Near-Linear Time Constant-Factor Approximation Algorithm
for Branch-Decomposition of Planar Graphs
Qian-Ping Gu and Gengchun Xu
School of Computing Science, Simon Fraser University
Burnaby BC Canada V5A1S6
qgu@cs.sfu.ca, gxa2@sfu.ca

Abstract: We give constant-factor approximation algorithms for branch-decomposition of planar graphs. Our main result is an algorithm which for an input planar graph $G$ of $n$ vertices and integer $k$, in $O(n \log^4 n)$ time either constructs a branch-decomposition of $G$ with width at most $(2 + \delta)k$, $\delta > 0$ is a constant, or a $(k + 1) \times \lceil \frac{k+1}{2} \rceil$ cylinder minor of $G$ implying $bw(G) > k$; $bw(G)$ is the branchwidth of $G$. This is the first $\tilde{O}(n)$ time constant-factor approximation for branchwidth/treewidth and largest grid/cylinder minors of planar graphs and improves the previous $\min\{O(n^{1+\epsilon}), O(nk^3)\}$ ($\epsilon > 0$ is a constant) time constant-factor approximations. For a planar graph $G$ and $k = bw(G)$, a branch-decomposition of width at most $(2 + \delta)k$ and a $g \times \beta$ cylinder/grid minor with $g = \frac{k}{\beta}$, $\beta > 2$ is constant, can be computed by our algorithm in $O(n \log^4 n \log k)$ time.

Key words: Branch-/tree-decompositions, grid minor, planar graphs, approximation algorithm.

1 Introduction

The notions of branchwidth and branch-decomposition introduced by Robertson and Seymour [30] in relation to the notions of treewidth and tree-decomposition have important algorithmic applications. The branchwidth $bw(G)$ and the treewidth $tw(G)$ of graph $G$ are linearly related: $bw(G) \leq tw(G) + 1 \leq \lfloor \frac{3}{2}bw(G) \rfloor$ for every $G$ with more than one edge, and there are simple translations between branch-decompositions and tree-decompositions that meet the linear relations [30]. A graph $G$ of small branchwidth (treewidth) admits efficient algorithms for many NP-hard problems [2, 6]. These algorithms first compute a branch-/tree-decomposition of $G$ and then apply a dynamic programming algorithm based on the decomposition to solve the problem. The dynamic programming step typically runs in polynomial time in the size of $G$ and exponential time in the width of the branch-/tree-decomposition computed.

Deciding the branchwidth/treewidth and computing a branch-/tree-decomposition of minimum width have been extensively studied. For an arbitrary graph $G$ of $n$ vertices, the following results have been known: Given an integer $k$, it is NP-complete to decide whether $bw(G) \leq k$ [33] (tw(G) \leq k [11]). If $bw(G)$ (tw(G)) is upper-bounded by a constant then both the decision problem and the optimal decomposition problem can be solved in $O(n)$ time [4, 7]. However, the linear time algorithms are mainly of theoretical importance because the constant behind the Big-Oh is huge. The best known polynomial time approximation factor is $O(\sqrt{bw(G)})$ for branchwidth and $O(\sqrt{\log tw(G)})$ for treewidth [15]. The
best known exponential time approximation factors are as follows: an algorithm giving a branch-decomposition of width at most \(3bw(G)\) in \(2^{O(bw(G))}n^2\) time \([31]\); an algorithm giving a tree-decomposition of width at most \(3tw(G)+4\) in \(2^{O(bw(G))}n\log n\) time \([5]\); and an algorithm giving a tree-decomposition of width at most \(5tw(G)+4\) in \(2^{O(tw(G))}n\) time \([5]\). By the linear relation between the branchwidth and treewidth, the algorithms for tree-decompositions are also algorithms of same approximation factors for branch-decompositions, while from a branchwidth approximation \(\alpha\), a treewidth approximation \(1.5\alpha\) can be obtained.

Better results have been known for planar graphs \(G\). Seymour and Thomas show that whether \(bw(G)\) can be decided in \(O(n^2)\) time and an optimal branch-decomposition of \(G\) can be computed in \(O(n^4)\) time \([33]\). Gu and Tamaki improve the \(O(n^4)\) time for optimal branch-decomposition to \(O(n^3)\) \([19]\). By the linear relation between the branchwidth and treewidth, the above results imply polynomial time 1.5-approximation algorithms for the treewidth and optimal tree-decomposition of planar graphs. It is open whether deciding \(tw(G)\) is NP-complete or polynomial time solvable for planar graphs \(G\).

Fast algorithms for computing small width branch-/tree-decompositions of planar graphs have received much attention as well. Tamaki gives an \(O(n)\) time heuristic algorithm for branch-decomposition \([35]\). Gu and Tamaki give an algorithm which for an input planar graph \(G\) of \(n\) vertices and integer \(k\), either constructs a branch-decomposition of \(G\) with width at most \((c + 1 + \delta)k\) or outputs \(bw(G) > k\) in \(O(n^{1+\frac{1}{c}})\) time, where \(c\) is any fixed positive integer and \(\delta > 0\) is a constant \([20]\). By this algorithm and a binary search, a branch-decomposition of width at most \((c + 1 + \delta)k\) can be computed in \(O(n^{1+\frac{1}{c}}\log k)\) time, \(k = bw(G)\). Recently, Kammer and Tholey give an algorithm which for input \(G\) and \(k\), either constructs a tree-decomposition of \(G\) with width at most \((9 + \delta)k\), \(\delta > 0\) is a constant, or outputs \(tw(G) > k\) in \(O(nk^3)\) time \([26, 27]\). This implies that a tree-decomposition of width at most \((9 + \delta)k\) can be computed in \(O(nk^3\log k)\) time, \(k = tw(G)\). Computational studies on branch-decomposition can be found in \([3, 4, 22, 23, 34, 35]\). Fast constant-factor approximation algorithms for branch-/tree-decompositions of planar graphs have important applications such as that in shortest distance oracles in planar graphs \([28]\).

Grid and cylinder minors of graphs are notions closely related to branch-/tree-decompositions \([12, 13, 21, 32]\). A \(k \times h\) cylinder is a Cartesian product of a cycle on \(k\) vertices and a path on \(h\) vertices. For a graph \(G\), let \(cm(G)\) be the largest integer \(k\) such that \(G\) has a \(k \times \lceil \frac{k}{2} \rceil\) cylinder as a minor. It is shown in \([21]\) that \(cm(G) \leq bw(G) \leq 2cm(G)\) for planar graphs. The \(O(n^{1+\frac{1}{c}})\) time algorithm in \([20]\) actually constructs a branch-decomposition of \(G\) with width at most \((c + 1 + \delta)k\) or a \((k + 1) \times \lceil \frac{k+1}{2} \rceil\) cylinder minor. Other work on the lower bound for the branchwidth/treewidth of planar graphs can be found in \([8, 18]\).

We propose an \(\tilde{O}(n)\) time (the \(\tilde{O}\) notation disregards poly-logarithmic terms) constant-factor approximation algorithm for branch-/tree-decompositions of planar graphs. This result is stated as follows.

**Theorem 1.** There is an algorithm which given a planar graph \(G\) of \(n\) vertices and an integer \(k\), in \(O(n\log^4 n)\) time either constructs a branch-decomposition of \(G\) with width at most \((2 + \delta)k\), \(\delta > 0\) is a constant, or a \((k + 1) \times \lceil \frac{k+1}{2} \rceil\) cylinder minor of \(G\).
Since a \((k+1) \times \left[ \frac{k+1}{2} \right]\) cylinder has branchwidth at least \(k+1\) [21], a cylinder minor given in Theorem 1 implies \(\text{bw}(G) > k\).

By the linear relation between the branchwidth and treewidth, Theorem 1 implies an algorithm which for an input planar graph \(G\) and integer \(k\), in \(O(n \log^4 n)\) time constructs a tree-decomposition of \(G\) with width at most \((3 + \delta)k\) or outputs \(\text{tw}(G) > k\). For a planar graph \(G\) and \(k = \text{bw}(G)\), by Theorem 1 and a binary search, a branch-decomposition of width at most \((2 + \delta)k\) can be computed in \(O(n \log^4 n \log k)\) time. This improves the previous result of a branch-decomposition of width at most \((c + 1 + \delta)k\) in \(O(n^{1+\frac{1}{2}} \log k)\) time [20]. Similarly, for a planar graph \(G\) and \(k = \text{tw}(G)\), a tree-decomposition of width at most \((3 + \delta)k\) can be computed in \(O(n \log^4 n \log k)\) time, improving the previous result of a tree-decomposition of width at most \((9 + \delta)k\) in \(O(nk^3 \log k)\) time [26, 27] when \(k > c'(\log n)^\frac{1}{4}\) for some constant \(c' > 0\). Our algorithm can also be used to compute a \(g \times \left[ \frac{g}{2} \right]\) cylinder (grid) minor with \(g = \frac{\text{bw}(G)}{\beta}\), \(\beta > 2\) is a constant, and a \(g \times g\) cylinder (grid) minor with \(g = \frac{\text{bw}(G)}{\beta}\), \(\beta > 3\) is a constant, of \(G\) in \(O(n \log^4 n \log k)\) time. This improves the previous results of \(g \times \left[ \frac{g}{2} \right]\) with \(g \geq \frac{\text{bw}(G)}{\beta}\), \(\beta > (c + 1)\), and \(g \times g\) with \(g \geq \frac{\text{bw}(G)}{\beta}\), \(\beta > (2c + 1)\), in \(O(n^{1+\frac{1}{2}} \log k)\) time. As an application, our algorithm removes a bottleneck in work of [28] for computing a shortest path oracle and reduces its preprocessing time complexity in Theorem 6.1 from \(O(n^{1+\frac{1}{2}} \log k \log n + S \log^2 n)\) to \(O(n \log^5 n \log k + S \log^2 n)\).

