Algorithmica, Vol. 33, pp. 461–493, 2002

Fixed Parameter Algorithms for Dominating Set and Related Problems on Planar Graphs*

Jochen Alber[†] Hans L. Bodlaender[‡] Henning Fernau[§] Ton Kloks[¶] Rolf Niedermeier[†]

Abstract

We present an algorithm that constructively produces a solution to the k-DOMINATING SET problem for planar graphs in time $O(c^{\sqrt{k}}n)$, where $c = 4^{6\sqrt{34}}$. To obtain this result, we show that the treewidth of a planar graph with domination number $\gamma(G)$ is $O(\sqrt{\gamma(G)})$, and that such a tree decomposition can be found in $O(\sqrt{\gamma(G)}n)$ time. The same technique can be used to show that the k-FACE COVER problem (find a size k set of faces that cover all vertices of a given plane graph) can be solved in $O(c_1^{\sqrt{k}}n)$ time, where $c_1 = 3^{36\sqrt{34}}$ and k is the size of the face cover set. Similar results can be obtained in the planar case for some variants of k-DOMINATING SET, e.g., k-INDEPENDENT DOMINATING SET and k-WEIGHTED DOMINATING SET.

Keywords. NP-complete problems, fixed parameter tractability, planar graphs, PLANAR DOMINATING SET, FACE COVER, outerplanarity, treewidth.

*An extended abstract of parts of this paper appeared under a slightly different title in the proceedings of the 7th Scandinavian Workshop on Algorithm Theory (SWAT 2000), Springer-Verlag, *LNCS* 1851, pages 97–110, held in Bergen, Norway, July 5–7, 2000.

[†]Universität Tübingen, Wilhelm-Schickard-Institut für Informatik, Sand 13, D-72076 Tübingen, Federal Republic of Germany;
email: {alber,niedermr}@informatik.uni-tuebingen.de

Work of Jochen Alber supported by the Deutsche Forschungsgemeinschaft (DFG), research project "PEAL"(Parameterized complexity and Exact ALgorithms), NI 369/1-1.

[‡]Utrecht University, Department of Computer Science, Padualaan 14, De Uithof, NL-3584 CH Utrecht, The Netherlands; email: hansb@cs.uu.nl.

The work of this author was partially supported by EC contract IST-1999-14186: Project ALCOM-FT (Algorithms and Complexity - Future Technologies).

[§]Work done while the author was with Universität Tübingen; Current affiliation: University of Newcastle, Department of Computer Science and Software Engineering, University Drive, NSW 2308 Callaghan, Australia; email: fernau@cs.newcastle.edu.au

[¶]email: ton@win.tue.nl

1 Introduction

A *k-dominating set* D of an undirected graph G is a set of k vertices of G such that each of the rest of the vertices has at least one neighbor in D. The minimum k such that the graph G has a k-dominating set is called the *domination number* of G, denoted by $\gamma(G)$. The k-DOMINATING SET problem, i.e., the task to decide, given a graph $G = (V, E)$ and a positive integer k, whether or not there exists a k-dominating set, is among the core problems in algorithms, combinatorial optimization, and computational complexity [8, 18, 28, 32, 42]. The problem is NP-complete, even when restricted to planar graphs with maximum vertex degree 3 and to planar graphs that are regular of degree 4 [28]. A comprehensive overview on domination problems is provided by the two volumes [30, 31].

Coping with NP-hard problems. Despite their intractability, many NP-hard problems are of great practical importance. Besides heuristic methods which often lack theoretical analysis, the main contribution of theoretical computer science on the attack of intractability so far has been to design and analyze approximation algorithms. The approximability of the DOMINATING SET problem has received considerable attention [8, 18, 32]. It is not known and it is not believed that DOMINATING SET for general graphs has a constant factor approximation algorithm (see, e.g., [8, 18] for details). More precisely, if $P \neq NP$, then DOMINATING SET has no polynomial time approximation scheme, see [6, p. 503], since it is hard for the approximation class *MAX SNP* defined in [41]. Feige has even shown [25, p. 637] that DOMINATING SET cannot be approximated within a ratio of $\ln n$ without causing unbelieved collapses of inclusions of complexity classes. For further complexity results for DOMINATING SET, we refer to [30, 36, 43]. The DOMINATING SET problem *restricted to planar graphs*, however, possesses a polynomial time approximation scheme [9]. That is, there is a polynomial time approximation algorithm with approximation factor $1 + \epsilon$, where ϵ is a constant arbitrarily close to 0. The degree of the polynomial grows with $1/\epsilon$. Another approach, which we will take in this paper, is to work towards an "efficient" exact algorithm working in "reasonable exponential time" for DOMINATING SET on planar graphs.

Fixed parameter tractability. Lately, it has become popular to attack computational intractability in a different way: studying parameterized complexity [4, 23, 24, 26]. Here, the basic observation is that, for many hard problems, the seemingly inherent combinatorial explosion can be restricted to a "small part" of the input, the *parameter*. For instance, the k-VERTEX COVER problem can be solved by an algorithm with running time $O(kn + 1.3^k)$ [17, 38], where the parameter k is a bound on the maximum size of the vertex cover set we are looking for. The fundamental assumption is $k \ll n$. As can easily be seen, this yields an efficient, practical algorithm for small values of k. In general, a problem is called *fixed parameter tractable* if it can be solved in time $f(k)n^{O(1)}$ for an arbitrary function f which depends only on k. Unfortunately, according to the theory of parameterized complexity, it is very unlikely that the k-DOMINATING SET problem is fixed parameter tractable. On the contrary, it was proven to be complete for $W[2]$ (see [21]), a "complexity class of parameterized intractability" (refer to Downey and Fellows [23] for details). However, k-DOMINATING SET on planar graphs is fixed parameter tractable.

This already easily follows from the work of Baker [9] on the approximation of DOMINATING SET (and other problems) on planar graphs. Note for this that a planar graph with a dominating set of size k is at most $3k$-outerplanar (see Section 2). Baker has given an algorithm for DOMINATING SET on planar graphs with bounded outerplanarity; together this implies an $O(8^{3k}n)$ time algorithm. Besides this, an $O(4^{9k}n)$ time algorithm can be derived from the observation mentioned above, the fact that an r-outerplanar graph has treewidth of at most $3r-1$ (as exhibited in Section 2), and the observation that we can use a dynamic programming approach to construct a dominating set for a graph that is given together with its tree decomposition (as outlined in Section 2.1). Alternatively, the general logical framework of Frick and Grohe [27] easily proves the fixed parameter tractability of k-DOMINATING SET on planar graphs. Downey and Fellows [22, 23] claimed an $O(11^k n)$ time bound. However, this result was flawed. Very recently, it could be corrected and improved to running time $O(8^k n)$ [1].

Our main result. We present fixed parameter tractability results for k-DOMINATING SET on planar graphs and related problems. Our main result is to prove a new structural relationship: We show that a planar graph with a dominating set of size k has treewidth $O(\sqrt{k})$. Up to now, only treewidth $O(k)$ was known, which is basically straightforward. As can be seen from the example of grid graphs, this derived bound is optimal for planar graphs. Note that our proof can be made constructive. The running time to construct the tree decomposition will be $O(\sqrt{k}n)$. Besides, we give an elaborated dynamic programming approach to solve the DOMINATING SET problem constructively in time $O(q^\ell N)$, where $q=4$, for a graph that is given together with a tree decomposition having width ℓ and N nodes. This improves previous results by Telle and Proskurowski [46, 47] who obtained $q=9$ for their approach. We then conclude that the k-DOMINATING SET problem on planar graphs can be solved in time $O(c^{\sqrt{k}}n)$, where $c=4^{6\sqrt{34}}$. Clearly, the proven constant c is huge and exhibits the limited usefulness of our result in practice. It is conceivable that this constant might be improved significantly due to refined analysis and/or additional algorithmic ideas. Also note that our analysis and, thus, the constant refers to a pure worst case scenario; it might say little about the average case performance or the behavior of our method on problem instances drawn from practical applications. Finally, and maybe most importantly, our result means a structural breakthrough. The best known running time for the k-DOMINATING SET problem on planar graphs was $O(c_1^k n)$ [1], which we improved to $O(c^{\sqrt{k}} n)$. Indeed, this seems to be the first non-trivial result for an NP-hard, fixed parameter tractable problem where the exponent of the exponential term is growing sublinearly. Very recently, on the one hand, the basic concepts of this paper could be generalized to a wider spectrum of planar graph problems [2] (also see [3] for an approach based on planar separator theorems instead of tree decompositions). On the other hand, Cai and Juedes [16] showed that our result is in a way optimal: They state that there is no time $2^{o(\sqrt{k})} n^{O(1)}$ algorithm unless $\mathit{3SAT} \in \mathit{DTIME}(2^{o(n)})$, which is considered to be unlikely.

Further contributions of our work. Our new method can also be used to improve a known bound for the k-FACE COVER problem [10, 23, 44]. The problem is defined as follows. Given a plane graph G, i.e., a graph with a fixed embedding in the plane and a positive integer k, is there a set of at most k faces (also called disks [10, 44]), such that all of the graph vertices are covered by the faces, i.e., each vertex is part of at least one of these k faces? The problem is NP-complete [10]. Downey and Fellows [23] claimed an $O(12^k n)$ algorithm for this problem. Unfortunately, the proof of [23, Lemma 3.3] (which is the basis of the given search tree algorithm) is flawed. For a slightly more general version of the problem, Bienstock and Monma [10] showed that there is a time $O(c_2^k n)$ algorithm, where c_2 is an unspecified constant. In this paper, we give an algorithm that solves k-FACE COVER in time $O(c_3^{\sqrt{k}} n)$, where $c_3 = 3^{36\sqrt{34}}$. Finally, it is easy to extend our results for DOMINATING SET to "DOMINATING SET WITH PROPERTY P" problems on planar graphs. For example, this includes the problems k-INDEPENDENT DOMINATING SET or k-TOTAL DOMINATING SET. In addition, our results can also be generalized to the weighted case with positive integer weights, i.e, k-WEIGHTED DOMINATING SET.

Outline of the paper. Our paper is structured as follows. In Section 2, we introduce some key concepts for our work, including r-outerplanarity, layer decompositions of r-outerplanar graphs, and treewidth. Moreover, we point out the relations between domination number and r-outerplanarity and between r-outerplanarity and treewidth. We show how to compute a minimum size dominating set given a tree decomposition of a graph in a way that improves previous results obtained in [46, 47]. In Section 3, our main result follows, showing that for planar graphs with domination number k, a tree decomposition of width $O(\sqrt{k})$ exists. In addition, we also show how to actually construct such a tree decomposition. In Section 4, we summarize our findings in describing an algorithm solving the DOMINATING SET problem in time $4^{O(\sqrt{k})} n$. In Section 5, we extend our results to variations of k-DOMINATING SET, k-WEIGHTED DOMINATING SET on planar graphs, and, in particular, to the k-face cover problem. We conclude the paper with some open problems and final remarks (Section 6).

2 Preliminaries: Domination, r-outerplanarity, and treewidth

In this section, we provide necessary notions and some known results. We assume familiarity with basic graph-theoretical notation as provided in [20, 37]. In particular, for a graph $G = (V, E)$ and a subset $V' \subseteq V$, the subgraph induced by the vertices $V \setminus V'$ will frequently be denoted by $G - V'$. If $V' = \{v\}$ is a singleton, we write $G - v$ instead of $G - \{v\}$. Throughout this paper, n always denotes the number of vertices in the given graph.

This section is split into three parts. The first subsection introduces the notion of r-outerplanarity and the concept of a layer decomposition of a planar graph. Moreover, we state a central observation relating the domination number and the outerplanarity number of a planar graph. In the second subsection, we provide the concepts of tree decompositions and treewidth. Also, we explicitly outline how the Dominating Set problem can be solved in this framework. This idea will be used as a main tool for our algorithm. In the third subsection, we outline how to construct a tree decomposition of small width for r-outerplanar graphs, which is used as an important lemma in later parts.

2.1 Domination and r-outerplanarity

Definition 1 A crossing-free embedding of a graph G in the plane is called *outerplanar* if each vertex lies on the boundary of the outer face. A graph G is called *outerplanar* if it admits an outerplanar embedding in the plane.

The following generalization of the notion of outerplanarity was introduced by Baker [9].

Definition 2 A crossing-free embedding of a graph G in the plane is called *r-outerplanar* if, for $r = 1$, the embedding is outerplanar, and, for $r > 1$, inductively, when removing all vertices on the boundary of the outer face and their incident edges, the embedding of the remaining subgraph is $(r-1)$-outerplanar. A graph G is called *r-outerplanar* if it admits an r-outerplanar embedding. The smallest number r, such that G is r-outerplanar is called the *outerplanarity* number.

In this way, we may speak of the layers $L_1, \ldots, L_r$ of an embedding of an r-outerplanar graph.

Definition 3 For a given r-outerplanar embedding of a graph $G = (V, E)$, we define the *ith layer* L_i inductively as follows. Layer L_1 consists of the vertices on the boundary of the outer face, and, for $i > 1$, the layer L_i is the set of vertices that lie on the boundary of the outer face in the embedding of the subgraph $G - (L_1 \cup \ldots \cup L_{i-1})$.

One easily observes the following central relation between the domination number and the outerplanarity number of a planar graph.

