Discrete Mathematics and Theoretical Computer Science DMTCS vol. **13:1**, 2011, 13–20

Tree-width and large grid minors in planar graphs

Alexander Grigoriev[†]

Department of Quantitative Economics, Maastricht University SBE, Maastricht, The Netherlands

received 22nd September 2009, accepted 27th January 2011.

We show that for a planar graph with no g-grid minor there exists a tree-decomposition of width at most $5g - 6$. The proof is constructive and simple. The underlying algorithm for the tree-decomposition runs in $O(n^2 \log n)$ time.

Keywords: Planar graph, tree decomposition, tree width, minor, grid graph

1 Introduction

We call H a *minor* of a graph G if H is obtainable from a subgraph of G by edge contractions. If $g \geq 2$, the g-*grid* is the simple graph with vertices v_{ij} $(1 \leq i, j \leq g)$ where v_{ij} and $v_{i'j'}$ are adjacent if $|i - i'| + |j - j'| = 1$; see Robertson and Seymour (1991). A *tree-decomposition* of a graph G , is a pair (T, S), where T is a tree and $S = \{S_t : t \in V(T)\}$ is a family of subsets of $V(G)$, called *bags*, such that

1. $\bigcup_{t \in V(T)} S_t = V(G)$;
2. for every edge $e \in E(G)$ there exists $t \in V(T)$ such that S_t contains both ends of e;
3. if $t, t', t'' \in V(T)$ and t' lies on the path of T between t and t'' then $S_t \cap S_{t''} \subseteq S_{t'}$.

The *width* of a tree-decomposition is the cardinality of the maximum size bag minus 1 and the *tree-width* of a graph G is the minimum width of a tree-decomposition of G; see Robertson, Seymour and Thomas (1994).

The close relationship between the tree-width and the size of the largest grid minor not only in planar but also in general graphs is known already for more than two decades. The problem of finding a tight upper bound for the tree-width in terms of the size of the largest grid minor was introduced and studied first by Robertson and Seymour (1984). In that paper Robertson and Seymour show that every planar graph with no g-grid minor has tree-width strictly less than $\frac{3}{2}\, g^2 + 3g - 2$. Ten years later, Robertson, Seymour and Thomas (1994) improved the bound to be at most $6g - 5$. The proof of this bound follows straightforwardly from two lemmas. First, Robertson, Seymour and Thomas prove that every planar graph

[†]Email: `a.grigoriev@ke.unimaas.nl`

with a *tangle* of order $4g - 3$ has a g-grid minor. Second, Robertson and Seymour (1991) prove that if G has no tangle of order $\geq \theta$, then its tree-width is at most $3\theta/2$. To find tangles in planar graphs the *ratcatcher* algorithm and its modifications can be used; see Gu and Tamaki (2008); Hicks (2005a,b); Seymour and Thomas (1994). Recently, Thomas (2007b) announced that the bound can be further improved to $5g - 1$ simply using the techniques by Alon, Seymour and Thomas (1994) for finding cut-sets in planar graphs, called also *planar separators*. Unfortunately, this result has not been published Thomas (2007a).

The main result of this paper is a short, simple and constructive proof of the following theorem.

Theorem 1 *Every simple planar graph with no minor isomorphic to a g-grid has tree-width at most* $5g - 6$.

To prove the result, we describe an algorithm which, for a given planar graph G with no minor isomorphic to a g-grid, returns a tree-decomposition of G of width at most $5g - 6$. Let g be the minimum integer such that G does not contain g-grid minor. Then, by definition of g, G contains $(g-1)$-grid minor, and therefore $g - 1$ is a lower bound to the tree-width of G. Thus, we straightforwardly claim that our algorithm is a 5-approximation algorithm to the problem of finding a tree-decomposition of minimum width. The best known algorithm to the planar tree-width is a 1.5-approximation algorithm running in time $O(n^4)$ with $n = V(G)$ based on the ratcatcher algorithm of Seymour and Thomas (1994). Recently Gu and Tamaki (2008) improved the theoretic running time of that algorithm to $O(n^3)$ time. Hicks (2005a,b) provided an extensive computational study showing that the ratcatcher algorithm is quite slow in practice even for some medium size planar graphs. Compared to the ratcatcher algorithm, the algorithm presented in this paper has worse performance guarantee but better running time as it runs in $O(n^2 \log n)$ time.