Our algorithm for Theorem 1 uses the approach in the previous work of [20] described below. Given a planar graph \(G\) and integer \(k\), let \(Z\) be the set of biconnected components of \(G\) with a normal distance (a definition is given in the next section) \(h = ak\), \(a > 0\) is a constant, from a selected edge \(e_0\) of \(G\). For each \(Z \in Z\), a minimum vertex-cut set \(\partial(A_Z)\) which partitions \(E(G)\) into edge subsets \(A_Z\) and \(\overline{A}_Z = E(G) \setminus A_Z\) is computed such that \(Z \subseteq A_Z\) and \(e_0 \in \overline{A}_Z\) (\(\partial(A_Z)\) separates \(Z\) and \(e_0\)). If \(|\partial(A_Z)| > k\) for some \(Z \in Z\) then \(\text{bw}(G) > k\) is concluded. Otherwise, a branch-decomposition of graph \(H\) obtained from \(G\) by removing all \(A_Z\) is constructed. For each subgraph \(G[A_Z]\) induced by \(A_Z\), a branch-decomposition is constructed or \(\text{bw}(G[A_Z]) > k\) is concluded recursively. Finally, a branch-decomposition of \(G\) with width \(O(k)\) is constructed from the branch-decomposition of \(H\) and those of \(G[A_Z]\) or \(\text{bw}(G) > k\) is concluded.

Our algorithm uses a recent result in computing minimum face separating cycles in planar graphs to find \(\partial(A_Z)\) for every \(Z \in Z\). Borradaile et al. give an algorithm which in \(O(n \log^4 n)\) time computes an oracle for the all pairs minimum face separating cycle problem in a planar graph \(G\) [10, 11]. For any pair of faces \(f\) and \(g\) in \(G\), the oracle in \(O(|C|)\) time returns a minimum \((f, g)\)-separating cycle \(C\) (\(C\) cuts the sphere on which \(G\) is embedded into two regions, one contains \(f\) and the other contains \(g\)). By this result, we show that a minimum vertex-cut set \(\partial(A_Z)\) for every \(Z \in Z\) in all recursive steps can be computed in \(O(n \log^4 n)\) time and get an algorithm for Theorem 1.

When \(|Z|\) is small, we show that \(\partial(A_Z)\) for every \(Z \in Z\) can be computed more efficiently. Let \(n_z\) be the total number of components to be separated in all recursive steps, we get the following results.

**Theorem 2** There is an algorithm which given a planar graph \(G\) of \(n\) vertices and integer...
of leaves of \( G \) and the vertices of \( G \) (separation \( \delta \)). It is convenient to view a vertex-cut set \( v \) as nodes. Consider a link \( v \) of \( G \), with \( \phi \) to be the largest order of the separations induced by links of \( G \) \( \phi \), \( T \) to decompose \( G \) into subgraphs and computes \( \partial(A_Z) \) on the recursive \( r \)-division. An algorithm for Theorem 3 uses the crest separators introduced by Kammer and Tholey \( \{26, 27\} \) to decompose \( G \) into subgraphs and computes \( \partial(A_Z) \) on the subgraphs. For graphs with \( n_z = O(\sqrt{n}) \), we get a \( O(n \log^3 n, O(nk)) \) time \((2 + \delta)\)-approximation algorithm.

The next section gives the preliminaries of the paper. We prove Theorems \( \{11, 12, 13\} \) in Sections 3, 4, and 5, respectively. The final section concludes the paper.

## 2 Preliminaries

It is convenient to view a vertex-cut set \( \partial(A_Z) \) in a graph as an edge in a hypergraph in some cases. A hypergraph \( G \) consists of a set \( V(G) \) of vertices and a set \( E(G) \) of edges, each edge is a subset of \( V(G) \) with at least two elements. A hypergraph \( G \) is a graph if for every \( e \in E(G), e \) has two elements. For a subset \( A \subseteq E(G) \), we denote \( \cup_{e \in A} e \) by \( V(A) \) and denote \( E(G) \setminus A \) by \( \overline{A} \). For \( A \subseteq E(G) \), the pair \((A, \overline{A})\) is a separation of \( G \) and we denote by \( \partial(A) \) the vertex set \( V(A) \cap V(\overline{A}) \). The order of separation \((A, \overline{A})\) is \( |\partial(A)| \). A hypergraph \( H \) is a subgraph of \( G \) if \( V(H) \subseteq V(G) \) and \( E(H) \subseteq E(G) \). For \( A \subseteq E(G) \) and \( W \subseteq V(G) \), we denote by \( G[A] \) and \( G[W] \) the subgraphs of \( G \) induced by \( A \) and \( W \), respectively.

The notions of branchwidth and branch-decomposition are introduced by Robertson and Seymour \( \{30\} \). A branch-decomposition of hypergraph \( G \) is a pair \((\phi, T)\) where \( T \) is a ternary tree and \( \phi \) is a bijection from the set of leaves of \( T \) to \( E(G) \). We refer the edges of \( T \) as links and the vertices of \( T \) as nodes. Consider a link \( e \) of \( T \) and let \( L_1 \) and \( L_2 \) denote the sets of leaves of \( T \) in the two respective subtrees of \( T \) obtained by removing \( e \). We say that the separation \((\phi(L_1), \phi(L_2))\) is induced by this link \( e \) of \( T \). We define the width of the branch-decomposition \((\phi, T)\) to be the largest order of the separations induced by links of \( T \). The branchwidth of \( G \), denoted by \( bw(G) \), is the minimum width of all branch-decompositions of \( G \). In the rest of this paper, we identify a branch-decomposition \((\phi, T)\) with the tree \( T \), leaving the bijection implicit and regarding each leaf of \( T \) as an edge of \( G \).

A walk in graph \( G \) is a sequence of edges \( e_1, e_2, ..., e_k \), where \( e_i = \{v_{i-1}, v_i\} \). We call \( v_0 \) and \( v_k \) the end vertices and other vertices the internal vertices of the walk. A walk is a path if all vertices in the walk are distinct. A walk is a cycle if it has at least three vertices, \( v_0 = v_k \) and \( v_1, ..., v_k \) are distinct.

\( k \), in \( O((n + n_z \sqrt{n}) \log^3 n) \) time either constructs a branch-decomposition of \( G \) with width at most \((2 + \delta)k \) or a \((k + 1) \times \lceil \frac{k + 1}{2} \rceil \) cylinder minor of \( G \), where \( \delta > 0 \) is a constant.

**Theorem 3** There is an algorithm which given a planar graph \( G \) of \( n \) vertices and integer \( k \), in \( O(nk + n_z k^3) \) time, either construct a branch-decomposition of \( G \) with width at most \((2 + \delta)k \) or a \((k + 1) \times \lceil \frac{k + 1}{2} \rceil \) cylinder minor of \( G \), where \( \delta > 0 \) is a constant.
Let $\Sigma$ be a sphere. For an open segment $s$ homeomorphic to $\{x|0<x<1\}$ in $\Sigma$, we denote by $\text{cl}(s)$ the closure of $s$. A planar embedding of a graph $G$ is a mapping $\rho : V(G) \cup E(G) \rightarrow \Sigma \cup 2^\Sigma$ such that

- for $u \in V(G)$, $\rho(u)$ is a point of $\Sigma$, and for distinct $u, v \in V(G)$, $\rho(u) \neq \rho(v)$;
- for each edge $e = \{u, v\} \in E(G)$, $\rho(e)$ is an open segment in $\Sigma$ with $\rho(u)$ and $\rho(v)$ the two end points in $\text{cl}(\rho(e)) \setminus \rho(e)$; and
- for distinct $e_1, e_2 \in E(G)$, $\text{cl}(\rho(e_1)) \cap \text{cl}(\rho(e_2)) = \{\rho(u) | u \in e_1 \cap e_2\}$.

A graph $G$ is planar if it has a planar embedding $\rho$, and $(G, \rho)$ is called a plane graph. We may simply use $G$ to denote the plane graph $(G, \rho)$, leaving the embedding $\rho$ implicit. For a plane graph $G$, each connected component of $\Sigma \setminus (\cup_{e \in E(G)} \text{cl}(\rho(e)))$ is a face of $G$. We denote by $V(f)$ and $E(f)$ the set of vertices and the set of edges incident to face $f$, respectively. We say face $f$ is bounded by the edges of $E(f)$.

A plane graph $G$ is biconnected if for any distinct vertices $u, v, w \in V(G)$, there is a path of $G$ between $u$ and $v$ that does not contain $w$. It suffices to prove Theorems 1, 2 and 3 for a biconnected $G$ because if $G$ is not biconnected, the problems of finding branch-decompositions and cylinder minors of $G$ can be solved individually for each biconnected component.

For a plane graph $G$, a curve $\mu$ on $\Sigma$ is normal if $\mu$ does not intersect any edge of $G$. The length of a normal curve $\mu$ is the number of connected components of $\mu \setminus \bigcup_{v \in V(G)} \{\rho(v)\}$. For vertices $u, v \in V(G)$, the normal distance $\text{nd}_G(u, v)$ is defined as the length of the shortest normal curve between $\rho(u)$ and $\rho(v)$. The normal distance between two vertex-subsets $U, W \subseteq V(G)$ is defined as $\text{nd}_G(U, W) = \min_{u \in U, v \in W} \text{nd}_G(u, v)$. We also use $\text{nd}_G(U, \{v\})$ for $\text{nd}_G(U, \{v\})$ and $\text{nd}_G(u, W)$ for $\text{nd}_G(\{u\}, W)$.

A noose of $G$ is a closed normal curve on $\Sigma$ that does not intersect with itself. A noose $\nu$ of $G$ separates $\Sigma$ into two open regions $R_1$ and $R_2$ and induces a separation $(A, \overline{A})$ of $G$ with $A = \{e \in E(G) | \rho(e) \subseteq R_1\}$ and $\overline{A} = \{e \in E(G) | \rho(e) \subseteq R_2\}$. We also say $\nu$ induces edge subset $A$ ($\overline{A}$). A separation (resp. an edge subset) of $G$ is called noose-induced if there is a noose which induces the separation (resp. edge subset). A noose $\nu$ separates two edge subsets $A_1$ and $A_2$ if $\nu$ induces a separation $(A, \overline{A})$ with $A_1 \subseteq A$ and $A_2 \subseteq \overline{A}$. We also say that the noose induced subset $A$ separates $A_1$ and $A_2$.