Proposition 4 *If a planar graph $G = (V, E)$ has a k-dominating set, then all plane embeddings of G can be at most $3k$-outerplanar.*

Proof. Note that, for a given crossing-free embedding of G in the plane, each vertex in the dominating set can dominate vertices from the previous, the next, or its own layer only. Hence, each vertex in the dominating set can contribute to at most 3 new layers. □

To understand the techniques used in the following sections, it is helpful to consider the concept of a *layer decomposition* of an r-outerplanar embedding of graph G. A layer decomposition of an r-outerplanar embedding of graph G is a forest of height $r - 1$. The nodes of the forest correspond to different connected components of the subgraphs of G induced by a layer. For each layer with vertex set L_i, suppose the connected components of the subgraph of G induced by L_i have vertex sets $C_{i,1}, \ldots, C_{i,\ell_i}$, i.e., $L_i = \bigcup_{j=1}^{\ell_i} C_{i,j}$. Then, we have ℓ_i nodes that represent the nodes of layer L_i, one for each such connected component. Such a set $C_{i,j}$ will be called a *layer component*. Each layer component node $C_{1,j}$ will be the root of a tree in the forest, so we will have one tree per connected component of G, representing the exterior vertices of the component. Two layer component nodes will be adjacent if they contain vertices that share a common face. This means that a layer component node $C_{i,j}$ can only be adjacent to layer component nodes of the form $C_{i-1,j'}$ or $C_{i+1,j''}$; if $C_{i,j}$ is adjacent to $C_{i-1,j'}$, then the vertices of $C_{i,j}$ lie within the area formed by the subgraph induced by $C_{i-1,j'}$. Note that the layer component nodes on the i'th level of the forest correspond to the layer components of the form $C_{i,j}$. One easily observes that the planarity of G implies that the layer decomposition must indeed be a forest.

We need some further notation for the construction in the next sections.

Definition 5 A layer component $C_{i,j}$ of layer L_i is called *non-vacuous* if there are vertices from layer L_{i+1} in the interior of $C_{i,j}$ (i.e., in the region enclosed by the subgraph induced by $C_{i,j}$). In other words, L_i is non-vacuous iff the corresponding component node in the layer decomposition has a child.

Lemma 6 *Let $\emptyset \neq C \subseteq C_{i,j}$ be a subset of a non-vacuous layer component $C_{i,j}$ of layer i, where $i \geq 2$. Then, there exists a unique smallest (in number of vertices)* *cycle $B(C)$ in layer L_{i-1}, such that C is contained in the region enclosed by $B(C)$. No other vertex of layer L_{i-1} is contained in this region.*

Proof. The existence of a cycle $B'(C)$ in layer L_{i-1} which encloses C follows easily from the definition of a layer. Note, that the region enclosed by $B'(C)$, again by the definition of a layer, cannot contain vertices of layer L_{i-1}. In order to obtain the smallest such cycle, we proceed as follows. Let $B'(C) = (w_1, \ldots, w_m)$. If there exists a pair of vertices w_i, w_j with $j > i$ and $|i - j| \notin \{1, m-1\}$ that happen to be neighbors in the graph, and if C is enclosed by one of the cycles $(w_i, \ldots, w_j)$ or $(w_j, \ldots, w_m, w_1, \ldots, w_i)$, then we exchange $B'(C)$ by this one. Continuing this procedure until no further such situation occurs it is not hard to verify that we end up with a smallest cycle that encloses C. $\square$

Definition 7 For each non-empty subset C of a non-vacuous layer component of layer i ($i \geq 2$), the set $B(C)$ as given in Lemma 6 is called the *boundary cycle* of C.

2.2 Domination and treewidth

The main tool we use in our algorithm is the concept of tree decompositions as, e.g., described in [12, 34].

Definition 8 Let $G = (V, E)$ be a graph. A *tree decomposition* of G is a pair $\langle \{X_i \mid i \in I\}, T \rangle$, where each X_i is a subset of V, called a *bag*, and T is a tree with the elements of I as nodes. The following three properties must hold:

1. $\bigcup_{i \in I} X_i = V$;
2. for every edge $\{u, v\} \in E$, there is an $i \in I$ such that $\{u, v\} \subseteq X_i$;
3. for all $i, j, k \in I$, if j lies on the path between i and k in T, then $X_i \cap X_k \subseteq X_j$.

The *width* of $\langle \{X_i \mid i \in I\}, T \rangle$ equals $\max\{|X_i| \mid i \in I\} - 1$. The *treewidth* of G is the minimum k such that G has a tree decomposition of width k.

A tree decomposition with a particularly simple structure is given by the following.

Definition 9 A tree decomposition $\langle \{X_i \mid i \in I\}, T \rangle$ is called a *nice tree decomposition* if the following conditions are satisfied:

1. Every node of the tree T has at most 2 children.
2. If a node i has two children j and k, then $X_i = X_j = X_k$ (in this case i is called a JOIN NODE).

3. If a node i has one child j, then either
   (a) $|X_i| = |X_j| + 1$ and $X_j \subset X_i$ (in this case i is called an INSERT NODE), or
   (b) $|X_i| = |X_j| - 1$ and $X_i \subset X_j$ (in this case i is called a FORGET NODE).

It is not hard to transform a given tree decomposition into a nice tree decomposition. More precisely, the following result holds (see [34, Lemma 13.1.3])

Lemma 10 *Given a tree decomposition of a graph G that has width k and $O(n)$ nodes, where n is the number of vertices of G. Then, we can find a nice tree decomposition of G that has also width k and $O(n)$ nodes in time $O(n)$.* □

In Section 2.3, we give a constructive proof of a result stated in [35, Table 2, page 550] and in [13, Theorem 83]:

Theorem 11 *An r-outerplanar graph has treewidth of at most $3r - 1$.*

Proposition 4 and Theorem 11 immediately imply the following relation between the domination number and the treewidth of a planar graph.

Corollary 12 *If a planar graph $G = (V, E)$ has a k-dominating set, then it has bounded treewidth, or, more precisely, its treewidth is bounded by $9k - 1$.* □

We will give a considerably stronger bound later.

The concept of tree decompositions can be used to solve various graph problems. Typically, treewidth based algorithms proceed in two stages: The first stage finds a tree decomposition of bounded width of the input graph, and the second stage solves the problem using dynamic programming approaches on the tree decomposition (see [12]).

As to the first stage, the problem to determine whether a graph has treewidth bounded by some constant ℓ and, if so, producing a corresponding tree decomposition, has, at the current state of research, no algorithmically feasible solution. Even though the problem is proven to be fixed parameter tractable, the constants in the algorithm presented in [11] are too large for practical purposes. However, in the case of k-DOMINATING SET on planar graphs, this first stage of the tree decomposition algorithm is less involved, as will be shown in Theorem 14.

The second stage can be executed in polynomial time with an exponential factor depending on the treewidth bound. The corresponding result for the DOMINATING SET problem is the following. Note that, in this situation, we do not need to assume planarity of the underlying graph.

The theorem can be proven by using dynamic programming techniques, as described in a more general context, e.g., in [12, 46, 47]. We want to point out that the running time of the algorithm described in the proof of Theorem 13 has an exponential base of $q = 4$, thus, improving the so far best known base of $q = 9$ that was achieved in [46, Theorem 4, Table 1],[47, Theorem 5.7]. This improvement is due to the observation that the mappings which have to be kept in the dynamic programming obey a certain "monotonicity." The same trick will allow further improvements of the results of Telle and Proskurowski [46, 47] for various other problems (see Section 5 and [5] for details).

Theorem 13 *If a tree decomposition of width ℓ of a graph is known, then a minimum dominating set can be determined in time $O(4^\ell N)$, where N is the number of nodes of the tree decomposition.*

Proof. Let $\mathcal{X} = \langle\{X_i \mid i \in I\}, T\rangle$ be a tree decomposition for the graph $G = (V, E)$. By Lemma 10, we can assume that $\mathcal{X}$ is a nice tree decomposition.

Suppose $V = \{x_1, \ldots, x_n\}$. We assume that the vertices in the bags are given in increasing order of the vertices when used as indices of the dynamic programming mappings, i.e., $X_i = (x_{i_1}, \ldots, x_{i_{n_i}})$ with $i_1 \leq \ldots \leq i_{n_i}$. In the following, we use three different "colors" that will be assigned to the vertices in the bag:

- "black" (represented by 1, meaning that the vertex belongs to the dominating set),
- "white" (represented by 0, meaning that the vertex is already dominated at the current stage of the algorithm), and
- "grey" (represented by $\hat{0}$, meaning that, at the current stage of the algorithm, we still ask for a domination of this vertex).

More precisely, a vector $c = (c_1, \ldots, c_{n_i}) \in \{0, \hat{0}, 1\}^{n_i}$ will be called a *coloring* for the bag $X_i = (x_{i_1}, \ldots, x_{i_{n_i}})$, and the *color* assigned to vertex x_{i_t} by the coloring c is given by the coordinate c_t.

For each bag X_i (with $|X_i| = n_i$), we will use a mapping

$$A_i : \{0, \hat{0}, 1\}^{n_i} \longrightarrow \mathbb{N} \cup \{+\infty\}.$$

For a coloring $c = (c_1, \ldots, c_{n_i}) \in \{0, \hat{0}, 1\}^{n_i}$, the value $A_i(c)$ stores how many vertices are needed for a minimum dominating set (of the graph visited up to the current stage of the algorithm) under the restriction that the color assigned to vertex x_{i_t} is c_t ($t = 1, \ldots, n_i$).

On the color set $\{0, \hat{0}, 1\}$, let $\prec$ be the partial ordering given by $\hat{0} \prec 0$ and $d \prec d$ for all $d \in \{0, \hat{0}, 1\}$. This ordering naturally extends to colorings: For $c = (c_1, \ldots, c_m), c' = (c'_1, \ldots, c'_m) \in \{0, \hat{0}, 1\}^m$, we let $c \prec c'$ iff $c_t \prec c'_t$ for all $t = 1, \ldots, m$.

It is essential for the correctness of our algorithm as well as for the claimed running time that the mappings A_i are monotonous from $(\{0, \hat{0}, 1\}, \prec)$ to

$(\mathbb{N} \cup \{+\infty\}, \leq)$, i.e., that for $c, c' \in \{0, \hat{0}, 1\}^{n_i}$, $c \prec c'$ implies $A(c) \leq A(c')$. We indicate the approximate reasoning for showing this property by structural induction below.

A coloring $c \in \{0, \hat{0}, 1\}^{n_i}$ is *locally invalid* for a bag X_i if

$$\big(\exists s \in \{1, \dots, n_i\} : c_s = 0\big) \wedge \big(\nexists t \in \{1, \dots, n_i\} : (x_{i_t} \in N(x_{i_s}) \wedge c_t = 1)\big).$$

In other words, a coloring is locally invalid if there is some vertex in the bag that is colored white, but this color is not "justified" within the bag, i.e., not dominated by a vertex within the bag using this coloring.[1] Also, for a coloring $c = (c_1, \dots, c_m) \in \{0, \hat{0}, 1\}^m$ and a color $d \in \{0, \hat{0}, 1\}$, we use the notation $\#_d(c) := |\{t \in \{1, \dots, m\} : c_t = d\}|$.

Step 1: In the first step of the algorithm, for each leaf node i of the tree decomposition, we initialize the mapping A_i:

`for all` $c \in \{0, \hat{0}, 1\}^{n_i}$ `do`

$$A_i(c) \leftarrow \begin{cases} +\infty & \text{if } c \text{ is locally invalid for } X_i \\ \#_1(c) & \text{otherwise} \end{cases} \tag{1}$$

By this initialization step, we make sure that only colorings are taken into consideration where an assignment of color 0 is justified.

Since the check for local invalidity takes time $O(n_i)$, this step can be carried out in time $O(3^{n_i} \cdot n_i) \subseteq O(4^\ell)$.

Trivially, the mappings of the leaves are monotonous, yielding the induction base.

Step 2: After this initialization, we visit the bags of our tree decomposition from the leaves to the root, evaluating the corresponding mappings in each step according to the following rules.

FORGET NODES: Suppose i is a FORGET NODE with child j and suppose that $X_i = (x_{i_1}, \dots, x_{i_{n_i}})$. W.l.o.g.[2], we may assume that $X_j = (x_{i_1}, \dots, x_{i_{n_i}}, x)$. Evaluate the mapping A_i of X_i as follows:

`for all` $c \in \{0, \hat{0}, 1\}^{n_i}$ `do`

$$A_i(c) \leftarrow \min_{d \in \{0,1\}} A_j(c \times \{d\}) \tag{2}$$

Note that a coloring $c \times \{\hat{0}\}$ for X_j means that the vertex x is assigned color $\hat{0}$, i.e., not yet dominated by a vertex. Since, by condition (3) of

[1] A locally invalid coloring still may be a correct coloring if the white vertex whose color is not "justified" *within* the bag already is dominated by a vertex from bags that have been considered earlier.

[2] Possibly after rearranging the vertices in X_j and the entries of A_j accordingly.

Definition 8, the vertex x will never appear in a bag for the rest of the algorithm, a coloring $c \times \{\hat{0}\}$ will remain unresolved and it will not lead to a dominating set. That is why the minimum in the assignment (2) is taken over colors 1 and 0 only.

Clearly, the evaluations can be carried out in time $O(3^{n_i}) \subseteq O(4^{\ell})$.

It is trivial to observe that monotonicity of the table A_j implies that A_i also is monotonous.

INSERT NODES: Suppose i is an INSERT NODE with child j and suppose that $X_j = (x_{j_1}, \ldots, x_{j_{n_j}})$. W.l.o.g.[3], we may assume that $X_i = (x_{j_1}, \ldots, x_{j_{n_j}}, x)$. Let $N(x) \cap X_i = \{x_{j_{p_1}}, \ldots, x_{j_{p_s}}\}$ be the neighbors of the "introduced" vertex x that appear in the bag X_i. We now define a function $\phi : \{0, \hat{0}, 1\}^{n_j} \to \{0, \hat{0}, 1\}^{n_j}$ on the set of colorings of X_j. For $c = (c_1, \ldots, c_{n_j}) \in \{0, \hat{0}, 1\}^{n_j}$, we define $\phi(c) := (c'_1, \ldots, c'_{n_j})$ such that

$$c'_t = \begin{cases} \hat{0} & \text{if } t \in \{p_1, \ldots, p_s\} \text{ and } c_t = 0, \\ c_t & \text{otherwise.} \end{cases}$$

Then, evaluate the mapping A_i of X_i as follows:

`for all` $c = (c_1, \ldots, c_{n_j}) \in \{0, \hat{0}, 1\}^{n_j}$ `do`

$$A_i(c \times \{0\}) \leftarrow \begin{cases} A_j(c) & \text{if } x \text{ has a neighbor } x_{j_q} \text{ in } X_i \text{ with } c_q = 1, \\ +\infty & \text{otherwise} \end{cases} \tag{3}$$

$$A_i(c \times \{1\}) \leftarrow A_j(\phi(c)) + 1 \tag{4}$$

$$A_i(c \times \{\hat{0}\}) \leftarrow A_j(c) \tag{5}$$

For the correctness of the assignments (3) and (4), we remark the following: It is clear that, if we assign color 0 to vertex x (see assignment (3)), we again (as already done in the initializing step of assignment (1)) have to check whether this color can be justified at the current stage of the algorithm. Such a justification is given if and only if the coloring under examination already assigns a 1 to some neighbor of x in X_i. This is true, since condition (3) of Definition 8 implies that no neighbor of x has been considered in previous bags, and, hence, up to the current stage of the algorithm, x can only be dominated by a vertex in X_i (as checked in assignment (3)).

