Electronic Colloquium on Computational Complexity, Revision 01 of Report No. 023 (2001)

Parameterized Complexity: Exponential Speed-Up for Planar Graph Problems*

Jochen Alber† Henning Fernau Rolf Niedermeier

Wilhelm-Schickard-Institut für Informatik, Universität Tübingen,

Sand 13, D-72076 Tübingen, Fed. Rep. of Germany

{alber,fernau,niedermr}@informatik.uni-tuebingen.de

Abstract

A parameterized problem is called *fixed parameter tractable* if it admits a solving algorithm whose running time on input instance (I, k) is $f(k) \cdot |I|^{\alpha}$, where f is an arbitrary function depending only on k. Typically, f is some exponential function, e.g., $f(k) = c^k$ for some constant c. We describe general techniques to obtain then growth of the form $f(k) = c^{\sqrt{k}}$ for a large variety of planar graph problems. The key to this type of algorithm is what we call the "Layerwise Separation Property" of a planar graph problem. Problems having this property include PLANAR VERTEX COVER, PLANAR INDEPENDENT SET, and PLANAR DOMINATING SET. Extensions of our speed-up technique to basically all fixed parameter tractable planar graph problems are also exhibited.

Moreover, on our way to design fixed parameter algorithms with sublinear exponents, we derive some theoretical results relating, e.g., the domination number or the vertex cover number, with the treewidth of a plane graph.

Keywords: Planar graph problems, fixed parameter tractability, parameterized complexity, tree decomposition, graph separators.

*A preliminary version of the paper appears in the Proceedings of the 28th International Colloquium on Automata, Languages and Programming (ICALP 2001), held in Crete, Greece, July 8–12, 2001.

†Supported by the Deutsche Forschungsgemeinschaft (research project PEAL (Parameterized complexity and Exact ALgorithms), NI 369/1-1).

ISSN 1433-8092

1 Introduction

While many problems of practical interest tend to be intractable from a standard complexity-theoretic point of view, in many cases such problems have natural "structural" parameters, and practically relevant instances are often associated with "small" values of these parameters. The notion of fixed parameter tractability [16] tries to capture this intuition. This is done by taking into account solving algorithms that are exponential with respect to the parameter, but otherwise have polynomial time complexity. That is, on input instance (I, k) one terms a (parameterized) problem *fixed parameter tractable* if it allows for a solving algorithm running in time $f(k)n^{O(1)}$, where f is an arbitrary function only depending on k and $n = |I|$. The associated complexity class is called FPT. As fixed parameter tractability explicitly allows for exponential time complexity concerning the parameter, the pressing challenge is to keep the related "combinatorial explosion" as small as possible. In this paper, we provide a general framework for NP-hard planar graph problems that allows us to go from typically time $c^k n^{O(1)}$ algorithms to time $c^{\sqrt{k}} n^{O(1)}$ algorithms (subsequently briefly denoted by "$c^{\sqrt{k}}$-algorithms"), meaning an exponential speed-up.[1] The main contributions of our work, thus, are

- to provide new results and a "structural breakthrough" for the parameterized complexity of a large class of problems,
- to parallel and complement results for the approximability of planar graph problems obtained by Baker [7],
- to methodize and extend previous work on concrete graph problems [1], and
- to systematically take care of the bases occurring in the (exponential terms of the) running times.

Fixed parameter tractability. Many hard computational problems have the general form: given an object I and a positive integer k, does I have some property that depends on k? For instance, the NP-complete VERTEX COVER problem is: given an undirected graph $G = (V, E)$ and k, does G

[1] Actually, whenever we can construct a so-called problem kernel of polynomial size in polynomial time (which is often the case for parameterized problems), then we can replace the term $c^{\sqrt{k}} n^{O(1)}$ by $c^{\sqrt{k}} k^{O(1)} + n^{O(1)}$.

have a vertex cover of size at most k? In parameterized complexity theory, this positive integer k is called the *parameter*. In many applications, the parameter k can be considered to be "small" in comparison with the size $|I|$ of the given object I. The basic observation of parameterized complexity is that for many hard problems, the seemingly inherent "combinatorial explosion" really can be restricted to a "small part" of the input, the parameter. For instance, VERTEX COVER allows for an algorithm with running time $O(1.3^k + kn)$ [13, 28]. To put it in more complexity-theoretic terms, consider the class of parameterized problems that can be solved in deterministic time $f(k)n^{O(1)}$, called FPT. The complexity class FPT is the set of *fixed parameter tractable* problems. Note that in the definition of FPT the function $f(k)$ may take unreasonably large values, e.g.,

$$2^{2^{2^{2^{2^{2^k}}}}}.$$

Thus, showing that a problem is a member of the class FPT does not necessarily bring along an efficient algorithm (not even for small k). Hence, the question naturally arises how "small" we can make the function $f(k)$ (see [5]). One direction in current research on parameterized complexity is to investigate problems with fixed parameter algorithms of running time $c^k n^{O(1)}$ and to try to get the constant c as small as possible (e.g., see [3] in the case of PLANAR DOMINATING SET). Getting small constant bases in the exponential factor $f(k)$ is also our concern, but, even more importantly, our primary focus is on investigations to get functions f (asymptotically) growing as slowly as possible. Doing so, we provide a general framework for a broad class of problems, namely planar graph problems. We indicate necessary conditions for planar graph problems that imply FPT-algorithms with running time $c^{\sqrt{k}}n^{O(1)}$ for constant c. Moreover, we discuss an extension of our technique which applies to basically all fixed parameter tractable graph problems.

Planar graph problems. Planar graphs build a natural and practically important graph class. Many problems that are NP-complete for general graphs (such as VERTEX COVER and DOMINATING SET) remain so when restricted to planar graphs. Whereas many NP-complete graph problems are hard to approximate in general graphs, Baker, in her well-known work [7], showed that many of them possess a polynomial time approximation scheme for planar graphs. That is, there is a polynomial time approximation algorithm with approximation factor $1+\epsilon$, where ϵ is a constant arbitrarily close

to 0. However, the degree of the polynomial grows with the quality of the approximation, i.e., with $1/\epsilon$. Hence, applying the approximation scheme does not necessarily lead to practical solutions. Alternatively, finding an "efficient" exact solution in "reasonable exponential time" is an interesting and promising research challenge. In particular, many graph problems seemingly fixed parameter intractable for general graph classes become fixed parameter tractable for planar graphs (cf. [16]).

Previous work and methodology. In recent work, algorithms were presented that constructively produce a solution for PLANAR DOMINATING SET and related problems in time $c^{\sqrt{k}}n$ [1, 2]. To obtain these results, it was proven that the treewidth of a planar graph with a dominating set of size k is bounded by $O(\sqrt{k})$ and that a corresponding tree decomposition can be found in time $O(\sqrt{k}n)$. Building on that problem-specific work with its rather tailor-made approach for dominating sets, here we take a much broader perspective. This comes to light by the following two main points. First, by introducing the so-called "Layerwise Separation Property" we provide an abstract, problem-independent tool for a fairly general design technique for $c^{\sqrt{k}}$-algorithms. Second, based on this key notion we give two main ways how to finally obtain these algorithms, one based on tree decompositions and the other one based on a bounded outerplanarity approach. Both these approaches do have their pros and cons, which will be discussed later on. (Please refer to Section 2 for a schematic picture of our methodology.) Note that, even though algorithms based on tree decompositions are widely considered to be impractical, because finding a tree decomposition, in general, is much too time-consuming, our approaches seem to provide first results where such decompositions of small width can be computed relatively quickly, hence, yielding efficient algorithms. The advantage of both approaches from a practioner's point of view clearly is that, since the algorithms developed here can be stated in a very general framework, only small parts have to be changed to adapt them to the concrete problem. In this sense, our work differs strongly from research directions, where running times of algorithms are improved in a very problem-specific manner (e.g., by extremely sophisticated case-distinctions as in the case of VERTEX COVER for general graphs). For example, once one can show that a problem has the so-called "Layerwise Separation Property," one can run a general algorithm which quickly computes a tree decomposition of guaranteed small width (independent of the concrete problem). In summary, the heart of our approach can roughly be

sketched as follows: If...

1. ...one can show that a graph problem carries some nice properties (e.g., the Layerwise Separation Property) and

2. ...one can determine some corresponding "problem-parameters" for these properties (e.g., the width and the size-factor of the Layerwise Separation Property).

Then, one gets an algorithm of running time $O(c^{\sqrt{k}}n^{O(1)})$, where we give concrete formulas on how to evaluate the constant c as a function of these problem-parameters.

Results. The main contribution of this work is to provide a general methodology for the design of $c^{\sqrt{k}}$-algorithms. A key to this is the notion of select&verify graph problems and the introduction of the Layerwise Separation Property (see Section 4) of such problems in connection with the concept of linear problem kernels (see Subsection 3.1). With these fundamental notions, we can derive our results. We show that problems that have the Layerwise Separation Property and admit either a tree decomposition based algorithm (cf., e.g., [31, 32]) or admit an algorithm based on bounded outerplanarity (cf. [7]), can be solved in time $c^{\sqrt{k}}n^{O(1)}$. For instance, these include PLANAR VERTEX COVER, PLANAR INDEPENDENT SET, PLANAR DOMINATING SET, or PLANAR EDGE DOMINATION and also variations of these, such as their weighted versions. Moreover, we give explicit formulas to determine the base c of the exponential term with respect to the problem specific parameters. For PLANAR VERTEX COVER, e.g., we obtain a time $O(2^{4\sqrt{3k}}n) \approx O(122^{\sqrt{k}})$ algorithm. The methods can be generalized in a way that basically all FPT-problems that admit tree-decomposition based algorithms can be attacked with our approach. A library containing implementations of various algorithms sketched in this paper is currently under development. It uses the LEDA package [26] for efficient data types and (graph) algorithms and the results obtained so far are encouraging.

Structure of the paper. In the next section, we sketch the overall structure of our methodology and point out how different parts fit together. Thus, in a sense, the figure provided there can also be seen as a road map of the whole paper. In Section 3, some fundamental, old and new definitions for our methods are provided. Our approach consists of two main phases. In Section 4, we describe the first phase, which deals with layerwise separation. In

Section 5, then we give (dynamic programming) algorithms for layerwisely separated graphs. In Section 6, we describe further extensions and implications of our approach, deviating from the "$c^{\sqrt{k}}$-viewpoint." Finally, in Section 7, we compare and evaluate the different strategies offered by our methodology and discuss prospects for future research.

2 Outline and overview

The approach of this paper is based on the so-called layer-model of a planar graph as introduced in Subsection 3.2. Using this model, one proceeds in two phases.

In a first phase, one separates the graph in a particular way ("layerwise"). The key property of a graph problem to allow such an approach will be the so-called "Layerwise Separation Property." The details herefore are presented in Section 4. It will be shown that such a property holds for quite a large class of graph problems including those which admit a linear problem kernel. This property assures that the planar graph can be separated nicely.

In a second phase, the problem is solved on the layerwisely separated graph. We present two independent ways to achieve this in Section 5. Either, using the separators to set up a tree decomposition of width $O(\sqrt{k})$ and solving the problem using this tree decomposition; or using a combination of a trivial approach on the separators and some algorithms working on graphs of bounded outerplanarity (see [7]) for the partitioned rest graphs. Figure 1 gives a general overview of our methodology presented in the following two sections.

As noted before, in the second phase (Section 5) we will describe two independent ways to solve the underlying graph problem on the layerwisely separated graph. Both the tree decomposition as well as the bounded outerplanarity approach do have their pros and cons, which is why we present both of them. As to the tree decomposition approach, its advantage is its greater generality (up to the tree decomposition it is the same for all graph problems). In particular, it is definitely easier to implement in practice, also due to its universality and mathematical elegance. We also like to mention in passing that our approach to tree decompositions results in a really practicable method, which usually is not to be expected when talking about algorithms using tree decompositions.