For plane graph $G$ and a noose $\nu$ induced $A \subseteq E(G)$, we denote by $G \mid A$ the plane hypergraph obtained by replacing all edges of $A$ with edge $\partial(A)$ (i.e., $V(G \mid A) = (V(G) \setminus V(A)) \cup \partial(A)$ and $E(G \mid A) = (E(G) \setminus A) \cup \{\partial(A)\}$). An embedding of $G \mid A$ can be obtained from $G$ with $\rho(\partial(A))$ an open disk (homeomorphic to $\{(x, y)|x^2 + y^2 < 1\}$) which is the open region separated by $\nu$ and contains $A$. For a collection $\mathcal{A} = \{A_1, ..., A_r\}$ of mutually disjoint edge-subsets of $G$, $\ldots(G \mid A_1) \ldots | A_r$ is denoted by $G \mid \mathcal{A}$.
3 \( O(n \log^4 n) \) time algorithm

We give an algorithm to prove Theorem 1. Our algorithm follows the approach of the work in [20]. Given a plane graph \( G \) of \( n \) vertices, an edge \( e_0 \) of \( G \) and integers \( k, h > 0 \), let \( Z \) be the set of biconnected components of \( G \) such that for each \( Z \in Z, \) nd\(_G\)(\( e_0, V(Z) \)) = \( h \) (notice that the subgraph of \( G \) induced by the vertices with normal distance at least \( h \) from \( e_0 \) may not be biconnected, and we handle every biconnected component of the subgraph). For each \( Z \in Z \), our algorithm computes a minimum noose induced subset \( A_Z \) separating \( Z \) and \( e_0 \).

If for some \( Z \in Z \), \( |\partial(A_Z)| > k \) then the algorithm constructs a \((k + 1) \times h\) cylinder minor of \( G \) in \( O(n) \) time by Lemma 1 proved in [20]. Otherwise, a set \( A \) of noose induced subsets with the following properties is computed: (1) for every \( A_Z \in A \), \(|\partial(A_Z)| \leq k \); (2) for every \( Z \in Z \), there is an \( A_Z \in A \) which separates \( Z \) and \( e_0 \) and (3) for distinct \( A_Z, A_Z' \in A \), \( A_Z \cap A_Z' = \emptyset \). Such an \( A \) is called a good-separator for \( Z \) and \( e_0 \).

Lemma 1 [20] Given a plane graph \( G \) and integers \( k, h > 0 \), let \( A_1 \) and \( A_2 \) be edge subsets of \( G \) satisfying the following conditions: (1) each of separations \((A_1, \overline{A_1})\) and \((A_2, \overline{A_2})\) is noose-induced; (2) \( G[A_2] \) is biconnected; (3) \( \text{nd}_G(V(A_1), V(A_2)) \geq h \); and (4) every noose of \( G \) that separates \( \overline{A_1} \) and \( A_2 \) has length \( > k \). Then \( G \) has a \((k + 1) \times h\) cylinder minor and given \((G[\overline{A_1}], A_2)\), such a minor can be constructed in \( O(|V(A_1 \cap \overline{A_2})|) \) time.

Given a good-separator \( A \) for \( Z \) and \( e_0 \), our algorithm constructs a branch-decomposition of plane hypergraph \( G[A] \) with width at most \( k + 2h \) by Lemma 2 shown in [21] [35]. For each \( A_Z \in A \), the algorithm computes a cylinder minor or a branch-decomposition for the plane hypergraph \( G[\overline{A_Z}] \) recursively. If a branch-decomposition of \( G[\overline{A_Z}] \) is found for every \( A_Z \in A \), the algorithm constructs a branch-decomposition of \( G \) with width at most \( k + 2h \) from the branch-decomposition of \( G[A] \) and those of \( G[\overline{A_Z}] \) by Lemma 3 which is straightforward from the definitions of branch-decompositions.

Lemma 2 [24] [35] Let \( k > 0 \) and \( h > 0 \) be integers. Let \( G \) be a plane hypergraph with each edge of \( G \) incident to at most \( k \) vertices. If there is an edge \( e_0 \) such that for any vertex \( v \) of \( G \), \( \text{nd}_G(e_0, v) \leq h \) then given \( e_0 \), a branch-decomposition of \( G \) with width at most \( k + 2h \) can be constructed in \( O(|V(G)| + |E(G)|) \) time.

The upper bound \( k + 2h \) is shown in Theorem 3.1 in [21]. The normal distance in [21] between a pair of vertices is twice of the normal distance in this paper between the same pair of vertices. Tamaki gives a linear time algorithm to construct a branch-decomposition of width at most \( k + 2h \) [35].

Lemma 3 Given a plane hypergraph \( G \) and a noose-induced separation \((A, \overline{A})\) of \( G \), let \( T_A \) and \( T_{\overline{A}} \) be branch-decompositions of \( G[\overline{A}] \) and \( G[A] \) respectively. Let \( T_A + T_{\overline{A}} \) to be the tree obtained from \( T_A \) and \( T_{\overline{A}} \) by joining the link incident to the leaf \( \partial(A) \) in \( T_A \) and the link incident to the leaf \( \partial(A) \) in \( T_{\overline{A}} \) into one link and removing the leaves \( \partial(A) \). Then \( T_A + T_{\overline{A}} \) is a branch-decomposition of \( G \) with width \( \max\{|\partial(A)|, k_A, k_{\overline{A}}\} \) where \( k_A \) is the width of \( T_A \) and \( k_{\overline{A}} \) is the width of \( T_{\overline{A}} \).
To make a concrete progress in each recursive step, the following technique in [20] is used to compute $\mathcal{A}$. For a plane hypergraph $G$, a vertex subset $e_0$ of $G$ and an integer $d \geq 0$, let

$$\text{reach}_G(e_0, d) = \bigcup \{v \in V(G) | nd_G(e_0, v) \leq d\}$$

denote the set of vertices of $G$ with normal distance at most $d$ from set $e_0$. Let $\alpha > 0$ be an arbitrary constant. For integer $k \geq 2$, let $d_1 = \lceil \frac{k\alpha}{2} \rceil$ and $d_2 = d_1 + \lceil \frac{k\alpha}{2} \rceil$. The layer tree $\text{LT}(G, e_0)$ is defined as follows:

1. the root of the tree is $G$;
2. each biconnected component $X$ of $G[V(G) \setminus \text{reach}_G(e_0, d_1 - 1)]$ is a node in level 1 of the tree and is a child of the root; and
3. each biconnected component $Z$ of $G[V(G) \setminus \text{reach}_G(e_0, d_2 - 1)]$ is a node in level 2 of the tree and is a child of the biconnected component $X$ in level 1 that contains $Z$.

For $h = d_2$, $Z$ is the set of leaf nodes of $\text{LT}(G, e_0)$ in level 2. For a node $X$ of $\text{LT}(G, e_0)$ in level 1, let $\mathcal{Z}_X$ be the set of child nodes of $X$. It is shown in [20] that for any $Z \in \mathcal{Z}_X$, if a minimum noose in the plane hypergraph $(G | \overline{X}) | \mathcal{Z}_X$ separating $Z$ and $\overline{X}$ has length greater than $k$ then $G$ has a $(k + 1) \times \lceil \frac{k\alpha}{2} \rceil$ cylinder minor. From this, a good-separator $\mathcal{A}_X$ for $\mathcal{Z}_X$ and $\overline{X}$ can be computed in hypergraph $(G | \overline{X}) | \mathcal{Z}_X$, and the union of $\mathcal{A}_X$ for every $X$ gives a good-separator $\mathcal{A}$ for $Z$ and $e_0$.

To compute $\mathcal{A}_X$, we convert $(G | \overline{X}) | \mathcal{Z}_X$ to a weighted plane graph and compute a minimum noose induced subset $A_Z$ separating $Z \in \mathcal{Z}_X$ and $\overline{X}$ by finding a minimum face separating cycle in the weighted plane graph. We use the algorithm by Borradaile et al. [10] [11] to compute the face separating cycles.

For an open disk $D$ in $\Sigma$, let $\text{cl}(D)$ be the closure of $D$ and $\text{bd}(D) = \text{cl}(D) \setminus D$ be the boundary of $D$. For edge $\partial(X)$ in $(G | \overline{X}) | \mathcal{Z}_X$, the embedding $\rho(\partial(X))$ is an open disk and $E_{\overline{X}} = \text{bd}(\rho(\partial(\overline{X}))) \setminus \{\rho(u) | u \in \partial(\overline{X})\}$ is a set of open segments. Similarly, for each edge $\partial(Z)$ in $(G | \overline{X}) | \mathcal{Z}_X$, $E_Z = \text{bd}(\rho(\partial(Z))) \setminus \{\rho(u) | u \in \partial(Z)\}$ is a set of open segments. We convert hypergraph $(G | \overline{X}) | \mathcal{Z}_X$ to a plane graph $G_X$ as follows: edge $\partial(\overline{X})$ is replaced by the set of edges which are segments in $E_{\overline{X}}$ and for each $Z \in \mathcal{Z}_X$, edge $\partial(Z)$ is replaced by the set of edges which are segments in $E_Z$.