The main difference between our algorithm and the ratcatcher algorithms is that we do not compute tangles but directly construct bags of a tree-decomposition using specific cut-sets of the graph. Recent studies by Gu and Tamaki (2009, 2010) show that using branch-decomposition approach the results of this paper can be further improved in both, run time and performance guarantee of the algorithms. It is noticeable, however, that compared to the ratcatcher algorithms and the algorithms of Gu and Tamaki (2009, 2010), our algorithm is extremely simple. To understand the algorithm no deep knowledge of graph minor theory is required. In fact, we use only text-book maximum flow techniques to construct desired cut-sets. On our opinion, this provides good insight to the planar tree-decomposition problem. Therefore, we see also an added value of this paper in methodology and simplicity of the findings and in popularization of graph minor theory.

2 An algorithm to compute a planar tree-decomposition

In an earlier paper, Bodlaender, Grigoriev and Koster (2008) presented an algorithm computing a lower bound to the side size of the largest grid minor in a planar graph. More explicitly, given a planar graph with $(g - 1)$-grid minor and with no minor isomorphic to g-grid, their algorithm is capable to recognize existence of a $\frac{g}{4}$-grid minor in the graph. In this way the authors in Bodlaender, Grigoriev and Koster (2008) derive a lower bound to the planar tree-width. In the present paper we extend this algorithm to find actually a tree-decomposition of width at most $5g - 6$.

To describe the algorithm, we need several standard graph theoretic notions. Let $G[X]$ denotes the subgraph of G induced by $X \subseteq V(G)$. We write $\overline{X}$ for $V(G) \setminus X$ and $G \setminus X$ for $G[\overline{X}]$. If H is a connected subgraph of G, we define G/H to be a minor of G obtained by contraction of H into a single vertex v_H. We treat v_H as a new vertex and we call this vertex a *vertex for* H in G/H. Let the *vertex connectivity* of non-adjacent vertices v and w in $V(G)$ be defined as the minimum size of a vertex set $C \subset V(G)$, called vw *cut-set*, such that v and w are in different components of $G \setminus C$. If a cut-set consists of a single vertex, we call such a cut-set *atomic*. Let a *walk* in a graph G be a sequence $W := v_0 e_1 v_1 \ldots, v_{\ell-1} e_\ell v_\ell$, whose terms are alternately vertices and edges of G such that v_{i-1} and v_i are the ends of e_i $(1 \leq i \leq \ell)$. If u and v are two vertices of a walk W, where u precedes v on W, the subsequence of W starting with u and ending with v is denoted by uWv and called the *segment* of W from u to v. A walk $W := v_0 e_1 v_1 \ldots, v_{\ell-1} e_\ell v_\ell$ is *closed* if $v_0 = v_\ell$; see, e.g., Bondy and Murty (2008). Finally, for a set of vertices $X \subseteq V(G)$ and for a connected subgraph H in G we define

$$N(X, H) = \{v \in X : \exists u \in V(H) \text{ such that } (v, u) \in E(G)\}, \tag{1}$$

which is simply a subset of vertices in X adjacent to vertices in H.

Algorithm $\mathcal{A}$

Input. A planar embedding of a simple planar graph G with no edge crossings. Such an embedding can be constructed in $O(n)$ time; see Hopcroft and Tarjan (1974).

Preprocessing and postprocessing. We assume that G is 2-connected. Notice that this assumption can be made without loss of generality. Indeed, if G is 1-connected, we decompose the graph into a collection of subgraphs where each subgraph is either a simple edge or it is 2-connected. Moreover, for any two subgraphs H and H' from this collection, $V(H) \cap V(H')$ is either an atomic cut-set or empty. Given a decomposition of G into subgraphs, we run the algorithm for each subgraph separately. Obtained subgraph tree-decompositions we straightforwardly amalgamate: consider two subgraphs H and H' sharing a single vertex $v \in V(G)$ and let the corresponding tree-decompositions be (T, S) and (T', S); find two nodes, $t \in V(T)$ and $t' \in V(T')$, such that $v \in S_t$ and $v \in S_{t'}$; connect T and T' introducing an edge $e = (t, t')$; redefine $H := (V(H) \cup V(H'), E(H) \cup E(H') \cup \{e\})$ and recurse on the remaining subgraphs disregarding H'. Finding collection of subgraphs and amalgamation of the tree-decompositions can be done in preprocessing and postprocessing, respectively, both in linear time.