By way of contrast, as to the bounded outerplanarity approach, in some

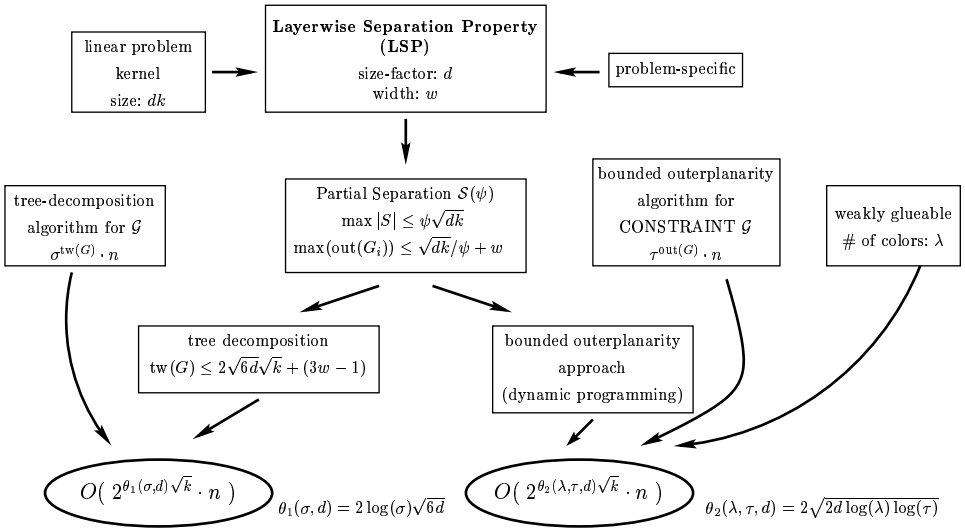

Figure 1: Road map of our methodology for planar graph problems.

cases we obtain better (theoretical worst case) time complexity bounds for our algorithms in comparison with the tree decomposition approach. Moreover, the space consumption is significantly smaller, because the tree decomposition approach works in its dynamic programming part with possibly large tables. To achieve this, however, we need more complicated formalism and more constraints concerning the underlying graph problems.

3 Basic definitions and preliminaries

We start with some basic notation used throughout the paper. We assume familiarity with elementary concepts of algorithms, complexity, and graph theory. We consider undirected graphs $G = (V, E)$, V denoting the vertex set and E denoting the edge set. For clarity, sometimes we refer to V by $V(G)$. Let $G[D]$ denote the subgraph induced by a vertex set $D \subseteq V$. We only consider simple (no double edges) graphs without self-loops. We study *planar* graphs, i.e., graphs that can be drawn in the plane without edge crossings. Let (G, ϕ) denote a *plane* graph, i.e., a planar graph G together with an embedding ϕ. A *face* of a plane graph is any topologically connected region surrounded by edges of the plane graph. The one unbounded face of a plane

graph is called the *exterior face*.

A *parameterized graph problem* is a language consisting of tuples (G, k), where G is a graph and k is an integer. A *parameterized graph problem on planar graphs* is a parameterized graph problem, where the graph G of an instance (G, k) is assumed to be a planar graph.[2] Among others, we study the following "graph numbers":

- A *vertex cover* C of a graph G is a set of vertices such that every edge of G has at least one endpoint in C; the size of a vertex cover set with a minimum number of vertices is denoted by $vc(G)$.
- An *independent set* of a graph G is a set of pairwise nonadjacent vertices; the size of an independent set with a maximum number of vertices is denoted by $is(G)$.
- A *dominating set* D of a graph G is a set of vertices such that each of the rest of the vertices in G has at least one neighbor in D; the size of a dominating set with a minimum number of vertices is denoted by $ds(G)$.

The corresponding problems are denoted by (PLANAR) VERTEX COVER, (PLANAR) INDEPENDENT SET, and (PLANAR) DOMINATING SET.

3.1 Linear problem kernels

Reduction to problem kernel is a core technique for the development of fixed parameter algorithms (see [16]). In a sense, the idea behind is to cut off the "easy parts" of a given problem instance such that only the "hard kernel" of the problem remains where, then, e.g., exhaustive search can be applied (with reduced costs). To get a problem kernel as small as possible is, therefore, a central goal from a practical as well as from a theoretical point of view.

More formally, reduction to problem kernel here is defined as follows.

Definition 1 Let $\mathcal{L}$ be a parameterized problem, i.e., $\mathcal{L}$ consists of pairs (I, k), where problem instance I has a solution of size k (the parameter).[3]

[2] If the instances of a parameterized graph problem on planar graphs were of the form $((G, \phi), k)$, where ϕ is an embedding of G, we would speak of a *parameterized graph problem on plane graphs*. However, in our setting, the planar graph G need not be given with an embedding.

[3] In this paper, we assume the parameter to be a positive integer, although, in general, it might also be from an arbitrary language (e.g., being a subgraph).

Reduction to problem kernel then means to replace problem (I, k) by a "reduced" problem (I', k') (which we call the *problem kernel*) such that

$$k' \leq d \cdot k, \quad |I'| \leq q(k')$$

with constant d,[4] polynomial q, and

$$(I, k) \in \mathcal{L} \text{ iff } (I', k') \in \mathcal{L}.$$

Furthermore, we require that the reduction from (I, k) to (I', k') (that we call *kernelization*) is computable in polynomial time $T_K(|I|, k)$.[5]

Clearly, concerning the development of *efficient* algorithms, it is highly desirable to have a fast reduction algorithm as well as a small size of the problem kernel I'. Usually, having constructed a size $k^{O(1)}$ problem kernel in time $n^{O(1)}$, one can improve the time complexity $f(k)n^{O(1)}$ of a fixed parameter algorithm to $f(k)k^{O(1)} + n^{O(1)}$. Subsequently, our focus is on decreasing $f(k)$, and we do not always refer to this simple fact. Often (cf. the subsequent example VERTEX COVER), the best one can hope for the problem kernel is size linear in k, a so-called *linear problem kernel*. For instance, using a theorem of Nemhauser and Trotter [27], (also cf. [8, 29]), Chen *et al.* [13] recently observed a problem kernel of size $2k$ for VERTEX COVER on general (not necessarily planar) graphs. According to the current state of knowledge, this is the best one could hope for because a problem kernel of size $(2 - \epsilon)k$ with constant $\epsilon > 0$ would probably imply a factor $2 - \epsilon$ polynomial time approximation algorithm for VERTEX COVER, which would mean a major breakthrough in approximation algorithms for VERTEX COVER [19]. As regards PLANAR VERTEX COVER, however, a problem kernel as small as $3k/2$ might be achievable, since there is a factor 3/2 approximation algorithm for PLANAR VERTEX COVER [8], as well as a polynomial time approximation scheme [7]. As a further example, note that due to the four color theorem for planar graphs and the corresponding algorithm generating a four coloring [30], it is easy to see that PLANAR INDEPENDENT SET has a problem

[4]Usually, $d \leq 1$. In general, however, it would even be allowed that $k' = g(k)$ for some arbitrary function g. For our purposes, however, we need that k and k' are linearly related. We are not aware of a concrete, natural parameterized problem with problem kernel where this is not the case.

[5]Again, one could allow for a more general definition here (i.e., allowing even an FPT reduction algorithm), but this would not fit our approach and we are not aware of a non-polynomial time problem kernelization.

kernel of size $4k$. If we are looking for an independent set of size $k \leq n/4$, then the four coloring algorithm basically does the job. Otherwise, we know that $k > n/4$, i.e., $n < 4k$, which gives a linear problem kernel.[6] In general, however, it is a reasonable and challenging task to try to construct a linear problem kernel. Ongoing work does this for PLANAR DOMINATING SET and it seems to require a rather complicated analysis.

Besides the positive effect of reducing the input size significantly by having small problem kernels, and all the already obvious consequences of that, this paper gives further justification, in particular, for the importance of linear problem kernels. The point is that once we have a linear problem kernel, e.g., for PLANAR VERTEX COVER or PLANAR INDEPENDENT SET, it is fairly easy to get $c^{\sqrt{k}}$-algorithms for these problems based upon the famous planar separator theorem [23, 24]. The constant factor in the problem kernel size directly influences the value of the exponential base and hence, lowering the kernel size means improved efficiency (see [4] for a detailed exposition). We will show alternative, more efficient ways (without using the planar separator theorem) of how to make use of linear problem kernels in a generic way in order to obtain $c^{\sqrt{k}}$-algorithms for planar graph problems.

3.2 Tree decomposition and layer decomposition

In this subsection, we briefly want to introduce two sorts of decompositions for a graph G, which are a fundamental basis for our approaches. Their use will become clear later on.

Definition 2 A *tree decomposition* of a graph $G = (V, E)$ is a pair $\langle \{X_i \mid i \in I\}, T \rangle$, where $X_i \subseteq V$ is called a *bag* and T is a tree with the elements of I as nodes, such that the following 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 $X_i \cap X_k \subseteq X_j$.

[6] Of course, to some extent this is kind of cheating, because in fact that means that the parameter is rather large, contradicting the very basic assumption of parameterized complexity. Hence, it would be more reasonable to look at the parameterized problem that asks for a independent set of size $n/4+k$ in the spirit of "parameterizing above guaranteed values" (cf. [25]).

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

Details on tree decompositions can be found in [10, 11, 21].

Let $G = (V, E)$ be a planar graph. The vertices of G can be decomposed according to the level of the "layer" in which they appear in an embedding ϕ, see [1, 7].

Definition 3 Let $(G = (V, E), \phi)$ be a plane graph.

a) The *layer decomposition* of (G, ϕ) is a disjoint partition of the vertex set V into sets $L_1, \ldots, L_r$, which are recursively defined as follows:

- L_1 is the set of vertices on the exterior face of G.
- L_i is the set of vertices on the exterior face of $G[V - \bigcup_{j=1}^{i-1} L_j]$ for all $i = 2, \ldots r$.

We will denote the layer decomposition of (G, ϕ) by

$$\mathcal{L}(G, \phi) := (L_1, \ldots, L_r).$$

b) The set L_i is called the *ith layer* of (G, ϕ).

c) The (uniquely defined) number r of different layers is called the *outerplanarity* of (G, ϕ), denoted by $\mathrm{out}(G, \phi) := r$.

d) We define $\mathrm{out}(G)$ to be the smallest outerplanarity possible among all plane embeddings, i.e., minimizing over all plane embeddings ϕ of G we set

$$\mathrm{out}(G) := \min_{\phi} \mathrm{out}(G, \phi).$$

Due to technical reasons, for a layer-decomposition $\mathcal{L}(G, \phi) := (L_1, \ldots, L_r)$, we set $L_i := \emptyset$ for all indices $i < 1$ and $i > r$.

Computing the layers of a plane graph can be done efficiently. For a detailed description of the algorithm, see [2, Section 2.3]:

Proposition 4 *Let $(G = (V, E), \phi)$ be a plane graph. The layer decomposition $\mathcal{L}(G, \phi) = (L_1, \ldots, L_r)$ can be computed in time $O(|V|)$.*

3.3 Algorithms based on separators in graphs

One of the most useful algorithmic techniques for solving computational problems is divide and conquer. To apply this method to planar graphs, we need graph separators and related notions.

To simplify notation, we use the notion $A+B$ for the disjoint union of two sets A and B. Graph separators are defined as follows.

Definition 5 Let $G = (V, E)$ be an undirected graph. A *separator* $S \subseteq V$ of G partitions V into two sets A_1 and A_2 such that

- $V = A_1 + S + A_2$ and
- no edge joins vertices in A_1 and A_2.

The triple (A_1, S, A_2) is called a *separation* of G, and the graphs $G[A_i]$ are called the *graph chunks* of the separation. Given a separation (A_1, S, A_2), we use the shorthands $\delta A_i := A_i + S$ for $i = 1, 2$.

In general, of course, A_1, A_2 and S will be non-empty. In order to cover boundary cases in some considerations below, we did not put this into the separator definition. In particular, we will also consider *u-v-separators* with $u, v \in V$, where we demand that such a separator cuts every path from u to v in G.

Already Lipton and Tarjan [23, 24] used their famous separator theorem in order to design algorithms with running time of $O(c^{\sqrt{n}})$ for certain planar graph problems, e.g., for glueable graph problems (for a formal definition see [4]). This naturally implies that, in the case of parameterized planar graph problems for which a linear kernel is known, algorithms with running time

$$O(c^{\sqrt{k}} + T_K(n, k))$$

can be derived, where $T_K(n, k)$ is the time to construct the problem kernel of an n-vertex input graph. As worked out in [4], it is possible to get an

$$O(2^{1.97\sqrt{dk}/(1-\sqrt{2/3+\epsilon})} + T_K(n, k))$$

algorithm for glueable planar graph problems with problem kernel of size dk, where $\epsilon \in (0, 1/3)$ can be chosen freely.[7] Unfortunately, the derived upper

[7] The constants 1.97 and 2/3 appearing in the exponent of the running time result from the best known $\sqrt{\cdot}$-separator theorem due to Djidjev and Venkatesan [15].

bound on the running time is only valid if $k \geq (1.97)^2/(d\epsilon^2)$. This means that, for "small" parameter values—as usually supposed for fixed parameter algorithms—, the involved constants in this algorithm get very bad. Even if we can assume $\epsilon = 0$ (this is feasible due to the nice properties of VERTEX COVER) we obtain an exponential growth of $c^{\sqrt{k}}$ with $c = 2^{1.97\sqrt{2}} \approx 37181$ for PLANAR VERTEX COVER, where $d = 2$ is known from [13, 27]. As worked out in [4], we can obtain divide and conquer algorithms with an exponential growth of $8564^{\sqrt{k}}$ by using separator theorems in a more clever way.