We denote the face in $G_X$ bounded by edges of $E_{\overline{X}}$ by $f_{\overline{X}}$. For each $Z \in \mathcal{Z}_X$, we denote the face in $G_X$ bounded by edges of $E_Z$ by $f_Z$. A face in $G_X$ which is not $f_{\overline{X}}$ or any of $f_Z$ is called a natural face in $G_X$. We convert $G_X$ to a weighted plane graph $H_X$ as follows: For each natural face $f$ in $G_X$ with $|V(f)| > 3$, we add a new vertex $u_f$ and new edges $\{u_f, v\}$ in $f$ for every vertex $v$ in $V(f)$. Each new edge $\{u_f, v\}$ is assigned the weight $1/2$. Each edge of $G_X$ is assigned the weight 1. The length of a cycle (resp. path) in $H_X$ is the sum of the weights assigned to the edges in the cycle (resp. path). For $Z \in \mathcal{Z}_X$, a minimum $(f_Z, f_{\overline{X}})$-separating cycle is a cycle separating $f_Z$ and $f_{\overline{X}}$ with the minimum length. A noose in $G_X$ is called a natural noose if it intersects only natural faces in $G_X$. It is shown in [20] that for each $Z \in \mathcal{Z}_X$, a minimum natural noose in $G_X$ separating $E_Z$ and $E_{\overline{X}}$ in $G_X$ is a
minimum noose separating \( Z \) and \( \overline{X} \) in \( (G|\overline{X})|Z_X \). By Lemma 4, such a natural noose \( \nu \) can be computed by finding a minimum \( (f_Z, f_{\overline{X}}) \)-separating cycle \( C \) in \( H_X \). The subset \( A_Z \) induced by \( \nu \) in \( (G|\overline{X})|Z_X \) is also called cycle \( C \) induced subset.

**Lemma 4** Let \( H_X \) be the weighted plane graph obtained from \( G_X \). For any \( (f_Z, f_{\overline{X}}) \)-separating cycle \( C \) in \( H_X \), there is a natural noose \( \nu \) which separates \( E_Z \) and \( E_{\overline{X}} \) in \( G_X \) with the same length as that of \( C \). For any minimum natural noose \( \nu \) in \( G_X \) separating \( E_Z \) and \( E_{\overline{X}} \), there is a \( (f_Z, f_{\overline{X}}) \)-separating cycle \( C \) in \( H_X \) with the same length as that of \( \nu \).

**Proof:** Let \( C \) be a \( (f_Z, f_{\overline{X}}) \)-separating cycle in \( H_X \). For each edge \( \{u, v\} \) in \( C \) with \( u, v \in V(G_X) \), \( \{u, v\} \) is incident to a natural face \( f \) because \( f_Z \) is not incident to \( f_{\overline{X}} \) by \( nd_{G[U]}(V(\overline{X}), V(Z)) = \lceil \frac{k+1}{2} \rceil \). We draw a simple curve with \( u, v \) as its end points in face \( f \). For each pair of edges \( \{u, u_f\} \) and \( \{u_f, v\} \) in \( C \) with \( u, v \in V(G_X) \) and \( u_f \in V(H_X) \setminus V(G_X) \), we draw a simple curve with \( u, v \) as its end points in the face \( f \) of \( G_X \) where the new added vertex \( u_f \) is placed. Then the union of the curves form a natural noose \( \nu \) which separates \( E_Z \) and \( E_{\overline{X}} \) in \( G_X \). Each of edge \( \{u, v\} \) with \( u, v \in V(G_X) \) is assigned weight 1. For a new added vertex \( u_f \), each of edges \( \{u, u_f\}, \{u_f, v\} \) is assigned weight 1/2. Therefore, the lengths of \( \nu \) and \( C \) are the same.

Let \( \nu \) be a minimum natural noose separating \( E_Z \) and \( E_{\overline{X}} \) in \( G_X \). It is obvious that \( \nu \) contains at most two vertices of every natural face because otherwise we can form a shorter separating noose. The vertices on \( \nu \) partitions \( \nu \) into a set of simple curves such that each curve has two vertices on \( \nu \) as its end points and does not intersect any vertex on \( \nu \) other than its end points. For each curve with end points \( u, v \), if \( \{u, v\} \) is an edge in \( G_X \) then we take \( \{u, v\} \) in \( H_X \) as a candidate. Assume that \( \{u, v\} \) is not an edge in \( G_X \). Then the curve is drawn in a face \( f \) of \( G_X \) incident to \( u, v \). We take edges \( \{u, u_f\}, \{u_f, v\} \) in \( H_X \) as candidates, where \( u_f \) is the vertex added in face \( f \) in getting \( H_X \). Since \( \nu \) contains at most two vertices of every natural face, these candidates form a \( (f_Z, f_{\overline{X}}) \)-separating cycle \( C \) in \( H_X \). Because each edge of \( G_X \) is given weight 1 and each added edge is given weight 1/2 in \( H_X \), the lengths of \( C \) and \( \nu \) are the same. \( \square \)

We assume that for every pair of vertices \( u, v \) in \( H_X \), there is a unique shortest path between \( u \) and \( v \). This can be realized by perturbing the edge weight \( w(e) \) of each edge \( e \) in \( H_X \) as follows. Assume that the edges in \( H_X \) are \( e_1, \ldots, e_m \). For each edge \( e_i \), let \( w'(e_i) = w(e_i) + \frac{1}{2^{i+1}} \). Then it is easy to check that for any pair of vertices \( u \) and \( v \) in \( H_X \), there is a unique shortest path between \( u \) and \( v \) w.r.t. to \( w' \); and the shortest path between \( u \) and \( v \) w.r.t. \( w' \) is a shortest path between \( u \) and \( v \) w.r.t. \( w \).

For a plane graph \( G \), a **minimum cycle base tree** (MCB tree) introduced in [10, 11] is an edge-weighted tree \( \tilde{T} \) such that

- There is a bijection from the faces of \( G \) to the nodes of \( \tilde{T} \);
- removing each edge \( e \) from \( \tilde{T} \) partitions \( \tilde{T} \) into two subtrees \( \tilde{T}_1 \) and \( \tilde{T}_2 \); this edge \( e \) corresponds to a cycle which separates every pair of faces \( f \) and \( g \) with \( f \) in \( \tilde{T}_1 \) and \( g \) in \( \tilde{T}_2 \); and

8
• for any distinct faces \( f \) and \( g \), the minimum-weight edge on the unique path between \( f \) and \( g \) in \( \tilde{T} \) has weight equal to the length of a minimum \((f, g)\)-separating cycle.

The next lemma gives the running time for computing a MCB tree of a plane graph and that for obtaining a cycle from the MCB tree.

**Lemma 5** \([10, 11]\) Given a plane graph \( G \) of \( n \) vertices with positive edge weights, a MCB tree of \( G \) can be computed in \( O(n \log^4 n) \) time. Further, for any distinct faces \( f \) and \( g \) in \( G \), given a minimum weight edge in the path between \( f \) and \( g \) in the MCB tree, a minimum \((f, g)\)-separating cycle \( C \) can be obtained in \( O(|C|) \) time, \( |C| \) is the number of edges in \( C \).

Using Lemma 5 for computing a MCB tree \( \tilde{T} \) of \( H_X \) and \( A_X \), our algorithm is summarized in Procedure Branch-Minor below. In the procedure, \( U \) is a noose induced edge subset and initially \( U = \{e_0\} \).

**Procedure** Branch-Minor(\( G|U \))

**Input:** A biconnected plane hypergraph \( G|U \) with \( \partial(U) \) specified, \( |\partial(U)| \leq k \) and every other edge has two vertices.

**Output:** Either a branch-decomposition of \( G|U \) of width at most \( k + 2h \), \( h = \frac{d_2}{2} \), or a \((k + 1) \times \lceil \frac{k+1}{2} \rceil \) cylinder minor of \( G \).

1. If \( nd_{G|U}(\partial(U), v) \leq h \) for every \( v \in V(G|U) \) then apply Lemma 2 to find a branch-decomposition of \( G|U \). Otherwise, proceed to the next step.

2. Compute the layer tree \( LT(G|U, \partial(U)) \).

   For every node \( X \) of \( LT(G|U, \partial(U)) \) in level 1, compute \( A_X \) as follows:
   
   (a) Compute \( H_X \) from \( (G|X)|Z_X \).
   
   (b) Compute a MCB tree \( \tilde{T} \) of \( H_X \) by Lemma 5.
   
   (c) Find a face \( f_Z, Z \in Z_X, \) in \( \tilde{T} \) by a breadth first search such that the path between \( f_Z \) and \( f_X \) in \( \tilde{T} \) does not contain \( f_{Z'} \) for any \( Z' \in Z_X \) with \( Z' \neq Z \). Find the minimum weight edge \( e_Z \) in the path between \( f_Z \) and \( f_X \), and the cycle \( C \) from edge \( e_Z \).

   If \( C \) has length \( > k \) then compute a \((k + 1) \times \lceil \frac{k+1}{2} \rceil \) cylinder minor by Lemma 1 and terminate.

   Otherwise, compute the cycle \( C \) induced subset \( A_Z \) and include \( A_Z \) to \( A_X \). For each node \( f \) of \( \tilde{T} \), if edge \( e_Z \) is in the path between \( f \) and \( f_X \) in \( \tilde{T} \) then delete \( f \) from \( \tilde{T} \).

   Repeat the above until \( \tilde{T} \) does not contain any \( f_Z \) for \( Z \in Z_X \).

   Let \( A = \bigcup_{X \text{ level 1 node}} A_X \) and proceed to the next step.

3. For each \( A \in \mathcal{A} \), call Branch-Minor(\( G|\overline{A} \)) to construct a branch-decomposition \( T_A \) or a cylinder minor of \( G|\overline{A} \).
If a branch-decomposition $T_A$ is found for every $A \in \mathcal{A}$, Lemma 2 is applied to $(G|U)|\mathcal{A}$ to construct a branch-decomposition $T_0$ of $(G|U)|\mathcal{A}$ and Lemma 3 is used to combine these branch-decompositions $T_A$, $A \in \mathcal{A}$, and $T_0$ into a branch-decomposition $T$ of $G|U$ and return $T$.

