Obtaining a Planar Graph by Vertex Deletion

Dániel Marx and Ildikó Schlotter[⋆]

Department of Computer Science and Information Theory,
Budapest University of Technology and Economics,
Budapest, H-1521, Hungary.
{dmarx,ildi}@cs.bme.hu

Abstract. In the k-Apex problem the task is to find at most k vertices whose deletion makes the given graph planar. The graphs for which there exists a solution form a minor closed class of graphs, hence by the deep results of Robertson and Seymour [34, 35], there is a cubic algorithm for every fixed value of k. However, the proof is extremely complicated and the constants hidden by the big-O notation are huge. Here we give a much simpler algorithm for this problem with quadratic running time, by iteratively reducing the input graph and then applying techniques for graphs of bounded treewidth.

Keywords: Planar graph, Apex graph, FPT algorithm, Vertex deletion.

1 Introduction

Planar graphs are subject of wide research interest in graph theory. There are many generally hard problems which can be solved in polynomial time when considering planar graphs, e.g., Maximum Clique, Maximum Cut, and Subgraph Isomorphism [16, 22]. For problems that remain NP-hard on planar graphs, we often have efficient approximation algorithms. For example, the problems Independent Set, Vertex Cover, and Dominating Set admit an efficient linear-time approximation scheme [3, 27]. The research for efficient algorithms for problems on planar graphs is still very intensive.

Many results on planar graphs can be extended to almost planar graphs, which can be defined in various ways. For example, we can consider possible embeddings of a graph in a surface other than the plane. The genus of a graph is the minimum number of handles that must be added to the plane to embed the graph without any crossings. Although determining the genus of a graph is NP-hard [37], the graphs with bounded genus are subjects of wide research. A similar property of graphs is their crossing number, i.e., the minimum possible number of crossings with which the graph can be drawn in the plane. Determining the crossing number is also NP-hard [20].

In [7] Cai introduced another notation to capture the distance of a graph G from a graph class $\mathcal{F}$, based on the number of certain elementary modification steps. He defines the distance of G from $\mathcal{F}$ as the minimum number of modifying

[⋆] Supported by the Hungarian National Research Fund OTKA 67651.

steps needed to make G a member of $\mathcal{F}$. Here, modification can mean the deletion or addition of edges or vertices. In this paper we consider the following question: given a graph G and an integer k, is there a set of at most k vertices in G, whose deletion makes G planar?

It was proven by Lewis and Yannakakis in [26] that the node-deletion problem is NP-complete for every non-trivial hereditary graph property decidable in polynomial time. As planarity is such a property, the problem of finding a maximum induced planar subgraph is NP-complete, so we cannot hope to find a polynomial-time algorithm that answers the above question. Therefore, following Cai, we study the problem in the framework of parameterized complexity developed by Downey and Fellows [14]. This approach deals with problems in which besides the input I an integer k is also given. The integer k is referred to as the parameter. In many cases we can solve the problem in time $O(|I|^{f(k)})$ for some function f. Clearly, this is also true for the problem we consider. Although this is polynomial time for each fixed k, these algorithms are practically too slow for large inputs, even if k is relatively small. Therefore, the standard goal of parameterized analysis is to take the parameter out of the exponent in the running time. A problem is called *fixed-parameter tractable* (FPT) if it can be solved in time $f(k)p(|I|)$, where p is a polynomial not depending on k, and f is an arbitrary computable function. An algorithm with such a running time is also called FPT. For more on fixed-parameter tractability see e.g. [14], [30] or [19].

The standard parameterized version of our problem is the following: given a graph G and a parameter k, the task is to decide whether deleting at most k vertices from G can result in a planar graph. Such a set of vertices is sometimes called a set of *apex vertices* or *apices*, so we will denote the class of graphs for which the answer is 'yes' by Apex(k). We note that Cai [7] used the notation Planar $+ kv$ to denote this class.

In the parameterized complexity literature, numerous similar node-deletion problems have been studied. A classical result of this type by Bodlaender [4] and Downey and Fellows [13] states that the Feedback Vertex Set problem, asking whether a graph can be made acyclic by the deletion of at most k vertices, is FPT. The parameterized complexity of the directed version of this problem has been a long-standing open question, and it has only been proved recently that it is FPT as well [8]. Fixed-parameter tractability has also been proved for the problem of finding k vertices whose deletion results in a bipartite graph [32], or in a chordal graph [29]. On the negative side, the corresponding node-deletion problem for wheel-free graphs was proved to be W[2]-hard [28].

Considering the graph class Apex(k), we can observe that this family of graphs is closed under taking minors. The celebrated graph minor theorem by Robertson and Seymour states that such families can be characterized by a set of excluded minors [35]. They also showed that for each graph H it can be tested in cubic time whether a graph contains H as a minor [34]. As a consequence, membership for such graph classes can be decided in cubic time. In particular, we know that there exists an algorithm with running time $f(k)n^3$ for some function f that can decide whether a graph on n vertices belongs to Apex(k).

However, the proof of the graph minor theorem is non-constructive in the following sense. It proves the existence of an algorithm for the membership test that uses the excluded minor characterization of the given graph class, but does not provide any algorithm for determining this characterization. The existential nature of the graph minor theorem is inherent in the sense that there is no algorithm that, given a Turing machine which is a membership test for a minor closed family of graphs, computes an excluded minor characterization of this family [17], and a similar theorem applies for minor closed graph classes determined by some monadic-second order formula [11].

Despite the fact that the graph minor theorem is non-constructive, the excluded minor characterization is already known in some cases. Recently, Adler et al. [1] presented an algorithm that, given an excluded minor characterization of a minor closed graph class $\mathcal{F}$, computes the set of excluded minors for the graph class $\mathcal{F} + kv$, containing those graphs G for which there exists a set S of at most k vertices whose deletion yields a graph in $\mathcal{F}$. We remark that this result follows also from [17], as pointed out by Fellows. Given the excluded minor characterization of planar graphs by Kuratowski, this yields a way to explicitly construct a cubic recognition algorithm for the class Apex(k).

Although these results provide a general tool that can be applied to our specific problem, no direct FPT algorithm has been proposed for it so far. In this paper we present an algorithm which decides membership for Apex(k) in $f(k)n^2$ time for any input graph on n vertices, for some function f. Note that the presented algorithm runs in quadratic time, and hence yields a better running time than any algorithm using the minor testing algorithm that is applied in the above mentioned approaches. Moreover, if $G \in$ Apex(k) then our algorithm also returns a solution, i.e., a set $S \in V(G)$, $|S| \leq k$ such that $G - S$ is planar.