We will see algorithms with much better constants in this paper. For example, in the case of PLANAR VERTEX COVER, we obtain an algorithm with an exponential growth of $2^{4\sqrt{3k}} \approx 122^{\sqrt{k}}$. In addition, the advantages of the approach pursued in this paper also lies in weaker assumptions. Thus, in some cases we may drop requirements such as linear problem kernels by replacing it with the so-called "Layerwise Separation Property," a seemingly less restrictive demand.

4 Phase 1: Layerwise separation

In this section, we will exploit the layer-structure of a plane graph in order to gain a "nice" separation of the graph. It is important that a "yes"-instance (G, k) (where G is a plane graph) of the graph problem $\mathcal{G}$ admits a so-called "layerwise separation" of small size. By this, we mean, roughly speaking, a separation of the plane graph G (i.e., a collection of separators for G), such that each separator is contained in the union of constantly many subsequent layers (see conditions 1 and 2 of the following definition). For (fixed parameter) algorithmic purposes, it will be important that the corresponding separators are "small" (see condition 3 of the definition).

Definition 6 Let $(G = (V, E), \phi)$ be a plane graph of outerplanarity $r := \mathrm{out}(G, \phi)$, and let $\mathcal{L}(G, \phi) = (L_1, \ldots, L_r)$ be its layer decomposition. A *layerwise separation of width w and size s* of (G, ϕ) is a sequence $(S_1, \ldots, S_r)$ of subsets of V, with the properties that, for $i = 1, \ldots, r$:[8]

1. $S_i \subseteq \bigcup_{j=i}^{i+(w-1)} L_j$,
2. S_i separates layers L_{i-1} and L_{i+w},

[8]By default, we let $S_i := \emptyset$ for all $i < 1$ and $i > r$.

3. $\sum_{j=1}^{r} |S_j| \leq s$.

The crucial property that makes the algorithms developed in this paper work is what we call the "Layerwise Separation Property."

Definition 7 A parameterized problem $\mathcal{G}$ for planar graphs is said to have the *Layerwise Separation Property* (abbreviated by: LSP) of width w and size-factor d if for each $(G,k) \in \mathcal{G}$ and every planar embedding ϕ of G, the plane graph (G,ϕ) admits a layerwise separation of width w and size dk.

4.1 How can layerwise separations be obtained?

The LSP can be shown directly for many parameterized graph problems.

Example 8

1. As an example, consider PLANAR VERTEX COVER. Here, we get constants $w = 2$ and $d = 2$. In fact, for $(G,k) \in$ VERTEX COVER (and any planar embedding ϕ of G) with a "witnessing" vertex cover V' of size k, the sets $S_i := (L_i \cup L_{i+1}) \cap V'$ form a layerwise separation, given the layer decomposition $\mathcal{L}(G,\phi) = (L_1, \ldots, L_r)$.

2. In [1], the non-trivial fact is proven that for PLANAR DOMINATING SET, this property holds, yielding constants $w = 3$ and $d = 51$.[9]

A large class of parameterized graph problems for which the LSP holds is given whenever there exists a reduction to a linear problem kernel.

Lemma 9 *Let $\mathcal{G}$ be a parameterized problem for planar graphs that admits a problem kernel of size dk. Then, the parameterized problem $\mathcal{G}'$ where each instance is replaced by its problem kernel has the LSP of width 1 and size-factor d.*

Proof. Let $(G',k') \in \mathcal{G}'$ with $k' \leq dk$ be the problem kernel of $(G,k) \in \mathcal{G}$, and let $\mathcal{L}(G',\phi') = (L'_1, \ldots, L'_{r'})$ be the layer decomposition of (G',ϕ') (where ϕ' is any embedding). Let $r' = \mathrm{out}(G',\phi')$. Observe that $r' \leq \frac{dk}{3}$ since each layer has to consist of at least 3 vertices. Then, clearly, the sequence $S_i := L'_i$

[9] Note that in the case of PLANAR DOMINATING SET a construction of this form (i.e., obtaining the separators S_i by intersecting a "witnessing" dominating set V' of G with a sequence of subsequent layers, e.g, $S_i := (L_{i-1} \cup L_i \cup L_{i+1}) \cap V'$), does not fulfill the conditions of a layerwise separation, since, in general, S_i need not be a separator.

for $i = 1, \ldots, r'$ is a layerwise separation of width 1 and size dk of (G', ϕ'). □

Example 10 1. With Lemma 9 and the size $2k$ problem kernel for VERTEX COVER (see Subsection 3.1), we derive that PLANAR VERTEX COVER has the LSP of width 1 and size-factor 2 (which is even better than what was shown in Example 8).

2. Using the $4k$ problem kernel for PLANAR INDEPENDENT SET, we see that this problem has the LSP of width 1 and size-factor 4 on the set of reduced instances.

3. Since we are (not yet) aware of a linear problem kernel for the PLANAR DOMINATING SET problem, there seems, at the current state of research, to be not yet a way of proving the LSP for this problem simply by using the Lemma 9. Instead, we still rely on a direct proof as mentioned in Example 8.

4.2 What are layerwise separations good for?

The idea of the following is that, from a layerwise separation of small size (say bounded by $O(k)$), we are able to choose a set of separators such that their size is bounded by $O(\sqrt{k})$ and—at the same time—the subgraphs into which these separators cut the original graph have outerplanarity bounded by $O(\sqrt{k})$. In order to formalize such a choice of separators from a layerwise separation, we give the following definition.

Definition 11 Let $(G = (V, E), \phi)$ be a plane graph with layer decomposition $\mathcal{L}(G, \phi) = (L_1, \ldots, L_r)$. A *partial layerwise separation of width w* is a sequence $\mathcal{S} = (S_1, \ldots, S_q)$ such that there exist $i_0 = 1 < i_1 < \ldots < i_q < r = i_{q+1}$ such that for $i = 1, \ldots, q$:[10]

1. $S_j \subseteq \bigcup_{\ell=i_j}^{i_j+(w-1)} L_\ell$,

2. $i_j + w \leq i_{j+1}$ (so, the sets in $\mathcal{S}$ are pairwise disjoint) and

3. S_j separates layers L_{i_j-1} and L_{i_j+w}.

[10] Again, by default, we set $S_i := \emptyset$ for $i < 1$ and $i > q$.

The sequence $\mathcal{C}_{\mathcal{S}} = (G_0, \ldots, G_q)$ with

$$G_j := G[(\bigcup_{\ell=i_j}^{i_{j+1}+(w-1)} L_\ell) - (S_j \cup S_{j+1})], \quad j = 0, \ldots, q,$$

is called the *sequence of graph chunks* obtained by $\mathcal{S}$.

With this definition at hand, we can state the key result needed to establish the algorithms that will be presented in Subsection 5.

Proposition 12 *Let $(G = (V, E), \phi)$ be a plane graph that admits a layerwise separation of width w and size dk. Then, for every $\psi \in \mathbb{R}_+$, there exists a partial layerwise separation $\mathcal{S}(\psi)$ of width w such that*

1. $\max_{S \in \mathcal{S}(\psi)} |S| \leq \psi\sqrt{dk}$ *and*
2. $\mathrm{out}(H) \leq \frac{\sqrt{dk}}{\psi} + w$ *for each graph chunk H in $\mathcal{C}_{\mathcal{S}(\psi)}$.*

Moreover, there is an algorithm with running time $O(\sqrt{k}n)$ which, for a given ψ,

- *recognizes whether (G, ϕ) admits a layerwise separation of width w and size dk and, if so, computes $\mathcal{S}(\psi)$;*
- *computes a partial layerwise separation of width w that fulfills the conditions above.*

Proof. For $m = 1, \ldots, w$, consider the integer sequences $I_m = (m + jw)_{j=0}^{\lfloor r/w \rfloor - 1}$ and the corresponding sequences of separators $\mathcal{S}_m = (S_i)_{i \in I_m}$. Note that each $\mathcal{S}_m$ is a sequence of pairwise disjoint separators. Since $(S_1, \ldots, S_r)$ is a layerwise separation of size dk, this implies that there exists a $1 \leq m' \leq w$ with

$$\sum_{i \in I_{m'}} |S_i| \leq \frac{dk}{w}. \tag{1}$$

For a given ψ, let $s := \psi\sqrt{dk}$. Define $\mathcal{S}(\psi)$ to be the subsequence of $\mathcal{S}_{m'}$ such that $|S| \leq s$ for all $S \in \mathcal{S}(\psi)$, and $|S| > s$ for all $S \in \mathcal{S}_{m'} - \mathcal{S}(\psi)$. This yields condition 1. As to condition 2, suppose that $\mathcal{S}(\psi) = (S_{i_1}, \ldots, S_{i_q})$. How many layers are two separators S_{i_j} and $S_{i_{j+1}}$ apart? Herefore, note that the

number of separators in $\mathcal{S}_{m'}$ that appear between S_{i_j} and $S_{i_{j+1}}$ is $(i_{j+1}-i_j)/w$. Since all of these separators have size greater than s, their number has to be bounded by dk/ws, see Equation (1). Therefore, we get $i_{j+1} - i_j \leq \sqrt{dk}/\psi$ for all $j = 1, \ldots, q-1$. Hence, the chunks $G[(\bigcup_{\ell=i_j}^{i_{j+1}+w-1} L_\ell) - (S_{i_j} \cup S_{i_{j+1}})]$ have outerplanarity at most $\sqrt{dk}/\psi + w$.

The algorithm that computes a partial layerwise separation $\tilde{\mathcal{S}}$ proceeds as follows: For given ψ, compute $s := \psi\sqrt{dk}$. Then, for $j = 1, \ldots, r-w$, one checks whether the graph $\tilde{G}_j(v_s, v_t)$ admits a v_s-v_t–separator $\tilde{S}_j$ of size at most s. Here, $\tilde{G}_j(v_s, v_t)$ is the graph $G[\bigcup_{\ell=j}^{j+(w-1)} L_\ell]$ with two further vertices v_s and v_t and edges from v_s to all vertices in L_j and from v_t to all vertices in L_{j+w-1}. The separator $\tilde{S}_j$ can be computed in time $O(s \cdot n)$ using techniques based on maximum flow (see [20] for details).

Let $\tilde{\mathcal{S}} = (\tilde{S}_1, \ldots, \tilde{S}_q)$ be the sequence of all separators of size at most s found in this manner.[11] Suppose that $\tilde{S}_j \subseteq \bigcup_{\ell=i_j}^{i_j+(w-1)} L_\ell$ for some indices $1 \leq i_1 < \ldots < i_q \leq r$. Note that, by the arguments given above, no two such separators can be more than $\sqrt{dk}/\psi$ layers apart. Hence, if there was a j_0 such that $i_{j_0+1} - i_{j_0} > \sqrt{dk}/\psi$, the algorithms exits and returns "no." Otherwise, $\tilde{\mathcal{S}}$ is a partial layerwise separation of width w. □

In what follows, the positive real number ψ of Proposition 12 is also called *trade-off parameter*. This is because it allows us to optimize the trade-off between outerplanarity and separator size.

On the one hand, Proposition 12 will help to construct a tree decomposition of treewidth $\mathrm{tw}(G) = O(\sqrt{k})$, assuming that a layerwise separation of some constant width and size dk exists. Hence, for graph problems fulfilling this assumption and, moreover, allowing a time $\sigma^{\mathrm{tw}(G)}n$ algorithm when the graph is given together with a tree decomposition, we obtain a solving algorithm with running time $c^{\sqrt{k}}n$. This aspect will be outlined in Subsection 5.1.

On the other hand, similar running times can be achieved for and, moreover, can be solved in time $\tau^{\mathrm{out}(G)}n$ for planar graphs of bounded outerplanarity. Such type of problems can be found in [7].

Both approaches will be dealt with in detail in Section 5.

[11] Possibly, the separators $\tilde{S}_j$ in $\tilde{\mathcal{S}}$ found by the algorithm may differ from the separators in $\mathcal{S}(\psi)$.

5 Phase 2: Algorithms on layerwisely separated graphs

After Phase 1, we are left with a set of disjoint (layerwise) separators of size $O(\sqrt{k})$ separating the graph in components, each of which having outerplanarity bounded by $O(\sqrt{k})$. We now present two different ways how to obtain, in a second phase, a $c^{\sqrt{k}}$-algorithm that makes use of this separation. In both cases, there is the trade-off parameter ψ from Proposition 12 that can be used to optimize the running time of the resulting algorithms.