**Proof of Theorem 1** The input hypergraph $G|\overline{A}$ of our algorithm in each recursive step for $A \in \mathcal{A}$ is biconnected. For the $\mathcal{A}_X$ computed in Step 2, obviously (1) for every $A_Z \in \mathcal{A}_X$, $|\partial(A_Z)| \leq k$; (2) due to the way we find the cycles from the MCB tree, for every $Z \in \mathcal{Z}_X$, there is exactly one subset $A_Z \in \mathcal{A}_X$ separating $Z$ and $\overline{X}$; and (3) from the unique shortest path in $H_X$, for distinct $A_Z, A_Z' \in \mathcal{A}_X$, $A_Z \cap A_Z' = \emptyset$. Therefore, $\mathcal{A}_X$ is a good-separator for $\mathcal{Z}_X$ and $\overline{X}$. From this, $\mathcal{A}$ is a good separator for $\mathcal{Z}$ and $U$ and our algorithm computes a branch-decomposition or a $(k + 1) \times \left\lceil \frac{k+1}{2} \right\rceil$ cylinder minor of $G$. The width of the branch-decomposition computed is at most

$$k + 2h = k + 2(d_1 + \left\lceil \frac{k + 1}{2} \right\rceil) \leq k + 2(\alpha k) + (k + 1) \leq (2 + \delta)k,$$

where $\delta$ is the smallest constant with $\delta k \geq \alpha k + 3$.

Let $M, m_x, m$ be the numbers of edges in $G[\text{reach}_{G|U}(\partial(U), d_2)], (G|\overline{X})|\mathcal{Z}_X, H_X$, respectively. Then $m = O(m_x)$. In Step 2, the layer tree $LT(G|U, \partial(U))$ can be computed in $O(M)$ time. For each level 1 node $X$, it takes $O(m)$ time to compute $H_X$ and by Lemma 5, it takes $O(m \log^4 m)$ time to compute a MCB tree $\tilde{T}$ of $H_X$. In Step 2(c), it takes $O(m)$ time to compute a cylinder minor by Lemma 1. From Property (3) of a good-separator, each edge of $H_X$ appears in at most two cycles which induce the subsets in $\mathcal{A}_X$. So Step 2(c) takes $O(m)$ time to compute $\mathcal{A}_X$. Therefore, the total time for Steps 2(a)-(c) is $O(m \log^4 m)$. For distinct level 1 nodes $X$ and $X'$, the edge sets of subgraphs $(G|\overline{X})|\mathcal{Z}_X$ and $(G|\overline{X'})|\mathcal{Z}_X'$ are disjoint. From this, $\sum_{X: \text{level 1 node}} m_x = O(M)$. Therefore, Step 2 takes $\sum_{X: \text{level 1 node}} O(m_x \log^4 m_x) = O(M \log^4 M)$ time.

The time for other steps in Procedure Branch-Minor$(G|U)$ is $O(M)$. The number of recursive calls in which each vertex of $G|U$ is involved in the computation of Step 2 is $O(\frac{1}{a}) = O(1)$. Therefore, the running time of the algorithm is $O(n \log^4 n)$. □

### 4 An algorithm for Theorem 2

To get an algorithm for Theorem 2 we use a different approach to compute minimum face separating cycles in $H_X$. A connected subgraph of a plane graph $G$ is called a **piece** of $G$. For a piece $P$ of $G$, the vertex-cut set $\partial P$ partitioning $E(G)$ into $E(P)$ and $E(G) \setminus E(P)$ is called the **boundary** of $P$. In this algorithm, we decompose $H_X$ into pieces which form a **recursive r-division** of $H_X$ [25] and then compute minimum face separating cycles using the techniques in [10, 11].

For a positive integer $r < n$, an **r-division** of a plane graph $G$ is a decomposition of $G$ into $O(\frac{n}{r})$ pieces such that each piece has $O(r)$ vertices and $O(\sqrt{r})$ boundary vertices [17]. Let $r = (r_1, r_2, \ldots)$ be an exponentially decreasing sequence of positive integers. A recursive
The external dense distance graph that each edge \( i \) between \( u \) putting a minimum of a piece graph \([14]\). Two types of dense distance graphs are used in \([10, 11]\). The internal dense distance graph of a piece \( P \), denoted by \( \text{intDDG}(P) \), is a complete graph on \( \partial P \) that each edge \( \{u, v\} \) in \( \text{intDDG}(P) \) is assigned the length of the shortest path in \( P \) between \( u \) and \( v \) as weight. The external dense distance graph of \( P \), denoted by \( \text{extDDG}(P) \), is a complete graph on \( \partial P \) that each edge \( \{u, v\} \) is assigned the length of the shortest path in the graph \( G[E(G) \setminus E(P)] \) between \( u \) and \( v \) as weight. Given a recursive \( r \)-division \( R_G \) of \( G \), the \( \text{intDDG}(P) \) for all pieces in \( R_G \) can be computed in \( O(n \log^2 n) \) time by the multiple source shortest paths algorithm due to Klein \([24]\), and it is shown in \([10, 11]\) that the \( \text{extDDG}(P) \) for all pieces in \( R_G \) can be computed in \( O(n \log^3 n) \) time by FR-Dijkstra Algorithm.

Given a piece \( P \) and a face \( f \) of a plane graph \( G \), we say \( f \) is touched by \( P \) if there is an edge in \( P \) incident to \( f \) in \( G \). Let \( G \) be a plane graph with positive edge weights. We replace Steps 2(b)(c) in Procedure Branch-Minor with the following steps to get the algorithm.

- Compute a recursive \( r \)-division \( R_H \) of \( H_X \), where \( r = (r_1, r_2, ..., r_l) \) with \( r_1 = n/2 \), \( r_i = r_{i-1}/2 \) for \( 1 < i \leq l \) and \( r_l = \theta(\sqrt{m}) \).

- For every piece \( P \) in \( R_H \), compute \( \text{intDDG}(P) \) and \( \text{extDDG}(P) \).

- For every \( Z \in Z_X \), compute a minimum \((f_Z, \mu_X)\)-separating cycle \( C \) using \( R_H \) and \( \text{intDDG}(P)/\text{extDDG}(P) \).

  If the length of \( C \) is greater than \( k \) then compute a \((k+1) \times \lceil \frac{k+1}{2} \rceil \) cylinder minor by Lemma \([11]\) and terminate. Otherwise, keep this cycle.

- Compute \( A_X \) from the minimum face separating cycles.

The results of \([10, 11]\) summarized in the lemmas below give the running time for computing a minimum \((f_Z, \mu_X)\)-separating cycle in \( H_X \) using a recursive \( r \)-division of \( H_X \) and \( \text{intDDG}(P)/\text{extDDG}(P) \).

**Lemma 6** \([11]\) Let \( P \) be a piece of \( G \) and \( f, g \) be two faces of \( G \) touched by \( P \). Given \( \text{intDDG}(P) \) and \( \text{extDDG}(P) \), a minimum \((f, g)\)-separating cycle \( C \) in \( G \) can be computed in \( O(|V(P)| + |\partial P|^2 \log^2 n) \) time.

**Lemma 7** \([11]\) Let \( P \) be a piece of \( G \) that is decomposed into sub-pieces \( Q_1, ..., Q_j \) in a recursive \( r \)-division of \( P \) and \( f, g \) be two faces of \( G \) with \( f \) touched by \( Q_i \) and \( g \) touched by \( Q_j \). Given \( \text{intDDG}(Q_i) \) for \( 1 \leq i \leq j \) and \( \text{extDDG}(P) \), a minimum \((f, g)\)-separating cycle \( C \) in \( G \) can be computed in \( O((|V(Q_1)| + |V(Q_j)|) + |\partial P| + \sum_{i=1}^{j} |\partial Q_i|) \log^3 n) \) time.
Figure 1: Pieces in $R_H$ used for computing a minimum $(f_Z, f_X)$-separating cycle.

Proof of Theorem 2:
We only show the time complexity of our algorithm. The other parts of the proof for the theorem are similar to those for Theorem 1.

Let $M, m_x, m$ be the numbers of edges in $G[\text{reach}(U, d_2), (G[X])_{E_X, H_X}$, respectively. Then $m = O(m_x)$. It takes $O(m)$ time to compute a recursive $r$-division $R_H$ of $H_X$ [25]. For every piece $P$ in $R_H$, $\text{intDDG}(P)$ and $\text{extDDG}(P)$ can be computed in $O(m \log^2 m)$ time [10, 11].

For a pair of faces $f_Z$ and $f_X$ to be separated, there is a piece in $R_H$ that touches $f_Z$ and $f_X$. A piece touching $f_Z$ and $f_X$ with a minimum size is called a minimum touching piece for $f_Z$ and $f_X$. We find a minimum touching piece for every pair of faces $f_Z$ and $f_X$ to be separated. The total time for this is $O(m \log^2 m)$. Let $P$ be a minimum touching piece for $f_Z$ and $f_X$. If $P$ is a leaf node of $R_H$, then a minimum $(f_Z, f_X)$-separating cycle can be computed in $O((|V(P)| + |\partial P|) \log^2 |P|) = O(\sqrt{m} \log^2 m)$ time (Lemma 6).

Assume that $P$ is not a leaf node. Let $P_Z$ and $P_X$ be the leaf nodes of $R_H$ that touches $f_Z$ and $f_X$, respectively. Assume that the path $L$ between $P_Z$ and $P_X$ in $R_H$ has the node sequence $P_Z, P_2, ..., P_i = P, ..., P_r, P_X$ (see Figure 1). Each node $P_i$ in $R_H$, $1 < i \leq r$, has a constant number of child nodes, one is in the path $L$ and the others are not. Let $\{Q_2, ..., Q_{j-1}\}$ be the set of all child nodes of $P_i$, $1 < i \leq r$, that are not in the path $L$, and let $Q_1 = P_Z$ and $Q_j = P_X$. Then $Q_1, Q_2, ..., Q_j$ form $P$. By Lemma 7, a minimum $(f_Z, f_X)$-separating cycle can be computed in $O((|V(Q_1)| + |V(Q_j)| + |\partial P| + \sum_{i=1}^{j} |\partial Q_i|) \log^3 m)$ time. Notice that for each level in $R_H$ there are at most $b = O(1)$ of nodes in $\{Q_1, Q_2, ..., Q_j\}$.
and on level \(i\) a piece has \(O(\sqrt{\frac{m}{2^i}})\) boundary vertices. So we have \(|\partial P| = O(\sqrt{m})\) and
\[
\sum_{i=1}^{j} |\partial Q_i| \leq b\left(\sqrt{\frac{m}{2}} + \sqrt{\frac{m}{4}} + \ldots + m^{\frac{1}{4}}\right) = O(\sqrt{m}).
\]
From this, \(|V(P_1)| = O(\sqrt{m})\) and \(|V(P_2)| = O(\sqrt{m})\), a minimum \((f_Z, f_X)\)-separating cycle can be computed in \(O(\sqrt{m}\log^3 m)\) time. If a \((f_Z, f_X)\)-separating cycle has length greater than \(k\) then a cylinder minor is computed in \(O(m)\) time by Lemma 1.

