Graph Separators: A Parameterized View

Jochen Alber
Henning Fernau
Rolf Niedermeier

WSI-2001-8

Wilhelm-Schickard-Institut für Informatik
Universität Tübingen
Sand 13
D-72076 Tübingen
Germany

E-Mail: alber,fernau,niedermr@informatik.uni-tuebingen.de
Telefon: (07071) 29-77569/5/8
Telefax: (07071) 29-5061

ISSN 0946-3852

Graph Separators: A Parameterized View*

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

Graph separation is a well-known tool to make (hard) graph problems accessible to a divide and conquer approach. We show how to use graph separator theorems in combination with (linear) problem kernels in order to develop fixed parameter algorithms for many well-known NP-hard (planar) graph problems. We coin the key notion of glueable select&verify graph problems and derive from that a prospective way to easily check whether a planar graph problem will allow for a fixed parameter algorithm of running time $c^{\sqrt{k}} \cdot n^{O(1)}$ for constant c.

Besides, we introduce the novel concept of "problem cores" that might serve as an alternative to problem kernels for devising parameterized algorithms. One of the main contributions of the paper is to exactly compute the base c of the exponential term and its dependence on the various parameters specified by the employed separator theorem and the underlying graph problem. We discuss several strategies to improve on the involved constant c.

Our findings also give rise to studying further refinements of the complexity class FPT of fixed parameter tractable problems.

Keywords: Planar graph problems, fixed parameter tractability, parameterized complexity, graph separators, separator theorems, divide and conquer algorithms.

*A preliminary version of this paper was presented at the 7th Annual International Computing and Combinatorics Conference (COCOON 2001), Springer-Verlag, LNCS 2108, pages 318–327, held in Guilin, China, August 2001.

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

1 Introduction

It is a common fact that algorithm designers are often faced with problems which, when viewed from classical computational complexity theory, are "intractable." More formally speaking, these problems can be shown to be *NP*-hard [24]. In many applications, however, a certain part (called the *parameter*) of the whole problem can be identified which tends to be of small size k when compared with the size n of the whole problem instance. This leads to the study of parameterized complexity [5, 18, 22].

Fixed parameter tractability. Formally, a parameterized problem is a (two-dimensional) language L over $\Sigma^* \times \mathbb{N}$, where Σ is some alphabet. The second coordinate of an element $(I, k) \in L$ is called the *parameter*. We say that L is *fixed parameter tractable* if there exists an algorithm that decides the word problem on input (I, k) running in time $f(k)n^{O(1)}$, where $n = |I|$ and f is an arbitrary function that captures the inherent combinatorial explosion of the problem and that only depends on k. The number k is also called the *parameter* of the problem (instance).[1] The associated complexity class is called FPT. We will also term such algorithms "$f(k)$-algorithms" for brevity, focusing on the exponential part of the running time bound. Typically in the literature, such functions f for fixed parameter problems are $f(k) = c^k$, $f(k) = k^k$, or $f(k) = c^{k^2}$. Of course, designing fixed parameter algorithms with a "small" function f is desirable. To our knowledge, so far, only one non-trivial fixed parameter tractability result where the corresponding function f is sublinear in the exponent, namely $f(k) = c^{\sqrt{k}}$ is known [1] : Dominating Set on planar graphs. Similar results hold for closely related problems on planar graphs such as Face Cover, Independent Dominating Set, Weighted Dominating Set, etc. [1]. In a companion paper, we improved this result to apply to a much broader class of planar graph problems, presenting a general methodology based on concepts such as tree decompositions and bounded outerplanarity and introducing the novel so-called "Layerwise Separation Property" as a key unifying tool [3]. Here, we will also discuss a rather general approach for obtaining parameterized graph algorithms running in time $O(c^{e(k)} \cdot q(n))$ for sublinear functions e, i.e., $e(k) \in o(k)$. By way of contrast, however, we investigate the usefulness of (planar) separator theorems in this context, yielding a new, alternative and conceptually rather different framework in comparison with [3].

Scope of the paper. Up to now, several interesting but specialized fixed-

[1] In the applications encountered in this paper, the parameter will always be a number. This number is encoded in unary, so that we need not distinguish between the parameter and its size.

parameter algorithms have been developed. The thrust has been to improve running times in a problem-specific manner, e.g., by extremely sophisticated case distinctions as can be seen in the case of VERTEX COVER and its variants [12, 23, 31, 32, 33, 39]. It is a crucial goal throughout the paper not to narrowly stick to problem-specific approaches, but to try to widen the techniques as far as possible. More specifically, we show how to use separator theorems for different graph classes, such as, e.g., the well-known planar separator theorem due to Lipton and Tarjan [28], in combination with known algorithms for obtaining linear size problem kernels (such "small" kernels are known, e.g., for VERTEX COVER, see Subsection 2.1), in order to obtain fixed parameter algorithms.

Our approach can be sketched as follows: We will apply the problem kernel reduction and, then, use (planar) separator theorems as already Lipton and Tarjan [29] do in order to pursue a divide and conquer strategy on the set of reduced instances. We do, however, take much more care for the dependence of the "graph separator parameters" on the recurrences in the running time analysis. In addition, we obviously consider a broader class of problems that can be attacked by this approach (namely, in principle, all so-called glueable select&verify problems such as, e.g., DOMINATING SET[2]). Doing so, we exhibit the importance of a special form of separators, so-called cycle separators and their influence on the running time analysis. Moreover, we show how to employ different separator-finding strategies in order to get divide and conquer algorithms with constants which are better than those corresponding to a direct use of the best known (planar) separator theorems. Finally, we discuss possible combinations of separator techniques with other solving methods (like search tree based algorithms), typically leading to $c^{k^{2/3}}$-algorithms for problems with linear kernels. The running time of our algorithms is mostly bounded by $c^{\sqrt{k}}q(n)$ or by $c^{k^{2/3}}q(n)$ for some constant $c > 1$ and some polynomial $q(\cdot)$.[3] Although the constants achieved in our setting so far seem to be too large in order to yield practical worst case algorithms, it provides a general and sound mathematical formalization of a rich class of problems that allow for divide and conquer solutions based on separators and it provides a strong link to fixed parameter tractability. Also, our methodology seems to leave much room for improvement in many directions. For instance, we introduce the novel concept of "problem cores" that can replace problem kernels in our setting. Furthermore, our findings em-

[2]Lipton and Tarjan only describe in details a solution for the structurally much simpler INDEPENDENT SET.

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

phasize the importance of small (linear size) problem kernels, and they give a push to the study of subclasses of the parameterized complexity class FPT. In this sense, our work might serve as a starting point for more algorithmic (including graph theory with respect to separator theorems), as well as more structural complexity-theoretic lines of future research in parameterized complexity.

2 Basic definitions and preliminaries

We start with some basic notation used throughout the paper assuming familiarity with elementary concepts of algorithms, complexity, and graph theory. We consider undirected graphs $G = (V, E)$, where V denotes the vertex set and E denotes the edge set. In our setting, all graphs are simple (i.e., with no double edges) without self-loops. Sometimes, we refer to V by $V(G)$ in order to emphasize that V is the vertex set of graph G; by $N(v)$ we refer to the set of vertices adjacent to v. $G[D]$ denotes the subgraph induced by vertex set D. For graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, by $G_1 \cap G_2$, we denote the graph with vertex set $V_1 \cap V_2$ and edge set $E_1 \cap E_2$. A graph $G' = (V', E')$ is a subgraph of $G = (V, E)$, denoted by $G' \subseteq G$, if $V' \subseteq V$ and $E' \subseteq E$. The *radius* of a graph G is the minimal height of a rooted spanning tree of G.

In this paper, we only consider graph classes, denoted by $\mathbb{G}$, that are closed under taking subgraphs. The most important among the graph classes considered in this paper is that of *planar* graphs, i.e., graphs that have a drawing in the plane without edge crossings. A *plane* graph, or a *planar embedding,* is any drawing of a planar graph in the plane without edge crossing.

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 planar.[4]

If $G = (V, E)$ is a planar graph, a *triangulation* $\hat{G} = (V, \hat{E})$, where $E \subseteq \hat{E}$, of G is a planar graph such that, for any additional edge $\{v, v'\} \notin \hat{E}$ with $v, v' \in V$, $(V, \hat{E} \cup \{v, v'\})$ is not planar. The *faces* of a plane graph are the maximal regions of the plane that contain no point used in the embedding. Among others, we study the following "graph numbers" $vc(\cdot)$, $is(\cdot)$, and $ds(\cdot)$:

- A *vertex cover* C of a graph G is a set of vertices such that every edge

[4]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 needs not be given with an embedding.

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 VERTEX COVER, INDEPENDENT SET, and DOMINATING SET.

Finally, to simplify notation, we write $A + B$ to denote the disjoint union of sets A and B.

2.1 Linear problem kernels

Compared with other definitions of problems kernels, we seemingly have to be a bit more restrictive and precise in the following definition.

Definition 1. Let $\mathcal{L}$ be a parameterized problem, that is, $\mathcal{L}$ consists of pairs (I, k), where problem instance I has a solution of size k (the parameter). *Reduction to problem kernel*, then, means to replace instance (I, k) by a "reduced" instance (I', k') (which we call *problem kernel*) such that

$$k' \leq c \cdot k, \qquad |I'| \leq p(k)$$

with a constant c,[5] some function p only depending on k, 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') is computable in polynomial time $T_K(|I|, k)$.[6] The *size* of the problem kernel is given by $p(k)$.

[5] Usually, $c \leq 1$. In general, 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.

[6] 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.

Often (cf. the subsequent example VERTEX COVER), the best one can hope for is that the problem kernel has size linear in k, a so-called *linear problem kernel*. For instance, using a theorem of Nemhauser and Trotter [30], (also cf. [10, 34]), Chen *et al.* [12] 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 $(2-\varepsilon)k$ with constant $\varepsilon > 0$ would probably imply a factor $2-\varepsilon$ polynomial time approximation algorithm for VERTEX COVER, which would mean a major breakthrough in approximation algorithms for VERTEX COVER [27]. More precisely, we can observe:

Remark 2. Consider VERTEX COVER. If the reduction of (G, k) to (G', k') guarantees that $G' \subseteq G$, $k' \leq k$, $|V(G')| \leq dk$ for some $d < 2$ and when the reduction algorithm yields, in addition, a set $\tilde{C}$ of vertices of G which belongs to some optimal cover of G, then $V(G') \cup \tilde{C}$ is a factor d polynomial time approximation.

This observation was basically exploited by Hochbaum [26] in her factor 2 approximation algorithm which uses the mentioned theorem of Nemhauser and Trotter, see also Bar-Yehuda and Even for related applications of the mentioned theorem [10].

Remarkably, for VERTEX COVER on planar graphs, better approximation algorithms are known [10], as well as a polynomial time approximation scheme [9].

By making use of the four color theorem for planar graphs and its corresponding algorithm generating a four coloring [36], it easily follows [3] that INDEPENDENT SET on planar graphs has a problem kernel of size $4k$. In general, however, it is a reasonable and challenging task to try to construct a linear problem kernel. Ongoing work tries to do this for DOMINATING SET.

Besides the positive effect of reducing the input size significantly and all obvious consequences of that, this paper gives further justification, in particular, for the importance of size $O(k)$ problem kernels. The point is that, once having a linear size problem kernel, e.g., for VERTEX COVER or INDEPENDENT SET on planar graphs, it is fairly easy to use our framework to get $c^{\sqrt{k}}$-algorithms for these problems based upon the famous planar separator theorem [28, 29]. The constant factor in the problem kernel size directly influences the value of the exponential base. Hence, lowering the kernel size means improved efficiency.

2.2 Classical separator theorems

Definition 3. Let $G = (V, E)$ be an undirected graph. A *separator* $S \subseteq V$ of G divides V into two *parts* $A_1 \subseteq V$ and $A_2 \subseteq V$ such that[7]

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

Later, we will write δA_1 (or δA_2) as shorthand for $A_1 + S$ (or $A_2 + S$, respectively). The triple (A_1, S, A_2) is also called a *separation* of G.

Clearly, this definition can be generalized to the case where a separator partitions the vertex set into ℓ subsets instead of only two. We refer to such separators simply by *ℓ-separator*. The techniques we develop here all are based on the existence of "small" graph separators. Here, "small" means that $|S|$ is bounded by $o(|V|)$.

Definition 4. According to Lipton and Tarjan [28], an *$f(\cdot)$-separator theorem (with constants $\alpha < 1$, $\beta > 0$)* for a class $\mathbb{G}$ of graphs which is closed under taking subgraphs is a theorem of the following form: If G is any n-vertex graph in $\mathbb{G}$, then there is a separation (A_1, S, A_2) of G such that

- neither A_1 nor A_2 contains more than αn vertices, and
- S contains no more than $\beta f(n)$ vertices.

Again, this definition easily generalizes to ℓ-separators with $\ell > 2$. We will be more concrete on 3-separators in Subsection 5.3.