5.1 Using tree decompositions

We use the concept of tree decompositions as, e.g., described in [10, 21] (also cf. Subsection 3.2). We will show how the existence of a layerwise separation of small size helps to constructively obtain a tree decomposition of small width. The following result can be found in [22, Table 2, page 550] or [11, Theorem 83]. The corresponding algorithm is outlined in [2, Theorem 12].

Proposition 13 *For a plane graph (G, ϕ), we have* $\mathrm{tw}(G) \leq 3 \cdot \mathrm{out}(G) - 1$. *Such a tree decomposition can be found in $O(\mathrm{out}(G) \cdot n)$ time.*

With this proposition at hand, we can prove our central result in this context.

Theorem 14 *Let (G, ϕ) be a plane graph that admits a layerwise separation of width w and size dk. Then, we have* $\mathrm{tw}(G) \leq 2\sqrt{6dk} + (3w - 1)$. *Such a tree decomposition can be computed in time $O(k^{3/2}n)$.*

Proof. By Proposition 12, for each $\psi \in \mathbb{R}_+$, there exists a partial layerwise separation $\mathcal{S}(\psi) = (S_1, \ldots, S_q)$ of width w with corresponding graph chunks $\mathcal{C}_{\mathcal{S}(\psi)} = (G_0, \ldots, G_q)$, such that $\max_{S \in \mathcal{S}(\psi)} |S| \leq \psi\sqrt{dk}$ and $\mathrm{out}(G_i) \leq \frac{\sqrt{dk}}{\psi} + w$ for all $i = 0, \ldots, q$. The algorithm that constructs a tree decomposition $\mathfrak{X}_\psi$ is given as follows:

- Construct a tree decomposition $\mathcal{X}_i$ of width at most $3\,\mathrm{out}(G_i) - 1$ for each of the graphs G_i (using the algorithm from Proposition 13).
- Add S_i and S_{i+1} to every bag in $\mathcal{X}_i$ $(i = 0, \ldots, q)$.
- Let T_i be the tree of $\mathcal{X}_i$. Then, successively add an arbitrary connection between the trees T_i and T_{i+1} in order to obtain a tree T.

It is easy to show (see [1, Proposition 4]) that the tree T, together with the constructed bags, gives a tree decomposition of G. Clearly, its width $\mathrm{tw}(\mathfrak{X}_\psi)$ is upperbounded by

$$\begin{aligned}\mathrm{tw}(\mathfrak{X}_\psi) &\leq 2 \max_{S\in\mathcal{S}(\psi)} |S| + \max_{i=0,\dots,q} \mathrm{tw}(G_i) \\ &\leq 2 \max_{S\in\mathcal{S}(\psi)} |S| + 3(\max_{i=0,\dots,q} \mathrm{out}(G_i)) - 1 \\ &\leq (2\psi + 3/\psi)\sqrt{dk} + (3w-1).\end{aligned}$$

This upper bound is minimized for $\psi = \sqrt{3/2}$. Therefore, $\mathrm{tw}(\mathfrak{X}_\psi) \leq 2\sqrt{6dk}+ (3w-1)$. □

By [6, Proposition 4.5], a graph G that has no K_h minor has treewidth bounded by $h^{3/2}\sqrt{n}$. In particular, this implies that a planar graph has treewidth bounded by $11.2\sqrt{n}$. In the case of the existence of linear problem kernels for a given graph problem $\mathcal{G}$, this method might be used in order to obtain $c^{\sqrt{k}}$-algorithms. From our results, we can derive upper bounds of the treewidth of a planar graph in terms of several graph specific numbers. As the reader may verify, these problem-specific treewidth bounds tend to outperform the numbers obtainable via [6, Proposition 4.5]. For example, Theorem 14 and Example 8 imply the following inequalities for a planar graph G, relating the treewidth with the vertex cover and dominating set number:

$$\begin{aligned}\mathrm{tw}(G) &\leq 4\sqrt{3\,vc(G)} + 5, \quad \text{and} \\ \mathrm{tw}(G) &\leq 6\sqrt{34\,ds(G)} + 8.\end{aligned}$$

Note that for general graphs, no relation of the form

$$\mathrm{tw}(G) \leq f(ds(G))$$

(for any function f) holds; consider, e.g., the clique K_n with n vertices, where $\mathrm{tw}(K_n) = n-1$, but $ds(K_n) = 1$. For VERTEX COVER, only the linear relation

$$\mathrm{tw}(G) \leq vc(G)$$

can be easily shown. This estimate is sharp (which becomes clear by, again, considering the graph K_n, where $vc(K_n) = n-1$).

In addition, Theorem 14 yields a $c^{\sqrt{k}}$-algorithm for certain graph problems.

Theorem 15 *Let $\mathcal{G}$ be a parameterized problem for planar graphs. Suppose that*

1. *$\mathcal{G}$ has the LSP of width w and size-factor d, and*
2. *there exists a time $\sigma^{\ell}n$ algorithm that decides $(G,k) \in \mathcal{G}$, if G is given together with a tree decomposition of width ℓ.*

Then, there is an algorithm to decide $(G,k) \in \mathcal{G}$ in time $O(\sigma^{3w-1} \cdot 2^{\theta_1(\sigma,d)\sqrt{k}}n)$, where $\theta_1(\sigma,d) = 2(\log \sigma)\sqrt{6d}$.

Proof. Given an instance (G,k), in linear time we can compute some planar embedding ϕ of G (for details see [14]). In time $O(\sqrt{k}n)$ (see Proposition 12), we can check whether the plane graph (G,ϕ) admits a layerwise separation of width w and size dk.

If so, the algorithm of Theorem 14 computes a tree decomposition of width at most $2\sqrt{6dk} + (3w-1)$, and we can decide $(G,k) \in \mathcal{G}$ by using the given tree decomposition algorithm in time $O(\sigma^{2\sqrt{6dk}+(3w-1)}n)$.

If (G,ϕ) does not admit such a layerwise separation, we know that $(G,k) \notin \mathcal{G}$, by definition of the LSP. □

Example 16 Going back to our running examples, it is well-known that VERTEX COVER and INDEPENDENT SET admit such a tree decomposition based algorithm with $\sigma = 2$ and, in the case of DOMINATING SET, with $\sigma = 4$.

1. For PLANAR VERTEX COVER, by Example 8.1 we have seen that the LSP of width 1 and size-factor $d = 2$ holds. Hence, Theorem 15 guarantees an $O(2^{4\sqrt{3k}}n)$ algorithm for this problem.
2. For PLANAR INDEPENDENT SET, we have a linear problem kernel of size $4k$, hence, the LSP of width 1 and size-factor $d = 4$ holds, which yields an $O(2^{4\sqrt{6k}}n)$ algorithm.
3. By Example 8.2 we obtain the result from [1], namely, an $O(4^{6\sqrt{34k}}n)$ algorithm for PLANAR DOMINATING SET.

In this subsection, we have seen that, for plane graphs, the notion of the LSP gives us a sufficient condition to upperbound the treewidth of the "yes"-instance graphs of a problem. Moreover, this property led to fast

computations of the corresponding tree decompositions. All in all, we came up with algorithms of running time $O(c^{\sqrt{k}}n)$ for a wide class of problems.

The next subsection aims to show similar results in a different context.

5.2 Using bounded outerplanarity

We now turn our attention to certain parameterized graph problems for which we know that a solving algorithm of linear running time on the class of graphs of bounded outerplanarity exists. This issue was addressed in [7]; several examples can be found therein. In this subsection, we introduce the so-called "select&verify" problems and examine how, in this context, the notion of the LSP will lead to $c^{\sqrt{k}}$-algorithms. Since this will be a purely separator based approach, we will restrict ourselves to parameterized graph problems that can be solved easily on separated graphs. We will introduce the notion of weakly glueable select&verify problems in a first paragraph and present the design of $c^{\sqrt{k}}$-algorithms for these problems afterwards (see Paragraph 5.2.2).

5.2.1 Select&verify problems and weak glueability

For the approach outlined in this section, we want to describe necessary properties for graph problems that allow for separator based algorithms. The key property will be what we call "weak glueability" for select&verify problems. The notion of select&verify graph problems as introduced in the next paragraph is exactly the same as in [4]. There, one also finds the more restrictive concept of "glueability," which is a property tailored for divide and conquer algorithms using graph separators. In our setting, however, we are interested in an iterative dynamic programming approach for which a weaker form of "glueability" is sufficient.

Select&verify graph problems

Definition 17 A set $\mathcal{G}$ of tuples (G, k), G an undirected graph with vertex set $V = \{v_1, \ldots, v_n\}$ and k a positive real number, is called a *select&verify (graph) problem* if there exists a pair $(P., \mathrm{opt})$ with $\mathrm{opt} \in \{\min, \max\}$, such that $P.$ is a function that assigns to G a polynomial time computable function of the form $P_G = P_G^{\mathrm{sel}} + P_G^{\mathrm{ver}}$, where $P_G^{\mathrm{sel}} : \{0,1\}^n \to \mathbb{R}_+$, $P_G^{\mathrm{ver}} : \{0,1\}^n \to \{0, \pm\infty\}$, and

$$(G,k) \in \mathcal{G} \quad \Leftrightarrow \quad \begin{cases} \mathrm{opt}_{\vec{x}\in\{0,1\}^n} P_G(\vec{x}) \leq k & \text{if } \mathrm{opt} = \min \\ \mathrm{opt}_{\vec{x}\in\{0,1\}^n} P_G(\vec{x}) \geq k & \text{if } \mathrm{opt} = \max. \end{cases}$$

For $\vec{x} = (x_1, \ldots, x_n) \in \{0,1\}^n$ with $P_G(\vec{x}) \leq k$ if opt = min and with $P_G(\vec{x}) \geq k$ if opt = max, the vertex set *selected* by $\vec{x}$ and *verified* by P_G is $\{v_i \in V \mid x_i = 1, 1 \leq i \leq n\}$. A vector $\vec{x}$ is called *admissible* if $P_G^{\text{ver}}(\vec{x}) = 0$.

The intuition behind the term $P_\cdot = P_\cdot^{\text{sel}} + P_\cdot^{\text{ver}}$ is that the "selecting function" $P_\cdot^{\text{sel}}$ counts the size of the selected set of vertices and the "verifying function" $P_\cdot^{\text{ver}}$ verifies whether this choice of vertices is an admissible solution.

It is an easy observation that every select&verify graph problem that additionally admits a problem kernel of size $p(k)$ is solvable in time $O(2^{p(k)}p(k) + T_K(n,k))$.

Example 18 We now give some examples for select&verify problems by specifying the function $P_G = P_G^{\text{sel}} + P_G^{\text{ver}}$. In all cases below, the "selecting function" P_G for a graph $G = (V, E)$ will be

$$P_G^{\text{sel}} = \sum_{v_i \in V} x_i.$$

Also, we use the convention that $0 \cdot (\pm\infty) = 0$.

1. In the case of VERTEX COVER, we have opt = min and choose

$$P_G^{\text{ver}}(\vec{x}) = \sum_{\{v_i, v_j\} \in E} \infty \cdot (1 - x_i)(1 - x_j),$$

   where this sum brings $P_G(\vec{x})$ to infinity whenever there is an uncovered edge. In addition, $P_G(\vec{x}) \leq k$ then guarantees a vertex cover set of size at most k. Clearly, P_G is polynomial time computable.

2. Similarly, in the case of INDEPENDENT SET, we let opt = max and choose

$$P_G^{\text{ver}}(\vec{x}) = \sum_{\{v_i, v_j\} \in E} \infty \cdot x_i \cdot x_j.$$

3. DOMINATING SET is another example for a select&verify graph problem. Here, for $G = (V, E)$, we have

$$P_G^{\text{ver}}(\vec{x}) = \sum_{v_i \in V} (\infty \cdot (1 - x_i) \cdot \prod_{\{v_i, v_j\} \in E} (1 - x_j)),$$

   where this sum brings $P_G(\vec{x})$ to infinity whenever there is a non-dominated vertex which is not in the selected dominating set. In addition, $P_G(\vec{x}) \leq k$ then guarantees a dominating set of size at most k.

4. Similar observations as for VERTEX COVER, INDEPENDENT SET, and DOMINATING SET do hold for many other graph problems and, in particular, weighted variants of these.[12] As a source of problems, consider the variants of DOMINATING SET listed in [32]. In particular, the TOTAL DOMINATING SET problem is defined by

$$P_G^{\mathrm{ver}}(\vec{x}) = \sum_{v_i \in V} (\infty \cdot \prod_{\{v_i, v_j\} \in E} (1 - x_j)).$$

Moreover, graph problems where a small (or large) *edge set* is sought for can often be reformulated into vertex set optimization problems by introducing an additional artificial vertex on each edge of the original graph. In this way, the EDGE DOMINATING SET [33] problem can be handled. Similarly, planar graph problems where a small (or large) *face set* is looked for are expressible as select&verify problems of the dual graphs.

