doi:10.1093/comjnl/bxh000

Parameterized complexity and approximation algorithms

Dániel Marx

Institut für Informatik, Humboldt-Universität zu Berlin, Unter den Linden 6, 10099, Berlin, Germany.
Email: dmarx@informatik.hu-berlin.de

Approximation algorithms and parameterized complexity are usually considered to be two separate ways of dealing with hard algorithmic problems. In this paper, our aim is to investigate how these two fields can be combined to achieve better algorithms than what any of the two theories could offer. We discuss the different ways parameterized complexity can be extended to approximation algorithms, survey results of this type, and propose directions for future research.

1. INTRODUCTION

Many of the computational problems that arise in practice are optimization problems: the task is to find a solution where the cost, quality, size, profit, or some other measure is as large or small as possible. The NP-hardness of an optimization problem implies that, unless P = NP, there is no polynomial-time algorithm that finds the exact value of the optimum.

Of course, the unfortunate fact that we cannot find the optimum in polynomial time does not give us an excuse to ignore the problem. After all, in practice, *some* solution is required. If we want to design and analyze algorithms for such problems in a mathematically rigorous way, then there are several options ahead of us. The field of *exact algorithms* relaxes the requirement that the running time is polynomial, and here our aim is to find the algorithm with the fastest running time, which is usually exponential in the size of the input. In *parameterized complexity* the running time is analyzed in finer detail: instead of expressing it as a function of the input size, one or more parameters of the instance are defined, and we investigate the effect of these parameters on the running time. The goal is to design algorithms that work efficiently if the parameters of the input instance are small (even if the size of the input is large). When designing *approximation algorithms*, we relax the requirement that the algorithm produces an optimum solution, and our aim is to devise a polynomial-time algorithm such that the solution it produces is not necessarily optimal, but there is some worst-case bound on the solution quality.

The motivation for studying approximation algorithms is twofold. Firstly, if we have an approximation algorithm whose error guarantee is really good (say, the maximum error is 1%), then in practice it can be as good as an optimum solution. However, for many approximation algorithms in the literature, the error guarantee is much higher (50%, 100%, 1000%, 10000%, or even worse). In this case we cannot argue that this approximation algorithm is almost as good as an exact algorithm. Nevertheless, such algorithms are still important from the theoretical point of view, as they allow us to better understand and classify problems and their variants.

In the literature on approximation algorithms, the goal is almost always to find a *polynomial-time* approximation algorithm. However, the question of approximability makes sense also in the framework of parameterized complexity. By applying ideas from both theories, we might be able to tackle problems that are intractable to both theories. There can be problems that are both inapproximable and fixed-parameter intractable, but have parameterized approximation algorithms running in fpt-time.

The aim of this survey paper is to investigate the various ways the notion of approximability can appear in parameterized complexity. We review the results on this topic from the literature, and try to interpret these results in a common framework. To motivate further research, we discuss the different directions that can be pursued. The following four main issues are identified:

- **Approximation with instance parameters.** Consider an optimization problem, where an integer parameter k is associated with every input instance. Is it possible to find an approximation algorithm with running time $f(k) \cdot |x|^{O(1)}$? For example, in the Partial Vertex Cover problem the task is to cover as many edges as possible with k vertices (where k is part of the input). Is it possible to find a 2-approximation in time $f(k) \cdot |x|^{O(1)}$? Another example: Graph Coloring is fixed-parameter

intractable if the parameter is the genus of the graph. However, there is a 2-approximation algorithm with running time $f(g) \cdot |x|$, if g is the genus of the graph [38].

- **Approximation parameterized by cost.** The most direct application of parameterized complexity to optimization problems is to parameterize by the optimum value. That is, we try to find an exact algorithm that solves the problem in $f(\mathrm{OPT}) \cdot |x|^{O(1)}$ time, where OPT is the value of the optimum solution. The motivation is that such an algorithm can work efficiently if the optimum value is very small compared to the size of the input, which might be a reasonable assumption in certain applications. We can investigate the analogous question for approximation algorithms as well. For example, we might be interested in the question whether there is a 2-approximation algorithm for MAXIMUM CLIQUE with running time $f(\mathrm{OPT}) \cdot |x|^{O(1)}$.

- **Performance functions instead of performance ratios.** The usual aim of approximation theory is to find an algorithm such that the ratio of the optimum value and the value of the solution provided by the algorithm can be bounded. For example, in the case of a minimization problem, the aim is to ensure that the cost of the solution is at most $c \geq 1$ times the optimum, for a constant c as small as possible. We investigate a more general notion of approximability: instead of the requirement that the value of the solution is at most c times the optimum, we require that it is at most $\varrho(\mathrm{OPT})$ times the optimum for some function ϱ.

- **Quality of approximation as parameter.** For certain problems, the approximation ratio can be as close to 1 as we would like: for every $\epsilon > 0$, there is a polynomial-time algorithm with approximation ratio $1+\epsilon$. A polynomial-time approximation scheme (PTAS) is an algorithm that has a parameter ϵ in the input, and produces a $1+\epsilon$-approximate solution in time $|x|^{f(1/\epsilon)}$. Typically, an approximation scheme has running time such as $|x|^{1/\epsilon^4}$ or $2^{1/\epsilon} \cdot |x|^2$. To investigate the dependence of the running time on ϵ, we study the parameterized version of the problem with $1/\epsilon$ being the parameter. We are especially interested in the question whether $1/\epsilon$ can be taken out of the exponent of the input size (as in $2^{1/\epsilon} \cdot |x|^2$).