Initialization. In the initialization phase of the algorithm we create a root node 0 of T with bag S_0. Further, we associate each node $t \in V(T)$ with a connected planar graph G_t constructed in a certain way.

Let F_0 be the outer face of the planar embedding of G. Let W_0 be a closed walk along F_0. Since G is 2-connected, W_0 forms a simple cycle. Assume W_0 contains at least four vertices. Let v_1, v_2, v_3, v_4 be any four distinct vertices appearing in W_0 in this particular order, i.e., $v_2 \in v_1 W_0 v_3$ and $v_3 \in v_2 W_0 v_4$. Now, define $S_0 = \{v_1, v_2, v_3, v_4\}$. If W_0 contains only three vertices, say v_1, v_2, v_3, define $S_0 = \{v_1, v_2, v_3\}$. Notice, that W_0 contains at least three vertices as G is simple. Let G_0 be the graph obtained from G by renaming vertices v_1, v_2, v_3, v_4 into $v_\mathcal{N}, v_\mathcal{E}, v_\mathcal{S}, v_\mathcal{W}$, respectively. In particular, this means $V(G_0) \cap S_0 = \emptyset$.

Basic step. We view the algorithm as construction of a rooted tree T with root 0. At each basic step of the algorithm, given a node $t \in V(T)$, bag $S_t \subseteq V(G)$, and a planar embedding of G_t, we create a number of child nodes of t and we recurse on the child nodes unless $|V(G_t)| \leq 5$. We assume that in the outer face F_t of the planar embedding of G_t there is at least one of the *special* vertices $v_\mathcal{N}, v_\mathcal{E}, v_\mathcal{S}$ or $v_\mathcal{W}$. We also assume that none of the special vertices belongs to the interior of the embedding. Moreover, if G_t

is 2-connected and all four special vertices are present, there exists a closed walk W_t along F_t such that $v_{\mathcal{E}} \in v_{\mathcal{N}} W_t v_{\mathcal{S}}$ and $v_{\mathcal{S}} \in v_{\mathcal{E}} W_t v_{\mathcal{W}}$. The graph associated with a child node of t is a minor of G_t. This minor is either a component of G_t, or it is obtained by deletion of an edge between two special vertices, or it is obtained by contraction of all edges in some connected subgraph of G_t into a special vertex. We consider the following six mutually exclusive cases. Here, case 1 is most general and crucial, while cases 2-6 are rather special.

Case 1. Suppose G_t is 2-connected, all four special vertices are present and not adjacent with each other, both graphs, $G_t \setminus \{v_{\mathcal{N}}, v_{\mathcal{S}}\}$ and $G_t \setminus \{v_{\mathcal{E}}, v_{\mathcal{W}}\}$, are connected. In this case the child nodes of t are constructed as follows.

Let c_t be the vertex connectivity of $v_{\mathcal{N}}$ and $v_{\mathcal{S}}$ in $G_t \setminus \{v_{\mathcal{E}}, v_{\mathcal{W}}\}$ and let d_t be the vertex connectivity of $v_{\mathcal{W}}$ and $v_{\mathcal{E}}$ in $G_t \setminus \{v_{\mathcal{N}}, v_{\mathcal{S}}\}$. Notice, by Menger (1927) the vertex connectivity of any two vertices v and u in a graph equals the number of vertex disjoint paths between v and u. Using the maximum flow technique, e.g., from Section 8.4 of Ahuja, Magnanti and Orlin (1993), we find c_t vertex disjoint $v_{\mathcal{N}} - v_{\mathcal{S}}$ paths in $G_t \setminus \{v_{\mathcal{E}}, v_{\mathcal{W}}\}$ together with the corresponding $v_{\mathcal{N}} v_{\mathcal{S}}$ cut-set C_t. Similarly, in $G_t \setminus \{v_{\mathcal{N}}, v_{\mathcal{S}}\}$ we find d_t vertex disjoint $v_{\mathcal{E}} - v_{\mathcal{W}}$ paths together with $v_{\mathcal{E}} v_{\mathcal{W}}$ cut-set D_t. If $c_t \leq d_t$, we create a single child node t^0 of t with bag $S_{t^0} = S_t \cup C_t$. We do not recurse on this node but we do recurse on its children that we construct next.