Stated in this framework, the planar separator theorem due to Lipton and Tarjan [28] is a $\sqrt{\cdot}$-separator theorem with constants $\alpha = 2/3$ and $\beta = 2\sqrt{2} \approx 2.83$. Later, Djidjev [13] showed an improved planar $\sqrt{\cdot}$-separator theorem with constants $\alpha = 2/3$ and $\beta = \sqrt{6} \approx 2.45$, which was furhter improved to $\alpha = 2/3$ and $\beta = \sqrt{4.5} \approx 2.12$ by Alon *et.al.* [7]. The current record for $\alpha = 2/3$ is $\beta = \sqrt{2/3} + \sqrt{4/3} \approx 1.97$ [17]. Djidjev has also shown a lower bound of $\beta \approx 1.55$ for $\alpha = 2/3$ [13]. For $\alpha = 1/2$, the "record" of $\beta = 7 + 1/\sqrt{3} \approx 7.58$ due to Venkatesan [43] was recently outperformed by Bodlaender [11], yielding $\beta = 2\sqrt{6} \approx 4.90$. A lower bound of $\beta \approx 1.65$ is known in this case [38]. For $\alpha = 3/4$, the best known value for β is $\sqrt{2\pi/\sqrt{3}} \cdot (1 + \sqrt{3})/\sqrt{8} \approx 1.84$ with a known lower bound of $\beta \approx 1.42$, see [38]. The results are summarized in Table 1.

[7] 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.

	$\alpha = \frac{2}{3}$		$r(\frac{2}{3}, \beta)$	$\alpha = \frac{1}{2}$		$r(\frac{1}{2}, \beta)$	$\alpha = \frac{1}{3}$		$r(\frac{1}{3}, \beta)$
upper bounds for β	$2\sqrt{2}$	[28]	15.41	$7 + \frac{1}{\sqrt{3}}$	[43]	25.87	$\sqrt{\frac{2\pi}{\sqrt{3}}} \cdot \frac{1+\sqrt{3}}{\sqrt{8}}$	[38]	13.73
	$\sqrt{6}$	[13]	13.35	$\sqrt{24}$	[11]	16.73			
	$\sqrt{4.5}$	[7]	11.56						
	$\sqrt{\frac{2}{3}} + \sqrt{\frac{4}{3}}$	[17]	10.74						
lower bounds for β	1.55	[13]	8.45	1.65	[38]	5.63	1.42	[38]	10.60

Table 1: Summary of various $\sqrt{\cdot}$-separator theorems with their constants α and β. Here, $r(\alpha, \beta)$ denotes the ratio $r(\alpha, \beta) = \beta/(1 - \sqrt{\alpha})$, which is of central importance to the running time analysis of our algorithms, cf. Proposition 20.

In order to develop a flexible framework, we will do our calculations below always with the parameters α and β left unspecified up to the point where we try to give concrete numbers in the case of Vertex Cover on planar graphs, which will serve as a running example. Also, we point out how the existence of ℓ-separators for $\ell > 2$ might improve the running time. In principle, our results also apply to graph problems for graphs from other graph classes with $\sqrt{\cdot}$-separator theorems as listed above. As indicated in [35], separator based techniques can be also used to solve counting problems instead of decision problems.

As we will see, for developing efficient fixed parameter algorithms, $\sqrt{\cdot}$-separator theorems are especially interesting in the case when a linear size problem kernel is known.

Variants of separator theorems

Cycle separators. In the literature, there are many separator theorems for planar graphs which guarantee that all the vertices of the separator lie on a simple cycle, provided that the given graph is biconnected or even triangulated. In fact, the current "record holder" in the case of $\alpha = 2/3$ yields a cycle separator, see [17]. From an algorithmic perspective, as explained below, the requirements of having biconnected or triangulated graphs are rarely met: even if the original graph was biconnected or triangulated, subgraphs which are obtained by recursive applications of separator theorems to a larger graph are not biconnected or triangulated in general. Therefore, we consider the following definition appropriate for our purposes:

Definition 5. We will call a separator S of a planar graph G *cycle separator*

if there exists a triangulation $\hat{G}$ of G such that S forms a simple cycle in $\hat{G}$.

Note that some triangulation of a given planar graph can be computed in linear time.

Remark 6. It will turn out that it is of special value (concerning the design of divide and conquer algorithms) to have separators that form simple cycles (within some triangulation of the given graph G), since then the Jordan curve theorem applies (for planar graphs), which basically means that the separator S splits G into an "inside"-part A_1 and an "outside"-part A_2. If the graph is a subgraph of a larger planar graph $\tilde{G}$, then this implies that each vertex v of $\tilde{G}$ that has neighbors in A_1 has no neighbors in A_2 and vice versa. This observation is important, since it means that a local property pertaining to vertex v of $\tilde{G}$ (like: v belongs to a dominating set or not) can only influence vertices in δA_1 or vertices in δA_2.

Weighted separation. It is also possible to incorporate *weights* in most separator theorems. For our purposes, weights are nonnegative reals assigned to the vertices in a graph such that the sum of all weights in a graph is bounded by one. For weighted graphs, an $f(\cdot)$-separator theorem with constants α and β for graph class $\mathbb{G}$ guarantees, for any n-vertex graph $G \in \mathbb{G}$, the existence of a separation (A_1, S, A_2) of G such that

- neither A_1 nor A_2 has weight more than α, and
- S contains no more than $\beta f(n)$ vertices.

Remark 7. It might be the case that, for fixed α, good $\sqrt{\cdot}$-separator theorems for weighted graphs have worse constants β than their unweighted counterparts. For example, the current record for $\alpha = 2/3$ is $\beta = 2$ for weighted graphs [17].

Other graph classes with separator theorems. Similar to the case of planar graphs, $\sqrt{\cdot}$-separator theorems are also known for other graph classes, e.g., for the class of graphs of bounded genus, see [15]. More generally, Alon, Seymour and Thomas proved a $\sqrt{\cdot}$-separator theorem for graph classes with an excluded complete graph minor [6, 8]. Many comments of this paper apply to these more general situations, too.

Conversely, to find separators is not possible in general (if arbitrary input graphs are permitted), as the example of the complete graph K_n with n vertices shows.

3 Glueable graph problems

Based on the notion of separators, we will give a characterization of a whole class of problems that can be attacked by the approach that will be described in the subsequent sections. To this end, we coin the notions of select&verify graph problems and glueability. These notions are central to this paper. In the companion paper [3], we also introduce select&verify graph problems. Since the algorithms from [3] are not recursive, only a simplified notion of glueability (termed "weak glueability") is needed there.

3.1 Select&verify graph problems

Definition 8. 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 $(P_\cdot, \mathrm{opt})$ with $\mathrm{opt} \in \{\min, \max\}$, such that $P_\cdot$ 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

$$P_G^{\mathrm{sel}} : \{0,1\}^n \to \mathbb{R}_+, \text{ (also called } \textit{selecting function}\text{)}$$
$$P_G^{\mathrm{ver}} : \{0,1\}^n \to \{0, \pm\infty\}, \text{ (also called } \textit{verifying function}\text{) and}$$

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

For $\vec{x} = (x_1, \ldots, x_n) \in \{0,1\}^n$, with $P_G(\vec{x}) \le k$ if opt $= \min$ and with $P_G(\vec{x}) \ge k$ if opt $= \max$, the vertex set *selected* by $\vec{x}$ and *verified* by P_G is

$$\{v_i \in V \mid x_i = 1, 1 \le i \le n\}.$$

A vector $\vec{x}$ is called *admissible* if $P_G^{\mathrm{ver}}(\vec{x}) = 0$.

The intuition behind the term $P_\cdot = P_\cdot^{\mathrm{sel}} + P_\cdot^{\mathrm{ver}}$ is that the selecting function $P_\cdot^{\mathrm{sel}}$ counts the size of the selected set of vertices and the verifying function $P_\cdot^{\mathrm{ver}}$ verifies whether this choice of vertices is an admissible solution. The "numbers" $\pm\infty$ indicate the non-admissibility of a candidate solution.

Remark 9. 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 10. We now give some examples for select&verify problems by specifying the function $P_G = P_G^{\mathrm{sel}} + P_G^{\mathrm{ver}}$. In all cases below, the selecting function P_G for a graph $G = (V, E)$ will be

$$P_G^{\mathrm{sel}}(\vec{x}) = \sum_{v_i \in V} x_i.$$

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

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

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

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

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

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

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

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.[8] As a source of problems, consider the variants of DOMINATING SET listed in [40, 41, 42]. In particular, the TOTAL DOMINATING SET problem is defined by

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

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 NP-complete EDGE DOMINATING SET [44] 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 or by introducing additional "face vertices."

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

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 11. 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

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

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

3.2 Glueability

We are going to solve graph problems recursively, 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. For example, in the case of Vertex Cover, this means that, for a separator S, all possible functions $S \to \{0,1\}$ are tested, where assigning the "color" 0 means that the corresponding vertex is *not* in the (partial) cover and assigning 1 means that the corresponding vertex lies in the (partial) cover. In the case of more involved problems like (variants of) Dominating Set, more sophisticated assignments are necessary, as detailed below. The separators will serve as boundaries between the different graph parts 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 remaining graph parts and then reconstruct a solution for the whole graph by "gluing" together the solutions for the graph parts. 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 handed on is pretty clear in the case of Vertex Cover, but it is much more involved in the case of Dominating Set and many others. This is the basic motivation for the formal framework we develop in this subsection. 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 Dominating Set), 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"). This is why the set of colors will be some union of finite sets $C_0 + C_1$, instead of $\{0,1\}$ only. The usefulness of considering a colored version of "normal" graph problems is also testified in [2], where a colored version of Dominating Set

on planar graphs was employed for developing a parameterized search-tree algorithm.

Definition 12. Let $G = (V, E)$ be an undirected graph and C_0, C_1 be finite, disjoint sets. A C_0-C_1-*coloring* of G is a function $\chi : V \to C_0 + C_1 + \{\#\}$.

The symbol # will be used for the undefined (i.e., not yet defined) color. This means that, 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 13. Consider an instance (G, k) of a select&verify problem $\mathcal{G}$ and a vector $\vec{x} \in \{0,1\}^n$ with $V(G) = \{v_1, \ldots, v_n\}$. Let χ be a C_0-C_1-coloring of G. Then, $\vec{x}$ is *consistent* with χ, written $\vec{x} \sim \chi$, if

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

In the next section, when doing the divide and conquer approach with a given separator, we will deal with colorings on two different color sets: one color set $C^{\text{int}} := C_0^{\text{int}} + C_1^{\text{int}} + \{\#\}$ of *internal* colors that will be used for the separator assignments and a color set $C^{\text{ext}} := C_0^{\text{ext}} + C_1^{\text{ext}} + \{\#\}$ of *external* colors that will be used for handing down the information in the divide-step of the algorithm. The idea is that, in each recursive step, we will be confronted with a graph "precolored" with external colors. Our algorithm then finds a new separator and assigns internal colors to the vertices of this separator. These assignments of internal colors should be, in some sense, "compatible" with the precolored graph. Moreover, we want to be able to "recolor" the graph for the next divide-step. That means that we somehow have to be able to merge an external coloring (from the precolored graph) with an internal coloring (i.e., an assignment of our current separator) in a way such that we obtain a new (compatible) external coloring that can be handed down in the next recursive step. These considerations are formalized as follows.

Definition 14. Let $G = (V, E)$ be a graph and let C_0^{int}, C_1^{int} and C_0^{ext}, C_1^{ext} be mutually disjoint, finite sets. Let $C^{\text{int}} := C_0^{\text{int}} + C_1^{\text{int}} + \{\#\}$ and let $C^{\text{ext}} := C_0^{\text{ext}} + C_1^{\text{ext}} + \{\#\}$.

If χ is a C_0-C_1-coloring of G and if χ' is a C_0'-C_1'-coloring of G, then χ is *preserved* by χ', written $\chi \rightsquigarrow \chi'$, if

$$\forall v \in V\, \forall i = 0, 1\, (\chi(v) \in C_i \Rightarrow \chi'(v) \in C_i').$$

Every function $\oplus$ that assigns to a pair $(\chi^{\text{ext}}, \chi^{\text{int}})$ with $\chi^{\text{ext}} : V \to C^{\text{ext}}$, $\chi^{\text{int}} : V \to C^{\text{int}}$, $\chi^{\text{ext}} \rightsquigarrow \chi^{\text{int}}$, a $(C_0^{\text{ext}}$-$C_1^{\text{ext}})$-coloring $\chi^{\text{ext}} \oplus \chi^{\text{int}}$ is called a *recoloring* if $\chi^{\text{int}} \rightsquigarrow \chi^{\text{ext}} \oplus \chi^{\text{int}}$.

From the point of view of recursion, χ^{ext} is the pre-coloring which a certain recursion instance "receives" from the calling instance and χ^{int} represents coloring which this instance assigns to a certain part of the graph. The coloring $\chi^{\text{ext}} \oplus \chi^{\text{int}}$ is handed down in the recursion. The notions $\chi^{\text{ext}} \rightsquigarrow \chi^{\text{int}}$ and $\chi^{\text{int}} \rightsquigarrow \chi^{\text{ext}} \oplus \chi^{\text{int}}$ express that any vector $\vec{x} \in \{0,1\}^{|V|}$ that is consistent with χ^{ext} is also consistent with the colorings χ^{int} and $\chi^{\text{ext}} \oplus \chi^{\text{int}}$.