These four issues are not completely separate from each other. There are many ways in which they overlap and interact. However, we would like to emphasize here that parameterized approximation is a complex field, and there are many subareas to investigate.

Section 2 reviews the basic notions of parameterized complexity and approximation algorithms. The four issues are discussed in Sections 3–6. Conclusions are given in Section 7. Most of the results presented in this paper are taken from the literature. Some of the cited results are mentioned only informally, while others are presented as theorems. To maintain the flow of the text, only those results are highlighted as theorems that are directly related to both parameterization and approximability.

2. PRELIMINARIES

Parameterized complexity. We follow [54] for the standard definitions of parameterized complexity. Let Σ be a finite alphabet. A decision problem is represented by a set $Q \subseteq \Sigma^*$ of strings over Σ. A *parameterization* of a problem is a polynomial-time computable function $\kappa : \Sigma^* \to \mathbb{N}$. A *parameterized decision problem* is a pair (Q, κ), where $Q \subseteq \Sigma^*$ is an arbitrary decision problem and κ is a parameterization.[1] Intuitively, we can imagine a parameterized problem as being a decision problem where each input instance $x \in \Sigma^*$ has a positive integer $\kappa(x)$ associated with it. A parameterized problem (Q, κ) is *fixed-parameter tractable (FPT)* if there is an algorithm that decides whether $x \in Q$ in time $f(\kappa(x)) \cdot |x|^c$ for some constant c and computable function f. An algorithm with such running time is called an *fpt-time algorithm* or simply *fpt-algorithm.* The theory can be extended to problems having multiple parameterizations; in this case we say that a problem $(Q, \kappa_1, \ldots, \kappa_p)$ is fixed-parameter tractable with combined parameters $\kappa_1, \ldots, \kappa_p$ if it can be solved in time $f(\kappa_1(x), \ldots, \kappa_p(x)) \cdot |x|^c$.

Many NP-hard problems were investigated in the parameterized complexity literature, with the goal of identifying fixed-parameter tractable problems. It turns out that several of these problems, e.g., MINIMUM VERTEX COVER, LONGEST PATH, DISJOINT TRIANGLES, etc., are indeed fixed-parameter tractable. There is a powerful toolbox of techniques for designing fpt-algorithms: kernelization, bounded search trees, color coding, well-quasi ordering, just to name some of the more important ones. On the other hand, certain problems resisted every attempt at obtaining fpt-algorithms. Analogously to NP-completeness in classical complexity, the theory of W[1]-hardness can be used to give strong evidence that certain problems are unlikely to be fixed-parameter tractable. W[1]-hardness is usually proved by presenting a *parameterized reduction.* For the technical details, the reader is referred to [45, 54].

Optimization problems and approximation. For each input instance of an optimization problem there is a set of feasible solutions associated to it, and a cost measure is defined for each feasible solution. The task is to

[1] Some authors (e.g., [45]) prefer to define parameterized problems such that the parameter value is not a function of the instance, but it is a number explicitly given in the instance. Each approach has its strengths and weaknesses. We prefer to define the parameter as a function, to emphasize that a problem can have many possible parameterizations.

find a feasible solution where the measure is as good as possible. Following [10], an NP optimization problem is formally defined as a 4-tuple $(I, \text{sol}, \text{cost}, \text{goal})$, where

- I is the set of instances.
- For an instance $x \in I$, $\text{sol}(x)$ is the set of feasible solutions of x. The length of each $y \in \text{sol}(x)$ is polynomially bounded in $|x|$, and it can be decided in polynomial time whether $y \in \text{sol}(x)$ holds for given x and y.
- Given an instance x and a feasible solution y, $\text{cost}(x, y)$ is a polynomial-time computable positive integer.
- goal is either min or max.

The goal is to find a feasible solution y that achieves the best objective value, i.e.,

$$\text{cost}(x, y) = \text{goal}\{\text{cost}(x, y') : y' \in \text{sol}(x)\}.$$

The cost of the optimum solution for instance x is denoted by $\text{opt}(x)$. If y is a solution for instance x, then the *performance ratio* of y is defined as

$$R(x, y) = \begin{cases} \text{cost}(x, y)/\text{opt}(x) & \text{if goal} = \min, \\ \text{opt}(x)/\text{cost}(x, y) & \text{if goal} = \max. \end{cases}$$

Thus $R(x, y)$ is always at least 1; the closer it is to 1, the closer the solution is to the optimum. For a real number $c > 1$, we say that an algorithm is a *c-approximation algorithm,* if it always produces a solution with performance ratio at most c.

Approximation schemes. It is a very common situation that after finding the first constant factor approximation for some problem, improved algorithms with better and better approximation ratios are published in subsequent papers. For certain problems, an endless series of improvements is possible, as there is no "best" approximation ratio. We say that a problem X admits a *polynomial-time approximation scheme (PTAS)* if for every $\epsilon > 0$, there is a polynomial-time $(1 + \epsilon)$-approximation algorithm for X. More precisely, we want this to hold in a uniform way: there is one algorithm that can produce an arbitrary good approximation. That is, there is an algorithm $\mathbb{A}$ such that given an instance x of X and an $\epsilon > 0$, $\mathbb{A}$ produces a $(1 + \epsilon)$-approximate solution in time $|x|^{f(1/\epsilon)}$ for some function f. Clearly, such an algorithm runs in polynomial time for every fixed value of ϵ. Approximation schemes are very abundant for geometric problems, but the literature contains many examples for other types of problems as well, e.g., for scheduling, packing, or pattern matching problems.