The presented algorithm is strongly based on the ideas used by Grohe in [21] for computing crossing number. Grohe uses the fact that the crossing number of a graph is an upper bound for its genus. Since the genus of a graph in Apex(k) cannot be bounded by a function of k, we need some other ideas. As in [21], we exploit the fact that in a graph with large treewidth we can always find a large grid minor [36]. Examining the structure of the graph with such a grid minor, we can reduce our problem to a smaller instance. Applying this reduction several times, we finally get an instance with bounded treewidth. Then we make use of Courcelle's Theorem [9], which states that every graph property that is expressible in monadic second-order logic can be decided in linear time on graphs of bounded treewidth.

It is worth mentioning that for every fixed k there is a linear-time algorithm by Kawarabayashi and Reed that decides whether a given graph has crossing number at most k [25]. In the same paper, the authors also present a linear-time FPT algorithm for the edge deletion version of the k-Apex problem, which given some graph G and some integer k, asks if G can be made planar by deleting at most k edges from it.

Remark 1. Very recently, a paper by Ken-ichi Kawarabayashi with title *Planarity allowing few error vertices in linear time* has been presented at FOCS 2009

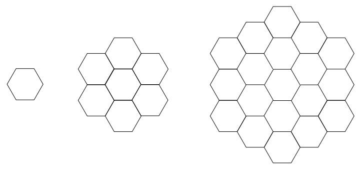

Fig. 1. The hexagonal grids H_1, H_2, and H_3.

proposing a linear-time algorithm for the k-Apex problem [24]. This result resolves an issue that has been posed as an open question also in [25], by solving the k-Apex problem in linear time.

The paper is organized as follows. Section 2 summarizes our notation, Section 3 outlines the algorithm, Sections 4 and 5 describe the two phases of the algorithm.

2 Notation

Graphs in this paper are assumed to be simple, since both loops and multiple edges are irrelevant in the k-Apex problem. The vertex set and edge set of a graph G are denoted by $V(G)$ and $E(G)$, respectively, and we use n for $|V(G)|$. The edges of a graph are unordered pairs of its vertices. If G' is a subgraph of G then $G-G'$ denotes the graph obtained by deleting G' from G. For a set of vertices S in G, we will also use $G-S$ to denote the graph obtained by deleting S from G.

A graph H is a *minor* of a graph G if it can be obtained from a subgraph of G by contracting some of its edges. Here *contracting an edge* e with endpoints a and b means deleting e, and then identifying vertices a and b.

A graph H is a *subdivision* of a graph G if H can be obtained from G by replacing some of its edges with newly introduced paths such that the inner vertices of these paths have degree two in H. We refer to these paths in H corresponding to edges of G as *edge-paths*. A graph H is a *topological minor* of G if G has a subgraph that is a subdivision of H. We say that G and G' are *topologically isomorphic* if they both are subdivisions of a graph H.

The $g \times g$ *grid* is the graph $G_{g\times g}$ where $V(G_{g\times g}) = \{v_{ij} \,|\, 1 \le i, j \le g\}$ and $E(G_{g\times g}) = \{v_{ij}v_{i'j'} \,|\, |i-i'| + |j-j'| = 1\}$. Instead of giving a formal definition for the *hexagonal grid* of radius r, which we will denote by H_r, we refer to the illustration shown in Figure 1. A *cell* of a hexagonal grid is one of its cycles of length 6.

A *tree decomposition* of a graph G is a pair $(T, (V_t)_{t\in V(T)})$ where T is a tree, $V_t \subseteq V(G)$ for all $t \in V(T)$, and the following are true:

- for all $v \in V(G)$ there exists a $t \in V(T)$ such that $v \in V_t$,
- for all $xy \in E(G)$ there exists a $t \in V(T)$ such that $x, y \in V_t$,

– if t lies on the path connecting t' and t'' in T, then $V_t \supseteq V_{t'} \cap V_{t''}$.

The *width* of such a tree decomposition is the maximum of $|V_t| - 1$ taken over all $t \in V(T)$. The *treewidth* of a graph G, denoted by $\mathrm{tw}(G)$, is the smallest possible width of a tree decomposition of G. For an introduction to treewidth see e.g. [6, 12].

3 Problem Definition and Overview of the Algorithm

We are looking for the solution of the following problem:

k-Apex problem:	
Input:	A graph $G = (V, E)$ and an integer k.
Task:	Find a set X of at most k vertices in V such that $G - X$ is planar.

Here we give an algorithm $\mathcal{A}$ which solves this problem in time $f(k)n^2$ for some function f, where n is the number of vertices in the input graph. Algorithm $\mathcal{A}$ works in two phases. In the first phase (Section 4) we compress the given graph repeatedly, and finally either conclude that there is no solution for our problem or construct an equivalent problem instance with a graph having bounded treewidth. In the latter case we solve the problem in the second phase of the algorithm (Section 5) by applying Courcelle's Theorem which gives a linear-time algorithm for the evaluation of MSO-formulas on bounded treewidth graphs.

To describe the first step of our algorithm, we need some deep results from graph minor theory. The following result states that every graph having large treewidth must contain a large grid as a minor.

Theorem 1 (Excluded Grid Theorem, [33]). *For every fixed integer r there exists an integer $w(r)$ such that if $\mathrm{tw}(G) > w(r)$ then G contains $G_{r\times r}$ as a minor.*

The grid minor guaranteed by this theorem in the case when the treewidth of the graph G is large can be found in cubic time. However, we need a linear-time algorithm for finding a large grid minor, so we have to make use of the following result, which states that if the graph is planar, then the bound on $w(r)$ is linear:

Theorem 2 (Excluded Grid Theorem for Planar Graphs, [36]). *For every integer r and every planar graph G, if $\mathrm{tw}(G) > 6r - 5$ then G contains $G_{r\times r}$ as a minor.*

Also, we will use the following algorithmic results:

Theorem 3 ([5, 31]). *For every fixed integer w there exists a linear-time algorithm that, given a graph G, does the following:*

- *either produces a tree decomposition of G of width at most w, or*
- *outputs a subgraph G' of G with $\mathrm{tw}(G') > w$, together with a tree decomposition of G' of width at most $2w$.*

Theorem 4 ([2]). *For every fixed graph H and integer w there exists a linear-time algorithm that, given a graph G and a tree decomposition for G of width w, returns a minor of G isomorphic to H, if this is possible.*

Now, we are ready to state our first lemma, which provides the key structures for the mechanism of our algorithm. In this lemma, we focus on hexagonal grids instead of rectangular grids. The reason for this is the well-known fact that if a graph of maximum degree three is a minor of another graph, then it is also contained in it as a topological minor [12]. This property of the hexagonal grid will be very useful later on.