As C_t is a $v_{\mathcal{N}} v_{\mathcal{S}}$ cut-set in $G_t \setminus \{v_{\mathcal{E}}, v_{\mathcal{W}}\}$, graph $G_t \setminus (C_t \cup \{v_{\mathcal{E}}, v_{\mathcal{W}}\})$ has at least two components: one containing $v_{\mathcal{N}}$ and one containing $v_{\mathcal{S}}$. Let H be a component in $G_t \setminus (C_t \cup \{v_{\mathcal{E}}, v_{\mathcal{W}}\})$ containing $v_{\mathcal{N}}$. Consistently, we create two child nodes, t^1 and t^2, of t^0 with bags $S_{t^1} = N(S_{t^0}, H)$ and $S_{t^2} = N(S_{t^0}, \overline{H})$. For associated graphs, let $G_{t^1} = G_t / \overline{H}$ where $v_{\mathcal{S}} := v_{\overline{H}}$. Respectively, let $G_{t^2} = G_t / H'$ where $H' = G_t[V(H) \cup C_t]$ and $v_{\mathcal{N}} := v_{H'}$.

If $c_t > d_t$, we create child nodes corresponding to $v_{\mathcal{E}} v_{\mathcal{W}}$ cut-set D_t.

This completes the construction in case 1. Now, we consider the cases when the assumptions of case 1 are not valid, i.e., either some special vertices are adjacent in G_t (case 2), or G_t has low connectivity (cases 3 and 4), or some special vertices are missing (case 5), or $G_t \setminus \{v_{\mathcal{N}}, v_{\mathcal{S}}\}$ or $G_t \setminus \{v_{\mathcal{E}}, v_{\mathcal{W}}\}$ are not connected (case 6).

Case 2. If there exists an edge e between any two special vertices in G_t, we create a child node t' of t with $S_{t'} = S_t$. Let $G_{t'}$ be a minor of G_t obtained by deletion of edge e.

Case 3. Suppose there are no edges between special vertices in G_t and graph G_t is not connected. Since, when taking minors, we delete edges only between special vertices, each component in G_t must contain at least one special vertex. For each component H in G_t we create a child node t^H of t defining $S_{t^H} = N(S_t, H)$ and $G_{t^H} = H$.

Case 4. Suppose there are no edges between special vertices and G_t is 1-connected. Then, like in the preprocessing, we decompose G_t into a collection of subgraphs $\mathcal{H}$, where each subgraph is either a simple edge or it is 2-connected. Again, any two subgraphs from $\mathcal{H}$ can share at most one vertex, which is an atomic cut-set in G_t.

First, assume that a special vertex v has degree 1, i.e., vertex w adjacent to v forms an atomic cut-set. Then, a single child node t^0 of t is created with $S_{t^0} = S_t \cup \{w\}$. In turn, we create a child node t^1 of t^0 with $S_{t^1} = N(S_{t^0}, V(G_t) \setminus \{v, w\})$ and $G_{t^1} = G_t/(v, w)$. Let the vertex obtained by contracting (v, w) be again special vertex v.

Now, assume that a special vertex v is an atomic cut-set in G_t. For each component H of $G_t \setminus \{v\}$ we create a child node t^H of t defining $S_{t^H} = N(S_t, H)$ and $G_{t^H} = H$. Notice that G_{t^H} is a minor of G_t where all components of $G_t \setminus \{v\}$ but H are contracted into v.

