertson, N., Seymour, P.: Graphs minors. X. Obstruction to tree-decomposition. *Journal of Combinatorial Theory Series B* **52** (1991) 153–190
4. Seymour, P., Thomas, R.: Call routing and the ratcatcher. *Combinatorica* **14**(2) (1994) 217–241

5. Kloks, T., Kratochvíl, J., Müller, H.: New branchwidth territories. In: Proceedings 16th Annual Symposium on Theoretical Aspects of Computer Science (STACS'99). Volume 1563 of Lecture Notes in Computer Science., Springer-Verlag (1999) 173–183
6. Mazoit, F.: Décomposition algorithmique des graphes. PhD thesis, École Normale Supérieure de Lyon (2004)
7. Fomin, F., Mazoit, F., Todinca, I.: Computing branchwidth via efficient triangulations and blocks. In: Proceedings 31th International Workshop on Graphs, WG 2005. Volume 3787 of Lecture Notes in Computer Science., Springer-Verlag (2005) 374 – 384
8. Fomin, F., Kratsch, D., Todinca, I.: Exact (exponential) algorithms for treewidth and minimum fill-in. In: Proceedings 31th International Colloquium on Automata, Languages, and Programming (ICALP'04). Volume 3142 of Lecture Notes in Computer Science., Springer-Verlag (2004) 568–580
9. Gavril, F.: The intersection graphs of a path in a tree are exactly the chordal graphs. *Journal of Combinatorial Theory* **16** (1974) 47–56
10. Parra, A., Scheffler, P.: Characterizations and algorithmic applications of chordal graph embeddings. *Discrete Applied Mathematics* **79**(1-3) (1997) 171–188
11. Sundaram, R., Sher Singh, K., Pandu Rangan, C.: Treewidth of circular-arc graphs. *SIAM Journal Discrete Mathematics* **7** (1994) 647–655
12. McConnell, R.M.: Linear-time recognition of circular-arc graphs. *Algorithmica* **37** (2003) 93–147