Lemma 1. *For every pair of fixed integers r and k there is a linear-time algorithm $\mathcal{B}$, that, given an input graph G, does the following:*

- *either produces a tree decomposition of G of width $w(r,k) = 24r - 11 + k$, or*
- *finds a subdivision of H_r in G, or*
- *correctly concludes that $G \notin \mathrm{Apex}(k)$.*

Proof. Let r and k be arbitrary fixed integers. Run the algorithm provided by Theorem 3 for $w = w(r,k)$ on graph G. If it produces a tree decomposition of width $w(r,k)$ for G, then we output it. Otherwise let G' be the subgraph of G with $\mathrm{tw}(G') > w(r,k)$ that has been provided together with a tree decomposition $\mathcal{T}'$ for it having width at most $2w(r,k)$.

On the one hand, if $G' \notin \mathrm{Apex}(k)$, then $G \notin \mathrm{Apex}(k)$ also holds as G' is a subgraph of G. On the other hand, if $G' \in \mathrm{Apex}(k)$, then there exists a set $S \subseteq V(G)$ with $|S| \leq k$ such that $G' - S$ is planar. Deleting a vertex of a graph can only decrease its treewidth by at most one, so $\mathrm{tw}(G' - S) > w(r,k) - k = 6(4r - 1) - 5$. Now, Theorem 2 implies that $G' - S$ contains $G_{(4r-1)\times(4r-1)}$ as a minor. Since the hexagonal grid with radius r is a subgraph of the $(4r-1)\times(4r-1)$ grid, we get that $G' - S$ must also contain H_r as a minor, and hence as a topological minor.

Thus, we get that either $G \notin \mathrm{Apex}(k)$, or G' (and hence G) contains H_r as a (topological) minor. Now, using the algorithm of Theorem 4 for G' and $\mathcal{T}'$, we can find H_r as a minor in G in linear time, if possible. Such a minor can be easily used to obtain a subgraph of G' isomorphic to a subdivision of H_r in linear time. If the algorithm produces such a subgraph, then we output it, otherwise we can correctly conclude that $G \notin \mathrm{Apex}(k)$. □

In algorithm $\mathcal{A}$ we will run $\mathcal{B}$ several times. As long as the result is a hexagonal grid of radius r as topological minor, we will run Phase I of algorithm $\mathcal{A}$, which compresses the graph G. If at some step algorithm $\mathcal{B}$ gives us a tree decomposition of width $w(r,k)$, we run Phase II. (The constant r will be fixed later.) And of course if at some step $\mathcal{B}$ finds out that $G \notin \mathrm{Apex}(k)$, then algorithm $\mathcal{A}$ can stop with the output “No solution.”

Clearly, we can assume without loss of generality that the input graph is simple, and it has at least $k+3$ vertices. So if $G \in \text{Apex}(k)$, then deleting k vertices from G (which means the deletion of at most $k(|V(G)|-1)$ edges) results in a planar graph, which has at most $3|V(G)|-6$ edges. Therefore, if $|E(G)| > (k+3)|V(G)|$ then surely $G \notin \text{Apex}(k)$. Since this can be detected in linear time, we can assume that $|E(G)| \leq (k+3)|V(G)|$.

4 Phase I of Algorithm $\mathcal{A}$

In Phase I we assume that after running $\mathcal{B}$ on G we get a subgraph H'_r that is a subdivision of H_r. Our goal is to find a set of vertices X such that $G-X$ is planar, and $|X| \leq k$. Let $\text{ApexSets}(G,k)$ denote the family of sets of vertices that have these properties, i.e., let $\text{ApexSets}(G,k) = \{X \subseteq V(G) \mid |X| \leq k \text{ and } G-X \text{ is planar}\}$. Since the case $k=1$ is very simple we can assume that $k>1$.

Reduction A: Flat zones. In the following we regard the grid H'_r as a fixed subgraph of G. Let us define z zones in it. Here z is a constant depending only on k, which we will determine later. A *zone* is a subgraph of H'_r which is topologically isomorphic to the hexagonal grid H_{2k+5}. We place such zones next to each other in the well-known radial manner with radius q, i.e., we replace each hexagon of H_q with a subdivision of H_{2k+5}. It is easy to show that in a hexagonal grid with radius $(q-1)(4k+9)+(2k+5)$ we can define this way $3q(q-1)+1$ zones that only intersect in their outer circles. So let $r=(q-1)(4k+9)+(2k+5)$, where we choose q big enough to get at least z zones, i.e. q is the smallest integer such that $3q(q-1)+1 \geq z$. Let the set of these innerly disjoint zones be $\mathcal{Z}$, and the subgraph of these zones in H'_r be R.

Let us define two types of *grid-components*. An edge which is not contained in R is a grid-component if it connects two vertices of R. A subgraph of G is a grid-component if it is a (maximal) connected component of $G-R$. A grid-component K is *attached* to a vertex v of the grid R if it has a vertex adjacent to v, or (if K is an edge) one of its endpoints is v. The *core* of a zone is the (unique) subgraph of the zone which is topologically isomorphic to H_{2k+3} and lies in the middle of the zone. Let us call a zone $Z \in \mathcal{Z}$ *open* if there is a vertex in its core that is connected to a vertex v of another zone in $\mathcal{Z}$, $v \notin V(Z)$, through a grid-component. A zone is *closed* if it is not open.

For a subgraph H of R we let $T(H)$ denote the subgraph of G induced by the vertices of H and the vertices of the grid-components which are only attached to H. Let us call a zone Z *flat* if it is closed and $T(Z)$ is planar. Let Z be such a flat zone. See Figure 2 (a) for an illustration of a flat zone together with its grid-components. A grid-component is an *edge-component* if it is either only attached to one edge-path of Z or only to one vertex of Z. Otherwise, it is a *cell-component* if it is only attached to vertices of one cell. As a consequence of the fact that all embeddings of a 3-connected graph are equivalent (see e.g. [12]), and Z is a subdivision of such a graph, every grid-component attached to some vertex in the core of Z must be one of these two types. Note that we can assume that in an embedding of $T(Z)$ in the plane, all edge-components are embedded

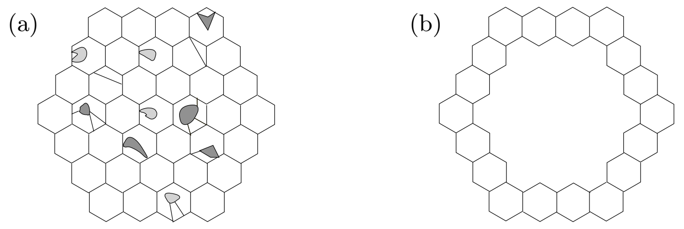
Fig. 2. (a) An induced subgraph of a flat zone, together with its grid components. Among them, there are two edges, four edge-components (shown in light gray) and five cell-components (dark gray). (b) The ring R_3 of Z.

in an arbitrarily small neighborhood of the edge-path (or vertex) which they belong to.