We now introduce the central notion of "glueable" select&verify problems. This formalizes those problems that can be solved with separator based divide and conquer techniques as described above. We apply this rather abstract notion to concrete graph problems afterwards (Lemma 16). The following definition is best understood with the help of a concrete example as given in the proof of Lemma 16.

Definition 15. A select&verify problem $\mathcal{G}$ given by $(P., \text{opt})$ is *glueable with σ colors* if there exist

- a color set $C^{\text{int}} := C_0^{\text{int}} + C_1^{\text{int}} + \{\#\}$ of internal colors with $|C_0^{\text{int}} + C_1^{\text{int}}| = \sigma$;
- a color set $C^{\text{ext}} := C_0^{\text{ext}} + C_1^{\text{ext}} + \{\#\}$ of external colors;
- a polynomial time computable function $h : (\mathbb{R}_+ \cup \{\pm\infty\})^3 \to \mathbb{R}_+ \cup \{\pm\infty\}$;

and if, for every n-vertex graph $G = (V, E)$ and subgraphs $G^{\text{ver}}, G^{\text{sel}} \subseteq G$ with a separation (A_1, S, A_2) of G^{ver}, we find

- recolorings $\oplus_X$ for each $X \in \{A_1, S, A_2\}$, and
- for each internal coloring $\chi^{\text{int}} : S \to C^{\text{int}}$,

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

  subgraphs $G_S^{\text{ver}}(\chi^{\text{int}})$ of G^{ver} with $G_S^{\text{ver}}(\chi^{\text{int}}) \subseteq G^{\text{ver}}[S]$

such that, for each external coloring $\chi^{\text{ext}} : V \to C^{\text{ext}}$,

$$\begin{aligned} &\text{opt}_{\substack{\vec{x} \in \{0,1\}^n \\ \vec{x} \sim \chi^{\text{ext}}}} P_{G^{\text{ver}}}(\vec{x} \mid G^{\text{sel}}) \qquad (1) \\ &= \text{opt}_{\substack{\chi^{\text{int}} : S \to C_0^{\text{int}} + C_1^{\text{int}} \\ \chi^{\text{ext}} \rightsquigarrow \chi^{\text{int}}}} h\big(\text{Eval}_{A_1}(\chi^{\text{int}}), \text{Eval}_S(\chi^{\text{int}}), \text{Eval}_{A_2}(\chi^{\text{int}})\big). \end{aligned}$$

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

$$\text{Eval}_X(\chi^{\text{int}}) \quad = \quad \text{opt}_{\substack{\vec{x} \in \{0,1\}^n \\ \vec{x} \sim (\chi^{\text{ext}} \oplus_X \chi^{\text{int}})}} P_{G_X^{\text{ver}}(\chi^{\text{int}})}(\vec{x} \mid G^{\text{ver}}[X] \cap G^{\text{sel}}). \qquad (2)$$

Lemma 16. Vertex Cover *and* Independent Set *are glueable with 2 colors and* Dominating Set *is glueable with 4 colors.*

Proof. For Vertex Cover (see Example 10.1)), we use the color sets $C_i^\ell := \{i^\ell\}$ for $\ell \in \{\text{int}, \text{ext}\}$ and $i = 0, 1$. The function h is $h(x, y, z) = x + y + z$. The subgraphs $G_X^{\text{ver}}(\chi^{\text{int}})$ for $X \in \{A_1, S, A_2\}$ and $\chi^{\text{int}} : S \to C_0^{\text{int}} + C_1^{\text{int}}$ are $G_X^{\text{ver}}(\chi^{\text{int}}) := G^{\text{ver}}[X]$. In this way, the subroutine $\text{Eval}_S(\chi^{\text{int}})$ checks whether the coloring χ^{int} yields a vertex cover on $G^{\text{ver}}[S]$ and the subroutines $\text{Eval}_{A_i}(\chi^{\text{int}})$ compute the minimum size vertex cover on $G^{\text{ver}}[A_i]$. However, we still need to make sure that all edges going from A_i to S are covered.[9] If a vertex in S is assigned a 1^{int} by χ^{int}, the incident edges are already covered. In the case of a 0^{int}-assignment for a vertex $v \in S$, we can color all neighbors in $N(v) \cap A_i$ to belong to the vertex cover. This is done by the following recolorings $\oplus_{A_i}$. Define

$$(\chi^{\text{ext}} \oplus_{A_i} \chi^{\text{int}})(v) = \begin{cases} 0^{\text{ext}} & \text{if } \chi^{\text{int}}(v) = 0^{\text{int}}, \\ 1^{\text{ext}} & \text{if } \chi^{\text{int}}(v) = 1^{\text{int}} \text{ or} \\ & \text{if } \exists w \in N(v) \text{ with } \chi^{\text{int}}(w) = 0^{\text{int}}, \\ \#, & \text{otherwise.} \end{cases}$$

By this recoloring definition, an edge between a separator vertex and a vertex in A_i which is not covered by the separator vertex (due to the currently considered internal covering) will be covered by the vertex in A_i. Our above reasoning shows that—with these settings—Equation (1) in Definition 15 is satisfied.

Independent Set (see Example 10.2)) is shown to be glueable with 2 colors by a similar idea.

To show that Dominating Set (see Example 10.3)) is glueable with 4 colors, we use the following color sets

$$C_0^{\text{int}} := \{0_{A_1}^{\text{int}}, 0_{A_2}^{\text{int}}, 0_S^{\text{int}}\}, \quad C_1^{\text{int}} := \{1^{\text{int}}\},$$
$$C_0^{\text{ext}} := \{0^{\text{ext}}\}, \qquad\qquad C_1^{\text{ext}} := \{1^{\text{ext}}\}.$$

The semantics of these colors is as follows. Assigning the color 0_X^{int}, for $X \in \{A_1, A_2, S\}$, to vertices in a current separation $V = A_1 + S + A_2$ means that the vertex is not in the dominating set and will be dominated by a vertex in X. Clearly, 1^{int} will mean that the vertex belongs to the dominating set. The external colors simply hand down the information whether a vertex

[9] We could have coped with this by letting $G_{A_i}^{\text{ver}}(\chi^{\text{int}}) := G^{\text{ver}}[A_i \cup (\chi^{\text{int}})^{-1}(\{0^{\text{int}}\})]$. In this way, the computation of $\text{Eval}_{A_i}(\chi^{\text{int}})$ checks whether all neighbors in A_i of a vertex in $(\chi^{\text{int}})^{-1}(\{0^{\text{int}}\})$ are covered. The disadvantage of this solution is that the graphs $G_{A_i}^{\text{ver}}(\chi^{\text{int}})$ that are handed down to the subroutines become unnecessarily big.

belongs to the dominating set, represented by 1^{ext}, or whether it is not in the dominating set *and* still needs to be dominated, represented by 0^{ext}.[10] The function h simply is addition, i.e., $h(x,y,z) = x+y+z$. When handing down the information to the subproblems, for a given internal coloring $\chi^{\text{int}} : S \to C_0^{\text{int}} + C_1^{\text{int}}$, we define

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

The recolorings $\oplus_X$ for $X \in \{A_1, S, A_2\}$ are chosen to be

$$(\chi^{\text{ext}} \oplus_X \chi^{\text{int}})(v) = \begin{cases} 0^{\text{ext}} & \text{if } \chi^{\text{int}}(v) \in C_0^{\text{int}}, \\ 1^{\text{ext}} & \text{if } \chi^{\text{int}}(v) = 1^{\text{int}}, \\ \#, & \text{otherwise.} \end{cases}$$

Let us explain in a few lines why—with these settings—Equation (1) in Definition 15 is satisfied. If an internal coloring χ^{int} assigns color 0^{int}_X ($X \in \{A_1, S, A_2\}$) to a vertex in S, then this vertex needs to be dominated by a neighbor in X. This will be checked in $\text{Eval}_X(\chi^{\text{int}})$ using the graph $G^{\text{ver}}_X(\chi^{\text{int}})$. To this end, vertices assigned the color 0^{int}_X (i.e., the set $(\chi^{\text{int}})^{-1}(\{0^{\text{int}}_X\})$) are included in $G^{\text{ver}}_X(\chi^{\text{int}})$. The vertices assigned color 1^{int} (i.e., $(\chi^{\text{int}})^{-1}(\{1^{\text{int}}\})$) also need to be handed down to the subroutines, since such a vertex may already dominate vertices in X. The recolorings merge the given external coloring χ^{ext} with the current internal coloring χ^{int} in a way that already assigned colors from C_i^{int} or C_i^{ext} ($i = 0, 1$) become i^{ext}. The terms $\text{Eval}_{A_i}(\chi^{\text{int}})$ then compute (for each internal coloring χ^{int}) 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 $\delta A_i \cap (\chi^{\text{ext}} \oplus_{A_i} \chi^{\text{int}})^{-1}(0^{\text{ext}})$) and some vertices in δA_i can already be assumed to be in the dominating set (namely, the vertices in $\delta A_i \cap (\chi^{\text{ext}} \oplus_{A_i} \chi^{\text{int}})^{-1}(1^{\text{ext}})$). The term $\text{Eval}_S(\chi^{\text{int}})$ checks the correctness of the internal coloring χ^{int} of S. □

Note that, from the point of view of divide and conquer algorithms, three colors are enough for Dominating Set, since the color 1^{int} already determines the color 0^{int}_S. This issue is detailed in [3].
We illustrate the ideas of the dominating set algorithm by using an example.

Example 17. Consider Dominating Set for the separated graph in Fig. 1. Beginning with the external coloring $\chi^{\text{ext}} \equiv \#$ and $G^{\text{ver}} = G^{\text{sel}} = G$, we need

[10] A vertex that is not in the dominating set but is already guaranteed to be dominated, e.g., by a vertex in the current separator, will never be handed down, since these vertices are of no use in the sequel of the recursion.

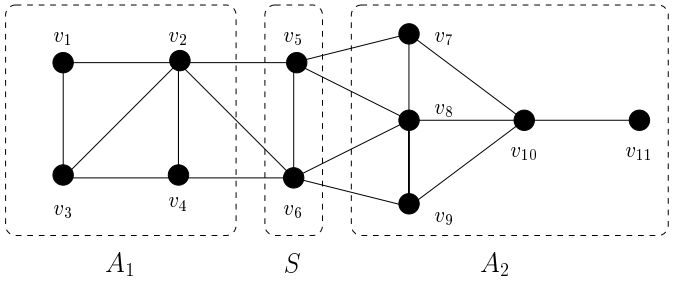

Figure 1: A partitioned graph.

to go over all $4^2 = 16$ internal colorings $\chi^{\text{int}} : S \to \{0^{\text{int}}_{A_1}, 0^{\text{int}}_{S}, 0^{\text{int}}_{A_2}, 1^{\text{int}}\}$ (which trivially satisfy $\chi^{\text{ext}} \rightsquigarrow \chi^{\text{int}}$). As an example, we choose χ^{int} with $v_5 \mapsto 0^{\text{int}}_{A_1}$, $v_6 \mapsto 0^{\text{int}}_{A_2}$. In this case, we get $G^{\text{ver}}_{A_1}[\chi^{\text{int}}] = G^{\text{ver}}[\{v_1, \ldots, v_5\}]$, $G^{\text{ver}}_{S}[\chi^{\text{int}}] = \emptyset$, and $G^{\text{ver}}_{A_2}[\chi^{\text{int}}] = G^{\text{ver}}[\{v_6, \ldots, v_{11}\}]$. In the recursive steps, we will use these graphs for the verifying function and the graphs $G^{\text{ver}}[A_1]$, $G^{\text{ver}}[S]$, and $G^{\text{ver}}[A_2]$ for the selecting function. Moreover, the external colorings that will be handed down to the subproblems after the recoloring look as follows: On the graph $G^{\text{ver}}_{A_1}[\chi^{\text{int}}]$, we have $\chi_1(v_i) := (\chi^{\text{ext}} \oplus_{A_1} \chi^{\text{int}})(v_i) = \#$ for $i = 1, \ldots, 4$, and $\chi_1(v_5) = 0^{\text{ext}}$. On the graph $G^{\text{ver}}_{A_2}[\chi^{\text{int}}]$, we have $\chi_2(v_i) := (\chi^{\text{ext}} \oplus_{A_2} \chi^{\text{int}})(v_i) = \#$ for $i = 7, \ldots, 11$, and $\chi_2(v_6) = 0^{\text{ext}}$. It is easy to see that

$$\text{Eval}_{A_1}(\chi^{\text{int}}) = 2, \quad \text{Eval}_{S}(\chi^{\text{int}}) = 0, \quad \text{and} \quad \text{Eval}_{A_2}(\chi^{\text{int}}) = 2.$$

The minimum in Equation (2) for $X = A_1$ is obtained, e.g., by choosing the vertices v_1 and v_2 (note that the latter needs to be chosen, since $\chi_1(v_5) = 0^{\text{ext}}$, meaning that v_5 is forced to be dominated in this term). The minimum for A_2 is obtained, e.g., by choosing the vertices v_8 and v_{10} (again, $\chi_2(v_6) = 0^{\text{ext}}$ forces either v_8 or v_9 to be in the dominating set). Hence,

$$h(\text{Eval}_{A_1}(\chi^{\text{int}}), \text{Eval}_{S}(\chi^{\text{int}}), \text{Eval}_{A_2}(\chi^{\text{int}})) = 2 + 0 + 2 = 4.$$

We obtain an optimal result, e.g., for the choice of the internal coloring χ^{int} with $\chi^{\text{int}}(v_5) = \chi^{\text{int}}(v_6) = 0^{\text{int}}_{A_2}$. Here, we get $\text{Eval}_{A_1}(\chi^{\text{int}}) = 1$, $\text{Eval}_{S}(\chi^{\text{int}}) = 0$, and $\text{Eval}_{A_2}(\chi^{\text{int}}) = 2$, for the possible choices of v_3, v_8, v_{10} as dominating set vertices.