If we assign color 1 to vertex x (see assignment (4)), we already dominate all vertices $\{x_{j_{p_1}}, \ldots, x_{j_{p_s}}\}$. Suppose now we want to evaluate $A_i(c \times \{1\})$ and suppose some of these vertices are assigned color 0 by c, say $c_{p'_1} = \ldots = c_{p'_q} = 0$ (where $(p'_1, \ldots, p'_q)$ is a subsequence of $(p_1, \ldots, p_s)$). Since the "1-assignment" of x already justifies the "0-values" of $c_{p'_1}, \ldots, c_{p'_q}$, and

[3] Possibly after rearranging the vertices in X_i and the entries of A_i accordingly.

since our mapping A_j is monotonous, we obtain $A_i(c \times \{1\})$ by taking entry $A_j(c')$, where $c'_{p'_1} = \ldots = c'_{p'_q} = \hat{0}$, i.e., where $c' = \phi(c)$.

Since we need time $O(n_i)$ in order to check whether a coloring is locally invalid, the evaluation of A_i can be carried out in time $O(3^{n_i} \cdot n_i) \subseteq O(4^{\ell})$.

Considering assignments (3) and (5), it is easy to see that A_i is monotonous if A_j is monotonous.

JOIN NODES: Suppose i is a JOIN NODE with children j and k and suppose that $X_i = X_j = X_k = (x_{i_1}, \ldots, x_{i_{n_i}})$.

Let $c = (c_1, \ldots, c_{n_i}) \in \{0, \hat{0}, 1\}^{n_i}$ be a coloring for X_i. We say that $c' = (c'_1, \ldots, c'_{n_i})$, $c'' = (c''_1, \ldots, c''_{n_i}) \in \{0, \hat{0}, 1\}^{n_i}$ *divide* c if

1. $(c_t \in \{\hat{0}, 1\} \;\Rightarrow\; c'_t = c''_t = c_i)$, and
2. $\big(c_t = 0 \;\Rightarrow\; \big[(c'_t, c''_t \in \{0, \hat{0}\}) \;\wedge\; (c'_t = 0 \vee c''_t = 0)\big]\big)$.

Then, evaluate the mapping A_i of X_i as follows:

`for all` $c \in \{0, \hat{0}, 1\}^{n_i}$ `do`

$$A_i(c) \;\leftarrow\; \min\{A_j(c') + A_j(c'') - \#_1(c) \mid c' \text{ and } c'' \text{ divide } c\,\} \tag{6}$$

In other words, in order to evaluate the value $A_i(c)$ we look up the corresponding values for coloring c in A_j (which gives us the minimum dominating set for c needed for the bags considered up to this stage in the left subtree) and in A_k (the minimum dominating set for c needed according to the right subtree), add the corresponding values, and subtract the number of "1-assignments" in c, since they would be counted twice, otherwise.

Clearly, if coloring c of node i assigns the colors 1 or $\hat{0}$ to a vertex x in X_i, we have to make sure that we use colorings c' and c'' of the children j and k that also assign the same color to x. However, if c assigns color 0 to x, it is sufficient to justify this color by *at least one* of the colorings c' or c''. Observe that, by the monotonicity of A_j and A_k we obtain the same "min" in assignment (6), if we replace condition (2) in the definition of "divide" by:

2'. $\big(c_t = 0 \;\Rightarrow\; (c'_t, c''_t \in \{0, \hat{0}\} \wedge c'_t \neq c''_t)\big)$.

Note that, for given $c \in \{0, \hat{0}, 1\}^{n_i}$, with $z := \#_0(c)$, we have 2^z many pairs (c', c'') that divide c (if we use condition (2') instead of (2) (sic!)). Since there are $2^{n_i - z}\binom{n_i}{z}$ many colorings c with $\#_0(c) = z$, we obtain

$$|\{(c', c'') : c \in \{0, \hat{0}, 1\}^{n_i},\ c' \text{ and } c'' \text{ divide } c\,\}| = \sum_{z=0}^{n_i} 2^{n_i - z}\binom{n_i}{z} \cdot 2^z = 4^{n_i}.$$

This shows that evaluating A_i can be done in time $O(4^{n_i}) \subseteq O(4^{\ell})$.

Again, it is not hard to see that A_i is monotonous if A_j and A_k are monotonous. This basically follows by the definition of "divide."

Step 3: Let r denote the root of T. For the domination number $\gamma(G)$, we finally get

$$\gamma(G) = \min\{A_r(c) \mid c \in \{0,1\}^{n_r}\}. \tag{7}$$

The minimum in Equation (7) is taken only over colorings containing colors 1 and 0, since a valid dominating set does not contain "unresolved" vertices of color $\hat{0}$.

The total running time of the algorithm is $O(4^{\ell} N)$. For the correctness of the algorithm, we observe the following. Firstly, property (1) of a tree decomposition (see Definition 8) guarantees that each vertex is assigned a color. Secondly, in our initialization Step 1, as well as in the updating process for INSERT NODES and JOIN NODES of Step 2, we made sure that the assignment of color 0 to a vertex x always guarantees that at the current stage of the algorithm x is already dominated by a vertex from previous bags. Since, by property (2) of a tree decomposition (see Definition 8), any pair of neighbors appears in at least one bag, the validity of the colorings was checked for each such pair of neighbors. And, thirdly, property (3) of a tree decomposition (see Definition 8), together with the comments given in Step 2 of the algorithm, implies that the updating of each mapping is done consistently with all mappings that have been visited earlier in the algorithm.

Also, note that, when bookkeeping how the minima in the assignments (2), (6), and (7) of Step 2 and Step 3 were obtained, this algorithm constructs a dominating set D corresponding to $\gamma(G)$. □

We remark that Aspvall *et al.* [7] addressed the memory requirement problem arising in the type of algorithms described above. They suggested a method to minimize the sum of the sizes of the tables that need to be stored simultaneously.

Using Corollary 12 and Theorem 13, a straightforward solution to the k-DOMINATING SET problem on planar graphs using tree decompositions leads to an algorithm which runs in time $O(4^{9k}n)$, which, of course, is inferior to the best known search tree algorithm that has running time $O(8^k n)$ [1].

In Subsection 3, we show that a planar graph with a k-dominating set has treewidth $O(\sqrt{k})$. Moreover, we prove that a corresponding tree decomposition can be found in time $O(\sqrt{k}n)$. Combining this with Theorem 13 gives a significant asymptotic improvement of the search tree result.

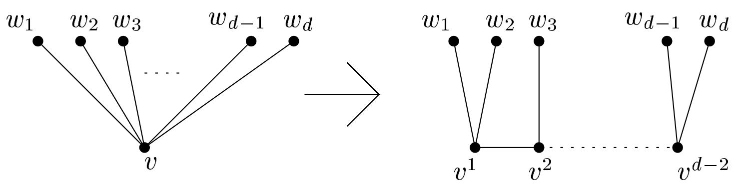

Figure 1: Replacing a vertex by a path with vertices of degree 3.

2.3 Treewidth and r-outerplanarity

In this section, we give a constructive proof of the result stated in Theorem 11. The proof is based upon the proof in [13, Theorem 83]. To be more precise, we show:

Theorem 14 *Let an r-outerplanar graph $G = (V, E)$ be given together with an r-outerplanar embedding. Then, a tree decomposition $\langle \{X_i \mid i \in I\}, T\rangle$, with width at most $3r - 1$ and with $|I| = O(n)$ of G, can be found in $O(rn)$ time.*

We make the assumption that with the embedding, for each vertex v, the incident edges of v are given in clockwise order as they appear in the embedding. Most (linear time) graph planarity testing and embedding algorithms yield such orderings of the edge lists (see [19]).

Now, we discuss the proof of Theorem 14. For the construction of the desired tree decomposition we proceed in several steps. Firstly, we determine the layers of the given graph G. Secondly, G is embedded in a graph H, whose degree is bounded by 3 and whose outerplanarity number doesn't exceed the one of G. Then, a suitable maximal spanning forest is constructed for H. This step is done inductively proceeding along the different layers of H in a way that the so-called "edge and vertex remember numbers" are kept small. Using this spanning tree, in a further step, we determine a tree decomposition of H. Finally, this tree decomposition can be turned into a tree decomposition of G.

Determining the layers of the embedded graph. Without loss of generality, we assume G is connected. Suppose G is given with an r-outerplanar embedding with, for every vertex, a clockwise ordering of its adjacent edges. With the help of these orderings, one can build, in $O(|V|)$ time, the dual graph G^*, with pointers from edges of G to the two adjacent faces in the dual. We can first partition the set of faces into 'layers': put a face f in layer L^*_{i+1} if the distance of this face in the dual graph to the exterior face is i. This distance can be determined in linear time using breadth-first search on the dual graph.

Now, a vertex v of G belongs to layer L_i for the smallest i such that v is adjacent to a face f_v in L^*_i. Note that faces can belong to layer L^*_{r+1}, but not to layers L^*_s with $s > r + 1$.

Embedding G in a graph H of degree three. The next step is to construct an r'-outerplanar (where $r' \leq r$) graph $H = (V_H, E_H)$ of degree at most three that contains G as a minor, i.e., G can be obtained from H by a series of vertex deletions, edge deletions, and edge contractions.

In order to do this, every vertex with degree $d \geq 4$ is replaced by a path of $d - 2$ vertices of degree 3, as shown in Fig. 1. This is done in such a way that the graph stays r'-outerplanar. More precisely, for each vertex v of layer L_i we determine the face f_v in L_i^* as described above. We then find two successive edges $\{x, v\}$, $\{v, y\}$ that are border to f_v. Now, let x take the role of w_1 and y take the role of w_d in Fig. 1. Observe that, in this manner, all vertices v^i on the newly formed path are adjacent to face $f_v \in L_i^*$. Let H be the graph obtained after replacing all vertices of degree at least four in this manner.

Note that H has the same set of faces as G (i.e., H^* and G^* share the same set of vertices) and that faces adjacent in G are still adjacent in H (i.e., G^* is a subgraph of H^*). Hence, H^* can be obtained from G^* by possibly adding some further edges. Clearly, the minimum distance of a vertex in H^* to the exterior face vertex may only decrease (not increase) compared to the corresponding distance in G^*. Since the layer to which a vertex belongs is exactly one more than the minimum distance of the adjacent faces to the exterior face, the outerplanarity number r' of H is bounded by r.

Constructing a suitable maximal spanning forest for H. At this point, we have an r'-outerplanar graph $H = (V_H, E_H)$ of maximum degree 3. We now construct a maximal spanning forest T for H that yields small so-called "edge and vertex remember numbers." This step is done inductively along the different layers proceeding from inside towards the exterior.

Observe that when removing all *edges* on the exterior face of an s-outerplanar graph of maximum degree three, we obtain an $(s-1)$-outerplanar graph, when $s > 1$. When we remove all edges on the exterior face of an outerplanar graph, we obtain a forest.

Thus, we can partition the edges into $r' + 1$ sets $E_1, \ldots, E_{r'+1}$, with E_1 the edges on the exterior face, and E_i the edges on the exterior face when all edges in $E_1 \cup \ldots \cup E_{i-1}$ are removed. Again, using the dual, this partition can be computed in $O(n)$ time.

Now, we form a sequence of forests. We start with forest $T_{r'+1}$, which consists of all edges in $E_{r'+1}$. (Note that these are the interior edges of an outerplanar graph of maximum degree 3, so $T_{r'+1}$ is acyclic.)

When we have T_i, $1 < i \leq r' + 1$, we form T_{i-1} in the following way: add a maximal set of edges from E_{i-1} to T_i such that no cycles are formed. Note that in this way, each T_i is a maximal spanning forest of the subgraph formed by the edges in $E_i \cup \ldots \cup E_{r'+1}$; we call this subgraph H_i. (Maximality is meant here with respect to set inclusion; a maximal spanning forest of an arbitrary graph hence can be found in $O(n)$ time using a standard depth first search approach, but here we need such a forest made in a specific way.)

It is not hard to see that one such step can be done in $O(n)$ time; as we do at most r such steps, the time to build T_1 becomes $O(rn)$.

Definition 15 For a graph $G = (V, E)$, and a forest $T = (V, F)$ that is a subgraph of G, define the *edge remember number* $\mathrm{er}(G, T, e)$ of an edge $e \in F$ (with respect to G and T) as the number of edges $\{v, w\} \in E - F$ such that there is a simple path in T from v to w that uses e. The edge remember number of T (with respect to G) is $\mathrm{er}(G, T) = \max_{e\in F} \mathrm{er}(G, T, e)$. The *vertex remember number* $\mathrm{vr}(G, T, v)$ of a vertex $v \in V$ (with respect to G and T) is the number of edges $\{v, w\} \in E - F$ such that there is a simple path in T from v to w that uses v. The vertex remember number of T (with respect to G) is $\mathrm{vr}(G, T) = \max_{v\in V} \mathrm{vr}(G, T, v)$.