We will also need a notion of select&verify problems where the "selecting function" and the "verifying function" operate on a subgraph of the given graph.

Definition 19 Let $P_\cdot = P_\cdot^{\mathrm{sel}} + P_\cdot^{\mathrm{ver}}$ be the function of a select&verify problem. For an n-vertex graph G and subgraphs $G^{\mathrm{ver}} = (V^{\mathrm{ver}}, E^{\mathrm{ver}})$, $G^{\mathrm{sel}} = (V^{\mathrm{sel}}, E^{\mathrm{sel}}) \subseteq G$, we let

$$P_{G^{\mathrm{ver}}}(\vec{x} \mid G^{\mathrm{sel}}) := P_{G^{\mathrm{ver}}}^{\mathrm{ver}}(\pi_{V^{\mathrm{ver}}}(\vec{x})) + P_{G^{\mathrm{sel}}}^{\mathrm{sel}}(\pi_{V^{\mathrm{sel}}}(\vec{x})),$$

where $\pi_{V'}$ is the projection of the vector $\vec{x} \in \{0,1\}^n$ to the variables corresponding to the vertices in V'.

Weakly glueable graph problems

We are going to solve graph problems, slicing the given graph into small pieces with the help of small separators. Within these separators, the basic strategy will be to test all possible assignments of the vertices. The separators will serve as boundaries between the different graph chunks into which the graph is split. For each possible assignment of the vertices in the separators, we want to—independently—solve the corresponding problems on the

[12] In the weighted case, one typically chooses a "selecting function" of the form $P_G^{\mathrm{sel}} = \sum_{v_i \in V} \alpha_i x_i$, where α_i is the weight of the vertex v_i.

graph chunks and then reconstruct a solution for the whole graph by "gluing" together the solutions for the graph chunks. In order to do so, all additional information necessary for solving the subproblems correctly has to be transported and coded within the separators. It turns out that the information to be passed on is pretty clear in the case of the VERTEX COVER problem, but it is much more involved in the case of the DOMINATING SET problem and many others. This is the basic motivation for the formal framework we develop here. We need to assign *colors* to the separator vertices in the course of the algorithm. Hence, our algorithm has to be designed in such a manner that it can also cope with colored graphs, even though the original problem may have been a problem on non-colored graphs. In general (e.g., in the case of the DOMINATING SET problem), it is not sufficient to simply use the two colors 1 (for encoding "in the selected set") and 0 (for "not in the selected set").

Definition 20 Let $G = (V, E)$ be an undirected graph and let C_0, C_1 be finite, disjoint sets. A *C_0-C_1-coloring* of G is a function $\chi : V \to C_0 + C_1 + \{\#\}$.[13] A C_0-C_1-coloring with $C_0 = \{0\}$ and $C_1 = \{1\}$ is called a *0-1-coloring*.

For a C_0-C_1-coloring χ, the *corresponding* 0-1-coloring $\hat{\chi}$ is given by

$$\hat{\chi}(v) = \begin{cases} i & \text{if } \chi(i) \in C_i, \quad i = 0, 1, \\ \# & \text{otherwise.} \end{cases}$$

For $V' \subseteq V$, a function $\chi : V' \to C_0 + C_1$ can naturally be extended to a C_0-C_1-coloring of G by setting $\chi(v) = \#$ for all $v \in V \setminus V'$.

Definition 21 For two 0-1-colorings $\chi_1, \chi_2 : V \to \{0, 1, \#\}$ with $\chi_1^{-1}(\{0,1\}) \cap \chi_2^{-1}(\{0,1\}) = \emptyset$, the *sum* $\chi_1 + \chi_2$ is defined by

$$(\chi_1 + \chi_2)(v) = \begin{cases} \chi_1(v) & \text{if } \chi_1(v) \neq \#, \\ \chi_2(v) & \text{if } \chi_2(v) \neq \#, \\ \# & \text{otherwise.} \end{cases}$$

Definition 22 Consider an instance (G, k) of a select&verify problem $\mathcal{G}$ and a vector $\vec{x} \in \{0,1\}^n$ with $n = |V(G)|$. Let χ be a 0-1-coloring of G. Then, $\vec{x}$ is *consistent* with χ, written $\vec{x} \sim \chi$, if

$$\chi(v_j) = i \;\Rightarrow\; x_j = i, \quad \text{for } i = 0, 1,\ j = 1, \ldots, n.$$

[13] The symbol # will be used for the undefined (i.e., not yet defined) color.

We now provide the central notion of "weakly glueable" select&verify problems, a weaker form of the glueability concept as introduced in [4]. We apply this rather abstract notion to concrete graph problems afterwards.

Definition 23 A select&verify problem $\mathcal{G}$ given by $(P., \text{opt})$ is *weakly glueable with λ colors* if there exist

- a color set $C := C_0 + C_1 + \{\#\}$ with $|C_0 + C_1| = \lambda$, and
- a polynomial time computable function $h : (\mathbb{R}_+ \cup \{\pm\infty\})^3 \to \mathbb{R}_+ \cup \{\pm\infty\}$;

and, for every n-vertex graph $G = (V, E)$ and subgraphs $G^{\text{sel}}, G^{\text{ver}} \subseteq G$ with a separation (A_1, S, A_2) of G^{ver}, we find, for each coloring $\chi : S \to C_0 + C_1$,

- subgraphs $G(A_i, \chi)$ of G^{ver} with $G^{\text{ver}}[A_i] \subseteq G(A_i, \chi) \subseteq G^{\text{ver}}[\delta A_i]$ for $i = 1, 2$,
- subgraphs $G(S, \chi)$ of G^{ver} with $G(S, \chi) \subseteq G^{\text{ver}}[S]$

such that, for each 0-1-coloring $\mu : V \to \{0, 1, \#\}$ with $\mu \mid_S \equiv \#$, we can write

$$\text{opt}_{\substack{\vec{x} \in \{0,1\}^n \\ \vec{x} \sim \mu}} P_{G^{\text{ver}}}(\vec{x} \mid G^{\text{sel}}) \;=\; \text{opt}_{\chi : S \to C_0 + C_1} h\big(\text{Eval}_{A_1}(\chi), \text{Eval}_S(\chi), \text{Eval}_{A_2}(\chi)\big). \tag{2}$$

Here, $\text{Eval}_X(\cdot)$ for $X \in \{A_1, S, A_2\}$ is of the form

$$\text{Eval}_X(\chi) \;=\; \text{opt}_{\substack{\vec{x} \in \{0,1\}^n \\ \vec{x} \sim (\mu + \chi)}} P_{G(X,\chi)}(\vec{x} \mid G[X] \cap G^{\text{sel}}).$$

Example 24 We give some examples of weakly glueable problems, where — for the time being—we restrict ourselves to the case where $G^{\text{ver}} = G^{\text{sel}} = G$. The subtlety of allowing different subgraphs G^{ver}, G^{sel} in the definition above is due to technical reasons that become clear later. All examples generalize in a straight-forward way to this case.

1. VERTEX COVER is weakly glueable with $\lambda = 2$ colors. We use the color sets $C_i := \{i\}$ for $i = 0, 1$. The function h is $h(x, y, z) = x + y + z$. The subgraphs $G(X, \chi)$ for $X \in \{A_1, S, A_2\}$ and $\chi : S \to C_0 + C_1$ are

$$\begin{aligned} G(A_i, \chi) &:= G[A_i \cup \chi^{-1}(0)] \quad \text{for } i = 1, 2, \quad \text{and} \\ G(S, \chi) &:= G[S]. \end{aligned}$$

In this way, the subroutine $\mathrm{Eval}_S(\chi)$ checks whether the coloring χ yields a vertex cover on $G[S]$ and the subroutines $\mathrm{Eval}_{A_i}(\chi)$ compute the minimum size vertex cover on $G[A_i]$ under the constraint that all neighbors in A_i of a vertex in $\chi^{-1}(0)$ are covered. The reader may verify that—with these settings— equation (2) is satisfied.

2. INDEPENDENT SET is weakly glueable with 2 colors by a similar argument.

3. DOMINATING SET is weakly glueable with $\lambda = 4$ colors. We use the color sets $C_0 := \{0_{A_1}, 0_{A_2}, 0_S\}$ and $C_1 := \{1\}$. The semantics of these colors is as follows. Assigning the color 0_X, for $X \in \{A_1, S, A_2\}$, to vertices in a separation (A_1, S, A_2) means that the vertex is not in the dominating set and will be dominated by a vertex from X. Clearly, 1 means that the vertex belongs to the dominating set. The function h simply is addition, i.e., $h(x, y, z) = x + y + z$. When passing the information to the subproblems, for a given coloring $\chi : S \to C_0 + C_1$, we define

$$\begin{aligned} G(A_i, \chi) &:= G[A_i \cup \chi^{-1}(\{1, 0_{A_i}\})] \\ G(S, \chi) &:= G[\chi^{-1}(\{1, 0_S\})]. \end{aligned}$$

   In this way, $\mathrm{Eval}_S(\chi)$ checks whether the assignments of the color 0_S are correct (i.e., whether all vertices assigned 0_S are dominated by a vertex from S). Also, Eval_{A_i} returns the size of a minimum dominating set in A_i under the constraint that some vertices in δA_i still need to be dominated (namely, the vertices in $\chi^{-1}(0_{A_i})$) and some vertices in δA_i can already be assumed to be in the dominating set (namely, the vertices in $\chi^{-1}(1)$).

   It is not hard to check that—with these settings— equation (2) is satisfied.

4. We want to mention in passing that—besides the problems given here—many more select&verify problems are weakly glueable. In particular this is true for the weighted versions and variations of the above mentioned problems. Note that, e.g., TOTAL DOMINATING SET is an example of a graph problem where a color set C_1 of more than one color is needed.

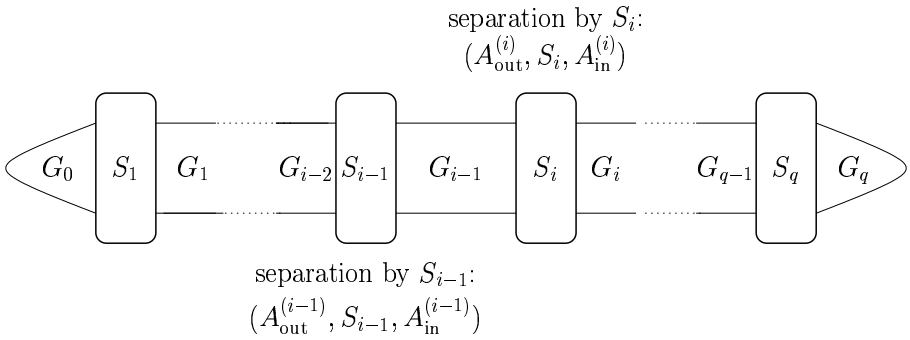

Figure 2: Dynamic programming on layerwisely separated graph.

5.2.2 The algorithm

Similar to Theorem 15, which is based on tree decompositions, we construct a partial layerwise separation $\mathcal{S}(\psi)$ with optimally adapted trade-off parameter ψ to guarantee an efficient dynamic programming algorithm. However, for our purposes here, we need to be able to deal with "precolored" graphs.

Definition 25 Let $\mathcal{G}$ be a select&verify graph problem defined by $(P., \text{opt})$. The problem CONSTRAINT $\mathcal{G}$ then is to determine, for an n-vertex graph $G = (V, E)$, two subgraphs $G^{\text{sel}}, G^{\text{ver}} \subseteq G$, and a given 0-1-coloring $\chi : V \to \{0, 1, \#\}$, the value

$$\text{opt}_{\substack{\vec{x}\in\{0,1\}^n \\ \vec{x}\sim\chi}} P_{G^{\text{ver}}}(\vec{x} \mid G^{\text{sel}}).$$

Theorem 26 *Let $\mathcal{G}$ be a select&verify problem for planar graphs. Suppose that*

1. *$\mathcal{G}$ has the LSP of width w and size-factor d,*
2. *$\mathcal{G}$ is weakly glueable with λ colors, and*
3. *there exists an algorithm that solves the problem* CONSTRAINT *$\mathcal{G}$ for a given graph G in time $\tau^{\text{out}(G)}n$.*

Then, there is an algorithm to decide $(G, k) \in \mathcal{G}$ in time $O(\tau^w \cdot 2^{\theta_2(\lambda,\tau,d)\sqrt{k}}n)$, where $\theta_2(\lambda, \tau, d) = 2\sqrt{2d\log(\lambda)\log(\tau)}$.