Let all special vertices have degree at least 2 and none of those vertices is an atomic cut-set in G_t. Then, there exists a vertex w which is an atomic cut-set in G_t such that at least two special vertices belong to different components in $G_t \setminus \{w\}$, for otherwise the original graph G is not 2-connected as the algorithm either deletes the edges between the special vertices or contracts connected subgraphs into special vertices. Without loss of generality assume, $v_{\mathcal{N}}$ and $v_{\mathcal{S}}$ are in different components of $G_t \setminus \{w\}$. Denote atomic cut-set $C_t = \{w\}$. A single child node t^0 of t is defined by $S_{t^0} = S_t \cup C_t$. We do not recurse on t^0 but we do recurse on its children. Two child nodes, t^1 and t^2, of t^0 are constructed as follows. Let H be a component of $G_t \setminus \{w\}$ containing $v_{\mathcal{N}}$. Consider a complement of H in G_t defined by $\overline{H} = G_t[\overline{V(H)}]$. Notice that $\overline{H}$ is connected and contains $v_{\mathcal{S}}$. Let $S_{t^1} = N(S_{t^0}, H)$, $G_{t^1} = G_t/\overline{H}$, and let vertex $v_{\overline{H}}$ for $\overline{H}$ in G_{t^1} be $v_{\mathcal{S}} := v_{\overline{H}}$. For the second child node, define $S_{t^2} = N(S_{t^0}, \overline{H})$, $G_{t^2} = G_t/G_t[V(H) \cup C_t]$, and let the vertex for $G_t[V(H) \cup C_t]$ in G_{t^2} be $v_{\mathcal{N}}$.

Case 5. Suppose G_t contains at most three special vertices, these vertices are not adjacent with each other, and G_t is 2-connected. Since G_t is 2-connected and $|V(G_t)| > 5$, any closed walk along the outer face F_t forms a simple cycle of length at least 3.

Assume there are at most two special vertices in G_t. Take any vertex v in F_t which is not special and define $C_t = \{v\}$. Let the child node t' of t be the node with $S_{t'} = S_t \cup C_t$ and let $G_{t'}$ be the graph obtained from G_t by renaming v into a special vertex different from the special vertices in G_t.

Assume there are exactly three special vertices in G_t. Without loss of generality assume that special vertex $v_{\mathcal{W}}$ is missing. Consider a closed walk W_t along F_t starting at $v_{\mathcal{N}}$ and having $v_{\mathcal{E}} \in v_{\mathcal{N}} W_t v_{\mathcal{S}}$. Notice that such a walk always exists. Since $v_{\mathcal{S}}$ and $v_{\mathcal{N}}$ are not adjacent, there is a vertex v in W_t such that $v_{\mathcal{S}} \in v_{\mathcal{E}} W_t v$. Notice, v is not a special vertex. Let $C_t = \{v\}$. For child node t' define $S_{t'} = S_t \cup C_t$, and let $G_{t'}$ be the graph obtained from G_t by renaming v into $v_{\mathcal{W}}$.

Case 6. Suppose in 2-connected graph G_t all four special vertices are present and these vertices are not adjacent with each other. Moreover, suppose either $G_t \setminus \{v_{\mathcal{N}}, v_{\mathcal{S}}\}$ or $G_t \setminus \{v_{\mathcal{E}}, v_{\mathcal{W}}\}$ is not connected. If $G_t \setminus \{v_{\mathcal{N}}, v_{\mathcal{S}}\}$ is not connected, $\{v_{\mathcal{N}}, v_{\mathcal{S}}\}$ is a $v_{\mathcal{E}} v_{\mathcal{W}}$ cut-set in G_t. Let H be a component of $G_t \setminus \{v_{\mathcal{N}}, v_{\mathcal{S}}\}$ containing $v_{\mathcal{E}}$. We create two child nodes of t: t^1 and t^2. Here, $S_{t^1} = N(S_t, H)$, $G_{t^1} = G_t/\overline{H}$ where $v_{\mathcal{W}} := v_{\overline{H}}$. In turn, $S_{t^2} = N(S_t, \overline{H})$, $G_{t^2} = G_t/H$ where $v_{\mathcal{E}} := v_H$. The case when $G_t \setminus \{v_{\mathcal{E}}, v_{\mathcal{W}}\}$ is not connected is treated similarly.

Leaves of the tree-decomposition. Consider a node $t \in V(T)$ such that $|V(G_t)| \leq 5$. Let $C_t = V(G_t) \setminus \{v_{\mathcal{N}}, v_{\mathcal{E}}, v_{\mathcal{S}}, v_{\mathcal{W}}\}$. We create a single child node t' of t with bag $S_{t'} = S_t \cup C_t$ and we stop recursing on it, i.e., t' is a leaf in T.