Let us define the *ring* R_i ($1 \leq i \leq 2k+4$) as the union of those cells in Z that have common vertices both with the i-th and the $(i+1)$-th concentric circle of Z. Let R_0 be the cell of Z that lies in its center. The zone Z can be viewed as the union of $2k+5$ concentric rings, i.e., the union of the subgraphs R_i for $0 \leq i \leq 2k+4$. Figure 2 (b) depicts the ring R_3.

Lemma 2. *Let Z be a flat zone in R, and let G' denote the graph $G - T(R_0)$. Then $X \in \text{ApexSets}(G', k)$ implies $X \in \text{ApexSets}(G, k)$.*

Proof. Suppose $X \in \text{ApexSets}(G', k)$. Since $G - T(R_0) - X$ is planar, we can fix a planar embedding ϕ of it. If $R_i \cap X = \emptyset$ for some i ($2 \leq i \leq 2k+2$) then let W_i denote the maximal subgraph of $G - T(R_0) - X$ for which $\phi(W_i)$ is in the region determined by $\phi(R_i)$ (including R_i). If $R_i \cap X$ is not empty then let W_i be the empty graph. Note that if $2 \leq i \leq 2k$ then W_i and W_{i+2} are disjoint. Therefore, there exists an index i for which $W_i \cap X = \emptyset$ and W_i is not empty. Let us fix this i.

Let Q_i denote $T(\bigcup_{j=0}^{i} R_j)$. We prove the lemma by giving an embedding for $G - X'$ where $X' = X \setminus V(Q_{i-1})$. The region $\phi(R_i)$ divides the plane in two other regions. As Z is flat, vertices of Q_{i-1} can only be adjacent to vertices of Q_i. Thus we can assume that in the finite region only vertices of Q_{i-1} are embedded, so $G - X' - (Q_{i-1} \cup W_i)$ is entirely embedded in the infinite region. Let U denote those vertices in Q_{i-1} which are adjacent to some vertex in $G - Q_{i-1}$. Observe that the vertices of U lie on the i-th concentric circle of Z, hence, the restriction of ϕ to $G - X' - (Q_{i-1} - U)$ has a face whose boundary contains U.

Now let θ be a planar embedding of $T(Z)$, and let us restrict θ to Q_{i-1}. Note that U only contains vertices which are either adjacent to some vertex in R_i or are adjacent to cell-components belonging to a cell of R_i. But θ embeds R_i and its cell-components also, and therefore the restriction of θ to Q_{i-1} results in a face whose boundary contains U. Here we used also that R_i is a subdivision of a 3-connected graph whose embeddings are equivalent.

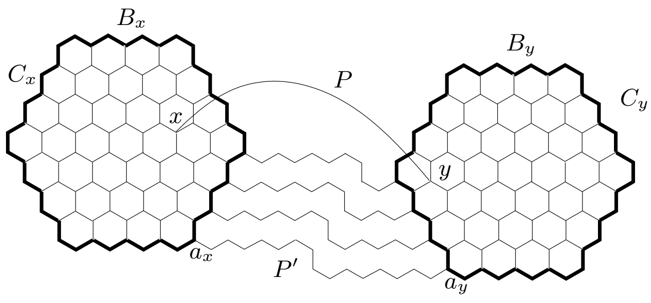

Fig. 3. Illustration for Lemma 3. The edges of C_x and C_y are shown in bold.

Now it is easy to see that we can combine θ and ϕ in such a way that we embed $G - X' - (Q_{i-1} - U)$ according to ϕ and, similarly, Q_{i-1} according to θ, and then "connect" them by identifying $\phi(u)$ and $\theta(u)$ for all $u \in U$. This gives the desired embedding of $G - X'$. Finally, we have to observe that $X' \in \text{ApexSets}(G, k)$ implies $X \in \text{ApexSets}(G, k)$, since $X' \subseteq X$ and $|X| \leq k$. □

This lemma has a trivial but crucial consequence: $X \in \text{ApexSets}(G, k)$ if and only if $X \in \text{ApexSets}(G - T(R_0), k)$, so deleting $T(R_0)$ reduces our problem to an equivalent instance. Let us denote this deletion as *Reduction A*.

Note that whether a zone Z is closed can be decided by a simple breadth first search, which can also produce the graph $T(Z)$. Planarity can also be tested in linear time [23]. Therefore we can test whether a zone is flat, and if so, we can apply Reduction A on it in linear time.

Later we will see that assuming that the graph G is contained in $\text{Apex}(k)$, a flat zone can always be found in G, unless G contains some easily recognizable vertices which must be included in every solution (Lemma 7). This yields an easy way to handle graphs with large treewidth: compressing our graph by repeatedly applying Reduction A we can reduce the problem to an instance with bounded treewidth.

Reduction B: Well-attached vertices. A subgraph of R is a *block* if it is topologically isomorphic to H_{k+3}. A vertex of a given block is called *inner vertex* if it is not on the outer circle of the block. (We define the outer circle of the block using the "standard" planar embedding of H_{k+3}. Instead of a formal definition, we refer to the illustration in Figure 3.)

Lemma 3. *Let $X \in \text{ApexSets}(G, k)$. Let x and y be inner vertices of the disjoint blocks B_x and B_y, respectively. If P is an $x - y$ path that (except for its endpoints) does not contain any vertex from B_x or B_y, then X must contain a vertex from B_x, B_y or P.*

Proof. See Figure 3 for the illustration of this proof. Let C_x and C_y denote the outer circle of B_x and B_y, respectively. Let us notice that since B_x and B_y are

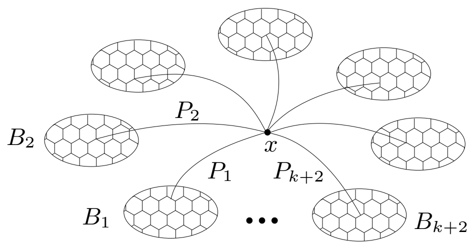

Fig. 4. A well-attached vertex.

disjoint blocks, there exist at least $k+3$ vertex disjoint paths between their outer circles, which—apart from their endpoints—do not contain vertices from B_x and B_y. Moreover, it is easy to see that these paths can be defined in a way such that their endpoints that lie on C_x are on the border of different cells of B_x. To see this, note that the number of cells which lie on the border of a given block is $6k+12$. At least three of these paths must be in $G-X$ also. Since x can lie only on the border of at most two cells having common vertices with C_x, we get that there is a path P' in $G-X$ whose endpoints are a_x and a_y (lying on C_x and C_y, resp.), and there exist no cell of B_x whose border contains both a_x and x.