We want to mention in passing that—besides the problems stated in the preceding Lemma 16—many more select&verify problems are glueable, for example, those which are listed in [40, 41, 42]. In particular, weighted versions and variations of the problems discussed in Lemma 16 are glueable.

Note that Total Dominating Set is an example of a graph problem where a color set C^{int}_1 of more than one color is needed.

4 Fixed parameter divide and conquer algorithms

In this section, we provide the basic framework for deriving fixed parameter algorithms based on the concepts we introduced so far. Moreover, we define problem cores, a potential alternative to problem kernels.

4.1 Using glueability for divide and conquer

The notion of glueable select&verify problems is tailored in a way such that divide and conquer approaches can be used to solve these kinds of problems.

Fix a graph class $\mathbb{G}$ for which a $\sqrt{\cdot}$-separator theorem with constants α and β (cf. Definition 4) is known. We consider a glueable select&verify graph problem $\mathcal{G}$ defined by $(P_\cdot, \mathrm{opt})$. The evaluation of the term $\mathrm{opt}_{\vec{x}\in\{0,1\}^n} P_G(\vec{x})$ (cf. Definition 8) can be done recursively according to the following strategy.

Algorithm 18. 1. Start the computation by evaluating

$$\mathrm{opt}_{\vec{x}\in\{0,1\}^n} P_G(\vec{x}) \;=\; \mathrm{opt}_{\vec{x}\in\{0,1\}^n,\, \vec{x}\sim\chi_0^{\mathrm{ext}}} P_{G^{\mathrm{ver}}}(\vec{x} \mid G^{\mathrm{sel}}),$$

where "$\chi_0^{\mathrm{ext}} \equiv \#$" is the everywhere undefined external coloring and $G^{\mathrm{ver}} = G^{\mathrm{sel}} = G$ (also cf. Definition 11).

2. When $\mathrm{opt}_{\vec{x}\in\{0,1\}^n,\, \vec{x}\sim\chi^{\mathrm{ext}}} P_{G^{\mathrm{ver}}}(\vec{x} \mid G^{\mathrm{sel}})$ needs to be calculated for some subgraphs $G^{\mathrm{sel}}, G^{\mathrm{ver}} \subseteq G$, and an external coloring $\chi^{\mathrm{ext}} : V(G) \to C_0^{\mathrm{ext}} + C_1^{\mathrm{ext}} + \{\#\}$, we do the following:

   (a) If G^{ver} has size greater than some constant c, then find a $\sqrt{\cdot}$-separator S for G^{ver} with $V(G^{\mathrm{ver}}) = A_1 + S + A_2$.

   (b) Define $\Phi := \{\chi^{\mathrm{int}} : S \to C_0^{\mathrm{int}} + C_1^{\mathrm{int}} \mid \chi^{\mathrm{ext}} \rightsquigarrow \chi^{\mathrm{int}}\}$.
   For all internal colorings $\chi^{\mathrm{int}} \in \Phi$ do:

      i. Determine $\mathrm{Eval}_{A_i}(\chi^{\mathrm{int}})$ recursively for $i = 1, 2$.

      ii. Determine $\mathrm{Eval}_S(\chi^{\mathrm{int}})$.

   (c) Return $\mathrm{opt}_{\chi^{\mathrm{int}}\in\Phi} h(\mathrm{Eval}_{A_1}(\chi^{\mathrm{int}}), \mathrm{Eval}_S(\chi^{\mathrm{int}}), \mathrm{Eval}_{A_2}(\chi^{\mathrm{int}}))$.

The sizes of the subproblems, i.e., the sizes of the graphs $G^{\mathrm{ver}}_{A_i}(\chi^{\mathrm{int}})$ which are used in the recursion, play a crucial role in the analysis of the running time of this algorithm. A particularly nice situation is given by the following problems.

Definition 19. A glueable select&verify problem is called *slim* if the subgraphs $G^{\text{ver}}_{A_i}(\chi^{\text{int}})$ are only by a constant number of vertices larger than $G^{\text{ver}}[A_i]$, i.e., if there exists an $\eta \geq 0$ such that $|V(G^{\text{ver}}_{A_i}(\chi^{\text{int}}))| \leq |A_i| + \eta$ for all internal colorings $\chi^{\text{int}} : S \to C^{\text{int}}$.

Note that the proof of Lemma 16 shows that both Vertex Cover and Independent Set are slim with $\eta = 0$, whereas Dominating Set is not.

The following proposition gives the running time of the above algorithm in terms of the parameters of the separator theorem used and the select&verify problem considered. In order to assess the time required for the above given divide and conquer algorithm, we use the following abbreviations for the running times of certain subroutines:

- $T_S(n)$ denotes the time to find a separator in an n-vertex graph from class $\mathbb{G}$.
- $T_M(n)$ denotes the time to construct the modified graphs $G^{\text{ver}}_X(\chi^{\text{int}}) \in \mathbb{G}$ and the modified colorings $(\chi^{\text{ext}} \oplus_X \chi^{\text{int}})$ (for $X \in \{A_1, S, A_2\}$ and each internal coloring $\chi^{\text{int}} \in \Phi$ from an n-vertex graph from class $\mathbb{G}$.
- $T_E(m)$ is the time to evaluate $\text{Eval}_S(\chi^{\text{int}})$ for any $\chi^{\text{int}} \in \Phi$ in a separator S of size m.
- $T_G(n)$ is the time for gluing the results obtained by two sub-problems each of size $O(n)$.

In the following, we assume that all these functions are polynomials.

Proposition 20. *Let $\mathbb{G}$ be a graph class for which a $\sqrt{\cdot}$-separator theorem with constants α and β is known and let $\mathcal{G}$ be a select&verify problem defined by $(P_\cdot, \text{opt})$ that is glueable with σ colors. Then, for every $G \in \mathbb{G}$, $\text{opt}_{\vec{x} \in \{0,1\}^n} P_G(\vec{x})$ can be computed in time*

$$c(\alpha', \beta, \sigma)^{\sqrt{n}} q(n), \quad \textit{where} \quad c(\alpha', \beta, \sigma) = \sigma^{\beta/(1-\sqrt{\alpha'})}.$$

Here, $\alpha' = \alpha + \varepsilon$ for any $\varepsilon \in (0, 1-\alpha)$ and $q(\cdot)$ is some polynomial; the running time analysis only holds for $n \geq n_0(\varepsilon)$.

If, however, $\mathcal{G}$ is slim or the $\sqrt{\cdot}$-separator theorem yields cycle separators (and $\mathbb{G}$ is the class of planar graphs), then the running time for the computation is $c(\alpha, \beta, \sigma)^{\sqrt{n}} q(n)$, which then holds for all n.

Proof. Let $T(n)$ denote the running time to compute

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

for $G^{\text{ver}} = (V^{\text{ver}}, E^{\text{ver}})$ with $n = |V^{\text{ver}}|$ (where $\chi^{\text{ext}} : V(G) \to C_0^{\text{ext}} + C_1^{\text{ext}} + \{\#\}$ is some external coloring and $G^{\text{sel}}, G^{\text{ver}} \subseteq G$). In general, the recurrence we have to solve in order to compute an upper bound on $T(n)$ for the above given divide and conquer algorithm then reads as follows:

$$T(n) \leq \sigma^{\beta\sqrt{n}} \cdot 2\, T(\alpha n + \beta\sqrt{n}) \cdot T_{M,E,G}(n) + T_S(n), \quad \text{where}$$

$$T_{M,E,G}(n) := T_M(n) + T_E(\beta\sqrt{n}) + T_G(\alpha n + \beta\sqrt{n}).$$

By assumption, a separation (A_1, S, A_2) of G^{ver} can be found in time $T_S(n)$. Since the size of the separator is upperbounded by $\beta\sqrt{n}$ and there are σ internal colors, there are $\sigma^{\beta\sqrt{n}}$ many passes through a loop which checks all possible assignments to separator vertices. In each pass through the loop, two instances of smaller subproblems have to be evaluated, namely for A_1 and A_2. This yields the term $2 \cdot T(\alpha n + \beta\sqrt{n})$. The time needed for modifying the graphs and colorings, for evaluating the separator and for gluing the obtained subproblems is covered by $T_{M,E,G}(n)$. Note that the functions $T_{M,E,G}(n)$ and $T_S(n)$ are polynomials by our general assumption. From the definition of glueability, we have that the size of the two remaining subproblems to be solved recursively is

$$|V(G^{\text{ver}}_{A_i}(\chi^{\text{int}}))| \leq |V(G^{\text{ver}}[\delta A_i])|\alpha n + \beta\sqrt{n}$$

for each $\chi^{\text{int}} \in \Phi$. For every $\varepsilon \in (0, 1-\alpha)$, there is, of course, an $n_0(\varepsilon)$ such that

$$(\alpha + \varepsilon)n \geq \alpha n + \beta\sqrt{n} \tag{3}$$

for $n \geq n_0(\varepsilon)$. Hence, by setting $\alpha' = \alpha + \varepsilon$, we can simplify the above recurrence to

$$T(n) \leq 2 \cdot \sigma^{\beta\sqrt{n}}\, T(\alpha' n)\, T_{M,E,G}(n) + T_S(n)$$

for some $\alpha' < 1$. Hence, the recursion depth is $n' = \log_{1/\alpha'}(n)$, and we get

$$\begin{array}{rcll} T(n) & \leq & \sigma^{\beta \sum_{i=0}^{n'} \sqrt{\alpha'^i n}} & 2^{n'} T(O(1)) \prod_{i=0}^{n'} T_{M,E,G}(\alpha'^i n) + n' \cdot T_S(n) \\ & \leq & \sigma^{\beta/(1-\sqrt{\alpha'})\cdot\sqrt{n}} & q(n) \end{array}$$

for the polynomial $q(n) := 2^{\log_{1/\alpha'} n} T(O(1)) \prod_{i=0}^{n'} p(\alpha'^i n) + \log_{1/\alpha'}(n) \cdot T_S(n)$.

Now, consider the case where $\mathcal{G}$ is slim. In this situation the recursive subproblems have size $|V(G^{\text{ver}}_{A_i}(\chi^{\text{int}}))| \leq |A_i| + \eta$. Hence, we have

$$T(n) \leq \sigma^{\beta\sqrt{n}} \cdot 2\, T(\alpha n + \eta) \cdot T_{M,E,G}(n) + T_S(n).$$

The size of the problem after r recursion steps obviously is:

$$\alpha^r n + \alpha^{r-1}\eta + \cdots + \alpha\eta + \eta \leq \alpha^r n + \frac{1}{1-\alpha}\eta.$$

Since the recursion will stop after $n' = \log_{1/\alpha}(n)$ recursive calls (leaving us with subproblems of at most constant size), we can further estimate:

$$\begin{aligned} T(n) &\leq \sigma^{\beta \sum_{i=0}^{n'} \sqrt{\alpha^i n + \eta/(1-\alpha)}}\, 2^{n'} T(O(1)) \\ &\quad \cdot \prod_{i=0}^{n'} T_{M,E,G}(\alpha^i n + \eta/(1-\alpha)) + n' \cdot T_S(n) \\ &\leq \sigma^{\beta \cdot (\sum_{i=0}^{n'} \sqrt{\alpha^i n} + (n'+1)\cdot\sqrt{\eta/(1-\alpha)})}\, 2^{n'} T(O(1)) \\ &\quad \cdot \prod_{i=0}^{n'} T_{M,E,G}(\alpha^i n + \eta/(1-\alpha)) \\ &\quad + n' \cdot T_S(n) \\ &\leq \sigma^{\beta/(1-\sqrt{\alpha})\cdot\sqrt{n}} q(n) \end{aligned}$$

for some polynomial $q(\cdot)$. More precisely, we can estimate

$$q(n) \leq \sigma^{\beta\sqrt{\eta/(1-\alpha)}(n'+1)} 2^{n'} T(O(1)) \prod_{i=0}^{n'} T_{M,E,G}(\alpha^i n + \eta/(1-\alpha)) + n' T_S(n)\,.$$

It remains to prove the claim in the case of the existence of a cycle separator theorem. Without going into detail, we want to sketch the key idea in this case. We consider the first recursive step of the algorithm, where we deal with a separation (A_1, S, A_2) of the input graph G. Suppose now (A_{11}, S_1, A_{12}) is a separation of $G[A_1]$, then, for $v \in S$, it is possible that $N(v) \cap A_{1i} \neq \emptyset$ for both $i = 1, 2$. This phenomenon basically forces us to find the separator in the first recursive step in $G_{A_1}(\chi^{\text{int}})$ (a possibly larger graph than $G[A_1]$) rather than in $G[A_1]$. In the case of cycle separators, i.e., if S_1 were a cycle separator, the above described phenomenon cannot occur anymore due to the Jordan curve theorem, also see Remark 6. That is why Algorithm 18 can be modified such that the new separator in step 2a is computed for G^{sel} (instead of G^{ver}). The corresponding recurrence equation for this modified algorithm then reads as

$$T(n) \leq \sigma^{\beta\sqrt{n}} \cdot 2\, T(\alpha n) \cdot T_{M,E,G}(n) + T_S(n),$$

the solution of which is given by $T(n) \leq \sigma^{\beta/(1-\sqrt{\alpha})\cdot\sqrt{n}} q(n)$ for some polynomial $q(\cdot)$. □

Remark 21. A similar proposition holds for graph classes on which an ℓ-separator theorem is known with constants α and β. It might turn out that such separator theorems have better ratio $\beta/(1-\sqrt{\alpha})$, which, in turn, would directly improve the running time in Proposition 20.