If ϵ is small then the exponent of the polynomial $|x|^{f(1/\epsilon)}$ can be very large. Two restricted classes of approximation schemes were defined that avoid this problem. An *efficient polynomial-time approximation scheme (EPTAS)* is a PTAS with running time of the form $f(1/\epsilon) \cdot |x|^{O(1)}$, while a *fully polynomial-time approximation scheme (FPTAS)* runs in time $(1/\epsilon)^{O(1)} \cdot |x|^{O(1)}$.

It is possible to prove negative results on the existence of approximation schemes using APX-hardness. APX is the class of optimization problems that have constant-factor approximation algorithms. A PTAS for an APX-hard problem would imply that there is a PTAS for every problem in APX. The celebrated PCP theorem (cf. [10]) states that this would also imply P = NP. Consequently, for every APX-hard optimization problem there exists a constant $c_0 > 1$ such that there is no polynomial-time c_0-approximation algorithm for the problem, unless P = NP. Often, there is a large gap between this lower bound c_0 and the best known approximation ratio. However, for the problem MAX E3SAT, a tight bound of $\frac{8}{7}$ is proved by Håstad [67].

Decision problems. In many cases, it is easier to work with decision problems than with optimization problems: most of complexity theory is developed for decision problems. There is a standard way in which an optimization problem can be turned into a more or less equivalent decision problem. If X is an optimization problem, then we define decision problem $X_{\leq}$ as

$X_{\leq}$	
Input:	An instance x of X, an integer k.
Decide:	$\text{opt}(x) \leq k$

The problems $X_{\geq}$ and $X_{=}$ can be defined analogously. If X can be solved optimally in polynomial time, then clearly all these decision problems can be solved in polynomial-time as well. The other direction is not that clear. If we can solve $X_{\leq}$ in polynomial time, then $\text{opt}(x)$ can be determined in polynomial time using binary search, but this does not give us a solution with optimum cost. However, for many problems, if we can determine the optimum, then we can actually find an optimum solution by repeatedly determining the optimum cost for slightly modified versions of the instance, a strategy generally known as *polynomial-time self-reducibility.*

In parameterized complexity, the problems $X_{\leq}$, $X_{\geq}$, $X_{=}$ are studied with the parameter k. If X is a minimization (resp., maximization) problem, then the *standard parameterization* of X is the problem $X_{\leq}$ (resp., $X_{\geq}$) parameterized by k. If the standard parameterization is fixed-parameter tractable, then this means that we have an efficient algorithm for determining the optimum for those instances where the optimum is small.

3. APPROXIMATION WITH INSTANCE PARAMETERS

The first way of studying parameterized approximation algorithms that we discuss here is to define a parameterization of the optimization problem, and instead of looking for a polynomial-time algorithm, our goal is to find an fpt-time algorithm that produces an approximate solution. That is, we define a parameterization κ that assigns a positive integer to each instance $x \in I$, and the approximation algorithm should run in time $f(\kappa(x)) \cdot |x|^{O(1)}$. In this case we say that the algorithm is an *fpt-approximation algorithm with parameterization κ.* If the performance ratio of the algorithm is c, then we say that it is an *fpt c-approximation algorithm.* Hopefully, such an algorithm can achieve a better approximation ratio than traditional polynomial-time approximation algorithms, as it has more time at its disposal.

The following two results are examples of this type of parameterized approximation algorithms (definitions will be given later):

- Metric TSP with Deadlines (ΔDLTSP) can be 2.5-approximated in time $O(|x|^3 + k!k)$, where k is the number of deadline vertices [14].
- Vertex Coloring can be 2-approximated in $f(g) \cdot |x|$ time for graphs with genus g [38].

Broadly speaking, there are two large categories of parameterizations: the parameter is either some obvious measure of the problem instance (e.g., the number of deadline vertices in ΔDLTSP), or it is some structural property of the input structure that describes (in some well-defined sense) how complicated the input is (e.g, tree width, genus, etc.) In both cases, it can be interesting to find approximation algorithms that are efficient for small parameter values. Sections 3.1 and 3.2 discuss results of these two types, respectively.

Parameterized approximation schemes can be defined in a similar manner. An *fpt-approximation scheme (fpt-AS) with parameterization κ* is an algorithm whose input is an instance $x \in I$ and an $\epsilon > 0$, and it produces a $(1+\epsilon)$-approximate solution in time $f(\epsilon, \kappa(x)) \cdot |x|^{O(1)}$ for some computable function f. For example, Har-Peled and Mazumdar [66] present an $f(\epsilon, k, d) \cdot |x|$ time approximation scheme for the Minimum k-Median problem in d dimensions, which means that the problem has an fpt-AS with combined parameters k and d. This can be considered as a generalization of EPTAS: the constant factor depends not only on ϵ, but also on the parameters of the instance. Sections 3.1 and 3.2 review this type of approximation results as well.

3.1. Measure parameters

In this section we overview parameterized approximation results where the parameter is some obvious measure of the input instance.