Let us suppose that $B_x \cup B_y \cup P$ is a subgraph of $G-X$. Since all embeddings of a 3-connected planar graph are equivalent, we know that if we restrict an arbitrary planar embedding of $G-X$ to B_x, then all faces having x on their border correspond to a cell in B_x. Since x and y are connected through P and $V(P) \cap V(B_x) = \{x\}$, we get that y must be embedded in a region F corresponding to a cell C_F of B_x. But this implies that B_y must entirely be embedded also in F.

Since $V(P' - a_x - a_y) \cap V(B_x) = \emptyset$ and P' connects $a_x \in V(B_x)$ and $a_y \in V(B_y)$ we have that a_y must lie on the border of F. But then C_F is a cell of B_x containing both a_x and x on its border, which yields the contradiction. $\square$

Using this lemma we can identify certain vertices that have to be deleted. Let x be a *well-attached* vertex in G if there exist paths $P_1, P_2, \ldots, P_{k+2}$ and disjoint blocks $B_1, B_2, \ldots, B_{k+2}$ such that P_i connects x with an inner vertex of B_i ($1 \le i \le k+2$), the inner vertices of P_i are not in R, and if $i \ne j$ then the only common vertex of P_i and P_j is x.

Lemma 4. *Let $X \in \mathrm{ApexSets}(G,k)$. If x is well-attached, then $x \in X$.*

Proof. If $x \notin X$, then after deleting X from G (which means deleting at most k vertices) there would exist indices $i \ne j$ such that no vertex from P_i, P_j, B_i, and B_j was deleted. But then the disjoint blocks B_i and B_j were connected by the path $P_i - x - P_j$, and by the previous lemma, this is a contradiction. $\square$

We can decide whether a vertex v is well-attached in time $f'(k)|E(G)|$ for some function f', using standard flow techniques. This can be done by simply testing for each possible set of $k + 2$ disjoint blocks whether there exist the required disjoint paths that lead from x to these blocks. Since the number of blocks in R depends only on k, and we can find p disjoint paths starting from a given vertex of a graph G in time $O(p|E(G)|)$, we can observe that this can be done indeed in time $f'(k)|E(G)|$.

Finding flat zones. Now we show that if there are no well-attached vertices in the graph G but $G \in \mathrm{Apex}(k)$, then a flat zone exists in our grid.

Lemma 5. *Let $X \in \mathrm{ApexSets}(G, k)$, and let G not include any well-attached vertices. If K is a grid-component, then there cannot exist $(k+1)^2$ disjoint blocks such that K is attached to an inner vertex of each block.*

Proof. Let us assume for contradiction that there exist $(k + 1)^2$ such blocks. Since $|X| \leq k$, at least $(k + 1)^2 - k$ of these blocks do not contain any vertex of X. So let $x_1, x_2, \ldots\ x_{(k+1)^2-k}$ be adjacent to K and let $B_1, B_2, \ldots, B_{(k+1)^2-k}$ be disjoint blocks of $G - X$ such that x_i is an inner vertex of B_i.

Since $G - X$ is planar, it follows from Lemma 3 that a component of $K - X$ cannot be adjacent to different vertices from $\{x_i | 1 \leq i \leq (k+1)^2 - k\}$. So let K_i be the connected component of $K - X$ that is attached to x_i in $G - X$. K is connected in G, hence for every K_i there is a vertex of $T = K \cap X$ that is adjacent to it in G. Since there are no well-attached vertices in G, every vertex of T can be adjacent to at most $k + 1$ of these subgraphs. But then $|T| \geq ((k + 1)^2 - k)/(k + 1) > k$ which is a contradiction since $T \subseteq X$. □

Let us now fix the constant $d = (k + 1)((k + 1)^2 - 1)$.

Lemma 6. *Let $X \in \mathrm{ApexSets}(G, k)$, let G not include any well-attached vertices, and let x be a vertex of the grid R. Then there cannot exist $B_1, B_2, \ldots, B_{d+1}$ disjoint blocks such that for all i $(1 \leq i \leq d+1)$ an inner vertex of B_i and x are both attached to some grid-component K_i.*

Proof. As a consequence of Lemma 5, each of the grid-components K_i can be attached to at most $(k+1)^2-1$ disjoint blocks. But since x is not a well-attached vertex, there can be only at most $k+1$ different grid-components among the grid-components K_i, $1 \leq i \leq d + 1$. So the total number of disjoint blocks that are attached to x through a grid-component is at most $(k+1)((k+1)^2-1) = d$. □

Lemma 7. *Let $X \in \mathrm{ApexSets}(G, k)$, and let G not include any well-attached vertices. Then there exists a flat zone Z in G.*

Proof. Let $Z \in \mathcal{Z}$ be an open zone which has a vertex w in its core that is attached to a vertex v of another zone in $\mathcal{Z}$ $(v \notin V(Z))$ through a grid-component K. By the choice of the size of the zones and their cores, we have disjoint blocks B_w and B_v containing w and v respectively as inner points. We can also assume that B_w is a subgraph of Z which does not intersect the outer circle of Z.

By Lemma 3 we know that B_w, B_v or K contains a vertex from X. Let $\mathcal{Z}_1$ denote the set of zones in $\mathcal{Z}$ with an inner vertex in X, let $\mathcal{Z}_2$ denote the set of open zones in $\mathcal{Z}$ with a core vertex to which a grid-component, having a common vertex with X, is attached, and finally let $\mathcal{Z}_3$ be the set of the remaining open zones in $\mathcal{Z}$. Since $|X| \leq k$ and a grid-component can be attached to inner vertices of at most $(k+1)^2$ disjoint blocks by Lemma 5, we have that $|\mathcal{Z}_1| \leq k$ and $|\mathcal{Z}_2| \leq k(k+1)^2$.

Let us count the number of zones in $\mathcal{Z}_3$. To each zone Z in $\mathcal{Z}_3$ we assign a vertex $u(Z)$ of the grid not in Z, which is connected to the core of Z by a grid-component. First, let us bound the number of zones Z in $\mathcal{Z}_3$ for which $u(Z) \in X$. Lemma 6 implies that any $v \in X$ can be connected this way to at most d zones, so we can have only at most kd such zones.