Proof. Let us first sketch the overall structure of the algorithm.

1. Compute some planar embedding ϕ of G, and find a "suitable" partial layerwise separation $(S_1, \ldots, S_q)$ for the plane graph (G, ϕ). A coarse sketch of the such obtained graph structure is depicted in Figure 2.

2. By using dynamic programming techniques, an optimal solution is found by sweeping over the graph from left to right, as illustrated in Figure 2. More detailed, we do the following:

   (a) For all possible "colorings" of S_1, find an optimal solution of CONSTRAINT $\mathcal{G}$ on G_0 (plus suitably chosen precolored vertices from S_1); store the obtained optima in a (large) table belonging to S_1.

   (b) For $j := 2$ to q do:

   - For all possible "colorings" of S_{j-1} and of S_j, find an optimal solution of CONSTRAINT $\mathcal{G}$ on G_{j-1} (plus suitably chosen precolored vertices from S_{j-1} as well as of S_j).
   - Store the obtained optima (for the subgraph of G with vertices from G_0 through G_{j-1} and S_1 through S_j) in a table belonging to S_j.
   - (For reasons of space efficiency, you might now forget about the table belonging to S_{j-1}.)

   (c) For all possible "colorings" of S_q, find an optimal solution of CONSTRAINT $\mathcal{G}$ on G_q (plus suitably chosen precolored vertices from S_q); store the obtained optima in a (large) table belonging to S_q.

   (d) From the table pertaining to S_q, the desired optimum can now be found.

We are now giving formal details of the sketched algorithms, thereby proving its correctness. Suppose $\mathcal{G}$ is defined by $(P., \text{opt})$.

Step 1: Given an instance (G, k), in linear time we can compute some planar embedding ϕ of G (see [14]). Compute a partial layerwise separation $\mathcal{S}(\psi)$ (ψ will be determined later) for (G, ϕ), using Proposition 12 and the assumption 1 (LSP). Suppose $\mathcal{S}(\psi) = (S_1, \ldots, S_q)$ and let $\mathcal{C}_{\mathcal{S}(\psi)} = (G_0, \ldots, G_q)$ denote the corresponding graph-chunks cut out by $\mathcal{S}(\psi)$.

For every separator S_i, we get a separation of the form

$$(A^{(i)}_{\text{out}}, S_i, A^{(i)}_{\text{in}}),$$

where $A^{(i)}_{\text{out}}$, and $A^{(i)}_{\text{in}}$, respectively, are the vertices of the graph chunks of lower order layers, and higher order layers, respectively. By default, we let $S_0 = S_{q+1} = \emptyset$ such that the corresponding separations are $(\emptyset, S_0, V)$ and $(V, S_{q+1}, \emptyset)$, respectively. The layerwise separation and the separations $(A^{(i)}_{\text{out}}, S_i, A^{(i)}_{\text{in}})$ are illustrated in Figure 2.

Step 2: Determine the value

$$\text{opt}_{\vec{x}\in\{0,1\}^n} P_G(\vec{x}).$$

This can be done by a dynamic programming approach that makes use of the weak glueability of $\mathcal{G}$ as follows.

We successively compute, for $i = 1, \ldots, q+1$, the values

$$M^{(i)}(\mu^{(i)}) \quad := \quad \text{opt}_{\vec{x}\sim\widehat{\mu^{(i)}}} P_{H_i^{\text{ver}}(\mu^{(i)})}(\vec{x} \mid H_i^{\text{sel}}) \tag{3}$$

for all C_0-C_1-colorings $\mu^{(i)} : S_i \to C_0 + C_1$, where

$$H_i^{\text{ver}}(\mu^{(i)}) \;:=\; G(A^{(i)}_{\text{out}}, \mu^{(i)}) \quad \text{and} \quad H_i^{\text{sel}} \;:=\; G[A^{(i)}_{\text{out}}].$$

Note that we have

$$\text{opt}_{\vec{x}\in\{0,1\}^n} P_G(\vec{x}) = M^{(q+1)}(\mu),$$

for the empty map $\mu : S_{q+1} = \emptyset \to C_0 + C_1$, because $H^{\text{sel}}_{q+1} = G[A^{(q+1)}_{\text{out}}] = G[V] = G$ and $H^{\text{ver}}_{q+1}(\mu) = G$ (since $H^{\text{sel}}_{q+1} \subseteq H^{\text{ver}}_{q+1}(\mu)$).

The computation of $M^{(i)}(\mu^{(i)})$ as defined in (3) can be done iteratively. To do so, note that $H_i^{\text{ver}}(\mu^{(i)})$ is separated by S_{i-1} in

$$(B^{(i-1)}_{\text{out}}, S_{i-1}, B^{(i-1)}_{\text{in}}),$$

where $B^{(i-1)}_{\text{out}} = A^{(i-1)}_{\text{out}}$ and $B^{(i-1)}_{\text{in}} = A^{(i-1)}_{\text{in}} \cap V(H_i^{\text{ver}}(\mu^{(i)}))$. Hence, by definition of weak glueability we have

$$\begin{aligned} M^{(i)}(\mu^{(i)}) \;=\; & \text{opt}_{\mu^{(i-1)}:S_{i-1}\to C_0+C_1} \\ & h(\text{Eval}_{B^{(i-1)}_{\text{out}}}(\mu^{(i-1)}), \text{Eval}_{S_{i-1}}(\mu^{(i-1)}), \text{Eval}_{B^{(i-1)}_{\text{in}}}(\mu^{(i-1)})) \end{aligned} \tag{4}$$

where

$$\text{Eval}_X(\mu^{(i-1)}) = \text{opt}_{\vec{x}\sim(\widehat{\mu^{(i-1)}}+\widehat{\mu^{(i)}})} P_{G(X,\mu^{(i-1)})}(\vec{x} \mid G[X] \cap H_i^{\text{sel}}). \tag{5}$$

for $X \in \{B^{(i-1)}_{\text{out}}, S_{i-1}, B^{(i-1)}_{\text{in}}\}$. Here, recall that $\widehat{\mu^{(i)}}$ denotes the 0-1-coloring corresponding to $\mu^{(i)}$. In particular, for the different choices of X, we get the following.

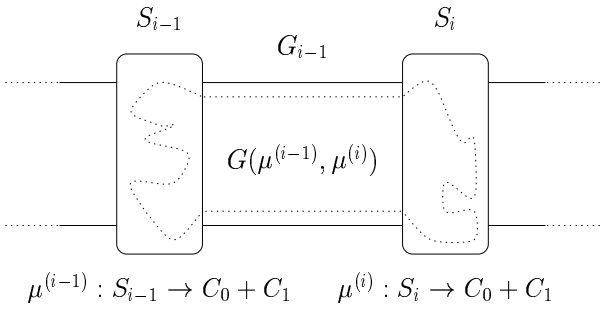

Figure 3: The graph $G(\mu^{(i)}, \mu^{(i-1)})$.

- For $X = B_{\text{out}}^{(i-1)}$, equation (5) becomes

$$\begin{aligned} \text{Eval}_{B_{\text{out}}^{(i-1)}}(\mu^{(i-1)}) &= \text{opt}_{\vec{x} \sim (\widehat{\mu^{(i-1)}} + \widehat{\mu^{(i)}})} P_{H_{i-1}^{\text{ver}}(\mu^{(i-1)})}(\vec{x} \mid H_{i-1}^{\text{sel}} \cap H_i^{\text{sel}}) \\ &= M^{(i-1)}(\mu^{(i-1)}), \end{aligned} \tag{6}$$

  where the last equation holds, since $H_{i-1}^{\text{sel}} \subseteq H_i^{\text{sel}}$ and since $\widehat{\mu^{(i)}} \equiv \#$ restricted to $G(B_{\text{out}}^{(i-1)}, \mu^{(i-1)}) = H_{i-1}^{\text{ver}}(\mu^{(i-1)})$.

  Hence, the value of $\text{Eval}_{B_{\text{out}}^{(i-1)}}(\mu^{(i-1)})$ is given by the previous step of the iteration.

- For $X = S_{i-1}$, equation (5) becomes

$$\begin{aligned} \text{Eval}_{S_{i-1}}(\mu^{(i-1)}) &= \text{opt}_{\vec{x} \sim \widehat{\mu^{(i-1)}}} P_{G(S_{i-1}, \mu^{(i-1)})}(\vec{x} \mid G[S_{i-1}]) \\ &= P_{G(S_{i-1}, \mu^{(i-1)})}(\vec{x}_{\widehat{\mu^{(i-1)}}} \mid G[S_{i-1}]), , \end{aligned} \tag{7}$$

  where $\vec{x}_{\widehat{\mu^{(i-1)}}} \in \{0,1\}^n$ is an arbitrary vector such that $\vec{x}_{\widehat{\mu^{(i-1)}}} \sim \widehat{\mu^{(i-1)}}$. The first equation above holds, since $G(S_{i-1}, \mu^{(i-1)}) \subseteq G[S_{i-1}] \subseteq H_i^{\text{sel}}$ and $\widehat{\mu^{(i)}} \equiv \#$ restricted to $G(S_{i-1}, \mu^{(i-1)})$. The second equation is true since the 0-1-coloring $\widehat{\mu^{(i-1)}}$ assigns color 0 or color 1 to all vertices in S_{i-1}, and since $G(S_{i-1}, \mu^{(i-1)}) \subseteq G[S_{i-1}]$.

  Hence, the value $\text{Eval}_{S_{i-1}}(\mu^{(i-1)})$ can be computed by a simple evaluation of the function P for the given vector $\vec{x}_{\widehat{\mu^{(i-1)}}}$.

- For $X = B_{\text{in}}^{(i-1)}$, equation (5) becomes

$$\begin{aligned}\text{Eval}_{B_{\text{in}}^{(i-1)}}(\mu^{(i-1)}) &= \text{opt}_{\vec{x}\sim(\widehat{\mu^{(i-1)}}+\widehat{\mu^{(i)}})} P_{G(B_{\text{in}}^{(i-1)},\mu^{(i-1)})}(\vec{x} \mid G[B_{\text{in}}^{(i-1)}] \cap H_i^{\text{sel}}) \\ &= \text{opt}_{\vec{x}\sim(\widehat{\mu^{(i-1)}}+\widehat{\mu^{(i)}})} P_{G(\mu^{(i)},\mu^{(i-1)})}(\vec{x} \mid G_{i-1}), \qquad (8)\end{aligned}$$

  where $G(\mu^{(i)}, \mu^{(i-1)}) := G(A_{\text{in}}^{(i-1)}, \mu^{(i-1)}) \cap G(A_{\text{out}}^{(i)}, \mu^{(i)})$. This graph is illustrated in Figure 3. In the evaluation above we used that $G[B_{\text{in}}^{(i-1)}] \cap H_i^{\text{sel}} = G_{i-1}$ and that $G(B_{\text{in}}^{(i-1)}, \mu^{(i-1)}) = G(\mu^{(i)}, \mu^{(i-1)})$.

  Hence, the value $\text{Eval}_{B_{\text{in}}^{(i-1)}}(\mu^{(i-1)})$ can be computed using the $\tau^{\text{out}(G)}$ time algorithm for CONSTRAINT $\mathcal{G}$.

Hence, plugging formulas (6), (7), and (8) in expression (4), we obtain

$$M^{(i)}(\mu^{(i)}) = \text{opt}_{\mu^{(i-1)}:S_{i-1}\to C_0+C_1} \; h\left(\begin{array}{c} M^{(i-1)}(\mu^{(i-1)}) \\ P_{G(S_{i-1},\mu^{(i-1)})}(\vec{x}_{\widehat{\mu^{(i-1)}}} \mid G[S_{i-1}]) \\ \text{opt}_{\vec{x}\sim\widehat{\mu^{(i-1)}}+\widehat{\mu^{(i)}}} P_{G(\mu^{(i)},\mu^{(i-1)})}(\vec{x} \mid G_{i-1}) \end{array}\right). \qquad (9)$$

This evaluation is done successively for all $i = 1, \ldots, q+1$. By induction, one sees that

$$\text{opt}_{\vec{x}\in\{0,1\}^n} P_G(\vec{x}) = M^{(q+1)}(\mu)$$

can be computed in this way.