The *Traveling Salesperson Problem* (TSP) is one of the most studied optimization problems. The input consists of n cities and a matrix describing the distances between the cities (the distances can be arbitrary nonnegative integers). The task is to start from one of the cities, visit every city in some order, and then return to the starting city. We have to find an ordering of the cities such that the length of this tour is as small as possible. The problem is NP-hard, in fact, there is no c-approximation algorithm for any $c \geq 1$ (unless P = NP). The *Metric Traveling Salesperson* problem is a restricted version of TSP where we assume that the distance matrix satisfies the triangle inequality $d(x, y) \leq d(x, z) + d(z, y)$ for any three cities x, y, z. This assumption obviously holds in applications where we know that the direct route is never longer than a route via multiple cities. Metric TSP can be 1.5-approximated using Christofides' Algorithm [31], but has no polynomial-time approximation scheme, unless P = NP [97].

Metric TSP with Deadlines (ΔDLTSP) is investigated in [14]. Here we have the further restriction that some subset D of the vertices have deadlines assigned to them, and we have to visit each such vertex before its deadline. That is, each $v \in D$ has a deadline $d(v)$, and the length of the tour from the starting city to v must not exceed $d(v)$. The task is to find the shortest tour that satisfies all the deadlines. The problem has no polynomial-time constant factor approximation algorithm, unless P = NP. Furthermore, it is not fixed-parameter tractable if the parameter is the number $|D|$ of deadline vertices; in fact, it is NP-hard even if there is only one deadline vertex. Thus neither approximation nor parameterized complexity alone can tackle this problem. However, [14] presents a $O(|x|^3 + k!k)$ time algorithm that produces a 2.5-approximate solution, if k is the number of deadline vertices. Therefore, the combination of approximation and parameterized complexity results in an efficient approximation algorithm for small values of k.

Theorem 3.1. *([14])* Metric TSP with Deadlines *parameterized by the number of deadline vertices has an fpt 2.5-approximation algorithm.*

The Vertex Cover problem (cover all the edges of a graph with as few vertices as possible) is well-studied both in the parameterized complexity and approximation algorithms literature. Vertex Cover is fixed-parameter tractable (see e.g., [29, 112]) and has a simple 2-approximation algorithm. However, the problem does not have a PTAS (unless P = NP); in fact, some recent conjectures imply that the factor 2 is best possible [76].

Partial Vertex Cover is in some sense the dual of Vertex Cover: given a graph G with an integer k, the task is to cover as many edges as possible with k vertices. It can be shown that this problem is W[1]-hard with parameter k [65]. The straightforward

greedy algorithm (repeatedly select a vertex that covers as many uncovered edges as possible) gives a 1.582-approximation [70], but the problem does not admit a PTAS [98]. Here we show that the problem admits an fpt-AS:

THEOREM 3.2. PARTIAL VERTEX COVER *admits an fpt-AS with parameter k, the number of vertices in the solution.*

Proof. We present an $f(\epsilon,k)\cdot|x|^{O(1)}$ time algorithm that produces a $(1+\epsilon)$-approximate solution for the problem. Let $D := 2\binom{k}{2}/\epsilon$ and let $v_1, \ldots, v_n$ be the vertices of the graph ordered by non-increasing degree, i.e., $d(v_i) \geq d(v_j)$ for $i < j$. We consider two cases:

Case 1: $d(v_1) \geq D$. In this case the algorithm outputs the set $S = \{v_1, \ldots, v_k\}$. These k vertices cover at least $\sum_{i=1}^{k} d(v_i) - \binom{k}{2}$ edges: there are at most $\binom{k}{2}$ edges between the vertices of S, thus at most $\binom{k}{2}$ edges are counted twice when the degrees are summed. The value of the optimum solution cannot be larger than $\sum_{i=1}^{k} d(v_i)$, hence the value of the constructed solution S is at least

$$\frac{\sum_{i=1}^{k} d(v_i) - \binom{k}{2}}{\sum_{i=1}^{k} d(v_i)} \geq 1 - \frac{\binom{k}{2}}{D} = 1 - \frac{\epsilon}{2} \geq \frac{1}{1+\epsilon}$$

times the optimum, that is, the performance ratio is at most $1+\epsilon$.

Case 2: $d(v_1) \leq D$. In this case the optimum value is at most kD, and we are able to determine it exactly in fpt-time. For each $1 \leq \ell \leq kD$, the algorithm checks whether it is possible to cover at least ℓ edges with k vertices. We use the method of color coding: random colors between 1 and ℓ are assigned to the edges, and we try to find a solution where the ℓ edges covered by the vertices have distinct colors. As shown below, the existence of such a solution can be checked in fpt-time for a given coloring. If it is not possible to cover ℓ edges, then obviously the algorithm does not find a solution for any random coloring. On the other hand, if there is a solution, then the random coloring assigns distinct colors to the ℓ edges with probability at least $\ell^{-\ell}$. Therefore, we have to repeat the algorithm on average at most ℓ^{ℓ} times to find a solution. Note that ℓ^{ℓ} can be bounded by a function of ϵ and k only. The algorithm can be derandomized using standard techniques, see [54, Section 13.3], [45, Section 8.3], and [3].