Now let $U = \{v \mid v = u(Z), Z \in \mathcal{Z}_3\}$. Let a and b be different members of U, and let a be connected through the grid-component K_a with the core vertex z_a of $Z_a \in \mathcal{Z}_3$. Let B_a denote a block which only contains vertices that are inner vertices of Z_a, and contains z_a as inner vertex. Such a block can be given due to the size of a zone and its core. Let us define K_b, z_b, Z_b, and B_b similarly. Note that $V(B_a) \cap X = V(B_b) \cap X = \emptyset$ by $Z_a, Z_b \notin \mathcal{Z}_1$.

Now let us assume that a and b are in the same component of $R - X$. Let P be a path connecting them in $R - X$. If P has common vertices with B_a (or B_b) then we modify P the following way. If the first and last vertices reached by P in Z_a (or Z_b, resp.) are w and w', then we swap the $w - w'$ section of P using the outer circle of Z_a (or Z_b, resp.). This way we can fix a path in $R - X$ that connects a and b, and does not include any vertex from B_a and B_b. But this path together with K_a and K_b would yield a path in $G - X$ that connects two inner vertices of B_a and B_b, contradicting Lemma 3.

Therefore, each vertex of U lies in a different component of $R - X$. But we can only delete at most k vertices, and each vertex in a hexagonal grid has at most 3 neighbors, thus we can conclude that $|U| \leq 3k$. As for different zones Z_1 and Z_2 in $\mathcal{Z}$ we cannot have $u(Z_1) = u(Z_2)$ (which is also a consequence of Lemma 3) we have that $|\mathcal{Z}_3| \leq 3k$. So if we choose the number of zones in $\mathcal{Z}$ to be $z = 7k + k(k+1)^2 + kd + 1$ we have that there are at least $3k + 1$ zones in $\mathcal{Z}$ which are not contained in $\mathcal{Z}_1 \cup \mathcal{Z}_2 \cup \mathcal{Z}_3$, indicating that they are closed. Since a vertex can be contained by at most 3 zones, $|X| \leq k$ implies that there exist a closed zone $Z^* \in \mathcal{Z}$, which does not contain any vertex from X, and all grid-components attached to Z^* are also disjoint from X. This immediately implies that $T(Z^*)$ is a subgraph of $G - X$, and thus $T(Z^*)$ is planar. □

Algorithm for Phase I. The exact steps of Phase I of the algorithm $\mathcal{A}$ are shown in Figure 5. It starts with running algorithm $\mathcal{B}$ on the graph G and integers $w(r,k)$ and r. If $\mathcal{B}$ returns a hexagonal grid as a topological minor, then the algorithm proceeds with the next step. If $\mathcal{B}$ returns a tree decomposition $\mathcal{T}$ of width $w(r,k)$, then Phase I returns the triple $(G, W, \mathcal{T})$. Otherwise G does not have H_r as minor and its treewidth is larger than $w(r,k)$, so by Lemma 1 we can conclude that $G \notin \text{Apex}(k)$.

In the next step the algorithm tries to find a flat zone Z. If such a zone is found, then the algorithm executes a deletion, whose correctness is implied

```
Phase I of algorithm $\mathcal{A}$:
 Input: $G = (V, E)$.
 Let $W = \emptyset$.
 1. Run algorithm $\mathcal{B}$ on $G$, $w(r, k)$, and $r$.
    If it returns a subgraph $H'_r$ topologically isomorphic to $H_r$ then go to
    Step 2. If it returns a tree decomposition $\mathcal{T}$ of $G$, then output($G$, $W$, $\mathcal{T}$).
    Otherwise output("No solution.").
 2. For all zones $Z$ do:
    If $Z$ is flat then $G := G - T(R_0)$, and go to Step 1.
 3. Let $U = \emptyset$. For all $x \in V$: if $x$ is well-attached, then $U := U \cup \{x\}$.
    If $|U| = \emptyset$ or $|W| + |U| > k$ then output("No solution.").
    Otherwise $W := W \cup U$, $G := G - U$ and go to Step 1.
```

Fig. 5. Phase I of algorithm $\mathcal{A}$.

by Lemma 2. Note that after altering the graph, the algorithm must find the hexagonal grid again and thus has to run $\mathcal{B}$ several times.

If no flat zone was found in Step 2, the algorithm removes well-attached vertices from the graph in Step 3. The vertices already removed this way are stored in W, and U is the set of vertices to be removed in the actual step. By Lemma 4, if $X \in \text{ApexSets}(G, k)$ then $W \cup U \subseteq X$, so $|W| + |U| > k$ means that there is no solution. By Lemma 7, the case $U = \emptyset$ also implies $G \notin \text{Apex}(k)$. In these cases the algorithm stops with the output " No solution." Otherwise it proceeds with updating the variables W and G, and continues with Step 1.

The output of the algorithm can be of two types: it either refuses the instance (outputting "No solution.") or it returns an instance for Phase II. For the above mentioned purposes the new instance is equivalent with the original problem instance in the following sense:

Theorem 5. *Let $(G', W, \mathcal{T})$ be the triple returned by $\mathcal{A}$ at the end of Phase I. Then for all $X \subseteq V(G)$ it is true that $X \in \text{ApexSets}(G, k)$ if and only if $W \subseteq X$ and $(X \setminus W) \in \text{ApexSets}(G', k - |W|)$.*

Now let us examine the running time of this phase. The first step can be done in time $f''(k)n$ for some function f''. according to [36, 5, 31]. Since the algorithm only runs algorithm $\mathcal{B}$ again after reducing the number of the vertices in G, we have that $\mathcal{B}$ runs at most n times. This takes $f''(k)n^2$ time. The second step requires only linear time (a breadth first search and a planarity test). Deciding whether a vertex is well-attached can be done in time $f'(k)|E(G)|$, so we need $f'(k)n|E(G)|$ time to check every vertex at a given iteration in Step 3. Note that the third step is executed at most $k + 1$ times, since at each iteration $|W|$ increases. Hence, this phase of algorithm $\mathcal{A}$ uses total time $f''(k)n^2 + f'(k)kn|E(G)| = f(k)n^2$ for some function f, as the number of edges is $O(kn)$.

5 Phase II of Algorithm $\mathcal{A}$

At the end of Phase I of algorithm $\mathcal{A}$ we either conclude that $G \notin \mathrm{Apex}(k)$, or we have a triple $(G', W, \mathcal{T})$ for which Theorem 5 holds. Here $\mathcal{T}$ is a tree decomposition for G' of width at most $w(r,k)$. This bound only depends on r which is a function of k. From the choice of the constants r, q, z, and d we can derive by a straightforward calculation that $\mathrm{tw}(G') \leq w(r,k) \leq 100(k+2)^{7/2}$.