4.2 How (linear) problem kernels help

If the considered parameterized problem has a problem kernel of size dk, we can use the considerations we have made up to this point in order to obtain fixed parameter algorithms whose exponential term is of the form $c^{\sqrt{k}}$ for some constant c. More generally, a problem kernel of size $p(k)$ yields exponential terms of the form $c^{\sqrt{p(k)}}$.

Theorem 22. *Assume the following:*

- *Let $\mathbb{G}$ be a graph class for which a $\sqrt{\cdot}$-separator theorem with constants α and β is known,*
- *let $\mathcal{G}$ be a select&verify problem defined by $(P., \text{opt})$ glueable with σ colors, and*
- *suppose that $\mathcal{G}$ admits a problem kernel of polynomial size $p(k)$ on $\mathbb{G}$ computable in time $T_K(n,k)$.*

Then, there is an algorithm to decide $(G,k) \in \mathcal{G}$, for a graph $G \in \mathbb{G}$, in time

$$c(\alpha', \beta, \sigma)^{\sqrt{p(k)}} q(k) + T_K(n,k), \quad \text{where} \quad c(\alpha', \beta, \sigma) = \sigma^{\beta/(1-\sqrt{\alpha'})}, \tag{4}$$

and $\alpha' = \alpha + \varepsilon$ for any $\varepsilon \in (0, 1-\alpha)$, holding only for $k \geq k_0(\varepsilon)$, where $q(\cdot)$ is some polynomial.

If, however, $\mathcal{G}$ is slim or the $\sqrt{\cdot}$-separator theorem yields cycle separators (on the class $\mathbb{G}$ of planar graphs), then the running time for the computation is

$$c(\alpha, \beta, \sigma)^{\sqrt{p(k)}} q(k) + T_K(n,k),$$

which then holds for all k.

Proof. The result directly follows from Proposition 20 applied to the problem kernel. □

In particular, Theorem 22 means that, for glueable select&verify problems for planar graphs that admit a *linear* problem kernel of size dk, we can get an algorithm of running time

$$O\big(c(\alpha, \beta, \sigma, d)^{\sqrt{k}} q(k) + T_K(n,k)\big), \quad \text{where} \quad c(\alpha, \beta, \sigma, d) = \sigma^{\sqrt{d}\beta/(1-\sqrt{\alpha})}.$$

Obviously, the choice of the separator theorem has a decisive impact on the constants of the corresponding algorithms. In particular, our running time analysis shows that the ratio $r(\alpha, \beta) := \beta/(1 - \sqrt{\alpha})$ has a direct and significant influence on the running time. In Table 1, this ratio is computed for the various $\sqrt{\cdot}$-separator theorems. In the following example we use these ratios explicitly.

Example 23. In the case of Vertex Cover on planar graphs, we can take $d = 2$, $\alpha = 2/3$, and $\beta = \sqrt{2/3} = \sqrt{4/3}$ (see [17]) with the ratio $r(\alpha, \beta) \approx 10.74$. In this way, we obtain an algorithm with running time $O(2^{\sqrt{2} \cdot 10.74 \cdot \sqrt{k}} + nk)$. Neglecting polynomial terms, we have such obtained a $c^{\sqrt{k}}$-algorithm with $c = 2^{15.19} \approx 37381$.

Taking $d = 2$, $\alpha = 3/4$, and $\beta = \sqrt{2\pi/\sqrt{3}} \cdot (1 + \sqrt{3})/\sqrt{8} \approx 1.84$ (see [38]) with $r(\alpha, \beta) \approx 13.73$, we get an algorithm with running time $O(2^{\sqrt{2} \cdot 13.73 \cdot \sqrt{k}} + nk)$. This means, we have a $c^{\sqrt{k}}$-algorithm with $c = 2^{19.42} \approx 701459$.

Even worse, choosing $d = 2$, $\alpha = 1/2$, and $\beta = \sqrt{24} \approx 7.58$ [11] with $r(\alpha, \beta) \approx 16.73$, we obtain a $c^{\sqrt{k}}$-algorithm with $c = 2^{23.66} \approx 13254694$.

In this place, let us mention that taking known 3-separator theorems would yield better constants here (at the cost of worse polynomial terms which we generally neglect in the course of this example). By a result due to Venkatesan [43] (also see Lemma 36 below), we could obtain $\beta = \sqrt{12} \approx 3.46$ for $\alpha = 1/2$ (here, $r(\alpha, \beta) \approx 11.83$, which yields a $c^{\sqrt{k}}$-algorithm with $c = 2^{16.73} \approx 108701$. Observe that this constant is comparable with the constant obtained via separator theorems for $\alpha = 2/3$.

The constants obtained by this first approach are admittedly bad. Section 5 is dedicated to present new strategies on how to substantially improve these constants.

Remark 24. Let us make some remarks on the importance of cycle separator theorems or slim graph problems. Assume that none of these two conditions is met in a given situation. Then, the claimed bound from Equation (4) of Theorem 22 is only true for some $\alpha' = \alpha + \varepsilon$ with $\varepsilon \in (0, 1 - \alpha)$. Now, there is a certain trade-off in the choice of ε:

1. The factor $\beta/(1 - \sqrt{\alpha'})$ in the exponent of $c(\alpha', \beta, \sigma)$ tends to infinity if α' tends to one, i.e., if ε is as large as possible.

2. The analysis of Theorem 22 is only valid if $p(k) \geq (\beta/\varepsilon)^2$. This bound is easily derived from Equation (3) in Proposition 20 when replacing n by $p(k)$ due to the assumption of a problem kernel of polynomial size.

Keeping in mind that typical values of $p(k)$ are not very large in practical cases, the second point means that, since β is fixed, ε should be comparatively large, otherwise, β/ε would be greater than $\sqrt{p(k)}$. This gives us very bad constants in the analysis due to the first point.

As explained in the following example, Theorem 22 is not only interesting in the case of planar graph problems.

Example 25. Since VERTEX COVER is a slim problem which has a linear size kernel, Theorem 22 yields a $c^{\sqrt{gk}}$-algorithm for $\mathbb{G}_g$, where $\mathbb{G}_g$ denotes the class of graphs of genus bounded by g; see [15], where the existence of a separator of size $O(\sqrt{gn})$ for n-vertex graphs from $\mathbb{G}_g$ was proven. For the same reason, we get a $c^{\sqrt{gk}}$-algorithm for INDEPENDENT SET on $\mathbb{G}_g$.

Note that these are the first examples of fixed parameter algorithms with sublinear exponents for bounded genus graphs.

4.3 Towards avoiding (linear) problem kernels: the core concept

We are going to introduce the novel notion of *problem cores*, which is closely related to that of problem kernels, but seemingly "incomparable" and tailored towards *unweighted minimization* select&verify problems.[11] The main distinguishing point between problem kernels and problem cores (the latter to be defined next) can be sketched as follows. In the case of problem kernels, we reduce an originally "big" problem instance to a "small" one, where then the remaining work has to be done exclusively on the small instance without taking care of the original big instance. The kernel that comes out of such a preprocessing might be completely unrelated to the original problem instance. By way of contrast, in the case of cores, we also reduce, in a sense, to a small instance, but we still allow that in order to solve the underlying problem, the original big instance may still be used for "checking;" the "guessing" (i.e., the search), however, can be restricted to the core. To make this work, we have to demand that the core itself really has to be a "sub-instance" of the original instance, being obtained from the original instance by simply omitting parts of it. (We make this formal in the rest of this subsection.) That is why we think that both notions are incomparable and why we hope that cores might prove useful.

Definition 26. Consider an unweighted select&verify minimization graph problem $\mathcal{G}$ specified by $(P., \min)$. A *corer of size* $p(k)$ is a polynomial time computable mapping $\phi : ((V, E), k) \mapsto V_c$, $V_c \subseteq V$, satisfying

[11] In fact, we do not know how to define cores for maximization problems.

- $|V_c| \leq p(k)$, and
- $\exists \vec{x} = (x_1, \ldots, x_{|V|}) \in \{0,1\}^{|V|}(P_G(\vec{x}) \leq k \wedge \{v_i \in V \mid x_i = 1\} \subseteq V_c)$.

The set V_c is also called the *problem core* of $\mathcal{G}$. If $p(k) = ak$, we call ϕ a *linear* corer. In this case, V_c is called a *factor-a problem core*.

Note that weighted minimization problems could also be treated similarly at the expense of further technical complications. Having a problem core automatically makes a select&verify problem a "simple" one from the viewpoint of parameterized complexity:

Lemma 27. *If an unweighted select&verify minimization graph problem $\mathcal{G}$ has a core of size $p(k)$, then it is fixed parameter tractable.*

Proof. For the problem core V_c, which can be computed in polynomial time, it is enough to check all k-element subsets, giving fixed parameter tractability. □

Moreover, Stirling's formula yields for linear kernels or corers:

Lemma 28. *If a select&verify problem $\mathcal{G}$ has a size ak problem kernel or if the core is factor-a, then there is a "c^k-algorithm" for $\mathcal{G}$, where $c = \min\{ea, 2^a\}$.*

Proof. In the case of $p(k) = ak$, by applying Stirling's formula, we can exploit the fact that ak choose k mostly is considerably smaller than 2^{ak}, namely

$$\begin{aligned}
\binom{ak}{k} &= \frac{(ak)!}{k!((a-1)k)!} \\
&\approx \frac{\sqrt{2\pi ak}}{\sqrt{2\pi k}\sqrt{2\pi(a-1)k}} \cdot \left(\frac{ak}{e}\right)^{ak} \cdot \left(\frac{e}{k}\right)^{k} \cdot \left(\frac{e}{(a-1)k}\right)^{(a-1)k} \\
&= \frac{\sqrt{a}}{\sqrt{2\pi(a-1)k}} a^k \left(\frac{a}{a-1}\right)^{(a-1)k} \\
&\leq a^k \left(\frac{a}{a-1}\right)^{(a-1)k} \\
&\leq (ea)^k
\end{aligned}$$

for large enough k. (If a is not an integer, we can use general binomial coefficients based on the Gamma function.) Since, by definition of select&verify problems, P. is polynomial time computable, the claim is shown. □

It is quite obvious that there is a close intuitive connection between corers and kernelizations, which, however, seems hard to grasp formally. For example, the already mentioned kernel of size $2k$ for Vertex Cover based on a theorem of Nemhauser and Trotter [30] is also a factor-2 core. Unfortunately, we do not know of any concrete problem having a small core but no (small) kernel, although we feel that such examples should exist.

Even though there seems to be no *general* interrelation between problem kernels and problem cores, for our purposes, the different concepts can be interchanged. The analogue to Theorem 22 reads as follows. Note, however, that we now inherently need separator theorems for weighted graphs, even though the originally given graph problem deals with unweighted graphs. This might lead to worse constants, see Remark 7.

Theorem 29. *Assume the following:*

- *Let $\mathbb{G}$ be a graph class for which a $\sqrt{\cdot}$-separator theorem for* weighted graphs *with constants α and β is known,*
- *let $\mathcal{G}$ be a select&verify problem defined by $(P., \min)$ glueable with σ colors, and*
- *suppose that $\mathcal{G}$ admits a corer of size $p(k)$ on $\mathbb{G}$, which can be computed in polynomial time $T_C(n)$.*[12]

Then, there is an algorithm to decide $(G,k) \in \mathcal{G}$, for a graph $G \in \mathbb{G}$, in time

$$c(\alpha', \beta, \sigma)^{\sqrt{p(k)}} q(k) + T_C(n), \quad \text{where} \quad c(\alpha', \beta, \sigma) = \sigma^{\beta/(1-\sqrt{\alpha'})},$$

and $\alpha' = \alpha + \varepsilon$ for any $\varepsilon \in (0, 1-\alpha)$, holding only for $k \geq k_0(\varepsilon)$.