For a particular coloring, we proceed as follows. We consider every possible partition $\mathcal{P} = \{P_1, \ldots, P_k\}$ of the color set $\{1, \ldots, \ell\}$ into k classes; there are at most k^{ℓ} such partitions. For a given partition $\mathcal{P}$, we try to find vertices $u_1, \ldots, u_k$ such that for each color $c \in P_i$, vertex u_i covers at least one edge with color c. This can be easily done in polynomial time. □

One of the most fruitful areas for developing approximation algorithms is the field of geometric problems. The geometric nature of the problem allows us to use all sorts of approximation techniques. Very often, the problem even admits a PTAS. The computational geometry community has been doing parameterized approximation analysis for several years: the quest for obtaining approximation algorithms where certain constants do not appear in the exponent of the input size is evident in the literature.

The MINIMUM k-CENTER problem arises as a fundamental problem in many applications such as facility location, clustering, and information retrieval. Given a set of points in the plane and an integer k, the task is to select k center points such that each input point is close to some selected point. More precisely, we have to minimize the maximum distance of a point to the closest selected point.

MINIMUM k-CENTER	
Input:	A set $S \subseteq \mathbb{R} \times \mathbb{R}$ of points, an integer k.
Find:	A subset $C \subseteq S$ of size k.
Goal:	Minimize $\max_{s \in S} \min_{c \in C} d(s,c)$.

There are several ways of interpreting distance in the plane. Here we discuss the two most common cases, where the distance means L_2 metric ($d((x_1,y_1),(x_2,y_2)) = \sqrt{|x_1-x_2|^2 + |y_1-y_2|^2}$) or L_∞ metric ($d((x_1,y_1),(x_2,y_2)) = \max\{|x_1-x_2|, |y_1-y_2|\}$). In both cases, the problem can be 2-approximated in polynomial time, but there is some constant $\alpha < 2$ such that it is NP-hard to find an α-approximation [56]. If the number of centers is a fixed constant k, then the problem can be solved exactly in time $|x|^{O(k)}$ by brute force. This cannot be improved to $f(k)\cdot|x|^{O(1)}$ time: parameterized by the number k of centers, the problem is W[1]-hard in the L_∞ metric [83]. However, there are efficient approximations for every fixed value of k: [1] shows that for every fixed k there is a linear-time approximation scheme, i.e., a $(1+\epsilon)$-approximate solution can be found in $f(\epsilon,k)\cdot n$ time. Using the terminology of parameterized approximation,

THEOREM 3.3. *([1])* MINIMUM k-CENTER *admits an fpt-AS with parameter k.*

MINIMUM k-MEDIAN is the variant of the problem where instead of minimizing the *maximum* distance to the closest center point, the optimization goal is to minimize the *sum* of these distances. Unlike MINIMUM k-CENTER, this problem admits an EPTAS in the L_2 metric (even if k is part of the input) [66].

Most geometric problems can be easily generalized to higher dimensions. If the dimension of the input can be arbitrarily large, then usually the problem becomes as hard as on any general metric, since we lose the nice geometric properties that make approximation easy (cf. [114]). However, in many cases the geometric approximation schemes can be generalized to any fixed dimension d. The approximation schemes for MINIMUM k-CENTER [1] and for MINIMUM k-MEDIAN [66] work for an arbitrarily fixed dimension d, and the running

time is $f(\epsilon,k,d)\cdot|x|^{O(1)}$ and $f(\epsilon,d)\cdot|x|^{O(1)}$, respectively. Therefore,

Theorem 3.4. *([1, 66])* Minimum k-Median *admits an fpt-AS with parameter d.* Minimum k-Center *admits an fpt-AS with combined parameters d and k.*

Dimension is a very natural parameter for the study of geometric problems. In the examples above, fpt-algorithms are possible if the dimension is taken as parameter. Conceivably, there can be problems where it is possible to show that the exponent of the input size has to increase as the dimension increases. There is very little work done in the literature on proving hardness results for problems parameterized by the dimension of the space. The only such result that we are aware of is on the Subset Congruence problem. Given two sets of points $A, B \subseteq \mathbb{R}^d$, the task is to decide if there is a distance-preserving transformation that makes A equal to a subset of B. The problem is polynomial-time solvable for every fixed d, but W[1]-hard parameterized by the dimension d [21].

3.2. Structural parameters

In the previous section we discussed parameters that measure the size of some part of the input or the solution. These parameters were "obvious." Parameterization can be more subtle than that: it can measure arbitrary properties of the input structures. In many cases, the parameter measures how complicated the input is, in some precise, well-defined sense. The motivation is to obtain algorithms that are efficient for less complex instances, where less complex means that the parameter value is small. Such an algorithm can be useful in applications where it is a reasonable assumption that the input is not completely unstructured, and the value of this particular parameter is usually small.

Below we list some parameters that describe complexity in graphs. The list is not exhaustive; depending on the problem or application, many such parameters can be defined.

- **Maximum degree.** The maximum degree of a graph is the maximum number of neighbors a vertex can have.
- **Diameter.** The diameter of a graph is the maximum distance between any two vertices.
- **Tree width.** The tree width is a parameter that describes how "tree-like" the graph is. We omit the technical definition, see e.g., [16] for more details. In many cases, problems on graphs with small tree width can be handled with dynamic programming techniques.
- **Genus.** The genus of a graph describes how close the graph is to being a planar graph. A graph has genus 0 if and only if it is planar (i.e., can be drawn on the sphere without crossing edges). A graph has genus at most k if it can be drawn on the sphere with k "handles" attached to the sphere (cf. [41, Appendix B]).
- **Distance from a class $\mathcal{F}$.** Let $\mathcal{F}$ be an arbitrary class of graphs (such as planar, bipartite, interval, etc.) The class $\mathcal{F}+kv$ (resp., $\mathcal{F}+ke$) contains those graphs that can be constructed from some $G\in\mathcal{F}$ with the addition of k new vertices[2] (resp., edges). The classes $\mathcal{F}-kv$ and $\mathcal{F}-ke$ are similarly defined. For any problem and any class $\mathcal{F}$, we can assume that the input is in, say, $\mathcal{F}+kv$ for some k, and define this k to be the parameter. This line of research was initiated by Cai [22], and later pursued in e.g., [64, 86].