Assume that minimum cycles separating all \(f_Z, Z \in Z_X\), and \(f_X\) are computed. This can be done in \(O(|Z_X|\sqrt{m}\log^3 m)\) time. Let \(\mathcal{C}\) be the set of cycles computed above. Then for every cycle in \(\mathcal{C}\), (1) the cycle induced subset \(A_Z\) has \(|\partial(A_Z)| \leq k\) and (2) for every \(Z \in Z_X\), there is a cycle induced subset \(A_Z\) separating \(Z\) and \(X\). For distinct cycles \(C, C' \in \mathcal{C}\), the \(C\) induced \(A_Z\) and \(C'\) induced \(A_Z\) satisfy \(A_Z \subseteq A_{Z'}\) or \(A_{Z'} \subseteq A_Z\) or \(A_Z \cap A_{Z'} = \emptyset\) due to the unique shortest path in \(H_X\). To get Property (3) \((A_Z \cap A_{Z'} = \emptyset)\) of a good-separator for \(A_X\), we need to select some cycles of \(\mathcal{C}\) to compute \(A_X\).

A cycle \(C\) separating \(f_Z\) and \(f_X\) cuts the sphere \(\Sigma\) into two regions, one contains \(f_Z\) and the other contains \(f_X\). We denote the region containing \(f_Z\) by \(\text{ins}(C)\). A cycle \(C \in \mathcal{C}\) is called maximal if for any \(C' \in \mathcal{C}\) with \(C' \neq C\), \(\text{ins}(C) \subseteq \text{ins}(C')\). We orient every cycle \(C\) such that \(\text{ins}(C)\) is on the right side when we proceed on the oriented \(C\). The orientation can be done in \(O(k)\) time when the cycle is computed. For a cycle \(C\), every edge in the subset \(A_Z\) induced by \(C\) is in \(\text{ins}(C)\) and every edge in \(\text{ins}(C)\) is in \(A_Z\). We say cycles \(C\) and \(C'\) are nested if \(\text{ins}(C) \subseteq \text{ins}(C')\) or \(\text{ins}(C') \subseteq \text{ins}(C)\). For cycles \(C\) and \(C'\) having a common edge \(e\), if the orientations of \(e\) in \(C\) and \(C'\) are the same then \(C\) and \(C'\) are nested due to the unique shortest path in \(H_X\), and we say \(C\) and \(C'\) have a common oriented edge, otherwise \(\text{ins}(C) \cap \text{ins}(C') = \emptyset\). For cycles \(C\) and \(C'\), if \(C'\) has an edge in \(\text{ins}(C)\) then \(\text{ins}(C') \subseteq \text{ins}(C)\) by the unique shortest path in \(H_X\).

We find all maximal cycles in \(\mathcal{C}\) for computing \(A_X\) as follows:

1. Assume that \(\mathcal{C} = \{C_1, \ldots, C_r\}\). We process \(C_1\), include \(C_1\) to \(D\).
2. We process cycle \(C_i\) for \(i = 2, 3, \ldots, r\):
   - Proceed on the oriented \(C_i\). If \(C_i\) does not have a common oriented edge with any cycle in \(D\) then include \(C_i\) to \(D\). Otherwise, \(C_i\) has a common oriented edge with a cycle \(C \in D\).
   - For every cycle \(C \in D\) having a common oriented edge with \(C_i\), there is a common edge \(\{u, v\}\) in \(C\) and \(C_i\) such that \(C\) has an edge \(\{v, w\}\), \(C_i\) has an edge \(\{v, x\}\) and \(\{v, w\} \neq \{v, x\}\). If edge \(\{v, w\}\) of \(C\) is in \(\text{ins}(C_i)\) then delete \(C\) from \(D\), otherwise, discard \(C_i\).
     - If \(C_i\) has not been discarded then include \(C_i\) to \(D\).
3. Perform an edge search in \(H_X\) starting from the edges of \(E(f_X)\) to find all maximal cycles in \(D\) and delete cycle \(C\) from \(D\) if \(C\) is not maximal.
At the end of Step (2) above, all cycles of \( C \) have been processed, every maximal cycle of \( C \) is in \( D \) and for any two distinct cycles \( C, C' \in D \) either \( \text{ins}(C) \cap \text{ins}(C') = \emptyset \) or \( C \) and \( C' \) are nested and do not have a common edge. Therefore, all maximal cycles of \( C \) can be found by searching edges from \( E(f_X) \). It takes \( O(k) \) time to process a cycle \( C_i \) because for each oriented edge \( e \) in \( C_i \), \( e \) appears in at most one cycle in \( D \), and takes \( O(k) \) time to delete a cycle from \( D \). From this, Step (2) takes \( O(|C|k) \) time. Step (3) takes \( O(m) \) time. Therefore and from \( k = \sqrt{n} \) \[16\], it takes \( O(m + |C|k) = O(m + |Z_X|\sqrt{n}) \) time to find all maximal cycles and \( A_X \). Thus, it takes

\[
\sum_{x: \text{ level } 1 \text{ node}} O((m_x + |Z_X|\sqrt{m_x}) \log^3 m_x + |Z_X|\sqrt{n}) = O((M + |Z|\sqrt{M}) \log^3 M + |Z|\sqrt{n})
\]
time to compute \( A \).

The number of recursive calls in which each vertex of \( G|U \) is involved in the algorithm is \( O(\frac{1}{\sqrt{n}}) = O(1) \). Therefore, the running time of the algorithm is \( O((n + n_z\sqrt{n}) \log^3 n) \). \( \square \)

5 \hspace{1em} An algorithm for Theorems 3

We first explain some notions and techniques introduced in \[26, 27\] that will be used in an algorithm for Theorem 3. For each face \( f_Z \), \( Z \in Z_X \), of \( H_X \), we add a new vertex in \( f_Z \), call this new vertex crest and denote it also by \( Z \). We add new edges \( \{Z, v\} \) for \( v \in V(f_Z) \). For each added edge, we assign a weight \( 1/2 \). For each vertex \( u \) in \( G_X \), let \( h(u) = nd_{G_X}(V(f_X), u) \). For each face \( f \) in \( G_X \), let \( h(f) = \min_{u \in V(f)} h(u) \). For each vertex \( u \) in \( H_X \) we define a label \( l(u) \): if \( u \) is a vertex in \( G_X \) then \( l(u) = h(u) \), otherwise \( u \) is an added vertex to a face \( f \) and \( l(u) = h(f) + 1/2 \).

For a path \( L \) in \( H_X \), the depth of \( L \) is the smallest \( l(u) \) of any vertex \( u \) in \( L \). A ridge between two crests \( Z \) and \( Z' \) is a path between \( Z \) and \( Z' \) with the largest depth. A crest separator on a ridge \( R \) between crests \( Z \) and \( Z' \) in \( H_X \) is a tuple \( S = (L_1, L_2) \), where \( L_1 \) is a selected shortest path from a vertex \( u \) on \( R \) with a smallest \( l(u) \) to a vertex in \( V(f_X) \) and \( L_2 \) is a selected shortest path from a neighbor \( v \) of \( u \) to a vertex in \( V(f_X) \) such that \( L_1, L_2 \) and edge \( \{u, v\} \) decomposes \( H_X \) into two pieces, one containing crest \( Z \) and the other having crest \( Z' \). Each of \( L_1 \) and \( L_2 \) is called a downpath in \( S \).

Given a set \( W_X \) of crests in \( H_X \) and a set \( S_X \) of crest separators, \( S_X \) decomposes \( G \) into pieces \( P_1, ..., P_r \). Since a crest separator does not contain any vertex of a crest, each crest is in one piece. We consider the graph \( T_S \) that \( V(T_S) = \{P_1, ..., P_r\} \) and there is an edge \( \{P_i, P_j\} \) if there is a crest separator in \( S_X \) on a ridge between a crest in \( P_i \) and a crest in \( P_j \) (we say the crest separator is on edge \( \{P_i, P_j\} \)). The tuple \( (H_X, S_X, W_X) \) is called a good mountain structure tree (GMST) if \( T_S \) is a tree and each node of \( T_S \) has exactly one crest of \( W_X \). We call \( T_S \) the underlying tree of the GMST \( (H_X, S_X, W_X) \). In a GMST \( (H_X, S_X, W_X) \), \( |S_X| = |W_X| - 1 \). We assume each piece \( P \) contains every crest separator on an edge incident to \( P \) in \( T_S \).
Given a GMST \((H_X, S_X, W_X)\), we choose a leaf node \(Q\) in \(T_S\) as the root. Each crest separator \(S \in S_X\) decomposes \(H_X\) into two pieces, one contains \(Q\), called the upper piece by \(S\), and the other does not, called the lower piece by \(S\). For two vertices \(u\) and \(v\) in \(S\), let \(l_S(u, v)\) be the minimum length of a path consisting of only edges in \(S\) between \(u\) and \(v\). We define \(\text{upDDG}(S)\) (resp. \(\text{lowDDG}(S)\)) to be the weighted complete graph on the vertices in \(S\) such that for every pair of vertices \(u\) and \(v\) in \(S\), if the length of a shortest path in the upper piece (resp. lower piece) between \(u\) and \(v\) is smaller than \(l_S(u, v)\) then the weight of edge \(\{u, v\}\) is the length of the shortest path in the upper piece (resp. lower piece), otherwise the weight of \(\{u, v\}\) is infinitive. Notice that if \(u\) and \(v\) are in the same down path of \(S\), the weight of \(\{u, v\}\) is infinitive.