In order to solve our problem, we only have to find out if there is a set $Y \in \mathrm{ApexSets}(G', k')$ where $k' = k - |W|$. For such a set, $Y \cup W$ would yield a solution for the original k-Apex problem.

A theorem by Courcelle states that every graph property defined by a formula in monadic second-order logic (MSO) can be evaluated in linear time if the input graph has bounded treewidth. Here we consider graphs as relational structures of vocabulary $\{V, E, I\}$, where V and E denote unary relations interpreted as the vertex set and the edge set of the graph, and I is a binary relation interpreted as the incidence relation. For instance, a formula stating that x and y are neighboring vertices is the following: $\exists e : Ixe \wedge Iye$. We will denote by U^G the universe of the graph G, i.e., $U^G = V(G) \cup E(G)$. Variables in monadic second-order logic can be element or set variables, and the containment relation between an element variable x and a set variable X is simply expressed by the formula Xx. For the complete description of MSO logic refer to [15], and for a survey on MSO logic on graphs see [10].

Following Grohe [21], we use a strengthened version of Courcelle's Theorem:

Theorem 6. ([18]) *Let $\varphi(x_1, \ldots, x_i, X_1, \ldots, X_j, y_1, \ldots, y_p, Y_1, \ldots, Y_q)$ denote a given MSO-formula and let $w \geq 1$. Then there is a linear-time algorithm that, given a graph G with $\mathrm{tw}(G) \leq w$ and $b_1, \ldots, b_p \in U^G, B_1, \ldots, B_q \subseteq U^G$, decides whether there exist $a_1, \ldots, a_i \in U^G, A_1, \ldots, A_j \subseteq U^G$ such that*

$$G \vDash \varphi(a_1, \ldots, a_i, A_1, \ldots, A_j, b_1, \ldots, b_p, B_1, \ldots, B_q),$$

and, if this is the case, computes such elements $a_1, \ldots, a_i$ and sets $A_1, \ldots, A_j$.

It is well-known that there is an MSO-formula φ_{planar} that describes the planarity of graphs, i.e. for every graph G the statement $G \vDash \varphi_{\text{planar}}$ holds if and only if G is planar. The following simple claim shows that we can also create a formula describing the $\mathrm{Apex}(k)$ graph class.

Theorem 7. *For every integer k', there is an MSO-formula $\mathrm{apex}(x_1, \ldots, x_{k'})$ such that the statement $G \vDash \mathrm{apex}(v_1, \ldots, v_{k'})$ holds for a set $\{v_1, \ldots, v_{k'}\}$ of vertices in G if and only if $\{v_1, \ldots, v_{k'}\} \in \mathrm{ApexSets}(G, k')$.*

Proof. We will use the simple characterization of planar graphs by Kuratowski's Theorem: a graph is planar if and only if it does not contain any subgraph topologically isomorphic to K_5 or $K_{3,3}$. To formulate the existence of these subgraphs as an MSO-formula, we need some more simple formulas.

First, it is easy to see that the following formula expresses the property that (X, Y) is a partition of the set Z:

$$\mathrm{partition}(X, Y, Z) := \forall z : (Zz \to ((Xz \to \neg Yz) \wedge (\neg Xz \to Yz)))$$

Using this, we can express that the vertex set Z contains a path connecting a and b, by saying that every partition of Z that separates a and b has to separate two neighboring vertices:

$$\begin{aligned}&\text{connected}(a,b,Z) := Za \wedge Zb \wedge \forall X \forall Y : \\ &\quad ((\text{partition}(X,Y,Z) \wedge Xa \wedge Yb) \rightarrow (\exists c \exists d \exists e : Xc \wedge Yd \wedge Ice \wedge Ide))\end{aligned}$$

The following two formulas express that two sets are disjoint, or their intersection is some given unique vertex.

$$\begin{aligned}&\text{disjoint}(X,Y) := \forall z : (Xz \rightarrow \neg Yz) \\ &\text{almost-disjoint}(X,Y,a) := \forall z : (Xz \rightarrow (\neg Yz \vee (z = a)))\end{aligned}$$

Now, we can state formulas expressing that a given subgraph has K_5 or $K_{3,3}$ as a topological minor. For brevity, we only give the formula which states that there is a subdivision of K_5 in the graph such that the vertices $v_1, v_2, \ldots, v_5$ correspond to the vertices of the K_5, and the vertex sets $P_{12}, P_{13}, \ldots, P_{45}$ contain the subdivisions of the corresponding edges of K_5.

$$\begin{aligned}&K_5\text{-top-minor } (v_1, v_2, \ldots, v_5, P_{12}, P_{13}, \ldots, P_{45}) := \\ &\quad \text{connected}(v_1, v_2, P_{12}) \wedge \ldots \wedge \text{connected}(v_4, v_5, P_{45}) \wedge \\ &\quad \text{almost-disjoint}(P_{12}, P_{13}, v_1) \wedge \ldots \wedge \text{almost-disjoint}(P_{35}, P_{45}, v_5) \wedge \\ &\quad \text{disjoint}(P_{12}, P_{34}) \wedge \ldots \wedge \text{disjoint}(P_{23}, P_{45})\end{aligned}$$

The formula $K_{3,3}$-top-minor can be similarly created. Now, we are ready to give the apex formula, which uses the fact that $G - X$ is planar if and only if every subdivision of K_5 or $K_{3,3}$ in G must involve at least one vertex from X.

$$\begin{aligned}&\text{apex}(v_1, v_2, \ldots, v_{k'}) := \\ &\quad \forall x_1 \forall x_2 \ldots \forall x_5 \forall X_1 \forall X_2 \ldots \forall X_{10} : (K_5\text{-top-minor}(x_1, \ldots, x_5, X_1, \ldots, X_{10}) \\ &\qquad \rightarrow ((x_1 = v_1) \vee \ldots \vee (x_5 = v_{k'}) \vee X_1 v_1 \vee \ldots \vee X_{10} v_{k'})) \wedge \\ &\quad \forall x_1 \forall x_2 \ldots \forall x_6 \forall X_1 \forall X_2 \ldots \forall X_9 : (K_{3,3}\text{-top-minor}(x_1, \ldots, x_6, X_1, \ldots, X_9) \\ &\qquad \rightarrow ((x_1 = v_1) \vee \ldots \vee (x_6 = v_{k'}) \vee X_1 v_1 \vee \ldots \vee X_9 v_{k'}))\end{aligned}$$

□

Now let us apply Theorem 6. Let $\mathcal{C}$ be the algorithm which, given a graph G of bounded treewidth, decides whether there exist $v_1, \ldots, v_{k'} \in U^G$ such that $G \vDash \text{apex}(v_1, \ldots, v_{k'})$ is true, and if possible, also produces such variables. By Theorem 7, running $\mathcal{C}$ on G' either returns a set of vertices $U \in \text{ApexSets}(G', k')$, or reports that this is not possible. Hence, we can finish algorithm $\mathcal{A}$ in the following way: if $\mathcal{C}$ returns U then output($U \cup W$), otherwise output("No solution").