The first problem that we investigate here is Vertex Coloring: given a graph G, find a coloring of the vertices with minimal number of colors such that adjacent vertices receive different colors. The minimum number of colors that is required to color the vertices of G is called the *chromatic number* of G and is denoted by $\chi(G)$. Chromatic number is not approximable within $|V|^{1-\epsilon}$ for any $\epsilon>0$, unless ZPP = NP [49]. The problem remains hard for planar graphs: it is NP-complete to decide whether a planar graph is 3-colorable [111]. However, by the celebrated Four Color Theorem [5], four colors are sufficient for every planar graph, and there is a polynomial-time algorithm that actually finds this coloring [104]. As it is easy to check whether a graph is 2-colorable, a $\frac{4}{3}$-approximate coloring can be found in polynomial time for planar graphs.

Is there a constant factor approximation algorithm for graphs that are "almost" planar graphs? We consider two classes of graphs that are close to being planar: bounded genus graphs and planar+kv graphs. For bounded genus graphs, the following result is implicit in [38]:

Theorem 3.5. *([38])* Vertex Coloring *has an fpt 2-approximation algorithm if the parameter is the genus of the graph.*

For planar+kv graphs, an easy brute force algorithm gives a $\frac{7}{3}$-approximation:

Theorem 3.6. Vertex Coloring *has an fpt $\frac{7}{3}$-approximation algorithm for planar+kv graphs.*

Proof. Let $X=\{v_1,\ldots,v_k\}$ be k vertices of G such that $G\setminus X$ is planar (here we gloss over the question how this set is found or whether it is given in the input together with the planar+kv graph, cf. [88]). We can determine $\chi(G[X])$ by trying all the k^k possible colorings of $G[X]$. As discussed above, a $\frac{4}{3}$-approximate coloring of the planar graph $G\setminus X$ can be found in polynomial time. The coloring on $G[X]$ and the coloring on $G\setminus X$ can

[2] More precisely, we add k new vertices and connect them with each other and with the old vertices arbitrarily.

be combined to obtain a coloring of the whole graph with $c = \chi(G[X]) + \frac{4}{3} \cdot \chi(G \setminus X)$ colors. It is easy to check that $c \leq \frac{7}{3}\chi(G)$; the worst case occurs when $\chi(G) = \chi(G[X]) = \chi(G \setminus X) = 3$, but the algorithm produces a coloring with $3 + 4 = 7$ colors. □

Note that Theorems 3.5 and 3.6 are incomparable: bounded genus graphs might not be planar+kv for any k (e.g., large toroidal grids) and planar+$1v$ graphs can have unbounded genus (e.g., a grid with an additional vertex connected to every vertex).

The MAXIMUM INDEPENDENT SET problem is NP-hard for planar graphs [78], but has a linear-time approximation scheme [79, 11]. It follows from [60] and [36] that this can be generalized to bounded-genus graphs:

THEOREM 3.7. *([60, 36])* MAXIMUM INDEPENDENT SET *admits a linear-time fpt-AS if the parameter is the genus of the graph.*

For planar+kv graphs, a $(1 + \epsilon)$-approximation of MAXIMUM INDEPENDENT SET can be found by trying all 2^k possibilities on the k-extra vertices, and then by finding a $(1+\epsilon)$-approximation on the remaining graph.

THEOREM 3.8. MAXIMUM INDEPENDENT SET *admits a linear-time fpt-AS on planar+kv graphs.* □

In VERTEX COLORING the goal is to minimize the number of different colors used. In other words, if we identify the set of colors with the set of positive integers, then we try to minimize the *maximum* of the colors assigned. In MINIMUM SUM COLORING, the optimization goal is to minimize the *sum* of the colors (positive integers) on the vertices. Besides its combinatorial interest, the problem is motivated by applications in scheduling and VLSI design [12, 92]. The MINIMUM SUM EDGE COLORING problem is defined analogously. MINIMUM SUM COLORING is linear-time solvable on bounded tree width graphs [73], but MINIMUM SUM EDGE COLORING is one of the few problems that are polynomial-time solvable on trees [55, 107], but NP-hard already on graphs with tree width 2 [81]. However, MINIMUM SUM EDGE COLORING has a linear-time PTAS for bounded tree width graphs (in fact, this is true even for the much more general multicoloring version):

THEOREM 3.9. *([84])* MINIMUM SUM EDGE COLORING *admits an fpt-AS if the parameter is the tree width.*

Note that without parameterization we cannot obtain an approximation scheme for MINIMUM SUM EDGE COLORING, since the problem is APX-hard [81].