Computation time: For fixed coloring $\mu^{(i)}$, computing $M^{(i)}(\mu^{(i)})$ according to equation (9) costs time

$$\lambda^{|S_{i-1}|} \cdot \tau^{\text{out}(G[S_{i-1}\cup V_{G_{i-1}}\cup S_i])}.$$

The first factor reflects the cost of computing the opt over all $\mu^{(i-1)} : S_{i-1} \to C_0+C_1$. The second factor arises by the evaluations on the graphs $G(S_{i-1}, \mu^{(i-1)})$, $G(\mu^{(i-1)}, \mu^{(i)}) \subseteq G[S_{i-1} \cup V(G_{i-1}) \cup S_i]$, where we use assumption 3 of the theorem. Thus, the running time for evaluating $M^{(i)}(\mu^{(i)})$ for all $\mu^{(i)} : S_i \to C_0 + C_1$ is bounded by

$$\lambda^{|S_i|} \cdot \lambda^{|S_{i-1}|} \cdot \tau^{\text{out}(G[S_{i-1}\cup V_{G_{i-1}}\cup S_i])}.$$

Hence, the total running time of the algorithm is bounded by $2^{\theta(\psi)}n$, where

$$\begin{aligned}
\theta(\psi) &\leq 2\log(\lambda) \max_{i=1,\dots,q} |S_i| \quad + \quad \log(\tau) \max_{i=0,\dots,q} \mathrm{out}(G[S_i \cup V_{G_i} \cup S_{i+1}]) \\
&\leq 2\log(\lambda)\psi\sqrt{dk} \quad + \quad \log(\tau)(\frac{\sqrt{dk}}{\psi} + w) \\
&= (2\log(\lambda)\psi + \frac{\log\tau}{\psi})\sqrt{dk} \quad + \quad \log(\tau)w
\end{aligned}$$

This upper bound is minimized for $\psi = \sqrt{\log(\tau)/2\log(\lambda)}$, which gives us the claimed value $\theta_2(\lambda, \tau, d) = 2\sqrt{2d\log(\lambda)\log(\tau)}$ and the constant τ^w for the running time. □

It remains to say for which problems there exists a solving algorithm of the problem CONSTRAINT $\mathcal{G}$ for a given graph G in time $\tau^{\mathrm{out}(G)}n$. Baker [7] presented several such problems $\mathcal{G}$, however, the algorithms as stated there do *not* cope with the constraint version of $\mathcal{G}$. Often, we can adapt them quite easily. In the case of PLANAR VERTEX COVER, as well as in the case of PLANAR INDEPENDENT SET, it is quite simple to handle a "precoloring" $\mu : V \to \{0, 1, \#\}$ for a graph $G = (V, E)$.

More formally, given an admissible[14] coloring μ, one likes to transform an instance (G, k) to an instance (G', k'), such that $(G, k) \in \mathcal{G}$ for some witnessing vector $\vec{x}$ with $\vec{x} \sim \mu$ iff $(G', k') \in \mathcal{G}$ for some witnessing vector $\vec{x'}$ (without any constraint).

For PLANAR VERTEX COVER, this can, e.g., be achieved by the following transformation:

$$\begin{aligned}
G' &= G[V - (C \cup N(\mu^{-1}(0)))] \quad \text{and} \\
k' &= k - |\mu^{-1}(1)| - |\{v \in V - C \mid \exists u \in \mu^{-1}(0) \cap N(v)\}|,
\end{aligned}$$

here $C := \mu^{-1}(\{0, 1\})$ denotes the set of vertices that are already assigned a color. Herefore, observe that a vertex $v \in C$, which is—by the coloring μ—assigned not to be in the vertex cover, i.e., $\mu(v) = 0$, forces its neighbors to be in a vertex cover. Hence, the set $N(\mu^{-1}(0))$ needs to be in any vertex cover (represented by $\vec{x}$) that fulfills the constraint $\vec{x} \sim \mu$. This justifies the definition of G'. It is clear that the parameter k becomes smaller by the number of vertices which are already assigned to be in the vertex cover, i.e.,

[14] Here, *admissible* means that there exists a vector $\vec{x} \in \{0, 1\}^n$ with $\vec{x} \sim \mu$, such that $P_G^{\mathrm{ver}}(\vec{x}) = 0$.

$\mu^{-1}(1)$, and the number of vertices that are forced to be in the vertex cover by μ, i.e., by the number of vertices in $V - C$ that have a neighbor in $\mu^{-1}(0)$. We can apply the non-constraint algorithm to (G', k'). Note that, clearly, $\text{out}(G') \leq \text{out}(G)$.

A similar observation helps to deal with PLANAR INDEPENDENT SET. However, it is not clear to us how to transform instances for PLANAR DOMINATING SET, such that the non-constraint problem can be solved instead and—at the same time—the outerplanarity number is maintained.[15]

Example 27 1. For PLANAR VERTEX COVER, we have $d = 2$, $w = 1$ (see Example 10.1), $\lambda = 2$ (see Example 24.1), and $\tau = 8$ (see the result of Baker [7] which can be adapted to the constraint case by the considerations above) and, hence, the approach in Theorem 26 yields an $O(2^{4\sqrt{3k}}n)$ time algorithm.

2. Similarly, for the problem kernel of the PLANAR INDEPENDENT SET problem, we can set $d = 4, w = 1$ (see Example 10.2), $\lambda = 2$ (see Example 24.2), and $\tau = 8$ (see [7] and our considerations for the constraint case), hence, we obtain a time $O(2^{4\sqrt{6k}}n)$ algorithm.

3. As already discussed above, we do not know whether the $8^{\text{out}(G)}n$ time algorithm in [7] extends to the constraint case for PLANAR DOMINATING SET.

5.3 Comparing the benefits of the different approaches

Which of the two approaches (presented in Subsections 5.1 and 5.2, respectively) for the algorithms on layerwisely separated graphs should be preferred? For that purpose, let us compare the results obtained in Theorems 15 and 26.

Clearly, both results rely on the LSP assumption. Besides that, Theorem 15 requires the existence of a linear time algorithm for bounded treewidth graphs, whereas in Theorem 26 one needs (besides the weak glueability assumption) the existence of a linear time algorithm for graphs with bounded

[15] Clearly, we can transform G into a graph G' where we attach to each vertex $v \in \mu^{-1}(1)$ a degree-1 vertex v' and, hence, force v to be contained in any optimal dominating set of the non-constraint case. But it is not clear what gadget is needed to replace vertices $v \in \mu^{-1}(0)$.

outerplanarity.[16] The interrelation inbetween these assumptions is given by the following observation, which is based on Proposition 13.

Lemma 28 *Let $\mathcal{G}$ be a parameterized problem for planar graphs. Suppose that there exists a time $\sigma^{\ell}n$ algorithm that solves* CONSTRAINT $\mathcal{G}$*, when graph G is given together with a tree decomposition of width ℓ. Then, there is an algorithm that solves* CONSTRAINT $\mathcal{G}$ *in time $\tau^{\text{out}(G)}n$ for $\tau = \sigma^3$.*

Proof. Applying the algorithm of Proposition 13 to input instance (G, k), we obtain a tree decomposition of width at most $3\,\text{out}(G)$. Then CONSTRAINT $\mathcal{G}$ can be solved using this tree decomposition by our assumption. □

The following easy corollary helps comparing the approach from Subsection 5.1 (i.e., Theorem 15) with the approach in Subsection 5.2 (i.e., Theorem 26).

Corollary 29 *Let $\mathcal{G}$ be a select&verify problem for planar graphs. Suppose that*

1. *$\mathcal{G}$ has the LSP of width w and size-factor d,*
2. *$\mathcal{G}$ is weakly glueable with λ colors, and*
3. *there exists a time $\sigma^{\ell}n$ algorithm that solves* CONSTRAINT $\mathcal{G}$ *for a graph G, if G is given together with a tree decomposition of width ℓ.*

Then, there is an algorithm to decide $(G, k) \in \mathcal{G}$ in time $O(\sigma^{3w} \cdot 2^{\theta_3(\lambda,\sigma,d)\sqrt{k}}n)$, where $\theta_3(\lambda, \sigma, d) = 2\sqrt{6d\log(\lambda)\log(\sigma)}$.

The exponential factor of the algorithm in Corollary 29, i.e., $\theta_3(\lambda, \sigma, d)$, is related to the corresponding exponent of Theorem 15, i.e., $\theta_1(\sigma, d)$ in the following way:

$$\sqrt{\log \lambda} \cdot \theta_1(\sigma, d) = \sqrt{\log \sigma} \cdot \theta_3(\lambda, \sigma, d).$$

From this we derive that

[16] Clearly, for both results, Theorem 15 and Theorem 26, respectively, we do not rely on a *linear* time algorithm for bounded treewidth graphs, and graphs with bounded outerplanarity, respectively. Assuming a polynomial time algorithms is sufficient to get a $c^{\sqrt{k}}$-algorithm. The results are formulated for linear time algorithm since we always do have linear time algorithms.

- if $\lambda > \sigma$, the algorithm in Theorem 15 outperforms the result of Corollary 29,
- if $\lambda < \sigma$, the algorithm in Corollary 29 outperforms the result of Theorem 15.

However, in order to apply Corollary 29, we need the three extra assumptions that we have a select&verify problem which is weakly glueable and that we can deal with the problem CONSTRAINT $\mathcal{G}$ in the treewidth algorithm.

Example 30 Since the tree decomposition based algorithms for VERTEX COVER, INDEPENDENT SET, and DOMINATING SET can deal with CONSTRAINT $\mathcal{G}$, we derive:

1. For PLANAR VERTEX COVER, we have $\lambda = \sigma = 2$ which yields the same result as in Example 27.1.
2. Similarly, for PLANAR INDEPENDENT SET the fact that $\lambda = \sigma = 2$ gives us the result obtained in Example 27.2.
3. In contrast to our considerations in Example 27.3, by Lemma 28 we conclude that CONSTRAINT DOMINATING SET (where $d = 51, w = 3$ by Example 8.2) can be solved in time $\sigma^{3\,\mathrm{out}(G)}n$ with $\sigma = 4$. As to the constant $\lambda = 4$ (see Example 24.3) we remark that—for algorithmic purposes—it can be dropped to $\lambda = 3$ by following argument: Recall the color sets $C_1 = \{1\}$ and $C_0 = \{0_{A_1}, 0_S, 0_{A_2}\}$ and their semantics from Example 24.3. Instead of evaluating $M^{(i)}(\mu^{(i)})$ in the algorithm of Theorem 26 (see Equation 9) for all $\mu^{(i)} : S_i \to C_0 + C_1$, we use only colorings of the form $\mu^{(i)} : S_i \to \{1, 0_{A_1}, 0_{A_2}\}$. For each such coloring we "overwrite" the set

   $$C := \{v \in S_i \mid \mu^{(i)}(v) \in \{0_{A_1}, 0_{A_2}\} \text{ and } \exists w \in N(v) :\ \mu^{(i)}(w) = 1\}$$

   with the color 0_S, since the vertices in C are the ones that are already dominated by vertices in S_i. This means that, instead of dealing with the color set $C_0 + C_1$, we can restrict ourselves to the 3 colors $\{1, 0_{A_1}, 0_{A_2}\}$.

   All in all, Corollary 29 gives us an $O(2^{12\sqrt{17 \cdot \log(3) \cdot k}}n)$ time algorithm for PLANAR DOMINATING SET, which slightly beats the $O(2^{12\sqrt{34k}}n)$ from Example 16.

We give a last remark on the space consumption of the two different approaches of Theorem 15, and Theorem 26, respectively. The tables that need to be maintained in order to realize a treewidth based algorithm of Theorem 15 (i.e., the tables corresponding to the bags of the underlying tree decomposition) have size bounded by

$$\sigma^{2\sqrt{6dk}+(3w-1)}.$$

In contrast, the analysis of the proof of Theorem 26 gives an upper bound on the size of the tables for the dynamic programming therein. The upper bound is given by

$$\max_{i=1,\ldots,q} \lambda^{|S_i|} \leq \lambda^{\sqrt{\frac{\log(\tau)dk}{2\log(\lambda)}}}.$$

Since all our examples have the property that $\sigma = \lambda$, in terms of space consumption, the approach of Theorem 26 clearly outperforms the algorithm in Theorem 15.

6 Further extensions

6.1 Beyond linear kernels

This short section sketches an extension of the methods presented so far.

As already mentioned after Definition 17, every select&verify problem $\mathcal{G}$ that admits a linear problem kernel is solvable in time $2^{O(k)}n^{O(1)}$. Also, the existence of a linear problem kernel implies the LSP. Assuming certain further properties of $\mathcal{G}$, we have seen results, which improve the running time of a solving algorithm to $2^{O(\sqrt{k})}n^{O(1)}$.