If, however, $\mathcal{G}$ is slim or the $\sqrt{\cdot}$-separator theorem yields cycle separators (on the class $\mathbb{G}$ of planar graphs), then the running time for the computation is

$$c(\alpha, \beta, \sigma)^{\sqrt{p(k)}} q(k) + T_C(n),$$

which then holds for all k.

Proof. Basically, the assertion can be proved similar to Proposition 20. Looking at the list of assumptions, the main difference is that we now suppose the existence of a core (instead of that of a kernel). How can we exploit the knowledge of a corer within divide and conquer algorithms? We will do

[12] Observe that, due to our definition of a corer, the size of a core depends polynomially on n, so that the polynomial function T_C need not include k as an explicit argument, as it was necessary in the analogous theorem dealing with problem kernels instead of cores.

this in two ways: firstly, we only need to select vertices from the core V_c, so that we can start with the graph $G[V_c] \in \mathbb{G}$ as the graph from which vertices have to be selected. Secondly, we are now interested in finding separators which split the graph in parts with "approximately the same number" of vertices from V_c, so that $\log_2 |V_c| \leq \log_2(p(k))$ upperbounds the depth of the recursion.

For given graph $G = (V, E) \in \mathbb{G}$, the algorithm proceeds as follows:

1. Compute a core $V_c \subseteq V$ containing at most $p(k)$ vertices in time $T_C(n)$.

2. Then, the graph $G[V_c] \in \mathbb{G}$ is the graph from which vertices have to be selected.

3. Find an optimal $\vec{x}$ satisfying $P_G(\vec{x} \mid G[V_c])$ by applying Algorithm 18. One modification of this algorithm, however, is necessary: we do not use a separator theorem in step 2a of Algorithm 18 for a separation of G^{ver}, but we add *weights* to G^{ver} as follows: V_c induces a weight function ω on G^{ver} by letting

$$\omega : V \to \mathbb{R}_+, v \mapsto \begin{cases} 0, & v \notin V_c, \\ 1/|V_c|, & v \in V_c. \end{cases}$$

   Then, we apply a separator theorem for weighted graphs to G^{ver} with the weight function ω.

The constants $c(\alpha, \beta, \sigma)$ and $c(\alpha', \beta, \sigma)$ are then derived as in Proposition 20. □

Again, we can specialize the above theorem for the linear case: The existence of a factor-a problem core for a glueable select&verify problem for planar graphs implies a solving algorithm running in time

$$O(c(\alpha, \beta, \sigma, a)^{\sqrt{k}} q(n) + T_C(n)), \quad \text{where} \quad c(\alpha, \beta, \sigma, a) = \sigma^{\beta\sqrt{a}/(1-\sqrt{\alpha})}.$$

As illustrated in [17], for $\alpha = 2/3$, we can obtain $\beta \approx 1.97$ for a cycle separator theorem for unweighted planar graphs, and $\beta = 2$ for a cycle separator theorem for weighted planar graphs.

5 Improving constants

In order to improve the constants obtained in Theorems 22 and 29, we will analyze how separator theorems are proven in the literature. For the ease of presentation, we will again restrict ourselves to planar graphs. Similar observations can be made for other graph classes, as well.

5.1 A brief sketch of separator theorem proofs

As pointed out by Venkatesan [43], a small separator, as obtained by Lipton and Tarjan's approach, consists of two ingredients:

- the first ingredient of the separator is composed of all vertices which have the same distance from the root of an assumed minimal height spanning tree (modulo some constant s), and
- the second ingredient is found according to a "special separator theorem" for planar graphs with radius of at most s.

The constant s is chosen such that the overall size of the separator is minimized.

Let us discuss the first ingredient in more details. Consider an n-vertex planar graph $G = (V, E)$, and let a spanning tree T of minimal height of G be given which has root r. A *level* i of a vertex v of G specifies the distance of v from r in the tree T. Choose some integer s "suitably" (this will become clearer later). Let $L_{T,j}$ denote the vertices of G which are at those levels i in T such that $i \bmod s = j$. In particular, $r \in L_{T,0}$. If s does not exceed the number of levels in T, then there exists a level L_{T,i_0} such that

$$|L_{T,i_0}| \leq \lfloor n/s \rfloor. \tag{5}$$

It will become clear later that the case when s exceeds the number of levels of T is an easy one.

The vertex set L_{T,i_0} will be the first ingredient of the separator we are going to construct. Observe that L_{T,i_0} alone is not a separator of G in general, since there may be edges not belonging to the spanning tree T which connect vertices of the remaining levels. Nevertheless, the graph $G - L_{T,i_0}$ is small when we look at its radius. Therefore, we will also call the procedure of cutting out L_{T,i_0} from G a *folding step*. More precisely, Venkatesan [43, Theorem 1] proved:

Lemma 30. *Assume that s does not exceed the radius h of G. Let T be a spanning tree of height h. For any level $L_{T,j}$ of G, the components of $G - L_{T,j}$ form a subgraph of another planar graph G' which contains $n - |L_{T,j}| + 1$ vertices and which has radius no more than s.* □

The second ingredient of the separator we are going to construct is obtained via a separator theorem for planar graphs of bounded radius, applied to G' form Lemma 30; of course, the obtained small separator will also separate $G - L_{T,j}$. Let us quote the needed "special separator theorem" in the

form of a lemma which is due to Lipton and Tarjan [28]. Observe that weights are given to vertices of a graph by means of a function ω which assigns nonnegative reals to vertices and, hence, to vertex sets. Setting $\omega(v) = 1/|V|$ for all $v \in V$ yields the unweighted case.

Lemma 31. *Let G be a planar graph of radius s, with nonnegative weights on its vertices adding in total to no more than 1. Then, a separation (A_1, S, A_2) of G can be found in linear time such that neither A_1 nor A_2 has weight exceeding $2/3$ and $|S| \leq 2s + 1$. Moreover, the vertices of S lie on a cycle when G is triangulated.* □

We remark that a statement similar to Lemma 31 is also valid for graphs of bounded genus, compare [14, 15], so that our approach can be easily generalized to these graph classes $\mathbb{G}_g$. In addition, if it can be shown that there is a function $\bar{h}$ such that $\bar{h}(k)$ is an upper bound on the radius of any graph instance (G, k) with $G \in \mathbb{G}_g$ for a select&verify graph problem, then this problem (restricted to graphs from $\mathbb{G}_g$) is fixed-parameter tractable.

It is now possible to derive the classical Lipton/Tarjan result from the above considerations:

- If the radius of a given n-vertex planar graph G is less than or equal to $\sqrt{2n}$, then we might apply Lemma 31 directly.
- Otherwise, choose $s = \sqrt{n/2}$. The first ingredient of the separator, delivered by the folding step, then contains at most $n/s = \sqrt{2n}$ vertices according to Equation (5) and the second ingredient less than $2s+1 = \sqrt{2n} + 1$ vertices according to Lemma 31 (which is applicable due to Lemma 30), summing up to at most $2\sqrt{2n} + 1$ vertices within the separator.

Here, the "magic number" s was determined as to minimize the sum

$$n/s + 2s \tag{6}$$

of the two ingredients of the separator.

By making more clever use of the sketched ideas, Djidjev and Venkatesan [17] recently derived the constant $\beta = 2$ for $\alpha = 2/3$ in the case of weighted graphs. For the unweighted case, they even obtained $\beta = \sqrt{2/3} + \sqrt{4/3} \approx 1.97$. This figure already comes quite close to the lower bound of $\beta = 1.55$ for $\alpha = 2/3$ due to Djidjev [13] (see Table 1).

5.2 Optimizing divide and conquer algorithms based on graph separators

We are now going to improve the constants involved in the proofs of Theorems 22 and 29 considerably by tailoring the use of separator theorems for the divide and conquer approach. Note that, from a rough algorithmic perspective, we will still suggest Algorithm 18. The only difference to Proposition 20 is the way separators are obtained in step 2a.

Our main two observations leading to the improvements explained in the next theorem are the following ones:

1. It is unnecessary to perform the folding step over and over again in the recursion, since the graph parts which occur after having applied a classical separator theorem will have small radius. Instead, one can apply Lemma 30 directly several times in a row in the course of the recursion.[13] We cannot expect good results if we only apply Lemma 31, since the radius (as condition of applying Lemma 31) would not decrease in general, which means that the separators we get after several recursive applications of Lemma 31 will be rather large.[14] Therefore, from time to time, a folding step has to be inserted. The goal is now to find the optimal number of recursive applications of Lemma 31 in a row.

2. It is not necessary to minimize a single separator occurring in the recursion, but it might be more reasonable to minimize the sum of all separators which occur on an arbitrary recursion path. This means that we could choose a new optimal radius s in a way similar to Equation (6), which expresses the idea of Lipton and Tarjan.

In order to keep our analysis within reasonable length, we omit discussing cycle separators but focus on recursive algorithms for slim select&verify graph problems in the following. Let us explain in a few lines why this restriction is not too hard from a practical point of view, which means that the next theorem will be also applicable even when we are faced with problems which are not slim.

1. Practical applications of the obtained algorithms will not go into really deep recursions, since only small parameter values can be tackled.

[13] This idea has been already used by Venkatesan [43] in another context. We will analyze the mentioned idea of Venkatesan—in the context where it appeared originally—more thoroughly in Subsection 5.4.

[14] It is exactly this situation which was tackled by Venkatesan and which will be analyzed in Subsection 5.4 from a parameterized point of view.

2. Lemma 31 (which is applied "most of the time" according to the proof of the next theorem) yields cycle separators. Moreover, insisting on cycle separators is only necessary from the second iteration of the divide and conquer method onwards. Note that, if we want to avoid additional colors when solving problems which are not slim, these additional colors are only necessary in steps which are not guaranteed to yield cycle separators.

Hence, in practical applications, only few (if any) additional colors will be necessary to solve problems which are not slim with the presented methods. Since in practice, no really deep recursions are to be expected, the number of additional colors will be negligible.

In the following, we are specializing the constants α and β as $\alpha = 2/3$ and $\beta = \sqrt{8}$, because the simple proof structure of the planar separator theorem of Lipton and Tarjan (as sketched in Subsection 5.1) allows for easy modifications. It remains a challenging future research topic to see which ideas from other separator theorems could be used in order to obtain better constants in the next theorem. In fact, the results of Subsection 5.3 indicate that other separator theorems can be preferable.

The basic idea of the proof of the following theorem is to exploit the fact that the running time of a divide and conquer algorithm based on separators in graphs basically depends only on the sum of the sizes of the seperators accumulated along a fixed recursion path, cf. the discussions preceding Equation (6). Thus, it can be advantageous to have larger separators at certain levels of the recursion tree if this buys us smaller separators at the other levels.

Theorem 32. *Let $\mathcal{G}$ be a select&verify problem on planar graphs defined by $(P., \mathrm{opt})$ which is glueable with σ colors, and suppose that $\mathcal{G}$ admits a problem kernel of polynomial size $p(k)$ computable in time $T_K(n,k)$.*

Then, there is an algorithm to decide $(G,k) \in \mathcal{G}$, for an n-vertex planar graph G, in time

$$c(\alpha', \sigma)^{\sqrt{p(k)}} q(k) + T_K(n,k), \quad \text{where} \quad c(\alpha', \sigma) \approx \sigma^{1.80665/(1-\sqrt{\alpha'})},$$

and $\alpha' = 2/3 + \varepsilon$ for any $\varepsilon \in (0, 1/3)$, holding only for $k \geq k_0(\varepsilon)$, where $q(\cdot)$ is some polynomial.

If $\mathcal{G}$ is slim, then the running time for the computation is

$$c(2/3, \sigma)^{\sqrt{p(k)}} q(k) + T_K(n,k),$$

which then holds for all k.

Proof. After kernelization, we are left with a graph with at most $p(k)$ vertices. For this graph G', we would like to answer the question whether $(G', k') \in \mathcal{G}$ for some k' linearly depending on k. We will do this in a recursive manner.

In the first step of the recursion, we will basically use the classical separator theorem of Lipton and Tarjan. Recall the outline of its proof in Subsection 5.1. Especially, note that the obtained separator consists of two ingredients. We will discuss these ingredients once more in the following.

Analogously to the considerations preceding Lemma 30, let L_{T,i_0} be a level in the vertex set induced by some spanning tree T of minimal height in G' such that $|L_{T,i_0}| \leq \lfloor p(k)/s \rfloor$, where s is an integer we are going to optimize in the following. We use L_{T,i_0} in a folding step. According to Lemma 30, we are now left with a planar graph with radius upperbounded by s. Therefore, we can now recurse the next ℓ steps in a row using only Lemma 31. Then, we will interleave another folding step in order to decrease the radius of the remaining graph parts, which are again handled by ℓ applications of Lemma 31 in a row, and so forth.