One may observe that the construction of the trees T_i is the one used in the proofs given in [13, Section 13]. The following result is proved in [13, Lemma 80]. Note that in order to obtain this result it is essential that the degree of H is bounded by 3:

Lemma 16 *(i) For every i, $1 \le i \le r' + 1$, $\mathrm{er}(H_i, T_i) \le 2(r' + 1 - i)$.*
(ii) For every i, $1 \le i \le r'$, $\mathrm{vr}(H_i, T_i) \le 3(r' + 1 - i) - 1$. □

Without loss of generality, we can suppose that G and, therefore, H is connected and, hence, we have a spanning tree T_1 of H with $\mathrm{er}(H, T_1) \le 2r'$ and $\mathrm{vr}(H, T_1) \le 3r' - 1$.

Deriving a tree decomposition from the spanning forest. We now apply the following result of [13, Theorem 71] to the graph H and the spanning forest T_1 in order to obtain a tree decomposition in time $O(rn)$ of width bounded by

$$\max(\mathrm{vr}(H, T_1), \mathrm{er}(H, T_1) + 1) \le 3r' - 1 \le 3r - 1.$$

For the sake of completeness of the whole construction, we outline the easy proof.

Theorem 17 *Let $T = (V, F)$ be a maximal spanning forest for the graph $G = (V, E)$. Then, a tree decomposition with width at most* $\max(\mathrm{vr}(G, T), \mathrm{er}(G, T)+1)$ *and $O(n)$ nodes can be determined in $O(\mathrm{vr}(G, T) \cdot n)$ time.*

Proof. Our aim is to construct a tree decomposition $\langle\{X_i \mid i \in I\}, T'\rangle$ of G. Let $T' = (V \cup F, F')$ with $F' = \{\{v, e\} \mid v \in V, e \in F, \exists w \in V : e = \{v, w\}\}$ be the tree obtained by subdividing every edge of T. The bags X_i for $i \in I := V \cup F$ are obtained as follows. For every $v \in V$, add v to X_v. For every $e = \{v, w\} \in F$, add v and w to X_e. Now, for every edge $e = \{v, w\}$ in E but not in F, add v to all sets X_u and X_e, with $u \in V$ or $e \in F$ on the path from v to w in T.

Using standard graph algorithmic techniques, the path between two vertices in a tree can be found in time proportional to the length of that path; since each vertex in T can contribute to at most $\mathrm{vr}(G,T)$ such paths, the running time is bounded by $O(\mathrm{vr}(G,T) \cdot n)$.

It is easy to check that this indeed yields a tree decomposition. Its bags have size $|X_v| \leq 1 + \mathrm{vr}(G,T)$ (for all $v \in V$) and $|X_e| \leq 2 + \mathrm{er}(G,T)$ (for all $e \in E$). Hence, the resulting treewidth is at most $\max(\mathrm{vr}(G,T), \mathrm{er}(G,T)+1)$. □

Undoing the minor operations. Finally, the tree decomposition of H can be turned into a tree decomposition of G of equal or smaller width by replacing every occurrence of a vertex v^i in a bag X_i by an occurrence of the corresponding vertex v (see e.g., [13, Lemma 16].) This again costs time linear in the total size of all bags X_i in the tree decomposition, i.e., $O(rn)$ time.

Altogether this establishes the correctness of Theorem 14.

3 The main result

In this section, we show that a planar graph with domination number k has treewidth of at most $O(\sqrt{k})$. This improves Corollary 12 considerably. In the next section, we will show how this new result can be turned into a constructive algorithm. Combining the results of this section with Theorem 13, we will present an algorithm having time complexity $4^{O(\sqrt{k})}n$. This obviously gives an asymptotic improvement of the $O(8^k n)$ search tree algorithm [1].

This section is organized as follows. In a first subsection, we make some general observations on how to construct tree decompositions using separators. The following two subsections show that, in a planar graph which admits a k-dominating set, we can find small separators layerwisely. Finally, the results are pieced together to prove our main result in this section.

3.1 Separators and treewidth

Here, the main idea is to find small separators of the graph and to merge the tree decompositions of the resulting subgraphs.

Definition 18 Let $G = (V,E)$ be a graph. A subset S of the vertex set V is called a *separator* of G, if the subgraph $G - S$ is disconnected.

For any given separator splitting a graph into different components, we obtain a simple upper bound for the treewidth of this graph which depends on the size of the separator and the treewidth of the resulting components.

Proposition 19 *If a connected graph can be decomposed into components of treewidth of at most t by means of a separator of size s, then the whole graph has treewidth of at most $t+s$.*

Proof. The separator splits the graph into different components. Suppose we are given the tree decompositions of these components of width at most t. The goal is to construct a tree decomposition for the original graph. This can be achieved by firstly merging the separator to every bag in each of these given tree decompositions. In a second step, add some arbitrary connections preserving acyclicity between the trees corresponding to the components. It is straightforward to check that this forms a tree decomposition of the whole graph of width at most $t+s$. □

For plane graphs, there is an iterated version of this observation.

Proposition 20 *Let G be a plane graph with layers L_i, $(i = 1, \dots, r)$. For $i = 1, \dots, \ell$, let $\mathcal{L}_i$ be a set of consecutive layers, i.e., $\mathcal{L}_i = \{L_{j_i}, L_{j_i+1}, \dots, L_{j_i+n_i}\}$, such that $\mathcal{L}_i \cap \mathcal{L}_{i'} = \emptyset$ for all $i \neq i'$. Moreover, suppose G can be decomposed into components, each of treewidth of at most t, by means of separators $S_1, \dots, S_\ell$, where $S_i \subseteq \bigcup_{L \in \mathcal{L}_i} L$ for all $i = 1, \dots, \ell$. Then, G has treewidth of at most $t + 2s$, where $s = \max_{i=1,\dots,\ell} |S_i|$.*

Proof. The proof again uses the merging-technique illustrated in the previous proposition: Suppose, w.l.o.g., the sets $\mathcal{L}_i$ appear in successive order, i.e., $j_i < j_{i+1}$. For each $i = 0, \dots, \ell$, consider the component G_i of treewidth at most t which is cut out by the separators S_i and S_{i+1} (by default, we set $S_0 = S_{\ell+1} = \emptyset$). We add S_i and S_{i+1} to every node in a given tree decomposition of G_i. In order to obtain a tree decomposition of G, we successively add an arbitrary connection between the trees T_i and T_{i+1} of the so-modified tree decompositions that correspond to the subgraphs G_i and G_{i+1}. □

3.2 Finding separators layerwisely

In the following, we assume that our graph G has a fixed plane embedding with r layers. We show that the treewidth cannot exceed $O(\sqrt{k})$ if a dominating set of size k exists.

We assume that we have a dominating set D of size at most k. Let k_i be the number of vertices of $D_i = D \cap L_i$. Hence, $\sum_{i=1}^{r} k_i = k$. In order to avoid case analysis, we set $k_0 = k_{r+1} = k_{r+2} = 0$. Moreover, let c_i denote the number of non-vacuous layer components of layer L_i.

Our approach is based on finding small separators in G layerwisely. More precisely, for each $i = 1, \dots, r-2$, we want to construct a set $S_i \subseteq L_{i-1} \cup L_i \cup L_{i+1}$

separating layer L_{i-1} from layer L_{i+2} in such a way[4] that the total size of these sets can be bounded by some linear term in k. The key idea for proving that S_i separates layers L_{i-1} from L_{i+2} relies on a close investigation of the paths leaving layer L_{i-1} to the interior of the graph. Each such path passes a "first" vertex in layer L_i. This particular vertex can be dominated by vertices from D_{i-1}, D_i, or D_{i+1}. It turns out that, in order to cut this particular path, the set S_i has to contain the vertices of the sets D_{i-1}, D_i, and D_{i+1} *plus* some suitably chosen pairs of neighbors of any of these vertices. This results in letting S_i be the union of so-called "upper," "lower," and "middle" triples. We will carry out the to some extent technically complicated step in what follows.

For this purpose, in the following, we write $N(x)$ to describe the set of neighbors of a vertex x and use the notion $B(\cdot)$ for boundary cycles as introduced in Definition 7.

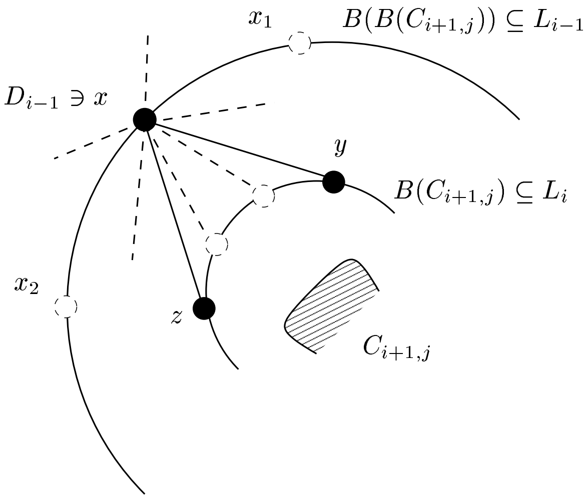

Figure 2: Upper triples.

Upper triples. An *upper triple* for layer L_i is associated to a non-vacuous[5] layer component $C_{i+1,j}$ of layer L_{i+1} and a vertex $x \in D_{i-1}$ that has a neighbor on the boundary cycle $B(C_{i+1,j})$ (see Fig. 2). Then, clearly, $x \in B(B(C_{i+1,j}))$, by definition of a boundary cycle. Let x_1 and x_2 be the neighbors of x on the boundary cycle $B(B(C_{i+1,j}))$. Starting from x_1, we go around x up to x_2 so that we visit all neighbors of x in layer L_i. We note the neighbors of x on the boundary cycle $B(C_{i+1,j})$. Going around gives two outermost neighbors y and z

[4]Intuitively, one might be tempted to construct a set of vertices separating layer L_{i-1} from layer L_{i+1}. However, for technical reasons that become clear in the proof of the forthcoming Proposition 25, it is more convenient to separate layers L_{i-1} and L_{i+2}.

[5]Note that here, as well as in the definitions of middle and upper triples, all vacuous components of layer L_{i+1} are of no interest to us, since we want to find a set of vertices separating levels L_{i-1} from L_{i+2}.

on this boundary cycle. The triple, then, is the three-element set $\{x, y, z\}$. In case x has only a single neighbor y in $B(C_{i+1,j})$, the "triple" consists of only $\{x, y\}$ (let, by default, $z = y$).

Definition 21 For each non-vacuous layer component $C_{i+1,j}$ of L_{i+1} and each vertex $x \in D_{i-1}$ with neighbors in $B(C_{i+1,j})$, the set $\{x, y, z\}$ as described above is called an *upper triple* of layer L_i.

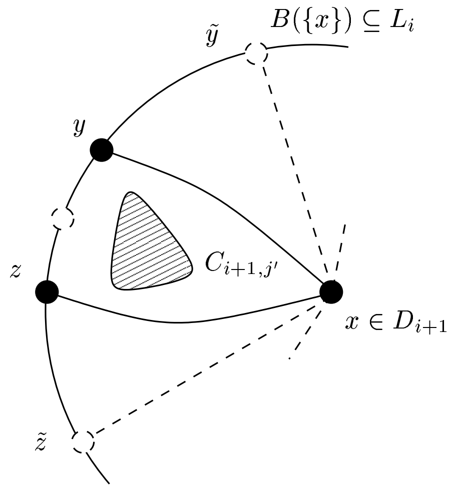

Figure 3: Lower triples.

Lower triples. A *lower triple* for layer L_i is associated to a vertex $x \in D_{i+1}$ and a non-vacuous layer component $C_{i+1,j}$ of layer L_{i+1} (see Fig. 3). We only consider layer components $C_{i+1,j}$ of layer L_{i+1} that are enclosed by the boundary cycle $B(\{x\})$. For each pair $\tilde{y}, \tilde{z} \in B(\{x\}) \cap N(x)$ (where $\tilde{y} \neq \tilde{z}$), we consider the path $P_{\tilde{y},\tilde{z}}$ from $\tilde{y}$ to $\tilde{z}$ along the cycle $B(\{x\})$, taking the direction such that the region enclosed by $\{\tilde{z}, x\}$, $\{x, \tilde{y}\}$, and $P_{\tilde{y},\tilde{z}}$ contains the layer component $C_{i+1,j}$. Let $\{y, z\} \subseteq B(\{x\}) \cap N(x)$ be the pair such that the corresponding path $P_{y,z}$ is shortest. The triple, then, is the three-element set $\{x, y, z\}$. If x has no or only a single neighbor y in $B(\{x\})$, then the "triple" consists only of $\{x\}$, or $\{x, y\}$ (by default, in these cases, we let $x = y = z$, or $y = z$, respectively).

Definition 22 For each vertex $x \in C_{i+1,j}$ of D_{i+1} and each non-vacuous layer component $C_{i+1,j}$ that is enclosed by $B(\{x\})$, the set $\{x, y, z\}$ as described above is called a *lower triple* of layer L_i.

Middle triples. A *middle triple* for layer L_i is associated to a non-vacuous layer component $C_{i+1,j}$ and a vertex $x \in D_i$ that has a neighbor in $B(C_{i+1,j})$ (see Fig. 4). Note that, due to the layer model, it is easy to see that a vertex $x \in D_i$ can have at most two neighbors y, z in $B(C_{i+1,j})$. Depending on whether x itself

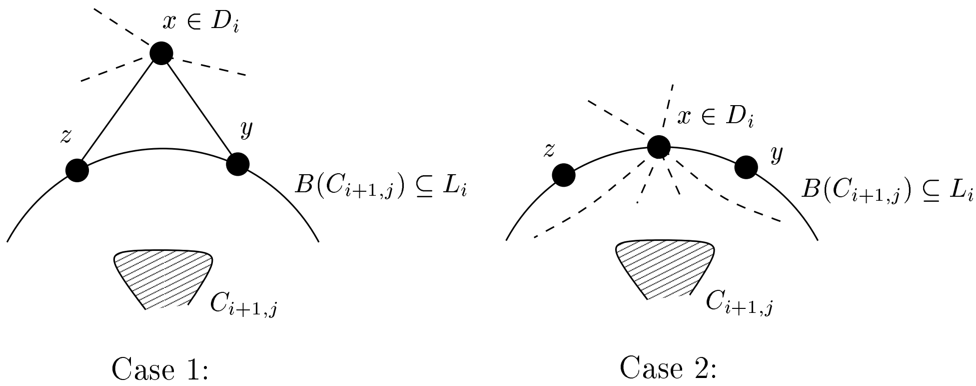

Figure 4: Middle triples.

lies on the cycle $B(C_{i+1,j})$ or not, we obtain two different cases which are both illustrated in Fig. 4. In either of these cases, the middle triple is defined as the set $\{x, y, z\}$. Again, if x has none or only a single neighbor y in $B(C_{i+1,j})$, then the "triple" consists only of $\{x\}$, or $\{x, y\}$, respectively (by default, in these cases, we let $x = y = z$, or $y = z$, respectively).