Cleaning and output. If there is an edge $(t', t) \in E(T)$ such that $S_{t'} \subseteq S_t$ we contract (t', t) identifying the resulting common bag with S_t. Then, we stop and output (T, S), where $S = \{S_t \,:\, t \in V(T)\}$.

3 Analysis of the algorithm

First, we proof correctness of the algorithm.

Lemma 2 *Pair (T, S) returned by algorithm $\mathcal{A}$ is a tree-decomposition of G.*

Proof: By construction, T is a tree; for each node $t \in V(T)$, bag S_t is a subset of $V(G)$; and $\bigcup_{t \in V(T)} S_t = V(G)$. Therefore, it only remains to verify tree-decomposition conditions (2) and (3).

For condition (2), consider an edge $(v, u) \in E(G)$. Let t' be the node in T such that none of the bags in $0 - t'$ path in T contains v or u, and there is a child node t'' of t' with a bag containing v. Without loss of generality we may assume existence of such a node, for otherwise there is a node t' in T such that none of the bags on $0 - t'$ path in T contains v or u, and there is a child node t'' of t' with a bag containing u. Now, if $u \in S_{t''}$, condition (2) is trivially satisfied. Assume $u \notin S_{t''}$. By definition of bags in (T, S) and by Equation 1, for any successive child t of t'', vertex v is an element of S_t unless vertices from $V(G_t)$ are not adjacent to v in G. Thus, there is a node in T, successor of t'', with a bag containing both ends of (v, u).

Finally, we argue that condition (3) is satisfied. Consider tree T before the cleaning phase of the algorithm. Let $t \in V(T)$ and let $t',\ t''$ be two child nodes of t in T. Notice that in cases 2 and 5, node t has a single child node. Therefore, t' and t'' are constructed in one of the cases: 1, 3, 4 or 6. In all these cases, $V(G_{t'}) \cap V(G_{t''}) = \emptyset$ as $G_{t'}$ and $G_{t''}$ are different components in graph $G_t \setminus X$ where $X \subseteq V(G_t)$. Hence, any vertex $v \in V(G)$ appears in exactly one branch of T. Let t be a node in T, closest to the root, such that $v \in S_t$. Again, by definition of bags in (T, S) and by Equation 1, for any successive child t' of t, vertex v is an element of $S_{t'}$ unless vertices from $V(G_t)$ are not adjacent to v in G. Thus, nodes of T with bags containing v form a connected subtree in T. $\square$

Now, we analyze the run time of the algorithm.

Lemma 3 *Algorithm $\mathcal{A}$ terminates in $O(n^2 \log n)$ time.*

Proof: As we mentioned in the algorithm description, initialization, preprocessing and postprocessing require $O(n)$ time. Recognition of the case of a basic step can also be done in linear time. Basic step cases 2-6 require $O(n)$ time. For case 1, cut-sets in planar networks can be computed in $O(n \log n)$ time; see Theorem 8.8 in Ahuja, Magnanti and Orlin (1993), p. 265. Given cut-sets, creating the child nodes takes only linear time. Therefore, one basic step of the algorithm takes at most $O(n \log n)$ time. Notice that processing the leaves of the tree-decomposition and clean-up require only linear time.

Let us argue that the number of basic steps of the algorithm is $O(n)$. First, we notice that any consecutive appearance of cases 2 and 3 can be observed in at most 6 basic steps. This is because we have at most four special vertices in the outer face of any considered graph, yielding at most 5 edges between the special vertices, i.e., 5 basic steps under case 2. Then, we can apply only one iteration with case 3 since in this case each of the components must contain a special vertex. Any other sequence of iterations with cases 2 and 3 is shorter. Now, consider basic step cases 1 and 4-6. In each of these cases, when specifying the bags of the child nodes of t, the algorithm finds a non-empty cut-set C_t in G_t. Notice that $V(G_t) \cap S_t = \emptyset$ as vertices from S_t are either renamed or contracted into the special vertices in G_t. Therefore, $C_t \cap S_t = \emptyset$. By construction, $C_t \subseteq S_{t'}$. Thus, for each child node t' of t, there exists a vertex $v \in C_t$ such that $v \in S_{t'}$ and $v \notin S_t$. Hence, the total number of basic steps with cases 1 and 4-6 is at most n. Since appearance of cases 1 and 4-6 can be interrupted by at most 6 consecutive cases 2-3, we derive that the total number of basic steps in $\mathcal{A}$ is at most $7n$. As each basic step is executed in $O(n \log n)$ time, the total run time of the algorithm is $O(n^2 \log n)$. $\square$

Finally, we analyze performance guarantee of the algorithm. The core of the analysis is the following three lemmas.