It is shown in \([26, 27]\) that for a plane graph \(G\), a GMST \((G, S, W)\) can be computed in \(O(|V(G)|k)\) time, and given a GMST \((G, S, W)\), \(\text{upDDG}(S)\) and \(\text{lowDDG}(S)\) for all \(S \in S\) can be computed in \(O(|V(G)|k + |S|k^3)\) time using a linear time algorithm by Thorup [36] for the single shortest path problem in graphs with integer edge weight as a subroutine. This requires that the edge weight in \(G\), \(\text{upDDG}(S)\) and \(\text{lowDDG}(S)\) can be expressed in \(O(1)\) words in integer form. So the perturbation technique used in the previous two algorithms cannot be used in the computation above. In the algorithm for Theorem 3, we do not assume uniqueness of shortest paths when we compute minimum separating cycles. For a minimum \((f_Z, f_X)\)-separating cycle \(C\), recall that \(C\) cuts \(\Sigma\) into two regions and \(\text{ins}(C)\) is the region containing \(f_Z\). We say two cycles \(C\) and \(C'\) intersect with each other if \(\text{ins}(C) \cap \text{ins}(C') \neq \emptyset\), \(\text{ins}(C) \setminus \text{ins}(C') \neq \emptyset\) and \(\text{ins}(C') \setminus \text{ins}(C) \neq \emptyset\). In the previous sections, we do not have intersected minimum separating cycles because of the uniqueness of shortest paths. However, two minimum separating cycles computed without the assumption of unique shortest paths may intersect with each other. We eliminate each intersected cycle pair using the properties of GMST and \(\text{upDDG}(S)/\text{lowDDG}(S)\), and then compute a good separator for \(A_X\).

Our algorithm for Theorem 3 decomposes \(H_X\) into a GMST \((H_X, S_X, W_X)\) by crest separators, computes \(\text{upDDG}(S)\) and \(\text{lowDDG}(S)\) for every crest separator \(S \in S_X\), and finds the minimum face separating cycles using the GMST and \(\text{upDDG}(S)/\text{lowDDG}(S)\). More specifically, we replace Steps 2(b)(c) in Procedure Branch-Minor with the following steps to get an algorithm for Theorem 3:

1. Decompose \(H_X\) by crest separators into a good mountain structure tree GMST \((H_X, S_X, W_X)\).
2. Compute \(\text{upDDG}(S)\) and \(\text{lowDDG}(S)\) for every crest separator \(S \in S_X\).
3. Mark every crest in \(W_X\) as non-separated, repeat the following until all crests are marked as separated.
   - If there exist a non-separated crest \(Z \in Z_X\), compute a minimum \((f_Z, f_X)\)-separating cycle \(C\) using the GMST and \(\text{upDDG}(S)/\text{lowDDG}(S)\). We call \(C\) the cycle computed for crest \(Z\).
   - If the length of \(C\) is greater than \(k\) then compute a \((k + 1) \times \lceil \frac{k+1}{2} \rceil\) cylinder minor by Lemma 1 and terminate. Otherwise, keep this cycle and mark every crest in \(\text{ins}(C)\) as separated.
(4) Compute $A_X$ from the minimum face separating cycles.

We first show the time complexity of our algorithm for computing the minimum face separating cycles for all crests.

**Lemma 8** The minimum face separating cycles for all crests can be computed in $O(nk + n_k k^3)$ time.

**Proof:** Let $M, m_x, m$ be the numbers of edges in $G[\text{reach}_{G[U]}(\partial(U), d)]$, $(G[U]|Z)|Z_X, H_X$, respectively. Then $m = O(m_x)$. We compute a GMST $(H_X, S_X, W_X)$, and $upDDG(S)$ and $lowDDG(S)$ for all $S \in S$. This can be done in $O(nk + |Z_X|k^3)$ time [26] [27].

For a non-separated crest $Z \in W_X$, let $P$ be the piece in the GMST $(H_X, S_X, W_X)$ containing $Z$. Then there is a shortest path $L$ between $Z$ and a vertex in $V(f_X)$ which is contained in $P$. This shortest path $L$ can be found in $O(k)$ time. Let $P(L)$ be the plane graph obtained from $P$ by cutting along $L$: duplicating every edge and internal vertex in $L$ and creating a new face bounded by edges of $L$ and their duplicates. Let $extDDG(P)$ be the external dense distance graph of $P$. For a vertex $u$ in path $L$, let $u'$ be the duplicate of $u$ and $C_u$ be a shortest path in the graph consisting of $P(L)$ and $extDDG(P)$ between $u$ and $u'$. Let $x$ be a vertex in $L$ such that $C_x$ has the minimum length among the paths $C_u$ for all vertices $u$ in $L$. Then $C_x$ gives a minimum $(f_z, f_{\overline{X}})$-separating cycle in $H_X$ [29].

Let $S'$ be the crest separator on the edge between $P$ and its parent node and let $S_P$ be the set of crest separators on an edge between $P$ and a child node of $P$ in underlying tree $T_S$. Then we can replace $extDDG(P)$ (which may have too many edges) by $upDDG(S')$ and $lowDDG(S)$, $S \in S_P$, for computing $C_x$ as shown below. For any two vertices $u$ and $v$ in $extDDG(P)$, the edge $\{u, v\}$ represents a shortest path between $u$ and $v$ in the subgraph induced by $E(H_X) \setminus E(P)$. If this path is not represented by an edge in $upDDG(S')$ or $lowDDG(S)$, $S \in S_P$, then the path is cut into several sub-paths by the crest-separators $S'$ and $S \in S_P$. Each of the sub-paths is represented by an edge in $upDDG(S')$ or $lowDDG(S)$, $S \in S_P$. So, by the approach described in [10] [11], it takes

$$O(|E(P)| + |upDDG(S)| + |\cup_{S \in S_P} lowDDG(S)|) \log^2 k)$$

$$O(|E(P)| + \Delta(P)k^2) \log^2 k)$$

time to compute a minimum $(f_z, f_{\overline{X}})$-separating cycle, where $\Delta(P)$ is the number of edges incident to $P$ in $T_S$. The total time for computing a minimum $(f_z, f_{\overline{X}})$-separating cycle for every $Z \in Z_X$ is

$$\sum_{P \in V(T_S)} O(|E(P)| + \Delta(P)k^2) \log^2 k) = O(|E(P)| + \Delta(P)k^2) \log^2 k).$$

Summing up as the proof of Theorem [11] the lemma holds. □

The next lemma gives a base for eliminating intersected cycle pairs.
Lemma 9 Let $C_1$ be the minimum $(f_Z, f_{\bar{X}})$-separating cycle computed for crest $Z$ and $W(C_1)$ be the set of crests in $\text{ins}(C_1)$. Let $Z'$ be a crest not in $W(C_1)$ and $C_2$ be the minimum $(f_{Z'}, f_{\bar{X}})$-separating cycle computed for $Z'$. If $C_1$ and $C_2$ intersect with each other, then there is a cycle $C$ such that $\text{ins}(C) = \text{ins}(C_1) \cup \text{ins}(C_2)$ and the length of $C$ is the same as that of $C_2$.

Proof: Let $\{Z = Z_1, Z_2\}, \{Z_2, Z_3\}, \ldots, \{Z_{r-1}, Z_r = Z'\}$ be the path between $Z$ and $Z'$ in the underlying tree $T_S$ of the GMST. By the property of the GMST, if $Z_i \in \text{ins}(C_1)$ then every $Z_j \in \text{ins}(C_1)$ for $j < i$, and if $Z_i \in \text{ins}(C_2)$ then every $Z_j \in \text{ins}(C_2)$ for $j > i$. Since $Z' \notin \text{ins}(C_1)$, there is a $Z_i$, $1 \leq i < r$, such that $Z_i \in \text{ins}(C_1)$ but $Z_j \notin \text{ins}(C_1)$ for $i < j$. Let $S$ be the crest separator on the ridge between $Z_i$ and $Z_{i+1}$ in the GMST (see Figure 2). We assume that $Z$ is in the upper piece by $S$ and $Z'$ is in the lower piece by $S$. If $C_1$ contains an edge $e = \{u, v\}$ in $\text{low}\text{DDG}(S)$ then the shortest path represented by $e$ in the lower piece has length smaller than $l_S(u, v)$, implying that $u$ and $v$ are not in the same downpath in $S$ and the shortest path does not intersect with the ridge between $Z_i$ and $Z_{i+1}$. These imply that $Z_{i+1} \notin \text{ins}(C_1)$, a contradiction. Therefore, $C_1$ does not have any edge in $\text{low}\text{DDG}(S)$ and intersects the ridge between $Z_i$ and $Z_{i+1}$. If $C_2$ does not have any edge in $\text{up}\text{DDG}(S)$ then $\text{ins}(C_1) \cap \text{ins}(C_2) = \emptyset$, a contradiction to that $C_1$ and $C_2$ are intersected with each other. So, $C_2$ has an edge $e = \{u, v\}$ in $\text{up}\text{DDG}(S)$, where $u$ and $v$ are not in the same downpath.

Let $C_2(u, v)$ be the shortest path in the upper piece by $S$ represented by edge $e = \{u, v\}$ in $\text{up}\text{DDG}(S)$. Let $u_1$ and $v_1$ be the first vertex and last vertex at which cycle $C_2$ intersects cycle $C_1$, respectively, when we proceed on $C_2$ from $u$ to $v$ (see Figure 2). Let $C_2(u_1, v_1)$ be the subpath of $C_2(u, v)$ connecting $u_1$ and $v_1$. Let $C_1(u_1, v_1)$ be the subpath of $C_1$ which connects $u_1$ and $v_1$ and intersects the ridge between $Z_i$ and $Z_{i+1}$. Since the length of $C_2(u, v)$ is smaller than $l_S(u, v)$, the length of $C_2(u_1, v_1)$ is smaller than that of $C_1(u_1, v_1)$. From this, $Z \in \text{ins}(C_2)$, because otherwise, we can replace $C_1(u_1, v_1)$ with $C_2(u_1, v_1)$ to get a separating cycle for $Z$ with a smaller length, a contradiction to that $C_1$ is a minimum separating cycle for $Z$.