Definition 23 For each non-vacuous layer component $C_{i+1,j}$ and each vertex $x \in D_i$, the set $\{x, y, z\}$ as described above is called a *middle triple* for layer L_i.

Definition 24 We define the set S_i as the union of all upper triples, lower triples and middle triples of layer L_i.

In the following, we will show that S_i is a separator of the graph. Note that the upper bounds on the size of S_i, which are derived afterwards, are crucial for the upper bound on the treewidth derived later on.

Proposition 25 *The set S_i separates vertices of layers L_{i-1} and L_{i+2}.*

Proof. Suppose there is a path P (with no repeated vertices) from layer L_{i+2} to layer L_{i-1} that avoids S_i. This clearly implies that there exists a path P' from a vertex x in a non-vacuous layer component $C_{i+1,j}$ of layer L_{i+1} to a vertex $z \in B(B(C_{i+1,j}))$ in layer L_{i-1} which has the following two properties:

- $P' \cap S_i = \emptyset$.
- All vertices inbetween x and z belong to layer L_i or to vacuous layer components of layer L_{i+1}.

This can be achieved by simply taking a suitable sub-path P' of P. Let y_1 (and y_2, respectively) be the first (last) vertex along the path P' from x to z that lies on the boundary cycle $B(C_{i+1,j}) \subseteq L_i$ (see Fig. 5).

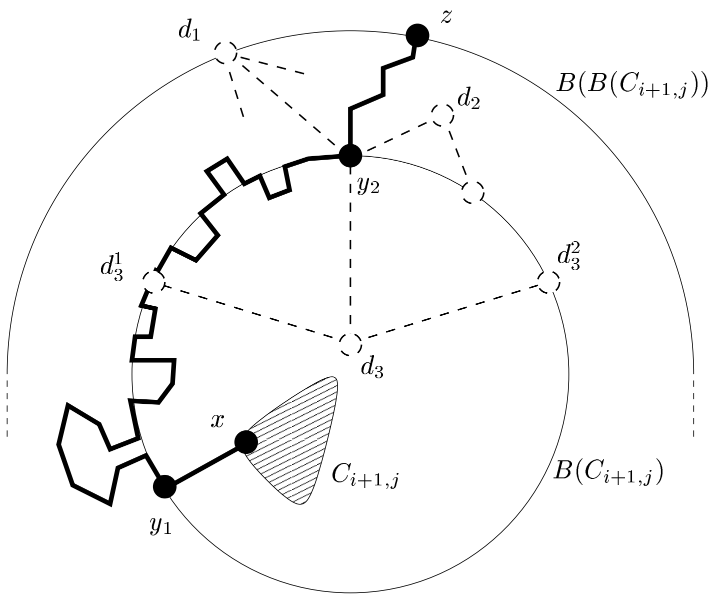

Figure 5: S_i separates L_{i-1} and L_{i+2}.

Obviously, y_2 cannot be an element of D since, then, it would appear in a middle triple of layer L_i and, hence, in S_i. We now consider the vertex that dominates y_2. This vertex can lie in layer L_{i-1}, L_i, or L_{i+1}.

Suppose first that y_2 is dominated by a vertex $d_1 \in L_{i-1}$. Then, d_1 is in $B(B(C_{i+1,j}))$, simply by definition of the boundary cycle (see Fig. 5). Since G is planar, this implies that y_2 must be an "outermost" neighbor of d_1 among all elements in $N(d_1) \cap B(C_{i+1,j})$. If this were not the case, then there would be an edge from d_1 to a vertex on $B(C_{i+1,j})$ that leaves the closed region bounded by $\{d_1, y_2\}$, the path from y_2 to z, and the corresponding path from z to d_1 along $B(B(C_{i+1,j}))$. Hence, y_2 would be in the upper triple of layer L_i which is associated to the layer component $C_{i+1,j}$ and d_1. This contradicts the assumption that P' avoids S_i.

Now, suppose that y_2 is dominated by a vertex $d_2 \in D_i$ (see Fig. 5). By definition of the middle triples, this clearly implies that y_2 is in the middle triple associated to $C_{i+1,j}$ and d_2. Again, this contradicts the fact that $P' \cap S_i = \emptyset$.

Consequently, the dominating vertex d_3 of y_2 has to lie in layer L_{i+1}. Let $\{d_3, d_3^1, d_3^2\}$, where $d_3^1, d_3^2 \in N(d_3) \cap B(C_{i+1,j})$, be the lower triple associated to layer component $C_{i+1,j}$ and d_3 (see Fig. 5). By definition, $C_{i+1,j}$ is contained in the region enclosed by $\{d_3^1, d_3\}, \{d_3, d_3^2\}$, and the path from d_3^2 to d_3^1 along $B(C_{i+1,j})$, which—assuming that $y_2 \notin \{d_3, d_3^1, d_3^2\}$—does not hit y_2 (see Fig. 5). We now observe that, whenever the path from y_1 to y_2 leaves the cycle $B(C_{i+1,j})$ to its exterior, say at a vertex q, then it has to return to $B(C_{i+1,j})$ at a vertex $q' \in N(q) \cap B(C_{i+1,j})$. Otherwise, we would violate the fact that, by definition,

$B(C_{i+1,j}) \subseteq L_i$. This, however, shows that the path P' has to hit either d_3^1 or d_3^2 on its way from y_1 to y_2. Since $d_3^1, d_3^2 \in S_i$, this case also contradicts the fact that $P' \cap S_i = \emptyset$. □

3.3 An upper bound for the size of the separators

In this subsection, we want to show that the sum of the cardinalities of all separators can be bounded by some linear term in k.

Lemma 26 $|S_i| \leq 5(k_{i-1} + k_i + k_{i+1}) + 12c_{i+1}$.

Proof. We give bounds for the number of vertices in upper, middle, and lower triples of layer i, separately.

Firstly, we discuss the upper triples of layer i, which were associated to a non-vacuous layer component $C_{i+1,j}$ of layer L_{i+1} and a vertex $x \in D_{i-1}$ with neighbors in $B(C_{i+1,j})$. Consider the bipartite graph G' which has vertices for each non-vacuous layer component $C_{i+1,j}$ and for each vertex in D_{i-1}. Whenever a vertex in D_{i-1} has a neighbor in $B(C_{i+1,j})$, an edge is drawn between the corresponding vertices in G'. Each edge in G', by construction, may correspond to an upper triple of layer L_i. Note that G' is a planar bipartite graph whose bipartition subsets consist of k_{i-1} and c_{i+1} vertices, respectively. Thus, the number of edges of G' is linear in the number of vertices; more precisely, it is bounded by $2(k_{i-1}+c_{i+1})$ (see [40, Corollary 1.2.]). From this, we obtain an upper bound for the number of vertices in upper triples of layer L_i as follows: Potentially, each vertex of D_{i-1} appears in an upper triple and, for each edge in G', we possibly obtain two further vertices in an upper triple. This shows that the total number of vertices in upper triples is bounded by $k_{i-1} + 4(k_{i-1} + c_{i+1})$.

A similar analysis can be used to show that the number of vertices in the lower triples is bounded by $k_{i+1} + 4(k_{i+1} + c_{i+1})$ and that the number of vertices in the middle triples can be bounded by $k_i + 4(k_i + c_{i+1})$.

By definition of S_i, this proves our claim. □

Note that, by a more detailed investigation, the bound given in Lemma 26 probably can be improved. One observes, e.g., that the planar bipartite graph G', which was constructed in the proof, has the special property that it is a "hyper-plane" bipartite graph, i.e., one of the bipartition subsets can be arranged on a line and all edges of the graph lie in one half-plane of this line. This property of G' is immediate from the fact that the upper triples associated to a non-vacuous layer component $C_{i+1,j}$ lie within the boundary cycle $B(B(C_{i+1,j}))$. For such graphs, our investigations indicate that one might obtain better estimates on the number of their edges than the ones used in the proof of Lemma 26.

A similar observation can be made for estimating the bounds for the lower triples.

For the number c_i of non-vacuous components in layer i, we have the following estimate.

Lemma 27 $c_i \leq k_i + k_{i+1} + k_{i+2}$.

Proof. By definition, c_i refers to only non-vacuous layer components in layer L_i, i.e., there is at least one vertex of layer L_{i+1} contained within each such layer component. Such a vertex can only be dominated by a vertex from layer L_i, L_{i+1}, or L_{i+2}. In this way, we get the claimed upper bound. □

Combining the two previous results yields the bound claimed at the beginning of this subsection.

Proposition 28 $\sum_{i=1}^{r} |S_i| \leq 51k$, *where r is the number of layers of the graph.*

Proof. This follows directly when we combine the previous two lemmas, noting that $\sum_{i=1}^{r} k_i = k$. □

3.4 An improved relation between domination number and treewidth

We are now ready to state our main result relating the domination number and the treewidth of a planar graph. Note that the following theorem gives a decisive asymptotic improvement of Corollary 12.

Theorem 29 *A planar graph with domination number k has treewidth of at most $6\sqrt{34}\sqrt{k} + 8$.*

Proof. Using the separators S_i as found in the previous subsections, we consider the following three sets of vertices: $\mathbb{S}_1 = S_1 \cup S_4 \cup S_7 \cup \ldots$, $\mathbb{S}_2 = S_2 \cup S_5 \cup S_8 \cup \ldots$, and $\mathbb{S}_3 = S_3 \cup S_6 \cup S_9 \cup \ldots$. Since, by Proposition 28, $|\mathbb{S}_1| + |\mathbb{S}_2| + |\mathbb{S}_3| \leq 51k$, one of these sets has size at most $\frac{51}{3}k$, say $\mathbb{S}_\delta$ (with $\delta \in \{1, 2, 3\}$).

Let δ and $\mathbb{S}_\delta$ be obtained as above. Let $d := \frac{3}{2}\sqrt{34}$. We now go through the sequence $S_{1+\delta}, S_{4+\delta}, S_{7+\delta}, \ldots$ and look for separators of size at most $s(k) := d\sqrt{k}$. Due to the estimate on the size of $\mathbb{S}_\delta$, such separators of size at most $s(k)$ must appear within each $n(k) := \frac{51}{3} \cdot \frac{1}{d\sqrt{k}} \cdot k = \frac{1}{3}\sqrt{34}\sqrt{k}$ sets in the sequence. In this manner, we obtain a set of disjoint separators of size at most $s(k)$ each, such that any two consecutive separators from this set are at most $3n(k)$ layers apart. Clearly, the separators chosen in this way fulfill the requirements in Proposition 20.

Observe that the components cut out in this way each have at most $3(n(k)+1)$ layers and, hence, their treewidth is bounded by $9(n(k)+1)-1$ due to Theorem 11.

Using Proposition 20, we can compute an upper bound on the treewidth $\mathrm{tw}(k)$ of the originally given graph with domination number k:

$$\begin{aligned}\mathrm{tw}(k) &\leq 2s(k) + 9(n(k)+1) - 1\\ &= 2(\frac{3}{2}\sqrt{34}\sqrt{k}) + 9(\frac{1}{3}\sqrt{34}\sqrt{k}) + 8\\ &= 6\sqrt{34}\sqrt{k} + 8.\end{aligned}$$

This proves our claim. □

How did we come to the constants? We simply computed the minimum of $2s(k) + 9(n(k) + 1) - 1$ (the upper bound on the treewidth) given the bound $s(k)n(k) \leq \frac{51}{3}k$. This suggests $s(k) = d\sqrt{k}$, and d is optimal when $2s(k) = 9n(k) = 9 \cdot \frac{51}{3} \cdot k \cdot s(k)^{-1}$, so, $2d = \frac{153}{d}$, i.e., $d = \frac{3}{2}\sqrt{34}$.

Observe that the tree structure of the tree decomposition obtained in the preceding proof corresponds to the structure of the layer decomposition forest.

Remark 30 Up to constant factors, the relation exhibited in Theorem 29 is optimal. This can be seen, for example, by considering a complete grid graph G_ℓ of size $\ell \times \ell$, i.e., with ℓ^2 vertices. It is known that $\mathrm{tw}(G_\ell) \geq \ell$ (see [13, Corollary 89]) and the domination number obeys $\gamma(G_\ell) \in \Theta(\ell^2)$, see [30, Theorem 2.39]. Therefore, there is an infinite family of planar graphs $\hat{G}_{k_i}$ with domination number k_i such that $\hat{G}_{k_i}$ has treewidth of $\Omega(\sqrt{k_i})$.

4 The algorithm

In this section, we outline our fixed parameter algorithm for solving k-dominating set on planar graphs constructively. The input instance to the algorithm consists of a planar graph G and a positive integer k. The algorithm determines whether G admits a dominating set of size at most k, and, if so, constructs such a set. The running time for the algorithm will be $4^{O(\sqrt{k})}n$.

Clearly, this generalizes to constructing a minimum size dominating set in a plane graph in time $4^{O(\sqrt{\gamma(G)})}n$, where $\gamma(G)$ is the domination number of the graph. If $\gamma(G)$ is not known in advance, one has to apply the fixed parameter algorithm and try different values of k—this can be done at the cost of an extra multiplicative factor of $O(\log \gamma(G))$ by using binary search.