Lemma 4 *Given a 2-connected planar graph G together with its planar embedding, let F be the outer face of the embedding, and let $v_{\mathcal{N}}, v_{\mathcal{E}}, v_{\mathcal{S}}, v_{\mathcal{W}}$ be four distinct vertices in $V(G)$ incident to F. Moreover, assume there exists a closed walk W along F such that $v_{\mathcal{E}} \in v_{\mathcal{N}}Wv_{\mathcal{S}}$ and $v_{\mathcal{S}} \in v_{\mathcal{E}}Wv_{\mathcal{W}}$. Finally, assume $G \setminus \{v_{\mathcal{N}}, v_{\mathcal{S}}\}$ and $G \setminus \{v_{\mathcal{E}}, v_{\mathcal{W}}\}$ are connected. Then, G has a $\min\{c, d\}$-grid minor, where c is the vertex connectivity of $v_{\mathcal{N}}$ and $v_{\mathcal{S}}$ in $G \setminus \{v_{\mathcal{E}}, v_{\mathcal{W}}\}$, and d is the vertex connectivity of $v_{\mathcal{E}}$ and $v_{\mathcal{W}}$ in $G \setminus \{v_{\mathcal{N}}, v_{\mathcal{S}}\}$.*

Proof: Under the lemma assumptions, vertex connectivity of both pairs, $(v_{\mathcal{N}}, v_{\mathcal{S}})$ and $(v_{\mathcal{E}}, v_{\mathcal{W}})$, in the corresponding graphs are well defined. Therefore, there are c vertex disjoint $v_{\mathcal{N}} - v_{\mathcal{S}}$ paths in $G \setminus \{v_{\mathcal{E}}, v_{\mathcal{W}}\}$, and there are d vertex disjoint $v_{\mathcal{E}} - v_{\mathcal{W}}$ paths in $G \setminus \{v_{\mathcal{N}}, v_{\mathcal{S}}\}$. Since $v_{\mathcal{N}}, v_{\mathcal{E}}, v_{\mathcal{S}}, v_{\mathcal{W}}$ are incident to the outer face and there exists a closed walk W along F such that $v_{\mathcal{E}} \in v_{\mathcal{N}}Wv_{\mathcal{S}}$ and $v_{\mathcal{S}} \in v_{\mathcal{E}}Wv_{\mathcal{W}}$, each of the c vertex disjoint $v_{\mathcal{N}} - v_{\mathcal{S}}$ paths crosses each of the d vertex disjoint $v_{\mathcal{E}} - v_{\mathcal{W}}$ paths. Thus, G has a $\min\{c, d\}$-grid minor. □

Lemma 5 *Let G be a planar graph that does not have a g-grid minor. At any basic step of algorithm $\mathcal{A}$, given a node $t \in V(T)$ and a graph G_t with k $(1 \le k \le 4)$ special vertices, the size of bag S_t is at most $k(g-1)$.*

Proof: Consider node $t \in V(T)$ and graph G_t with k $(1 \le k \le 4)$ special vertices. By definition, bag S_t is a subset of a union of k cut-sets contracted to the special vertices of G_t. By Lemma 4, the size of a cut-set is at most $g - 1$. Therefore, the size of S_t is at most $k(g-1)$ that proves the lemma. □

Now, we are ready to prove the last lemma, eventually implying Theorem 1.