Within this basic scheme of recursion, let us consider a fixed recursion path. The time spent in ℓ recursion steps using only Lemma 31 basically is $\sigma^{(2s)\ell}$, since the accumulated size of the separators along the recursion path is upperbounded by $2s\ell$. Namely, in each of these ℓ recursion steps, a new separator of size $2s$ has to be considered. Therefore, the size size_ℓ of all separators along the recursion path after one folding step and ℓ applications of Lemma 31 is upperbounded by:

$$\text{size}_\ell \leq |L_{T,i_0}| + 2s\ell \leq \lfloor p(k)/s \rfloor + 2s\ell. \tag{7}$$

In order to minimize size_ℓ (which directly influences the running time of the recursive algorithm, as noted in the discussion preceding this theorem), we should choose $s = s_\ell := \sqrt{p(k)/(2\ell)}$. This means that we obtain an optimal total separator size of

$$\text{size}_\ell \leq \sqrt{8\ell p(k)}. \tag{8}$$

Let us now compare this new mixed strategy with a direct application of known separator theorems for obtaining divide and conquer algorithms, as detailed in Proposition 20. Firstly, observe that, after ℓ recursion steps according to our new mixed strategy, we are left with a graph whose components contain at most $(2/3)^\ell n$ vertices, if we started with an n-vertex graph. This situation is completely the same if we would have repeatedly applied the currently best separator theorem known for planar graphs, see Proposition 20. Consider a separator theorem with general constants α and β. After

ℓ iterations of this separator theorem, along a fixed recursion path, we would have accumulated separators of total size upperbounded by

$$\beta \cdot \sqrt{n} + \beta \cdot \sqrt{\alpha n} + \cdots + \beta \cdot \sqrt{\alpha^{\ell-1} n},$$

if we started with an n-vertex graph. This geometric sum can be rewritten, yielding the general upper bound

$$\beta \cdot \frac{1 - \alpha^{\ell/2}}{1 - \sqrt{\alpha}} \cdot \sqrt{n} \tag{9}$$

Let us compare our approach therefore with Djidjev's separator theorem with constants $\beta = \sqrt{2/3} + \sqrt{4/3} \approx 1.97$ and $\alpha = 2/3$, see Table 1. After ℓ iterations of this separator theorem, along a fixed recursion path, we would have accumulated separators of total size $\overline{\text{size}}_\ell$, where

$$\overline{\text{size}}_\ell \leq (\sqrt{2/3} + \sqrt{4/3}) \cdot \frac{1 - (2/3)^{\ell/2}}{1 - \sqrt{2/3}} \cdot \sqrt{p(k)} \tag{10}$$

according to Equation (9). Comparing this approach with the previously outlined new mixed strategy can now be done by comparing size_ℓ and $\overline{\text{size}}_\ell$. Simple arithmetics shows that $\text{size}_\ell \leq \overline{\text{size}}_\ell$ iff $\ell \in \{1, \ldots, 12\}$. Therefore, our mixed strategy is better than the direct approach for $\ell \in \{1, \ldots, 12\}$.

Which $\ell \in \{1, \ldots, 12\}$ is the best choice within our mixed strategy? We answer this question by the following approach. Equating the right-hand side of Equation (8) and Expression (9) (in the case when $\alpha = 2/3$ and $n = p(k)$) allows us to compute a β_ℓ of a hypothetic separator theorem whose direct use for divide and conquer matches our mixed strategy. In this way, we obtain the following formula for β_ℓ:

$$\beta_\ell = \sqrt{8\ell} \cdot \frac{1 - \sqrt{2/3}}{1 - (2/3)^{\ell/2}}.$$

Some computations show that β_6 is the minimum of all such β_ℓ for $\ell \in \{1, \ldots, 12\}$. More precisely, we have $\beta_6 = 1.80665$ as required in the theorem. □

Note that one could replace the requirement of a polynomial size problem kernel by assuming a polynomial size problem core and obtain similar results by using Theorem 29, since also weights can be handled in the graph separator theorems which were used in the preceding proof.

Example 33. For Vertex Cover on planar graphs, we obtain by the previous theorem a $2^{13.9234\sqrt{k}} \approx 15537^{\sqrt{k}}$-algorithm, which obviously beats the figures in Example 23.

To further improve our constants, we need a generalized notion of separation as introduced in the following subsection.

5.3 Regular partitions

Djidjev [13] coined the notion of *regular γ-partition*. We will only need the following restricted notion (which Djidjev would term regular 1/2-partition):

Definition 34. A *regular partition* (A_1, A_2, A_3, S) of an n-vertex graph $G = (V, E)$ is a partition $V = A_1 + A_2 + A_3 + S$ such that

- A_i and A_j are not connected to each other for $1 \leq i < j \leq 3$ and
- $|A_i| \leq 1/2 \cdot n$ for $i = 1, 2, 3$.

Hence, in a regular partition, S can be viewed as a separator that splits G into *three* parts A_1, A_2, and A_3. Therefore, a regular partition is an example of a 3-separator in our notation introduced after Definition 4.

Djidjev has shown the following analogue of Lemma 31:

Lemma 35. *Let G be a planar graph of radius s. Then, a regular partition (A_1, A_2, A_3, S) exists such that $|S| \leq 3s + 1$.* □

With the help of Lemma 35, Venkatesan [43] proved:

Lemma 36. *Let G be a planar n-vertex graph. Then, there exists a regular partition (A_1, A_2, A_3, S) such that $|S| \leq \sqrt{12}\sqrt{n}$.* □

In principle, the proposition was shown in the same way as the separator theorem of Lipton and Tarjan. In the folding step, a level L_{T,i_0} of a spanning tree of G is chosen such that $|L_{T,i_0}| \leq n/s$; then, Lemmas 30 and 35 are used to get a regular partition with $|S| \leq n/s + 3s$; finally, s is chosen to minimize $n/s + 3s$ which yields the claimed constant.

Obviously, we can try the same trick as in Theorem 32 to find an optimal ℓ so that Lemma 35 is used ℓ subsequent times in the recursion. Some computations lead to:

Theorem 37. *Let $\mathcal{G}$ be a select&verify problem on planar graphs defined by $(P., \text{opt})$ which is glueable with σ colors, and suppose that $\mathcal{G}$ admits a problem kernel of polynomial size $p(k)$ computable in time $T_K(n, k)$.*

Then, there is an algorithm to decide $(G, k) \in \mathcal{G}$, for an n-vertex planar graph G, in time

$$c(\alpha', \sigma)^{\sqrt{p(k)}} q(k) + T_K(n, k), \quad \text{where} \quad c(\alpha', \sigma) \approx \sigma^{2.7056/(1-\sqrt{\alpha'})},$$

and $\alpha' = 1/2 + \varepsilon$ for any $\varepsilon \in (0, 1/2)$, holding only for $k \geq k_0(\varepsilon)$, where $q(\cdot)$ is some polynomial.

If $\mathcal{G}$ is slim, then the running time for the computation is

$$c(1/2,\sigma)^{\sqrt{p(k)}}q(k) + T_K(n,k),$$

which then holds for all k. □

Example 38. For VERTEX COVER on planar graphs, we obtain in this way a $2^{13.0639\sqrt{k}} \approx 8564^{\sqrt{k}}$-algorithm, which again beats the figure derived in Example 33.

Notice that, since there is no analogue of Lemma 35 known for *weighted* graphs, we do not know whether Theorem 37 is true when facing a graph problem for which we only know of a small core instead of a small kernel. To the contrary, such a "core-analogue" was true for Theorem 32.

Remark 39. In the case that a slim select&verify planar graph problem $\mathcal{G}$, which is glueable with σ colors, admits a kernel of size dk, where the kernelization yields a problem parameter $k' = k$ (for simplicity), two principle methods are immediately at hand:

- Employ the natural analogue of Lemma 28 in order to obtain a 2^{dk}- or $(ed)^k$-algorithm for $\mathcal{G}$.
- Employ Theorem 37 in order to get a $\sigma^{2.7056/(1-\sqrt{1/2})\sqrt{dk}} = \sigma^{9.2376\sqrt{dk}}$-algorithm.

Taking, e.g., the 2^{dk}-algorithm for $\mathcal{G}$ and equating 2^{dk} with $\sigma^{9.2376\sqrt{dk}}$ yields

$$k_{BE}(\sigma,d) = \left(\frac{9.2376 \cdot \log(\sigma)}{\log(2)}\right)^2 \frac{1}{d} = 177.61 \cdot (\log(\sigma))^2 d^{-1} \qquad (11)$$

as *break even point* of the separator-based algorithm, i.e., whenever $k \geq k_{BE}(\sigma,d)$, then the separator-based algorithm is better than the 2^{dk}-algorithm.

In many cases, as for example in the case of VERTEX COVER, there are known so-called search-tree algorithms which are much better than the 2^{dk}-algorithm. Therefore, the break even point of the separator-based algorithm tends to be greater than suggested by Equation (11). For example, in the case of VERTEX COVER, a 1.3^k-algorithm beats the suggested $8564^{\sqrt{k}}$-algorithm as long as $k \leq 1191$. Of course, $k \approx 1000$ is out of reach of current computer technology, although one has to keep in mind that we are always talking about worst case upper bounds. For example, the pieces obtained by the separator theorems might be much smaller than suggested by the theorems, which immediately yields an improved running time of these algorithms.

Nevertheless, in practice, it might be favorable to devise algorithms which have worse asymptotic behavior (since the exponential term contains a faster growing function) but smaller constants. We try to capture this sort of trade-off in the following.

5.4 n/ε separation

There are separator theorems which allow for arbitrary small weights (upper-bounded by ε) for each of the (many) graph components into which the given graph is split, at the expense of getting larger separators (of a size depending also on the chosen ε). Venkatesan proved the following result, which is, to our knowledge, the best of its kind [43]:

Lemma 40. *Let G be an n-vertex planar graph with nonnegative vertex weights adding to at most one, and let $0 < \varepsilon \leq 1$. Then, there exists some separator S with at most $\sqrt{24}\sqrt{n/\varepsilon}$ many vertices such that $G - S$ has no connected component of total weight greater than ε. The separator S can be found in $O(n \log n)$ time.* □

Let us briefly sketch the proof of Lemma 40: As in the proofs of previously shown separator theorems, in a preparatory folding step, the graph is sliced into pieces of bounded radius s by cutting off a layer L_{T,i_0} with $|L_{T,i_0}| \leq n/s$. Then, Lemma 31 is used repeatedly, until all remaining graph pieces are small enough. Choosing an optimal s then yields the claimed constant.

We want to apply Lemma 40 in order to design algorithms for certain select&verify planar graph problems that are glueable with σ colors. Thinking about ε as a function $\varepsilon : n \mapsto (0, 1]$ allows us to devise an algorithm of the following kind:

- Apply Algorithm 18, as long some of the graph pieces such obtained in the course of the recursion have more than $\varepsilon(n)n$ many vertices, and
- compute an optimal solution by using the best-known $\rho^{\varepsilon(n)n}$ algorithm (for the precolored problem!) on each graph piece. Of course, there are at most n such graph pieces.

Let us try to determine an optimal $\varepsilon(n)$, assuming that the constant ρ is known. Recall that only the size of the separators on some path of the divide and conquer recursion tree matters for the calculation of the running time. This size is upperbounded by

$$n/s + 2s \log_{3/2}(1/\varepsilon(n)) \tag{12}$$

according to Equation (7), since the depth of the recursion tree is bounded by $\log_{3/2}(1/\varepsilon(n))$. Expression (12) is minimal if

$$n/s = 2s \log_{3/2}(1/\varepsilon(n)). \tag{13}$$

This means that choosing

$$s(n) = \sqrt{\frac{\log(3/2)}{2}} \sqrt{\frac{n}{-\log(\varepsilon(n))}} \tag{14}$$

would be optimal. Tracing back the meaning of $s(n)$ as a natural number, it is clear that $\lfloor s(n) \rfloor$ or $\lceil s(n) \rceil$ has to be chosen. In the following, we will nonetheless work with the theoretical real-valued function $s(n)$. Now, the task would be to determine the optimal function $\varepsilon(n)$ in order to minimize the total running time, which is upperbounded by

$$(\sigma^{2n/s(n)} \cdot \rho^{\varepsilon(n)n})n. \tag{15}$$

For fixed n, the minimum of Expression (15) is assumed when

$$\sigma^{2n/s(n)} = \rho^{\varepsilon(n)n},$$

since $2n/s(n)$ is the overall size of all separators along a fixed recursion path according to Expression (12) and Equation (13). Unfortunately, we were not able to determine the optimal function $\varepsilon(n)$ corresponding to the $s(n)$ given by Equation (14) analytically. Therefore, we make the following coarse estimate: Of course, the size of the separators along one recursion path is upperbounded by the size of all separators which is upperbounded by $\sqrt{24}\sqrt{n/\varepsilon}$ according to Lemma 40. Therefore, we compute the $\varepsilon(n)$ given by

$$\sigma^{\sqrt{24}\sqrt{n/\varepsilon(n)}} = \rho^{\varepsilon(n)n}. \tag{16}$$

This means that $\varepsilon(n) = \Theta(n^{-1/3})$ is the best choice. More precisely, evaluating Equation (16) yields

$$\varepsilon(n) = \left(\frac{\log(\sigma) \cdot \sqrt{24}}{\log(\rho)}\right)^{2/3} n^{-1/3}. \tag{17}$$

This evaluation is valid if $\varepsilon(n) \leq 1$, since we like to apply Lemma 40, i.e., for

$$n \geq \frac{\log(\sigma)}{\log(\rho)} \cdot \sqrt{24}. \tag{18}$$

Hence, an algorithm which tries all possible colorings along a recursion path, applying the best known general algorithm to the remaining graph pieces, results in the following running time due to Expression (15):

$$(\sigma^{s(n)} \cdot \rho^{\varepsilon(n)n})n,$$

where we use $s(n) = \sqrt{24}\sqrt{n/\varepsilon(n)}$ and $\varepsilon(n)$ is given by Equation (17).