The key idea for our algorithm is to construct a tree decomposition of the stated width. However, Theorem 29 is a pure existence theorem which cannot be made constructive directly. There is one point that needs specific attention. As we do not start with the dominating set given, we cannot construct the upper, middle, and lower triples. Instead, we have to compute the minimum size separator $\hat{S}_i$ between L_{i-1} and L_{i+2} directly, and use that set instead of S_i as defined in the proof of Subsection 3.2. Such a minimum size separator can be computed with well known techniques based on maximum flow (see, e.g., [33]).

Our algorithm proceeds in the following steps:

1. Embed the planar graph $G = (V, E)$ crossing-free into the plane. Determine the outerplanarity number r of this embedding and all layers $L_1, \ldots, L_r$. By default, we set $L_i = \emptyset$ for all $i < 0$ and $i > r$.

2. If $r > 3k$ then `exit` (there exists no k-dominating set). This step of the algorithm is justified by Proposition 4.

3. For $\delta = 1, 2, 3$ and $i = 0, \ldots, \lfloor \frac{r}{3} \rfloor - 1$, find the minimum separator $\hat{S}_{3i+\delta}$, separating layers $L_{(3i+\delta)-1}$ and $L_{(3i+\delta)+2}$. Let $s_{3i+\delta} = |\hat{S}_{3i+\delta}|$.

4. Check whether there exists a $\delta \in \{1, 2, 3\}$ and an increasing sequence $(i_j)_{j=1,\ldots,t}$ of indices in $\{0, \ldots, \lfloor \frac{r}{3} \rfloor - 1\}$, such that

$$\begin{aligned} s_{3i_j+\delta} &\leq s(k) := \frac{3}{2}\sqrt{34}\sqrt{k} \quad \text{for all } j = 1, \ldots, t \text{ and} \\ |i_{j+1} - i_j| &\leq n(k) := \frac{1}{3}\sqrt{34}\sqrt{k} \quad \text{for all } j = 1, \ldots, t-1. \end{aligned}$$

   If the answer is "no," then `exit` (there exists no k-dominating set). This step of the algorithm is justified by the considerations in the proof of Theorem 29.

5. Consider the separators $\mathcal{S}_j := \hat{S}_{3i_j+\delta}$ for $j = 1, \ldots, t$ (by default, $\mathcal{S}_j = \emptyset$ for all other j) and, for each $j = 0, \ldots, t$, let G_j be the subgraph cut out by the separators $\mathcal{S}_j$ and $\mathcal{S}_{j+1}$ or, more precisely, let

$$G_j := G - \Big(\bigcup_{\ell=(3i_j+\delta)-1}^{(3i_{(j+1)}+\delta)+1} L_\ell \setminus (\mathcal{S}_j \cup \mathcal{S}_{j+1}) \Big).$$

   Note that G_j is at most $3(n(k)+1)$-outerplanar.

6. Construct tree decompositions $\mathcal{X}_j$ for G_j ($j = 0, \ldots, t$) with $O(n)$ nodes each. For this step, we refer to Theorem 14.

7. Construct a tree decomposition $\mathcal{X}$ of G with $O(n)$ nodes using the separators $\mathcal{S}_1, \ldots, \mathcal{S}_t$ and the tree decompositions $\mathcal{X}_j$ by the separator merging technique described in the proof of Proposition 20.

8. Solve the DOMINATING SET problem for G with tree decomposition $\mathcal{X}$ as described in Theorem 13.

Step 1 can be implemented with methods described in Section 2.3.

As to Step 3 of the algorithm, we want to remark that such a minimum size separator can be computed with well known techniques based on maximum flow (see, e.g., [33]) as follows: For given values of i and δ, we first, build the graph G', induced by the vertices in $L_{3i+\delta-1} \cup L_{3i+\delta} \cup L_{3i+\delta+1} \cup L_{3i+\delta+2}$. Then, we contract all vertices in $L_{3i+\delta-1}$ to one vertex s, and all vertices in $L_{3i+\delta+2}$ to one vertex t. We look for a minimum s-t-separator in the resulting graph but, if this separator has size more than $s(k)$, then we just note that no separator of size at most $s(k)$ exists (compare with Step 4). Finding the separator or determining that no small enough separator exists can be done with the following standard method. Build a directed graph G'' in the following way. Every vertex v in the graph is replaced by two vertices v^- and v^+, with an edge from v^- to v^+, and an edge $\{v, w\}$ is replaced by two edges (v^+, w^-), (w^+, v^-). As explained in [37, Lemma 11, page 83], the minimum s-t separator in G' can be found by applying a maximum flow algorithm in G''. To keep the running time in the stated bounds, we use the Ford-Fulkerson maximum flow algorithm, but stop as soon as a flow value of more than $s(k)$ is used since, in such a case, we can conclude that no s-t-separator of size at most $s(k)$ exists. As each flow augmentation step costs time linear in the number of vertices and edges of G'' and increases the flow value by one, the time used by this procedure is $O(n' \cdot s(k))$, with $n' = |L_{3i+\delta-1} \cup L_{3i+\delta} \cup L_{3i+\delta+1} \cup L_{3i+\delta+2}|$.

As for every vertex in G, there are at most four combinations of δ and i in Step 3 such that it belongs to $L_{3i+\delta-1} \cup L_{3i+\delta} \cup L_{3i+\delta+1} \cup L_{3i+\delta+2}$ and the total time of step 3 is bounded by $O(n \cdot s(k)) = O(n\sqrt{k})$.

The correctness of the algorithm above follows from our considerations in the previous sections.

Steps 1-7 allow us to construct a tree decomposition of width $6\sqrt{34}\sqrt{k} + 8$ of G in $O(\sqrt{k}n)$ time. The running time for the last step in the algorithm is $O(4^{6\sqrt{34}\sqrt{k}}n)$.

Summarizing these observations, we obtain the following theorem.

Theorem 31 *The* k-DOMINATING SET *problem on planar graphs can be solved in time* $O(c^{\sqrt{k}}n)$*, where* $c = 4^{6\sqrt{34}}$*. Moreover, if* $\gamma(G) \leq k$*, a minimum size dominating set can be constructed within the same time.* □

The constant c above is $4^{6\sqrt{34}}$, which is rather large. However, a more refined analysis should help to reduce this constant. Moreover, it is a worst case estimate, which might be far from what happens in practical applications.

We want to mention that an algorithm of similar running time could be obtained by combining our result (Theorem 29) with the $O(8^r n)$ time algorithm for solving DOMINATING SET on r-outerplanar graphs indicated by Baker [9]. To that end, one executes Steps 1-5 as presented above. Afterwards, Steps 6-8 are replaced by a dynamic programming approach, the idea of which we briefly want to sketch. One keeps mappings A_i for each of the separators $\mathcal{S}_i$ similar to the algorithm presented in the proof of Theorem 13. Again, one uses three distinct colors for the vertices in the separator $\mathcal{S}_i$:

- "black" (represented by 1, meaning that the vertex belongs to the dominating set),
- "white$_{\text{outer}}$" (represented by 0_{outer}, meaning that the vertex still needs to be dominated by the subgraph G_{i-1}), and
- "white$_{\text{inner}}$" (represented by 0_{inner}, meaning that the vertex still needs to be dominated by the subgraph G_i).

Note that an assignment of color 1 to some vertices $B \in \mathcal{S}_i$ already determines that vertices in $N(B) \cap \mathcal{S}_i$ are dominated. Hence, we only need to assign different white values to the remaining vertices in $\mathcal{S}_i \setminus N(B)$.

The mappings are updated starting from $i = 0$ up to $i = r - 1$ as follows. The new entry for a specific coloring $c^{(i+1)} \in \{1, 0_{\text{outer}}, 0_{\text{inner}}\}^{|\mathcal{S}_{i+1}|}$ in table A_{i+1} can be obtained from minimizing over all colorings $c^{(i)} \in \{1, 0_{\text{outer}}, 0_{\text{inner}}\}^{|\mathcal{S}_i|}$ in mapping A_i and applying Baker's algorithm to the graph G_i precolored according to $c^{(i)}$ and $c^{(i+1)}$. Working out the details of this approach yields a running time of $O(c^{\sqrt{k}}n)$ for some slightly better constant c than the one obtained in Theorem 31 (see [2]). However, we want to point out that the algorithm claimed by Baker is not worked out in detail for the DOMINATING SET problem, and, hence, it is hard to verify whether it can deal with precolored graphs, a basic assumption we need for this approach.

5 Variations of DOMINATING SET and the FACE COVER problem

For several variations of the DOMINATING SET problem on planar graphs, our technique can also help to obtain algorithms with a similar running time. In particular, we will consider the following problems: DOMINATING SET WITH PROPERTY P, WEIGHTED DOMINATING SET, and FACE COVER.

5.1 DOMINATING SET WITH PROPERTY P

In the following, a *property* P of a vertex set $V' \subseteq V$ of an undirected graph $G = (V, E)$ will be a Boolean predicate which yields true or false values when given as input V, E, and V'. Since V and E will always be clear from the context, we will simply write $P(V')$ instead of $P(V, E, V')$. Examples for such properties P are:

- V' is an independent set, i.e., the graph induced by V' has no edges, or
- the graph induced by V' is connected.

A *k-dominating set with property P* of an undirected graph G is a k-dominating set D of G which has the additional property $P(D)$ in G. The DOMINATING SET WITH PROPERTY P problem is the task to find a minimum size dominating set with property P. The k-DOMINATING SET WITH PROPERTY P problem is the task to decide, given a graph $G = (V, E)$, a property P, and a positive integer k, whether or not there exists a k-dominating set with property P.

Examples for such problems are:

- the k-INDEPENDENT DOMINATING SET problem, see [46, 47] or [23, p.464], where the property $P(D)$ of the k-dominating set D is that D is independent,
- the k-TOTAL DOMINATING SET problem, see [46, 47], where the property $P(D)$ of the k-dominating set D is that each vertex of D has a neighbor in D,
- the k-PERFECT DOMINATING SET problem [46, 47], where the property $P(D)$ of the k-dominating set D is that each vertex which is not in D has *exactly* one neighbor in D,
- the k-PERFECT INDEPENDENT DOMINATING SET problem, also known as the k-PERFECT CODE problem [23, 29, 46, 47], where the k-dominating set has to be perfect and independent, and
- the k-TOTAL PERFECT DOMINATING SET problem [46, 47], where the k-dominating set has to be total and perfect.

For all these instances, the condition of the following Theorem 32 holds and, hence, for these problems, we have an $O(c^{\sqrt{k}}n)$ time algorithm for some constant c. More precisely, a variant of Theorem 13 can be stated for each of these problems, leading to algorithms where the base q_i of the exponential term and the number λ_i of colors needed for the mappings in the dynamic programming are as follows:

- INDEPENDENT DOMINATING SET: $q_1 = 4$, $\lambda_1 = 3$;
  Here, in contrast to the algorithm given in the proof of Theorem 13, after each update of a mapping A_i for bag X_i, we check, for each coloring $c \in \{0, \hat{0}, 1\}^{n_i}$, if there exist two vertices $x, y \in X_i$ that both are assigned color 1 by c, and, if so, set $A_i(c) \leftarrow +\infty$.
- TOTAL DOMINATING SET: $q_2 = 5$, $\lambda_2 = 4$;
  Since one must also distinguish for the vertices in the domination set whether or not they have been dominated by other vertices from the dominating set, we may use 4 colors:

- 1, meaning that the vertex is in the dominating set and is already dominated;
- $\hat{1}$, meaning that the vertex is in the dominating set and still needs to be dominated;
- 0, meaning that the vertex is not in the dominating set and is already dominated;
- $\hat{0}$, meaning that the vertex is not in the dominating set and still needs to be dominated.

The corresponding partial ordering $\prec$ on $C := \{0, \hat{0}, 1, \hat{1}\}$, according to which our mappings will be monotonous, is given by $\hat{1} \prec 1$, $\hat{0} \prec 0$, and $d \prec d$ for all $d \in C$. The various steps of the algorithm for updating the mappings are similar the ones given in the algorithm of Theorem 13. The most cost-expensive part again is the updating of a JOIN NODE. Here, in the assignment (6), we have to adapt the definition of "divide" as follows: For a coloring $c = (c_1, \ldots, c_{n_i}) \in \{0, \hat{0}, 1, \hat{1}\}^{n_i}$ for X_i, we say that $c' = (c'_1, \ldots, c'_{n_i})$, $c'' = (c''_1, \ldots, c''_{n_i}) \in \{0, \hat{0}, 1, \hat{1}\}^{n_i}$ *divide* c if

1. $\big(c_t = 0 \ \Rightarrow \ (c'_t, c''_t \in \{0, \hat{0}\} \wedge c'_t \neq c''_t)\big)$, and
2. $\big(c_t = 1 \ \Rightarrow \ (c'_t, c''_t \in \{1, \hat{1}\} \wedge c'_t \neq c''_t)\big)$.

For a fixed coloring c, a combinatorial argument shows that the number of pairs that divide c, is given by

$$\sum_{\#_0(c)=0}^{n_i} \sum_{\#_1(c)=0}^{n_i - \#_0(c)} \binom{n_i}{\#_0(c)} \binom{n_i - \#_0(c)}{\#_1(c)} 2^{\#_0(c)} 2^{\#_1(c)} \quad = \quad 5^{n_i}.$$

This determines the running time of the algorithm.

- PERFECT DOMINATING SET: $q_3 = 4$, $\lambda_3 = 3$;
  Here, in contrast to the algorithm given in the proof of Theorem 13, a vertex is colored "grey" if it is dominated by exactly one "black" vertex which either lies in the "current" bag of the tree decomposition algorithm or in one of its child bags.

- PERFECT CODE: $q_4 = 4$, $\lambda_4 = 3$;
  This can be done by a combination of the arguments given for INDEPENDENT DOMINATING SET and PERFECT DOMINATING SET.