Lemma 6 *If graph G does not have a g-grid minor then the width of tree-decomposition (T, S) returned by algorithm $\mathcal{A}$ is at most $5g - 6$.*

Proof: By Lemma 5, for any $t \in V(T)$ it holds that $|S_t| \le k(g-1)$, where k $(1 \le k \le 4)$ is the number of special vertices in G_t. Therefore, at any basic step of the algorithm, when dealing with node $t \in V(T)$, bag S_t contains at most $4g - 4$ vertices. In addition to these nodes, in cases 1 and 4, we construct intermediate nodes $t^0 \in V(T)$. These nodes are necessary for (T, S) to satisfy condition (3) of tree-decomposition. By definition, $S_{t^0} = S_t \cup C_t$, where C_t is a cut-set in G_t. In case 4, this cut-set is atomic yielding $|S_{t^0}| \le 4g - 3$. In case 1, by Lemma 4, $|C_t| \le (g-1)$ implying $|S_{t^0}| \le 5g - 5$. For the leaves of T, we have associated graphs containing at most five vertices including the special ones. Applying the same arguments as in Lemma 5, we derive that the size of a leaf bag is at most $4g - 3$. Since $5g - 5$ is an upper bound for the number of vertices in any bag of (T, S), we conclude that the width of (T, S) is at most $5g - 6$. □

4 Conclusions

We present a new, fast and simple, constant approximation algorithm to the problem of finding a tree-decomposition of minimum width in planar graphs. Based on the algorithm, we derive the following structural result: if a planar graph does not contain a g-grid as a minor then its tree-width is at most $5g - 6$.

References

Ahuja, R.K., T.L. Magnanti, J.B. Orlin. 1993. *Network Flows: Theory, Algorithms, and Applications.* Prentice Hall, Inc., Upper Saddle River, New Jersey.

Alon, N., P.D. Seymour, R. Thomas. 1994. Planar Separators. *SIAM J. on Discrete Math.* 7, 184–193.

Bondy, J.A., U.S.R. Murty. 2008. *Graph Theory*. Springer, New York.

Bodlaender, H.L., A. Grigoriev, A.M.C.A. Koster. 2008. Treewidth lower bounds with brambles. *Algorithmica* 51, 81–98.

Gu, Q.P., H. Tamaki. 2009. Constant-factor approximations of branch-decomposition and largest grid minor of planar graphs in $O(n^{1+\varepsilon})$ time. In Y. Dong, D.-Z. Du, O.H. Ibarra (Eds.), *Proceedings of the 20th International Symposium (ISAAC'2009).* Lecture Notes in Computer Science 5878, Springer, 85–96.

Gu, Q.P., H. Tamaki. 2010. Improved bounds on the planar branchwidth with respect to the largest grid minor size. In O. Cheong, K.-Y. Chwa, K. Park (Eds.), *Proceedings of the 21st International Symposium (ISAAC'2010).* Lecture Notes in Computer Science 6507, Springer, 984–993.

Gu, Q.P., H. Tamaki. 2008. Optimal branch decomposition of planar graphs in $O(n^3)$ time. *ACM Trans. Algorithms* 4, 1–13.

Hicks, I.V. 2005a. Planar branch decompositions I: The ratcatcher. *INFORMS J. on Computing* 17, 402–412.

Hicks, I.V. 2005b. Planar branch decompositions II: The cycle method. *INFORMS J. on Computing* 17, 413–421.

Hopcroft, J.E., R.E. Tarjan. 1974. Efficient planarity testing, *J. ACM* 21, 549–568.

Menger, K. 1927. Zur allgemeinen Kurventheorie. *Fund. Math.* 10, 96–115.

Robertson, N., P.D. Seymour. 1984. Graph minors. III. Planar tree-width. *J. Combin. Theory Ser. B* 36, 49–64.

Robertson, N., P.D. Seymour. 1991. Graph minors. X. Obstructions to Tree-Decomposition. *J. Combin. Theory Ser. B* 52, 153–190.

Robertson, N., P.D. Seymour, R. Thomas. 1994. Quickly excluding a planar graph. *J. Combin. Theory Ser. B* 62, 323–348.

Seymour, P.D., R. Thomas. 1994. Call routing and the ratcatcher. *Combinatorica* 14, 217–241.

Thomas, R. 2007a. Personal communications, June-July 2007.

Thomas, R. 2007b. Tree-decompositions of graphs. Retrieved in June-July 2007 from `http://www.math.gatech.edu/ thomas/SLIDE/slide2.ps`, p. 32.