The techniques can be extended to the case where $\mathcal{G}$ admits a problem kernel of size $p(k)$. We briefly outline how, by a straightforward generalization of our methods, the running time for solving $\mathcal{G}$ can be sped up from $2^{O(p(k))}n^{O(1)}$ (as can be obtained trivially) to $2^{O(\sqrt{p(k)})}n^{O(1)}$. The key tool for that purpose is the extension of Proposition 12, which now becomes:

Proposition 31 *Let $(G = (V, E), \phi)$ be a plane graph that admits a layerwise separation of width w and size $p(k)$. Then, for every $\psi \in \mathbb{R}_+$, there exists a partial layerwise separation $\mathcal{S}(\psi)$ of width w such that*

1. $\max_{S \in \mathcal{S}(\psi)} |S| \leq \psi\sqrt{p(k)}$ *and*

2. $\mathrm{out}(H) \leq \frac{\sqrt{p(k)}}{\psi} + w$ *for each graph chunk* H *in* $\mathcal{C}_{\mathcal{S}(\psi)}$.

Moreover, there is an algorithm with running time $O(\sqrt{p(k)}n)$ *which, for a given* ψ,

- *recognizes whether* (G, ϕ) *admits a layerwise separation of width* w *and size* $p(k)$ *and, if so, computes* $\mathcal{S}(\psi)$;
- *computes a partial layerwise separation of width* w *that fulfills the conditions above.*

As in Section 5, the above result can be used to show similar results as those given in Subsection 5.1 (see Theorem 15) and in Subsection 5.2 (see Theorem 26). Again, one uses the trade-off parameter ψ to optimize the running time of the different approaches. The corresponding results obtained in this manner are the following.

Theorem 32 *Let* $\mathcal{G}$ *be a parameterized problem for planar graphs. Suppose that*

1. $\mathcal{G}$ *admits a problem kernel of size* $p(k)$, *and*
2. *there exists a time* $\sigma^{\ell}n$ *algorithm that decides* $(G, k) \in \mathcal{G}$, *if* G *is given together with a tree decomposition of width* ℓ.

Then, there is an algorithm to decide $(G, k) \in \mathcal{G}$ *in time* $O(\sigma^{3w-1} \cdot 2^{\vartheta_1(\sigma,d)\sqrt{p(k)}}n)$, *where* $\vartheta_1(\sigma, d) = 2\log(\sigma)\sqrt{6d}$.

Theorem 33 *Let* $\mathcal{G}$ *be a select&verify problem for planar graphs. Suppose that*

1. $\mathcal{G}$ *admits a problem kernel of size* $p(k)$,
2. $\mathcal{G}$ *is weakly glueable with* λ *colors, and*
3. *there exists an algorithm that solves the problem* CONSTRAINT $\mathcal{G}$ *for a given graph* G *in time* $\tau^{\mathrm{out}(G)}n$.

Then, there is an algorithm to decide $(G, k) \in \mathcal{G}$ *in time* $O(\tau^{w} \cdot 2^{\vartheta_2(\lambda,\tau,d)\sqrt{p(k)}}n)$, *where* $\vartheta_2(\lambda, \tau, d) = 2\sqrt{2d\log(\lambda)\log(\tau)}$.

Note that in both theorems it would be sufficient to replace condition 1 by a modified version of the LSP:

1'. for every $(G,k) \in \mathcal{G}$ the graph G has a layerwise separation of width w and size $p(k)$.

Clearly, using similar arguments as in the proof of Lemma 9, the existence of a problem kernel of size $p(k)$ implies this condition on the set of all problem kernels $\mathcal{G}'$.

6.2 Refining FPT

We finish this section with a more philosophical observation concerning the world of fixed parameter tractability. Since the typical running times we obtain by the approach of this paper have sublinear exponents in the exponential parts, it might be challenging to develop a more structural complexity theory yielding more fine grained subclasses of FPT. Due to [18], a parameterized language L obeys

$$L \in \mathrm{FPT} \quad \text{iff} \quad L \text{ admits a kernelization.}$$

This means that a fine-grained structure can be obtained by either relying on the asymptotic behavior of the exponential term in the running time or on the size of the kernels. As complexity classes, we hence propose: This leads to a refinement

$$\mathrm{FPT}(f) := \left\{ L \subseteq \Sigma \times \mathbb{N} \;\middle|\; \begin{array}{l} \exists \text{ an algorithm to decide } (x,k) \in L \text{ in time} \\ g(k) \cdot n^{\alpha}, \text{ for some } \alpha \in \mathbb{R}_+ \text{ and } g \in O(f) \end{array} \right\} \quad \text{or}$$

$$\mathrm{FPT}_K(f) := \left\{ L \subseteq \Sigma \times \mathbb{N} \;\middle|\; \begin{array}{l} L \text{ admits a reduction to a problem kernel} \\ \text{of size } g(k) \text{ for some } g \in O(f) \end{array} \right\}.$$

This issue is discussed more thoroughly in [4].

7 Conclusion

This paper presents new, improved results for the fixed parameter complexity of planar graph problems. To some extent, this paper can be seen as the "parameterized complexity counterpart" to what was developed by Baker [7] in the context of approximation algorithms. We describe two main ways

(namely linear problem kernels and problem-specific approaches) to achieve the novel concept of Layerwise Separation Property, from which again, two approaches (tree decomposition and bounded outerplanarity) lead to $c^{\sqrt{k}}$-algorithms for planar graph problems (see Figure 1 for an overview). A slight modification of our presented techniques can be used to extend our results to parameterized problems that admit a problem kernel of size $p(k)$ (not necessarily linear!). In this case, the running time can be sped up from $2^{O(p(k))}n^{O(1)}$ to $2^{O(\sqrt{p(k)})}n^{O(1)}$. As a matter of fact, basically all FPT-problems that admit treewidth based algorithms can be handled by our methods. Refer to [9, 10, 31, 32] for a list of such problems.

Future research topics raised by our work include

- to further improve the ("exponential") constants, e.g., by a further refined and more sophisticated "layer decomposition tree";
- to investigate and extend the availability of linear problem kernels for all kinds of planar graph problems;
- to find out in a systematic, methodological way which planar graph problems do not fit in our setting and why;
- to provide implementations of our approach (where we favor the tree decomposition approach in comparison with the bounded outerplanarity approach due to the greater elegance, clearness, and universality of the first) accompanied by sound experimental studies, thus taking into account that all our analysis is worst case and often overly pessimistic.

Finally, a more general question is whether there are other "problem classes" that allow for $c^{\sqrt{k}}$ fixed parameter algorithms. Cai and Juedes [12] addressed this issue and claimed that for a list of parameterized problems (e.g., VERTEX COVER on general graphs) $c^{o(k)}$-algorithms are impossible unless FPT $= W[1]$. However, very recently, a flaw was detected in their proof [17].

Acknowledgements. We'd like to mention that parts of this work were discussed at the first international Workshop on Parameterized Complexity (organized by Mike Fellows and Venkatesh Raman) in Chennai, India, December 7–9, 2000.

References

[1] J. Alber, H. L. Bodlaender, H. Fernau, and R. Niedermeier. Fixed parameter algorithms for planar dominating set and related problems. In *Proc. 7th Scandinavian Workshop on Algorithm Theory SWAT*, volume 1851 of *LNCS*, pages 97–110, 2000.

[2] J. Alber, H. L. Bodlaender, H. Fernau, and R. Niedermeier. Fixed parameter algorithms for planar dominating set and related problems. Technical Report UU-CS-2000-28, Utrecht University, 2000. Accepted for publication in *Algorithmica*.

[3] J. Alber, H. Fan, M. Fellows, H. Fernau, R. Niedermeier, F. Rosamond, and U. Stege. Refined search tree techniques for the planar dominating set problem. In *Proc. 26th International Symposium on Mathematical Foundations of Computer Science MFCS 2001*, volume 2136 of *LNCS*, pages 111–122. Springer-Verlag, 2001.

[4] J. Alber, H. Fernau, and R. Niedermeier. Graph separators: a parameterized view. In *Proc. 7th Annual International Computing and Combinatorics Conference COCOON 2001*, volume 2108 of *LNCS*, pages 318–327. Springer-Verlag, 2001. Long version available as Technical Report WSI-2001-8, Wilhelm-Schickard-Institut für Informatik, Universität Tübingen (Germany).

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

[6] N. Alon, P. D. Seymour, and R. Thomas. A separator theorem for graphs with an excluded minor and its applications. In *Proc. 22nd ACM Symposium on the Theory of Computing STOC'90*, pages 293–299, 1990.

[7] B. S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *Journal of the Association for Computing Machinery*, 41(1):153–180, 1994.

[8] R. Bar-Yehuda and S. Even. A local-ratio theorem for approximating the weighted vertex cover problem. *Annals of Discr. Math.*, 25:27–46, 1985.

[9] H. L. Bodlaender. Dynamic programming on graphs of bounded treewidth. In *Proc. 15th International Colloquium on Automata, Languages and Programming ICALP'88*, volume 317 of *LNCS*, pages 105–118. Springer-Verlag, 1988.

[10] H. L. Bodlaender. Treewidth: Algorithmic techniques and results. In *Proc. 22nd Mathematical Foundations of Computer Science MFCS'97*, volume 1295 of *LNCS*, pages 19–36. Springer-Verlag, 1997.

[11] H. L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209:1–45, 1998.

[12] L. Cai and D. Juedes. Subexponential parameterized algorithms collapse the W-hierarchy. In *Proc. 28th International Colloquium on Automata, Languages and Programming ICALP'01*, volume 2076 of *LNCS*, pages 273–284. Springer-Verlag, 2001.

[13] J. Chen, I. Kanj, and W. Jia. Vertex cover: Further observations and further improvements. In *Proc. 25th Workshop on Graph-Theoretic Concepts in Computer Science WG'99*, volume 1665 of *LNCS*, pages 313–324. Springer-Verlag, 1999.

[14] N. Chiba, T. Nishizeki, S. Abe, and T. Ozawa. A linear algorithm for embedding planar graphs using PQ-trees. *Journal of Computer and System Sciences*, 30:54–76, 1985.

[15] H. N. Djidjev and S. Venkatesan. Reduced constants for simple cycle graph separation. *Acta Informatica*, 34:231–243, 1997.

[16] R. G. Downey and M. R. Fellows. *Parameterized Complexity.* Springer-Verlag, 1999.

[17] R. G. Downey, M. R. Fellows, R. Niedermeier, and P. Rossmanith. Parameterized complexity. Dagstuhl-Seminar-Report No. 316, August 2001.

[18] 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.

[19] D. S. Hochbaum, editor. *Approximation algorithms for NP-hard problems*. Boston, MA: PWS Publishing Company, 1997.

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

[21] T. Kloks. *Treewidth: Computations and Approximations*, volume 842 of *LNCS*. Springer-Verlag, 1994.

[22] J. v. Leeuwen. Graph algorithms. In *Handbook of Theoretical Computer Science, Vol. A: Algorithms and Complexity*, pages 525–631. North Holland, 1990.

[23] R. J. Lipton and R. E. Tarjan. A separator theorem for planar graphs. *SIAM Journal of Applied Mathematics*, 36(2):177–189, 1979.

[24] R. J. Lipton and R. E. Tarjan. Applications of a planar separator theorem. *SIAM Journal on Computing*, 9(3):615–627, 1980.

[25] M. Mahajan and V. Raman. Parameterizing above guaranteed values: MaxSat and MaxCut. *Journal of Algorithms*, 31:335–354, 1999.

[26] K. Mehlhorn and S. Näher. *LEDA: A Platform of Combinatorial and Geometric Computing*. Cambridge University Press, Cambridge, England, 1999.

[27] G. L. Nemhauser and J. L. E. Trotter. Vertex packing: structural properties and algorithms. *Mathematical Programming*, 8:232–248, 1975.

[28] R. Niedermeier and P. Rossmanith. Upper bounds for Vertex Cover further improved. In *Proc. 16th Symposium on Theoretical Aspects of Computer Science STACS'99*, volume 1563 of *LNCS*, pages 561–570. Springer-Verlag, 1999.

[29] V. T. Paschos. A survey of approximately optimal solutions to some covering and packing problems. *ACM Computing Surveys*, 29(2):171–209, 1997.

[30] N. Robertson, D. P. Sanders, P. Seymour, and R. Thomas. Efficiently four-coloring planar graphs. In *Proc. 28th ACM Symposium on the Theory of Computing STOC'96*, pages 571–575, 1996.

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

[32] J. A. Telle and A. Proskurowski. Practical algorithms on partial k-trees with an application to domination-like problems. In *Algorithms and Data Structures, Proc. 3rd WADS'93*, volume 709 of *LNCS*, pages 610–621. Springer-Verlag, 1993.

[33] M. Yannakakis and F. Gavril. Edge dominating sets in graphs. *SIAM Journal on Applied Mathematics*, 38(3):364–372, 1980.

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'