4. PARAMETERIZATION BY COST

In Section 3, our aim was to develop approximation algorithms that work efficiently if some parameter of the input is small. Perhaps the most obvious parameter of an optimization problem instance is the optimum cost. What we would like to have is an algorithm that efficiently finds an approximation of the optimum, if this optimum is small. Notice that many of the standard, well-studied problems in the parameterized complexity literature are standard parameterizations of certain optimization problems (e.g., MINIMUM VERTEX COVER, MAXIMUM CLIQUE, MINIMUM DOMINATING SET, LONGEST PATH, etc.) By studying the fixed-parameter tractability of these problems, we are investigating the possibility of having efficient *exact* algorithms for these problems in the case when the optimum is small. When a W[1]-hardness result shows that such an exact algorithm is unlikely to exist, then it is natural to study whether it is possible to *approximate* the optimum, if it is small.

Recently, at least three papers tried to extend parameterized complexity into this directions [24, 30, 43]. In Section 3, the definition of fpt-approximability (when the parameter is some property of the optimization instance) was a straightforward generalization of polynomial-time approximability. The definition becomes technically more delicate if we want to parameterize by cost. The first complication is that we have to decide whether we want to parameterize by the optimum cost (which is somewhat counterintuitive, since presumably the cost is hard to determine), or we assume that the input contains a parameter k (the cost that should be reached), as in the standard parameterization of the problem. Here we follow the definition proposed by Chen et al.[30]:

DEFINITION 4.1. *Let $X = (I, \mathrm{sol}, \mathrm{cost}, \mathrm{goal})$ be an optimization problem. A* standard fpt-approximation algorithm with performance ratio c *for X is an algorithm that, given an input $(x, k) \in \Sigma^* \times \mathbb{N}$ satisfying*

$$\begin{cases} \mathrm{opt}(x) \leq k & \text{if } \mathrm{goal} = \min, \\ \mathrm{opt}(x) \geq k & \text{if } \mathrm{goal} = \max, \end{cases} \qquad (*)$$

computes a $y \in sol(x)$ in time $f(k) \cdot |x|^{O(1)}$ such that

$$\begin{cases} \mathrm{cost}(x, y) \leq k \cdot c & \text{if } \mathrm{goal} = \min. \\ \mathrm{cost}(x, y) \geq k/c & \text{if } \mathrm{goal} = \max, \end{cases} \qquad (**)$$

For inputs not satisfying condition (*), the output can be arbitrary.

The word "standard" signifies that this is the approximation version of the standard parameterization, and distinguishes it from the case when the parameter is unrelated to the cost (as in Section 3).

Unfortunately, we do not have a good example for this form of approximability, an example where a problem is not fixed-parameter tractable, but has a standard fpt-approximation algorithm with some performance ratio $c > 1$. Nevertheless, there are some examples of fpt-approximation algorithms appearing in the literature. Tree width is fixed-parameter tractable, in fact for every k, a tree decomposition of width k can be

computed in linear time if exists [17]. This algorithm is quite complicated and not practical. However, there are much simpler fpt-algorithms that produce tree decompositions having width at most a constant factor larger than the optimum [105, 102, 103, 4]. In the case of the rank width of graphs and branch width of matroids represented over a fixed finite field, we have the curious situation that it can be decided in fpt-time whether the width is at most k [33, 68], but it is not known whether a decomposition of width at most k (if it exists) can be constructed in fpt-time. However, standard fpt-approximation algorithms exist for these problems [96, 95, 68]

One might be tempted to define the notion of standard fpt-approximation scheme to capture the situation where a problem has a standard fpt-approximation algorithm for every $c > 1$. More precisely, we would like to have an analog of EPTAS, since approximation schemes where the exponent of input size increases as ϵ decreases is not very interesting from the parameterized complexity point of view. Thus the natural definition for a standard fpt-approximation scheme would be that for every $\epsilon > 0$, there is a standard fpt-approximation algorithm with performance ratio $1+\epsilon$, and the running time is $f(k,\epsilon)\cdot |x|^{O(1)}$. However, this notion is not very interesting to study, since it implies fixed-parameter tractability; therefore, it does not extend the class of problems that we can call tractable. The proof is essentially the same as the proof of Prop. 2.

PROPOSITION 1. *Assume that there is an algorithm that has an input parameter ϵ, and for every $\epsilon > 0$ it produces a $(1+\epsilon)$-approximate solution for optimization problem X in time $f(k,\epsilon)\cdot|x|^{O(1)}$. Then the standard parameterization of X is in FPT.* □

5. NON-CONSTANT PERFORMANCE FUNCTIONS

When designing approximation algorithms, the usual aim is to ensure that the cost of the solution differs from the optimum by at most a constant factor. This means that the algorithm has to perform equally well regardless of whether the optimum is "small" or "large." The analysis can be made more precise if the performance of the algorithm is bounded by a function of the optimum cost, rather than by an absolute constant. This is especially natural in the case of standard fpt-approximation (Section 4): if the goal is to have an algorithm that is efficient for small cost values, then it makes sense to require good performance only if the optimum is small.