For each connected region $R$ in $\text{ins}(C_1) \setminus \text{ins}(C_2)$, the boundary of $R$ consists of a subpath $C_1(R)$ of $C_1$ and a subpath $C_2(R)$ of $C_2$. The lengths of $C_1(R)$ and $C_2(R)$ are the same, otherwise, we can get a separating cycle for $Z$ or $Z'$ with length smaller than that of $C_1$ or $C_2$, respectively, a contradiction to that $C_1$ is a minimum separating cycle for $Z$ and $C_2$ is a
minimum separating cycle for $Z'$.

We construct the cycle $C$ for the lemma as follows: Initially $C = C_2$. For every connected region $R$ in $\text{ins}(C_1) \setminus \text{ins}(C_2)$, we replace $C_2(R)$ by $C_1(R)$. Then $\text{ins}(C) = \text{ins}(C_1) \cup \text{ins}(C_2)$ and has the same length as that of $C_2$. \hfill \square

Given a set of separating cycles computed by Step (3) in our algorithm, our next job is to eliminate the intersected cycle pairs and find all maximal separating cycle. The next lemma shows the time complexity for this.

**Lemma 10.** Given the set of separating cycles computed in Steps (3) in $H_X$, all maximal separating cycles can be found in $O(|V(H_X)|k)$ time.

**Proof:** Similar to what we do in the algorithm for Theorem 2, we orient every cycle $C$ such that $\text{ins}(C)$ is on the right side when we proceed on $C$. We create a plane graph $\Gamma$ containing all the given cycles. It can be computed in $O(|Z_X|k)$ time. Then we perform an edge search in $H_X$ starting from the edges of $E(f_X)$. Every time we reach an oriented edge, we perform a leftmost search from it in $\Gamma$ to find a maximal separating cycle $C$. According to Lemma 9 and the fact that every cycle has a length at most $k$, the length of $C$ is at most $k$. After finding $C$ we delete all the edges in $\text{ins}(C)$ in $H_X$ and continue the edge search. Every edge will be visited for at most twice so the time complexity of finding maximal cycles is $O(|V(H_X)|k)$. The total time is $O(|V(H_X)|k)$ time. \hfill \square

**Proof for Theorem 3:**

By Lemma 8 minimum separating cycles for all crests can be computed in $O(nk + n_zk^3)$ time. By Lemma 10 $A_X$ can be computed in $O(|V(H_X)|k)$ time. Thus the theorem holds. \hfill \square

## 6 Concluding remarks

If we modify the definition for $d_2$ in Section 3 from $d_2 = d_1 + \lceil \frac{k+1}{2} \rceil$ to $d_2 = d_1 + (k + 1)$, we get an algorithm which given a planar graph $G$ and integer $k > 0$, in $O(n \log^4 n)$ time either computes a branch-decomposition of $G$ with width at most $(3 + \delta)k$, where $\delta > 0$ is a constant, or a $(k + 1) \times (k + 1)$ cylinder minor (or grid minor). It is interesting to develop an $O(n)$ time constant factor approximation algorithm for the branchwidth and largest grid (cylinder) minors.

## References

[1] S. Arnborg and D. Cornell and A. Proskurowski. Complexity of finding embedding in a k-tree. SIAM J. Disc. Math. 8:277–284, 1987.

[2] S. Arnborg, J. Lagergren, and D. Seese. Easy problems for tree-decomposable graphs. Journal of Algorithms, 12:308–340, 1991.
[3] Z. Bian, Q.P. Gu, M. Marjan, H. Tamaki and Y. Yoshitake. Empirical Study on Branchwidth and Branch Decomposition of Planar Graphs. In Proc. Algorithm Engineering and Experimentation (ALENEX2008), pages 152-165. 2008.

[4] Z. Bian, Q.P. Gu. Computing branch decomposition of large planar graphs. In Experimental Algorithms, pages 87-100. 2008.

[5] H.L. Bodlaender, P.G. Drange, M.S. Dreg, F.V. Fomin, D. Lokshtanov and M. Pilipczuk. An $O(c^k n)$ 5-approximation algorithm for treewidth. In Proc. of the 2013 Annual Symposium on Foundation of Computer Science, (FOCS2013), pages 499-508, 2013.

[6] H.L. Bodlaender. A tourist guide through treewidth. Acta Cybernetica, 11:1–21, 1993.

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

[8] H.L. Bodlaender, A. Grigoriev and A. M. C. A. Koster. Treewidth lower bounds with brambles. Algorithmica. 51(1):81–98, 2008.

[9] H.L. Bodlaender and D. Thilikos. Constructive linear time algorithm for treewidth. In Proc. of the 24th International Colloquium on Automata, Languages, and Programming, 627–637, 1997.

[10] G. Borradaile and P. Sankowski and C. Wulff-Nilsen. Min st-cut oracle for planar graphs with near-linear time preprocessing time. In Proc. of the 51st Annual Symposium on Foundations of Computer Science (FOCS2010) 601–610, 2010 (also arXiv:1003.1320v2, April 2010).

[11] G. Borradaile and P. Sankowski and C. Wulff-Nilsen. Min st-cut oracle for planar graphs with near-linear time preprocessing time. arXiv:1003.1320v4 Oct 2013 (to appear in ACM TALG).

[12] E.D. Demaine and M.T. Hajiaghayi. Graphs excluding a fixed minor have grids as large as treewidth, with combinatorial and algorithmic applications through bidimensionality, In Proc. of the 2005 Symposium on Discrete Algorithms, (SODA 2005), pages 682-689, 2005.

[13] F. Dorn and F.V. Fomin and M.T. Hajiaghayi and D.M. Thilikos. Subexponential parameterized algorithms on bounded-genus graphs and $H$-minor-free graphs, Jour. of ACM 52(6) pages 866-893, 2005.

[14] J. Fakcharoenphol and S. Rao. Planar graphs, negative weight edges, shortest paths, and near linear time. Journal of Computer and System Sciences. 72: 868-889, 2006.

[15] U. Feige, M.T. Hajiaghayi and J.R. Lee. Improved approximation algorithms for minimum weight vertex separators. SIAM J. Comput. 38(2): 629-657, 2008.

[16] F.V. Fomin and D.M. Thilikos. New upper bounds on the decomposability of planar graphs. Journal of Graph Theory, 51(1):53–81, 2006.

[17] G.N. Frederickson. Fast algorithms for shortest paths in planar graphs, with applications. SIAM J. Comput. 16(6): 1004-1022, 1987.

[18] A. Grigoriev. Tree-width and large grid minors in planar graphs. Discrete Mathematics & Theoretical Computer Science. 13(1):13–20, 2011.
[19] Q.P. Gu and H. Tamaki. Optimal branch decomposition of planar graphs in $O(n^3)$ time. ACM Trans. Algorithms 4(3): article No.30, 1–13, 2008.

[20] Q.P. Gu and H. Tamaki. Constant-factor approximations of branch-decomposition and largest grid minor of planar graphs in $O(n^{1+\epsilon})$ time. Theoretical Computer Science Vol. 412, pages 4100-4109, 2011.

[21] Q.P. Gu and H. Tamaki. Improved bound on the planar branchwidth with respect to the largest grid minor size. Algorithmica, Vol. 64 pages 416-453, 2012.

[22] I. V. Hicks. Planar branch decompositions I: The ratcatcher. INFORMS Journal on Computing, 17.4 (2005): 402-412.

[23] I. V. Hicks. Planar branch decompositions II: The cycle method. INFORMS Journal on Computing, 17.4 (2005): 413-421.

[24] P.N. Klein. Multiple-source shortest paths in planar graphs. In Proc. of the 2005 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA2005) pages 146-155, 2005.

[25] P.N. Klein, S. Mozes and C. Sommer. Structural recursive separator decompositions for planar graphs in linear time. In Proc. of the 2013 Annual ACM Symposium on the Theory of Computing (STOC2013) pages 505-514, 2013.

[26] F. Kammer and T. Tholey. Approximate tree decompositions of planar graphs in linear time. In Proc. of the 2012 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA2012), 683–698, pages 2012.

[27] F. Kammer and T. Tholey. Approximate tree decompositions of planar graphs in linear time. arXiv:1104.2275v2, May 2013

[28] S. Mozes and C. Sommer. Exact distance oracles for planar graphs In Proc. of the 2012 Annual ACM-SIAM Symposium on Discrete Algorithms, (SODA 2012), pages 209-222, 2012.

[29] J.H. Reif. minimum s-t cut of a planar undirected network in $O(n \log^2 n)$ time. planar graphs in linear time. SIAM J. on Computing, 12(1): pages 71-81, 1983

[30] N. Robertson and P.D. Seymour. Graph minors X. Obstructions to tree decomposition. J. of Combinatorial Theory, Series B, 52:153–190, 1991.

[31] N. Robertson and P.D. Seymour. Graph minors XIII. The disjoint paths problem. of Combinatorial Theory, Series B, 63:65–110, 1995.

[32] N. Robertson and P.D. Seymour and R. Thomas. Quickly excluding a planar graph J. of Combinatorial Theory, Series B, 62:323–348, 1994.

[33] P.D. Seymour and R. Thomas. Call routing and the ratcatcher. Combinatorica, 14(2):217–241, 1994.

[34] J. C. Smith, E. Ulusal, and I. V. Hicks. A combinatorial optimization algorithm for solving the branchwidth problem. Computational Optimization and Applications, 51.3 (2012): 1211-1229.
[35] H. Tamaki. A linear time heuristic for the branch-decomposition of planar graphs. In Proc. of ESA 2003 765–775, 2003.

[36] M. Thorup. Undirected single-source shortest paths with positive integer weights in linear time. Journal of the ACM, 46(3):362–394, 1999.