The running time of Phase II is $g(k)n$ for some function g.

Remark 2. Phase II of the algorithm can also be done by applying dynamic programming, using the tree decomposition $\mathcal{T}$ returned by $\mathcal{B}$. This also yields a linear-time algorithm, with a double exponential dependence on $\text{tw}(G')$ (and hence on k). Since the proof is quite technical and detailed, we omit it.

References

1. I. Adler, M. Grohe, and S. Kreutzer. Computing excluded minors. In *SODA 2008: Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 641–650, 2008.
2. S. Arnborg, A. Proskurowski, and D. Seese. Monadic second order logic, tree automata and forbidden minors. In *CSL 1990: Proceedings of the 4th Workshop on Computer Science Logic*, volume 533 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 1991.
3. B. S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *J. ACM*, 41(1):153–180, 1994.
4. H. L. Bodlaender. On disjoint cycles. *Int. J. Found. Comput. Sci.*, 5(1):59–68, 1994.
5. H. L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25(6):1305–1317, 1996.
6. H. L. Bodlaender. Treewidth: Algorithmic techniques and results. In *MFCS 1997: Proceedings of the 22nd International Symposium on Mathematical Foundations of Computer Science*, volume 1295 of *Lecture Notes in Computer Science*, pages 19–36. Springer, 1997.
7. L. Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Inf. Process. Lett.*, 58(4):171–176, 1996.
8. J. Chen, Y. Liu, S. Lu, B. O'Sullivan, and I. Razgon. A fixed-parameter algorithm for the directed feedback vertex set problem. *J. ACM*, 55(5):1–19, 2008.
9. B. Courcelle. Graph rewriting: An algebraic and logic approach. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*, pages 193–242. Elsevier and MIT Press, 1990.
10. B. Courcelle. The expression of graph properties and graph transformations in monadic second-order logic. In G. Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformations, Volume 1: Foundations*, pages 313–400. World Scientific, 1997.
11. B. Courcelle, R. G. Downey, and M. R. Fellows. A note on the computability of graph minor obstruction sets for monadic second order ideals. *Journal of Universal Computer Science*, 3:1194–1198, 1997.
12. R. Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, Heidelberg, 2005.
13. R. G. Downey and M. R. Fellows. Fixed-parameter tractability and completeness. *Congressus Numerantium*, 87:161–187, 1992.
14. R. G. Downey and M. R. Fellows. *Parameterized complexity*. Monographs in Computer Science. Springer-Verlag, New York, 1999.
15. H.-D. Ebbinghaus, J. Flum, and W. Thomas. *Mathematical Logic*. Springer-Verlag, second edition, 1994.
16. D. Eppstein. Subgraph isomorphism in planar graphs and related problems. *J. Graph Algorithms Appl.*, 3(3):1–27, 1999.
17. M. R. Fellows and M. A. Langston. On search, decision, and the efficiency of polynomial-time algorithms. *J. Comput. Syst. Sci.*, 49(3):769–779, 1994.
18. J. Flum, M. Frick, and M. Grohe. Query evaluation via tree-decompositions. *J. ACM*, 49(6):716–752, 2002.
19. J. Flum and M. Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer-Verlag, New York, 2006.

20. M. R. Garey and D. S. Johnson. Crossing number is NP-complete. *SIAM J. Algebraic Discrete Methods*, 4:312–316, 1983.
21. M. Grohe. Computing crossing numbers in quadratic time. *J. Comput. Syst. Sci.*, 68(2):285–302, 2004.
22. F. Hadlock. Finding a maximum cut of a planar graph in polynomial time. *SIAM J. Comput.*, 4(3):221–225, 1975.
23. J. E. Hopcroft and R. E. Tarjan. Efficient planarity testing. *J. ACM*, 21(4):549–568, 1974.
24. K. Kawarabayashi. Planarity allowing few error vertices in linear time. In *FOCS 2009: Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 639–648, 2009.
25. K. Kawarabayashi and B. A. Reed. Computing crossing number in linear time. In *STOC 2007: Proceedings of the 39th Annual ACM Symposium on Theory of Computing*, pages 382–390, 2007.
26. J. M. Lewis and M. Yannakakis. The node-deletion problem for hereditary properties is NP-complete. *J. Comput. Syst. Sci.*, 20(2):219–230, 1980.
27. R. J. Lipton and R. E. Tarjan. Applications of a planar separator theorem. *SIAM J. Comput.*, 9(3):615–627, 1980.
28. D. Lokshtanov. Wheel-free deletion is W[2]-hard. In *IWPEC 2008: Proceedings of the Third International Workshop on Parameterized and Exact Computation*, volume 5018 of *Lecture Notes in Computer Science*, pages 141–147, Berlin, 2008. Springer.
29. D. Marx. Chordal deletion is fixed-parameter tractable. *Algorithmica*, 57(4):747–768, 2010.
30. R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*, volume 31 of *Oxford Lecture Series in Mathematics and its Applications*. Oxford University Press, Oxford, 2006.
31. L. Perkovic and B. A. Reed. An improved algorithm for finding tree decompositions of small width. *Int. J. Found. Comput. Sci.*, 11(3):365–371, 2000.
32. B. A. Reed, K. Smith, and A. Vetta. Finding odd cycle transversals. *Oper. Res. Lett.*, 32(4):299–301, 2004.
33. N. Robertson and P. D. Seymour. Graph minors. V. Excluding a planar graph. *J. Comb. Theory, Ser. B*, 41(1):92–114, 1986.
34. N. Robertson and P. D. Seymour. Graph minors. XIII. The disjoint paths problem. *J. Combin. Theory Ser. B*, 63(1):65–110, 1995.
35. N. Robertson and P. D. Seymour. Graph minors. XX. Wagner's conjecture. *J. Combin. Theory Ser. B*, 92(2):325–357, 2004.
36. N. Robertson, P. D. Seymour, and R. Thomas. Quickly excluding a planar graph. *J. Comb. Theory, Ser. B*, 62(2):323–348, 1994.
37. C. Thomassen. The graph genus problem is NP-complete. *J. Algorithms*, 10(4):568–576, 1989.