This reasoning yields the following statement.

Theorem 41. *Let $\mathcal{G}$ be a select&verify problem on planar graphs, defined by $(P., \text{opt})$. We make the following further assumptions:*

- *$\mathcal{G}$ is glueable with σ colors.*
- *$\mathcal{G}$ admits a problem kernel of linear size dk computable in time $T_K(n,k)$.*
- *There is an $O(\rho^n)$ algorithm for solving the (possibly precolored) graph decision problem under consideration for a planar graph with n vertices.*

Then, there is an algorithm to decide $(G,k) \in \mathcal{G}$, for an n-vertex planar graph G, in time

$$O(dk \cdot 2^{\theta(\sigma,\rho,d)\cdot k^{2/3}} + T_K(n,k)), \text{ where } \theta(\sigma,\rho,d) = 2\log(\rho)\left(\sqrt{24}\cdot d \cdot \frac{\log(\sigma)}{\log(\rho)}\right)^{2/3}.$$

According to Inequality (18), this calculation is valid for

$$k \geq \frac{\log(\sigma)}{\log(\rho)} \cdot \sqrt{24} \cdot \frac{1}{d}.$$

Observe that we indeed need no further assumptions as, e.g. slimness, since we rely on the cycle separator Lemma 31 from the second step in the recursion on. The first step of the recursion does not cause any problems.

In the case of VERTEX COVER, Robson's algorithm [37] gives $\rho \approx 1.21$, so that Theorem 41 yields a $c^{k^{2/3}}$-algorithm with

$$c = 2^{(2\cdot\log 1.21\cdot(2\sqrt{24}/\log 1.21)^{2/3})} \approx 2^{5.96} \approx 62.25.$$

Compare this running time with the best known $c^{k^{1/2}}$-algorithm ($c = 2^{4\sqrt{3}} \approx 121.79$ was shown in [3, 4]).

Similar remarks can be made for INDEPENDENT SET.

Let us reconsider the calculations which led to Theorem 41. In principle, there were two mutually dependent places where we could optimize:

1. We can optimize s (given ε) according to Equation (14); this leads us to the following simplified equation:

$$s_{\mathrm{opt}}(\varepsilon, n) = c \cdot \frac{\sqrt{n}}{\sqrt{\log(\varepsilon^{-1})}} \tag{19}$$

   for some constant c.

2. We can optimize ε (given s) in order to minimize Expression (15); this leads us to the following equation:

$$\varepsilon_{\mathrm{opt}}(s, n) = c' \cdot n/s \tag{20}$$

   for some constant c' which depends on σ and ρ.

These two competing requirements naturally lead us to a variational approach. Unfortunately, the corresponding computations and the resulting expressions are rather weird; therefore, we refrain from giving more details here.

6 Refining FPT

The results of this paper give rise to the following observation concerning the structure of the complexity class FPT. It is known (see [19, 22]) that, for a parameterized language L which is decidable in time $g(n)$ for an input instance (I, k) with $n = |I|$, we have:[15]

$$L \in \mathrm{FPT} \quad \text{iff} \quad L \text{ admits a reduction to problem kernel.}$$

More precisely, one can show that the existence of an $f(k) \cdot n^{O(1)}$-algorithm for a parameterized language $L \in \mathrm{FPT}$ yields a problem kernel of size $f(k)$. Conversely, a problem kernel of size $p(k)$ easily yields an $O(g(p(k), k) + T_K(n, k))$ algorithm, showing membership in FPT. Here, $T_K(n, k)$ is the time needed for the reduction to the problem kernel. This suggests a refinement of the class FPT according to the asymptotic behavior of the exponential term in the running time:

$$\mathrm{FPT}(f) := \left\{ L \subseteq \Sigma \times \mathbb{N} \;\middle|\; \begin{array}{l} \exists \text{ an algorithm to decide } (x,k) \in L \text{ in time} \\ 2^{O(\log f)} \cdot n^{\alpha}, \text{ for some } \alpha \in \mathbb{R}^{+} \end{array} \right\}.$$

Alternatively, FPT can be refined according to the size of the problem kernel:

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

[15] Typically, g is a function of the form $g(n) = 2^n$.

Then the above fact can be restated as

$$\bigcup_f \mathrm{FPT}(f) = \mathrm{FPT} = \bigcup_f \mathrm{FPT}_K(f).$$

More precisely, for every function f, according to the discussion above, we have

$$\mathrm{FPT}(f) \subseteq \mathrm{FPT}_K(f). \tag{21}$$

Moreover, if $L \in \mathrm{FPT}_K(f)$, then $L \in \mathrm{FPT}(g \circ f)$, where g is the time needed to decide L.

For select&verify problems, the function g trivially is $g(n) = 2^n$ simply checking all possible vertex assignments. In this sense, our results can be seen as an improvement of the general relations (derived above) towards

$$\mathrm{FPT}(f) \cap \mathcal{GP} \subseteq \mathrm{FPT}_K(f) \cap \mathcal{GP} \subseteq \mathrm{FPT}(g \circ \sqrt{f}) \cap \mathcal{GP}$$

for a large class $\mathcal{GP}$ of planar graph problems.

It would be interesting to see whether there are other classes of FPT problems for which similar (or even further improved) relations of this form can be shown. It is challenging to see whether these (or similar) refinements of FPT could lead to a reasonable structural complexity theory. To this end, one has to define also new reduction relations which are more sensible to the parameter k, a fact which is not reflected in currently used parameterized complexity reductions. We think that this could indeed give a new impact on parameterized complexity theory.

7 Conclusion

We exhibited the relations between (planar) separator theorems and their use in obtaining fixed parameter tractability results based on divide and conquer algorithms. To this end, we coined the key notion of glueable select&verify problems that captures intricate graph problems such as Dominating Set or Total Dominating Set. We showed that various glueable select&verify problems allow $c^{\sqrt{\cdot}}$-algorithms on graph classes that admit a $\sqrt{k}$-separator theorem. Then, the constant c is determined in terms of some problem-specific parameters. By exploiting further ideas on the use of separator theorems, we were able to lower these constants substantially. Finally, methods were presented that allow for $c^{k^{2/3}}$-algorithms with already reasonable constants c.

Future research topics stirred by our paper are, among others, to further investigate the newly introduced concept of cores and to further explore the structure inside FPT. Finally, we would like to emphasize that our work indicates that (at least for the context of fixed parameter tractability) research on improving constants in separator theorems with constants α and β might not only concentrate on bringing down β for fixed α (as, e.g., done for $\alpha = 2/3$ in [17]), but more importantly, on minimizing the function $\beta/(1-\sqrt{\alpha})$. This would directly improve the exponential terms in all graph problems captured by our methodology. Moreover, it is interesting to see whether the constants in Theorems 32 and 37 can be further improved. The idea behind these two theorems might also help to overcome the "lower bound barrier" imposed on the constants of several separator theorems, see [38]. Of course, an improvement of the presented algorithms can also be gained by increasing the number of parameterized problems with (small) linear problem kernel.

Let us finally mention that separator-based techniques for solving graph problems were also used in other recent papers [16, 20]. Last but not least, our techniques might be applicable to non-planar graphs, as well. This is strongly indicated by [6, 8, 21, 25].

Acknowledgment We thank the anonymous referees of the 7th Annual International Computing and Combinatorics Conference (COCOON 2001) for their remarks that helped improving the presentation of this paper.

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 2000*, volume 1851 of *LNCS*, pages 97–110. Springer-Verlag, 2000. Long version to appear in Algorithmica.

[2] J. Alber, H. Fan, M. R. Fellows, H. Fernau, R. Niedermeier, F. Rosamond, and U. Stege. Refined search tree techniques for the PLANAR DOMINATING SET problem. In J. Sgall, A. Pultr, and P. Kolman, editors, *Mathematical Foundations of Computer Science (MFCS 2001)*, volume 2136 of *LNCS*, pages 111–122. Springer-Verlag, 2001.

[3] J. Alber, H. Fernau, and R. Niedermeier. Parameterized complexity: exponential speed-up for planar graph problems. Technical Report TR01–023, ECCC Reports, Trier, March 2001.

[4] J. Alber, H. Fernau, and R. Niedermeier. Parameterized complexity: exponential speedup for planar graph problems. In F. Orejas, P. G. Spirakis, and J. v. Leeuwen, editors, *Proc. 28th International Colloquium on Automata, Languages and Programming ICALP 2001*, volume 2076 of *LNCS*, pages 261–272. Springer, 2001.

[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] N. Alon, P. D. Seymour, and R. Thomas. Planar separators. *SIAM J. Disc. Math.*, 2:184–193, 1990.

[8] N. Alon, P. D. Seymour, and R. Thomas. A separator theorem for nonplanar graphs. *Journal of the AMS*, 3:801–808, 1990.

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

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

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

[12] 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.

[13] H. N. Djidjev. On the problem of partitioning planar graphs. *SIAM J. Algebraic Discrete Methods*, 3(2):229–240, 1982.

[14] H. N. Djidjev. A linear algorithm for partitioning graphs of fixed genus. *Serdica*, 11:369–387, 1985.

[15] H. N. Djidjev. A separator theorem for graphs of fixed genus. *Serdica*, 11:319–329, 1985.

[16] H. N. Djidjev. Computing the girth of a planar graph. In *Proc. 27th International Colloquium on Automata, Languages and Programming ICALP 2000*, volume 1853 of *LNCS*, pages 821–831. Springer-Verlag, 2000.

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

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

[19] 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.

[20] D. Eppstein. Subgraph isomorphism in planar graphs and related problems. *Journal of Graph Algorithms and Applications*, 3(3):1–27, 1999.

[21] D. Eppstein. Diameter and treewidth in minor-closed graph families. *Algorithmica*, 27:275–291, 2000.

[22] M. R. Fellows. Parameterized complexity: new developments and research frontiers. In *Aspects of Complexity*. De Gruyter, 2001.

[23] H. Fernau and R. Niedermeier. An efficient exact algorithm for constraint bipartite vertex cover. *Journal of Algorithms*, 38(2):374–410, 2001.

[24] M. R. Garey and D. S. Johnson. *Computers and Intractability*. New York: Freeman, 1979.

[25] M. Grohe. Local tree-width, excluded minors, and approximation algorithms. *Combinatorica*, To appear, 2001.

[26] D. S. Hochbaum. Approximation algorithms for the set covering and vertex cover problems. *SIAM Journal on Computing*, 11(3):555–556, 1982.

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

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

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

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

[31] 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.

[32] R. Niedermeier and P. Rossmanith. On efficient fixed parameter algorithms for WEIGHTED VERTEX COVER. In *Proc. 11th Annual International Symposium on Algorithms And Computation ISAAC'00*, volume 1969 of *LNCS*, pages 180–191. Springer-Verlag, 2000.

[33] R. Niedermeier and P. Rossmanith. An efficient fixed parameter algorithm for 3-Hitting Set. *Journal of Discrete Algorithms*, 2001. To appear. A preliminary version appeared as technical report WSI-99-18, Wilhelm-Schickard-Institut für Informatik, Universität Tübingen.

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

[35] S. S. Ravi and H. B. Hunt. An application of the planar separator theorem to counting problems. *Information Processing Letters*, 25(6):317–322, 1987.

[36] 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.

[37] J. M. Robson. Algorithms for maximum independent sets. *Journal of Algorithms*, 7(3):425–440, 1986.

[38] D. A. Spielman and S.-H. Teng. Disk packings and planar separators. In *SCG 96: 12th Annual ACM Symposium on Computational Geometry*, pages 349–358, 1996.

[39] U. Stege and M. R. Fellows. An improved fixed-parameter-tractable algorithm for vertex cover. Technical Report 262, ETH Zürich, Department of Computer Sciene, April 1999.

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

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

[42] J. A. Telle and A. Proskurowski. Algorithms for vertex partitioning problems on partial k-trees. *SIAM Journal of Discrete Mathematics*, 10(4):529–550, 1997.

[43] S. M. Venkatesan. Improved constants for some separator theorems. *Journal of Algorithms*, 8:572–578, 1987.

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