- TOTAL PERFECT DOMINATING SET: $q_5 = 5$, $\lambda_5 = 4$;
  This can be done by a combination of the arguments given for TOTAL DOMINATING SET and PERFECT DOMINATING SET.

Observe that our updating technique which makes strong use of the monotonicity of the mappings involved yields a basis in the exponential term of the running time which outperforms the results of Telle and Proskurowski. The corresponding constants q'_i for the above listed problems that were derived in [46, Theorem 4, Table 1], and [47, Theorem 5.7] are $q'_1 = 9$, $q'_2 = 16$, $q'_3 = 9$, $q'_4 = 9$, and $q'_5 = 16$.

Note that all the problems listed above are defined via properties P which can be viewed as being "local properties." This means that essential information for deciding P can be maintained by the algorithms based on the tree decomposition approach. On the other hand, the CONNECTED DOMINATING SET problem (where it is required that the dominating set which is looked for should be connected) does not seem to be solvable in this way, since connectedness cannot be decided on the basis of knowing only a subset of the dominating set.

All the problem variants listed above are known to be NP-complete problems [46, 47] for general graphs. As regards their parameterized complexity, it is only known that k-PERFECT CODE (on general graphs) is a W[2]-problem which is W[1]-hard, and k-INDEPENDENT DOMINATING SET (on general graphs) is W[2]-complete, see [23]. The other variants are not mentioned in [23]. In spite of these hardness results, we can obtain algorithms which have a running time of the form $O(c^{\sqrt{k}}n)$ for some constant c for all of the instances of k-DOMINATING SET WITH PROPERTY P when restricting oneself to planar graphs. More generally, we can state:

Theorem 32 *Suppose that there is an algorithm that solves in $O(q^{\ell}N)$ time the* DOMINATING SET WITH PROPERTY P *problem on graphs, given a tree decomposition with treewidth ℓ and N nodes for some constant q. Then, the k-*DOMINATING SET WITH PROPERTY P *problem can be solved in $O(q^{d\sqrt{k}}n)$ time, given a planar graph $G = (V, E)$, where $d = 6\sqrt{34}$. Moreover, if a k-dominating set with property P exists, a minimum size dominating set with property P can be constructed within the same time.*

Proof. If the planar graph G admits a dominating set with property P of size at most k, then, clearly, G has domination number at most k. By Theorem 29, the treewidth of G is bounded by $6\sqrt{34}\sqrt{k} + 8$. According to the discussion in Section 4, a corresponding tree decomposition with $O(n)$ nodes can be found in time $O(\sqrt{k}n)$. The assumption on the existence of an $O(q^{\ell}N)$ time algorithm for given tree decomposition of width ℓ then yields the claim. □

5.2 Weighted dominating set

Here, we consider the following variant of DOMINATING SET: Take a graph $G = (V, E)$ together with a positive integer weight function $w : V \to \mathbb{N}$ with $w(v) > 0$ for all $v \in V$. The weight of a vertex set $D \subseteq V$ is defined as $w(D) = \sum_{v \in D} w(v)$. A *$k$-weighted dominating set* D of an undirected graph G is a dominating set D

of G with $w(D) \leq k$. The k-WEIGHTED DOMINATING SET problem is the task to decide, given a graph $G = (V, E)$, a weight function $w : V \to \mathbb{N}$, and a positive integer k, whether or not there exists a k-weighted dominating set.

Theorem 33 *The k-WEIGHTED DOMINATING SET problem on planar graphs can be solved in time $O(c^{\sqrt{k}}n)$, where $c = 4^{6\sqrt{34}}$. Moreover, if a k-weighted dominating set exists, a dominating set of minimum weight can be constructed within the same time.*

Proof. By definition of the weight function, a graph which possesses a k-weighted dominating set has a domination number bounded by k. Hence, its treewidth is bounded by $6\sqrt{34k} + 8$, see Theorem 29. Now, the techniques explained in Theorem 31 based on Theorem 13 work accordingly. In particular, only small modifications in the bookkeeping technique used in Theorem 13 are necessary in order to obtain a k-weighted dominating set. More precisely, we have to adapt the initialization (1) of the mappings A_i for the bag $X_i = (x_{i_1}, \ldots, x_{i_{n_i}})$ according to:

`for all` $c = (c_1, \ldots, c_{n_i}) \in \{0, \hat{0}, 1\}^{n_i}$ `do`

$$A_i(c) \leftarrow \begin{cases} +\infty & \text{if } c \text{ is locally invalid for } X_i \\ w(c) & \text{otherwise,} \end{cases} \tag{8}$$

where $w(c) := \sum_{t=0, c_t=1}^{n_i} w(x_{i_t})$. The updating of the mappings A_i in Step 2 in the algorithm of Theorem 13 is adapted similarly. □

Also weighted variants of domination problems with property P can be treated in this way. Observe that it is *not* straightforward to generalize the theorem above to domination problems with rational or "real" weights. The natural reduction to the integer weight problem treated above would also influence the parameter k, which then would depend on the whole problem instance, which is not feasible in the fixed parameter setting. Only if we fix beforehand a constant ϵ such that all admissible positive weights are greater than ϵ, this reduction would be feasible, showing that of the corresponding generalized problem instance for k-WEIGHTED DOMINATING SET on planar graphs is fixed parameter tractable (cf. similar issues for WEIGHTED VERTEX COVER on general graphs [39]).

5.3 FACE COVER

We now turn our attention to the FACE COVER problem (see [10, 23, 44]). A *k-face cover* C of an undirected plane graph $G = (V, E)$ (i.e., a planar graph with a fixed embedding) is a set of faces that cover all vertices of G, i.e., for every vertex $v \in V$, there exists a face $f \in C$ so that v lies on the boundary of f. The k-FACE COVER problem is the task to decide, given a plane graph $G = (V, E)$ and a positive integer k, whether or not there exists a k-face cover. The size of a face cover of minimum size of a plane graph G is called the *face cover number* of

G. The FACE COVER problem is the task to find a minimum size face cover for a given plane graph.

Downey and Fellows [23, p.38] claimed an $O(12^k n)$ algorithm for k-FACE COVER, which they call FACE COVER NUMBER FOR PLANAR GRAPHS. Unfortunately, there is a flaw in the correctness proof of this algorithm.

Before, Bienstock and Monma [10] had derived an $O(c^k n)$ algorithm (with an unspecified but large constant c) for a generalization of FACE COVER which they call DISK DIMENSION problem.

Basically, Downey and Fellows reduce FACE COVER to a problem which they call PLANAR RED/BLUE DOMINATING SET. Actually, we will use the same reduction technique in the following. This justifies why we also briefly treat this auxiliary problem.

An instance of PLANAR RED/BLUE DOMINATING SET is given by a planar bipartite graph $G = (V, E)$, where the bipartition is given by $V = V_{red} \cup V_{blue}$. The parameter is k. The question is whether there exists a set $V' \subseteq V_{red}$ with $|V'| \leq k$ such that every vertex of V_{blue} is adjacent to at least one vertex of V'.

Observe that PLANAR RED/BLUE DOMINATING SET is *not* a variant of planar dominating set in the sense of the first subsection, because a solution V' is not a dominating set, since red vertices can not and hence need not be dominated by red vertices. Downey and Fellows [23, p. 38] claim an $O(12^k n)$ algorithm for PLANAR RED/BLUE DOMINATING SET. Unfortunately, the proof of [23, Lemma 3.3] (which is the basis of the given search tree algorithm) is flawed.

Although it would surely be possible to develop an analogue to Theorem 31 for PLANAR RED/BLUE DOMINATING SET as well, we will only state and prove the following version here, since we are mainly interested in solving the FACE COVER problem.

Lemma 34 *Let a (planar) bipartite graph $G = (V, E)$ with bipartition $V = V_{red} \cup V_{blue}$ be given together with a tree decomposition of width ℓ. Then,* RED/BLUE DOMINATING SET *can be solved in time $O(3^\ell N)$, where N is the number of bags of the tree decomposition.*

Proof. (Sketch)Basically, the technique exhibited in Theorem 13 can be applied. Due to the bipartite nature of the graph, only two "states" have to be stored for each vertex: red vertices are either within the dominating set or not (represented by colors 1_{red} and 0_{red}, respectively), and blue vertices are either already dominated or not yet dominated (represented by colors 0_{blue} and $\hat{0}_{\mathrm{blue}}$, respectively).

We consider our bags as bipartite sets, i.e., $X_i := X_{i,\mathrm{red}} \cup X_{i,\mathrm{blue}}$, where $X_{i,\mathrm{red}} := X_i \cap V_{\mathrm{red}}$ and $X_{i,\mathrm{blue}} := X_i \cap V_{\mathrm{blue}}$. Let $n_{i,\mathrm{red}} := |X_{i,\mathrm{red}}|$ and $n_{i,\mathrm{blue}} := |X_{i,\mathrm{blue}}|$, i.e., $|X_i| =: n_i = n_{i,\mathrm{red}} + n_{i,\mathrm{blue}}$.

The partial ordering $\prec$ on the color set $C = C_{\mathrm{red}} \cup C_{\mathrm{blue}}$, where $C_{\mathrm{red}} := \{1_{\mathrm{red}}, 0_{\mathrm{red}}\}$ and $C_{\mathrm{blue}} := \{0_{\mathrm{blue}}, \hat{0}_{\mathrm{blue}}\}$, is given by $\hat{0}_{\mathrm{blue}} \prec 0_{\mathrm{blue}}$ and $d \prec d$ for all

$d \in C$.

A *valid* coloring for X_i is a coloring where we assign colors from C_{red} to vertices in $X_{i,\text{red}}$ and colors from C_{blue} to vertices in $X_{i,\text{blue}}$. The various steps of the algorithm for updating the mappings are similar the ones given in the algorithm of Theorem 13.

Again, the most cost-expensive part is the updating of a JOIN NODE. For a correct updating of JOIN NODES, we adapt the definition of "divide" that appears in the assignment (6) according to: For a valid coloring $c = (c_1, \dots, c_{n_i}) \in C^{n_i}$ of X_i, we say that the valid colorings $c' = (c'_1, \dots, c'_{n_i})$, $c'' = (c''_1, \dots, c''_{n_i}) \in C^{n_i}$ *divide* c if

1. $\big(c_t \neq 0_{\text{blue}} \;\Rightarrow\; (c'_t, c''_t = c_t)\big)$, and
2. $\big(c_t = 0_{\text{blue}} \;\Rightarrow\; (c'_t, c''_t \in \{0_{\text{blue}}, \hat{0}_{\text{blue}}\} \wedge c'_t \neq c''_t)\big)$.

For a fixed valid coloring c that contains $z := \#_{0_{\text{blue}}}(c)$ many colors 0_{blue}, the number of pairs that divide c is 2^z. Since there are $2^{n_{i,\text{red}}}\binom{n_{i,\text{blue}}}{z}$ many colorings with $\#_{0_{\text{blue}}}(c) = z$, the total number of pairs that divide a fixed coloring c is upper bounded by

$$\sum_{z=0}^{n_{i,\text{blue}}} 2^{n_{i,\text{red}}} \binom{n_{i,\text{blue}}}{z} \cdot 2^z \quad = \quad 2^{n_{i,\text{red}}} \cdot 3^{n_{i,\text{blue}}} \quad \leq \quad 3^{n_i}.$$

This determines the running time of the algorithm. □

In order to solve the FACE COVER problem with the previously established techniques, we need the following auxiliary notion. Let $G = (V, E)$ be a plane graph. Let F denote the set of faces of G. A mapping $r : F \to V$ is called a *c-bounded face representation* iff for all $v \in r(F)$:

1. for all $f \in r^{-1}(v)$, v lies on the boundary of f, and
2. $|r^{-1}(v)| \leq c$.

Lemma 35 *Each plane graph has a* 5*-bounded face representation. Moreover, such a face representation function can be constructed in time* $O(n)$.

Proof. Let $G_0 = (V_0, E_0)$ be a plane graph. We are going to define a 5-bounded face representation r of G_0 in a step-by-step fashion.

It is well-known that each planar graph has a vertex of degree of at most five [40, Corollary 1.4]. Hence, select one of these low-degree-vertices of G_0 and call it v_0. Since v_0 has degree at most five, there are at most five faces adjacent to v_0. For all these faces f, we define $r(f) = v_0$.

Consider, then, the graph $G_1 = (V_1, E_1)$, where $G_1 = G_0 - v_0$. We assume the "same" planar embedding for G_1 as for G_0. As before, we can find a vertex $v_1 \in V_1$ with at most five adjacent faces (in G_1). Therefore, v_1 has at most five adjacent faces in G_0 to which no vertices have yet been assigned. For all these faces f, we define $r(f) = v_1$.

Inductively, G_{i+1} is obtained as $G_i - v_i$, where v_i has degree of at most five in G_i. Here, v_i represents all adjacent faces in G_0 which are not already represented by the previously selected vertices $v_0, \ldots, v_{i-1}$.

This loop is repeated until all faces f of G_0 have one representing vertex $r(f)$, i.e., until r is completely defined.

We obtain time bound $O(n)$ as follows. Maintain for every vertex its degree and maintain a data structure that contains all vertices of degree at most 5. Now, repeatedly, pick a vertex v from the data structure, and take that as the vertex in the iteration; update the degrees of its neighbors ($O(1)$ work) and add these to the data structure if their degree is at most 5. $\square$

The preceding lemma will be the key in the proof of the following theorem.

Theorem 36 *The k-FACE COVER problem can be solved in time $O(c_1^{\sqrt{k}} n)$, where $c_1 = 3^{36\sqrt{34}}$. Moreover, if the face cover number of the input graph is at most k, then a minimum size face cover can be constructed within the same time.*

Proof. Let $G = (V, E)$ be a plane graph with face set F. Due to Lemma 35, we can find a 5-bounded face representation $r : F \to V$ in time $O(n)$.

Consider the following graph: Add a vertex to each face of G, and make each such "face vertex" adjacent to all vertices that are on the boundary of that face. These are the only edges of the bipartite graph $G' = (V', E')$. Write $V' = V \cup V_F$, where V_F is the set of face vertices, i.e., each $v \in V_F$ represents a face f_v in G. In other words, V and V_F form the bipartition of G'. Observe that G' can be viewed as an instance of RED-BLUE DOMINATING SET: the face vertices V_F are "red" and the other vertices V are "blue." Basically, this is the idea Downey and Fellows indicated in [23, Exercise 3.1.5]. Now, we would like to apply Lemma 34 to finish the proof.

To this end, consider the graph $\hat{G} = (V, \hat{E})$ obtained from G' by contracting each edge connecting a face vertex $v \in V_F$ of G' with $r(f_v)$. In other words, $\hat{G}$ is obtained from G by firstly removing all "original" edges from E, and then inserting, for every $f \in F$, edges between $r(f)$ and each other vertex v adjacent to f in G.

Now, assume that G has a k-face cover. Then, $\hat{G}$ has a k'-dominating set with $k' \leq k$. Namely, if $C \subseteq F$ is a face cover of G, then $r(C)$ is a dominating set of $\hat{G}$. Therefore, Theorem 29 yields $\mathrm{tw}(\hat{G}) \in O(\sqrt{k})$. We consider one tree decomposition of $\hat{G}$ of width $\leq 6\sqrt{34k}$. In order to be able to apply Lemma 34, we are going to construct a tree decomposition of G' of width $\leq 36\sqrt{34k}$ from the tree decomposition of $\hat{G}$. To this end, we enhance the bags of $\hat{G}$'s tree decomposition according to the following rule: if $r(f)$ is in some bag for some face $f \in F$, then put all $v \in V_F$ into that bag which satisfies $r(f_v) = r(f)$. The reader should verify that this, indeed, yields a tree decomposition of G', and the claimed width bound follows from Lemma 35. □

Hence, our algorithm for FACE COVER leads to an asymptotic improvement of the result stated by Downey and Fellows [23, p. 38]. The DISK DIMENSION problem treated by Bienstock and Monma [10] generalizes FACE COVER in two ways: firstly, they do not start with a fixed embedding of the planar input graph and, secondly, they have an additional input of their problem, namely a set D of designated vertices, where only these need to be covered. Both of these generalizations seem to be hard to treat within our framework.

6 Conclusion

In this paper, we presented a treewidth-based approach to improve the fixed parameter complexity of k-DOMINATING SET, k-FACE COVER and related problems on planar graphs drastically—we gained an exponential improvement over previous exact solutions for the problems. Seemingly for the first time, our results provide fixed parameter algorithms whose exponential factor has an exponent sublinear in the parameter.

The results. We show that for the treewidth of planar graphs, we have $\mathrm{tw}(G) \in O(\sqrt{\gamma(G)})$, where $\gamma(G)$ is the domination number of G. From this, we derive that k-DOMINATING SET on planar graphs can be solved in time $O(c^{\sqrt{k}}n)$, where $c = 4^{6\sqrt{34}}$. Analogously, on planar graphs, k-INDEPENDENT DOMINATING SET, k-PERFECT DOMINATING SET, k-PERFECT CODE, and k-WEIGHTED DOMINATING SET can be solved in the same time bounds and k-TOTAL DOMINATING SET and k-TOTAL PERFECT DOMINATING SET can be solved in time $O(c_1^{\sqrt{k}}n)$, where $c_1 = 6^{6\sqrt{34}}$. Finally, the k-FACE COVER problem can be solved in time $O(c_2^{\sqrt{k}}n)$, where $c_2 = 3^{36\sqrt{34}}$.

The constants. As already discussed earlier in the paper, our proven constants are huge. Hence, our result might be of only structural importance. There is hope for getting our results to be more practical, however: Our analysis only deals with worst case bounds. Even these might be improved significantly. A matter of special importance in this context is to note that our bounds heavily depend on the layer decomposition of the given planar graph, which seems to be a good starting point for the fight for better constants. In summary, ameliorating the constants needs *and* deserves future work.

The philosophical matters. As our results suffer from bad constants, besides discussing the opportunities to improve these, it is also valuable to investigate the "philosophical" perspectives opened by our work. Does our result mean that

k-DOMINATING SET is an "easy" fixed parameter tractable problem, e.g., easier than (the classical parameterized problem) k-VERTEX COVER on general graphs? In theory, seemingly yes, because the asymptotics of the known exponential term for PLANAR k-DOMINATING SET is much better than for k-VERTEX COVER ("sub-linear versus linear"); in practice, seemingly no, because for probably all reasonable sizes of k (say, e.g., $k \approx 100$) the exponential term for k-VERTEX COVER (currently already smaller than 1.3^k [17, 38]) is much better. Does this imply something for the theory of fixed parameter tractability [23] as a whole? Cai and Juedes [15] very recently addressed this issue by stating that if VERTEX COVER on graphs of maximum degree 3 has a $2^{o(k)}n^{O(1)}$ algorithm, then $FPT = W[1]$. From this they conducted that there is no time $2^{o(\sqrt{k})}n^{O(1)}$ algorithm for k-DOMINATING SET on planar graphs unless $FPT = W[1]$. However, there was a flaw in the proof for these results. In a revised version [16] of [15] they give similar but weaker results stating that a time $2^{o(k)}n^{O(1)}$ algorithm for k-VERTEX COVER on graphs of maximum degree 3 or a time $2^{o(\sqrt{k})}n^{O(1)}$ algorithm for k-DOMINATING SET on planar graphs are impossible unless $\mathit{3SAT} \in \mathit{DTIME}(2^{o(n)})$, which is also considered to be unlikely but a weaker condition than $FPT \neq W[1]$. Hence, from a structural point of view, there is good evidence that k-VERTEX COVER on general graphs indeed is harder than k-DOMINATING SET on planar graphs.

The concrete open questions. It would be interesting to investigate experimentally the practical usefulness of our result, since our estimates for the constants are worst case and very pessimistic ones. It also would be interesting to see if these results can be extended to more variants of DOMINATING SET and to other graph classes (e.g., graphs of bounded genus[6]). Another interesting open problem is how to use the techniques of this paper for the variant of the FACE COVER problem, where the embedding is not given as an input (i.e., for a given planar graph, find an embedding with minimum number of faces that cover all the vertices, cf. [10]). Moreover, the question whether k-DOMINATING SET on planar graphs admits a problem kernel of small (linear?) size still remains unanswered.

Acknowledgments. We thank Frederic Dorn and Peter Rossmanith for discussions on the topic. This paper also profited from the comments of anonymous referees of the conference *SWAT 2000* and of *Algorithmica.*

References

[1] J. Alber, H. Fan, M. R. Fellows, H. Fernau, R. Niedermeier, F. Rosamond, and U. Stege. Refined search tree techniques for the PLANAR DOMINATING

[6]The logical framework of Frick and Grohe [27] already guarantees fixed parameter tractability for this problem.

SET problem. In *Proceedings 26th MFCS 2001*, Springer-Verlag LNCS 2136, pp. 111-122, 2001.

[2] J. Alber, H. Fernau, and R. Niedermeier. Parameterized complexity: exponential speed-up for planar graph problems. In *Proceedings 28th ICALP 2001*, Springer-Verlag LNCS 2076, pp. 261–272, 2001. Long version available as Technical Report TR01-023, Electronic Colloquium on Computational Complexity (ECCC), Trier, March 2001.

[3] J. Alber, H. Fernau, and R. Niedermeier. Graph separators: a parameterized view. In *Proceedings 7th COCOON 2001*, Springer-Verlag LNCS 2108, pp. 318-327, 2001. Long version available as Technical Report WSI-2001-8, Wilhelm-Schickard Institut für Informatik, Universität Tübingen.

[4] J. Alber, J. Gramm, and R. Niedermeier. Faster exact solutions for hard problems: a parameterized point of view. *Discrete Mathematics*, **229**: 3–27, 2001.

[5] J. Alber and R. Niedermeier. Improved tree decomposition based algorithms for domination-like problems. In *Proceedings 5th LATIN 2002*, Springer-Verlag LNCS, April 2002.

[6] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM* **45**:501–555, 1998.

[7] B. Aspvall, A. Proskurowski, and J. A. Telle. Memory requirements for table computations in partial k-tree algorithms. *Algorithmica* **27**: 382–394, 2000.

[8] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation.* Springer-Verlag, 1999.

[9] B. S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *Journal of the ACM* **41**:153–180, 1994.

[10] D. Bienstock and C. L. Monma. On the complexity of covering vertices by faces in a planar graph. *SIAM J. Comput.* **17**:53–76, 1988.

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

[12] H. L. Bodlaender. Treewidth: Algorithmic techniques and results. In *Proceedings 22nd MFCS'97*, Springer-Verlag LNCS 1295, pp. 19–36, 1997.

[13] H. L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theor. Comp. Sci.* **209**:1–45, 1998.

[14] A. Brandstädt, V. B. Le, and J. P. Spinrad. *Graph Classes: a Survey.* SIAM Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics, 1999.

[15] L. Cai and D. Juedes. Subexponential parameterized algorithms collapse the W-hierarchy. In *Proceedings 28th ICALP 2001*, Springer-Verlag LNCS 2076, pp. 273–284, 2001. Flawed. Corrected in [16].

[16] L. Cai and D. Juedes. On the existence of subexponential-time parameterized algorithms. Manuscript, November 2001.

[17] J. Chen, I. Kanj, and W. Jia. Vertex cover: further observations and further improvements. In *Proceedings 25th WG*, Springer-Verlag LNCS 1665, pp. 313–324, 1999. Revised version to appear in *Journal of Algorithms.*

[18] P. Crescenzi and V. Kann. A compendium of NP optimization problems. Available at http://www.nada.kth.se/theory/problemlist.html, August 1998.

[19] G. Di Battista, P. Eades, R. Tamassia, and I.G. Tollis. *Graph Drawing; Algorithms for the Visualization of Graphs*, Prentice Hall, 1999.

[20] R. Diestel. *Graph Theory*, Graduate Texts in Mathematics 173, Springer-Verlag, 1997.

[21] R. G. Downey and M. R. Fellows. Fixed-parameter tractability and completeness. *Congr. Num.* **87**:161–187, 1992.

[22] R. G. Downey and M. R. Fellows. Parameterized computational feasibility. In *P. Clote, J. Remmel (eds.): Feasible Mathematics II*, pp. 219–244. Birkhäuser, 1995.

[23] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer-Verlag, 1999.

[24] R. G. Downey, M. R. Fellows, and U. Stege. Parameterized complexity: A framework for systematically confronting computational intractability. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 49:49–99, 1999.

[25] U. Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM* **45**:634–652, 1998. update

[26] M. R. Fellows. Parameterized complexity: new developments and research frontiers. In R. Downey, D. Hirschfeldt (editors), *Aspects of Complexity*, pp. 51–72, De Gruyter, 2001.

[27] M. Frick and M. Grohe. Deciding first-order properties of locally tree-decomposable structures. In *Proceedings 26th ICALP'99*, Springer-Verlag LNCS 1644, pp. 331–340, 1999.

[28] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness.* Freeman, 1979.

[29] M. M. Halldórsson, J. Kratochvíl, and J. A. Telle. Independent sets with domination constraints. In *Proceedings 25th ICALP'98*, Springer-Verlag LNCS 1443, pp. 176–187, 1998.

[30] T. W. Haynes, S. T. Hedetniemi, and P. J. Slater. *Fundamentals of Domination in Graphs*. Monographs and textbooks in pure and applied Mathematics Vol. 208, Marcel Dekker, 1998.

[31] T. W. Haynes, S. T. Hedetniemi, and P. J. Slater (eds.). *Domination in Graphs; Advanced Topics*. Monographs and textbooks in pure and applied Mathematics Vol. 209, Marcel Dekker, 1998.

[32] D. S. Hochbaum (ed.). *Approximation algorithms for NP-hard problems.* PWS Publishing Company, 1997.

[33] A. Kanevsky. Finding all minimum-size separating vertex sets in a graph. *Networks* **23**: 533–541, 1993.

[34] T. Kloks. *Treewidth. Computations and Approximations*. Springer-Verlag LNCS 842, 1994.

[35] J. van Leeuwen. Graph Algorithms. In *Handbook of Theoretical Computer Science, Vol. A: Algorithms and Complexity*, pp. 525–631, North Holland, 1990.

[36] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *Journal of the ACM* **41**:960–981, 1994.

[37] K. Mehlhorn. *Graph Algorithms and NP-Completeness.* Springer-Verlag, 1984.

[38] R. Niedermeier and P. Rossmanith. Upper bounds for Vertex Cover further improved. In *Proceedings 16th STACS'99*, Springer-Verlag LNCS 1563, pp. 561–570, 1999.

[39] R. Niedermeier and P. Rossmanith. On efficient fixed parameter algorithms for Weighted Vertex Cover. In *Proceedings 11th ISAAC 2000*, Springer-Verlag LNCS 1969, pp. 180–191, 2000.

[40] T. Nishizeki and N. Chiba. Planar graphs: theory and applications. *Annals of Discr. Math.*, **32**, North-Holland, 1988.

[41] C. H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, **43**:425–440, 1991.

[42] C. H. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

[43] A. Paz and S. Moran. Nondeterministic polynomial optimization problems and their approximations. *Theor. Comp. Sci.* **15**:251–277, 1981.

[44] R. C. Read. Prospects for graph theory algorithms. *Ann. Discr. Math.* **55**:201–210, 1993.

[45] J. A. Telle. Complexity of domination-type problems in graphs. *Nordic J. Comput.* **1**:157–171, 1994.

[46] J. A. Telle and A. Proskurowski. Practical algorithms on partial k-trees with an application to domination-like problems. In *Proceedings 3rd WADS'93*, Springer-Verlag LNCS 709, pp. 610–621, 1993.

[47] J. A. Telle and A. Proskurowski. Algorithms for vertex partitioning problems on partial k-trees. *SIAM J. Discr. Math.* **10(4)**:529–550, 1997.