Let $\varrho : \mathbb{N} \to \mathbb{N}$ be a nondecreasing function. We say that an approximation algorithm for optimization problem X has *performance ratio function* $\varrho(k)$ (or it is a $\varrho(k)$-approximation algorithm) if it produces a solution with performance ratio at most $\varrho(\mathrm{opt}(x))$ for every instance x. For example, if the solution has cost $\mathrm{opt}(x)\log\mathrm{opt}(x)$, then it is an log-approximation algorithm. Constant-factor approximability is the special case where $\varrho(k)$ is a constant function. The definition of parameterized fpt-approximation (Section 3) can be extended similarly. To define standard fpt-approximability with performance ratio function $\varrho(k)$, (**) of Definition 4.1 has to be replaced with

$$\begin{cases} \mathrm{cost}(x,y) \le k\cdot\varrho(k) & \text{if goal} = \min, \\ \mathrm{cost}(x,y) \ge k/\varrho(k) & \text{if goal} = \max. \end{cases}$$

If our goal is only to obtain an approximation algorithm with *some* performance function (no matter how bad it is), then in the case of maximization problems, fpt time does not give us more power than polynomial time. This surprising result was observed by Grohe and Grüber [61]:

THEOREM 5.1. *If a maximization problem X has a standard fpt-approximation algorithm with performance ratio function $\varrho(k)$, then there is a polynomial-time $\varrho'(k)$-approximation algorithm for X, for some function $\varrho'(k)$.*

Proof. Assume that X has a standard fpt-approximation algorithm with running time $f(k)\cdot|x|^{O(1)}$ and performance ratio function $\varrho(k)$. There is a computable nondecreasing function $b(n)$ such that every instance of X with size at most n has optimum at most $b(n)$. Denote by $c(n)$ the largest i such that $i \le n$, $f(i) \le n$ and $f(i)$ can be computed in time $O(n)$; clearly, $c(n)$ is unbounded, nondecreasing, and computable in time $n^{O(1)}$.

Given an instance x, the polynomial-time approximation algorithm proceeds as follows. For every $k = 1, 2, \ldots, \max(c(|x|), 1)$, we run the assumed fpt-approximation algorithm on x, and select the best solution returned by the different runs (since we try $k = 1$ and $\mathrm{opt}(x) \ge 1$ holds, some solution is always found). From $k \le \max(c(|x|), 1)$ it follows that the $f(k)\cdot|x|^{O(1)}$ time fpt-algorithm runs in polynomial time, hence the running time of the whole procedure is polynomial in $|x|$.

If $c(|x|) \ge \mathrm{opt}(x)$, then the fpt-approximation algorithm is invoked with $k = \mathrm{opt}(x)$, which means that a solution of cost at least $\mathrm{opt}(x)/\varrho(\mathrm{opt}(x))$ is produced. Assume therefore that $c(|x|) < \mathrm{opt}(x)$. Let $d(m)$ be the largest n such that $b(n) < m$. This means that $|x| > d(\mathrm{opt}(x))$ for every instance x, hence $c(|x|) \ge c(d(\mathrm{opt}(x))$. Thus when the fpt-approximation algorithm is invoked with $k = c(d(\mathrm{opt}(x))) < \mathrm{opt}(x)$, then it produces a solution with value at least $r(\mathrm{opt}(x)) := c(d(\mathrm{opt}(x)))/\varrho(c(d(\mathrm{opt}(x))))$. It is easy to see that $r(\mathrm{opt}(x))$ is an unbounded computable function of $\mathrm{opt}(x)$, hence the polynomial-time algorithm is a approximation algorithm with performance ratio function $\varrho'(k) = k/r(k)$. □

We remark that the analog of Theorem 5.1 does not seem to hold for minimization problems.

5.1. Examples (polynomial-time).

Expressing the performance ratio of an algorithm as a function of the optimum can be interesting even in the case of polynomial-time approximation algorithms, independently of parameterized complexity. We cite here three results of this type. Determining tree width is NP-hard [19], and no constant factor polynomial-time approximation algorithm is known (it is an important open question whether such an approximation is possible). However, the following approximation result was obtained recently by Feige et al. [48]:

THEOREM 5.2. *([48]) Given a graph with tree width k, a tree decomposition of width at most $O(k\sqrt{\log k})$ can be constructed in polynomial time.*

That is, tree width can be $O(\sqrt{\log k})$-approximated in polynomial time. (The previous best result was a $O(\log k)$-approximation [4, 20].)

Whether MINIMUM DIRECTED FEEDBACK VERTEX SET is in FPT or not is one of the most important open question of parameterized complexity. In this problem the task is to find a minimum set of vertices whose deletion makes the given directed graph acyclic. The undirected version of the problem is known to be in FPT (cf. [45, 34]), but apparently none of the techniques go through for the directed case. It seems that a much deeper understanding of the structure of directed graphs is required before this problem can be settled. However, it is observed in [30] that the results of [47, 108] imply an approximation algorithm with bounded performance ratio function:

THEOREM 5.3. *([30])* MINIMUM DIRECTED FEEDBACK VERTEX SET *has a polynomial-time approximation algorithm with performance ratio function $O(\log k \cdot \log\log k)$.*

In the MAXIMUM DISJOINT CYCLES problem the task is to find k pairwise vertex disjoint cycles in a given graph. This problem is fixed-parameter tractable (cf. [45, Section 8.1]). On the other hand, the directed version of the problem is W[1]-hard [110, 30], but polynomial-time solvable for every fixed value of k [100]. Based on the theoretical results proved in [100], Grohe and Grüber [61] presented an fpt-approximation algorithm. By Theorem 5.1, the algorithm can be made to work in polynomial time.

THEOREM 5.4. *([61]) There is a polynomial-time approximation algorithm for* MAXIMUM DISJOINT DIRECTED CYCLES *with performance ratio function $\varrho(k)$, for some computable function $\varrho(k)$.*

The function $\varrho(k)$ is not given explicitely in [61], but $k/\varrho(k)$ is very slowly growing (even in the fpt-time version of the algorithm, before applying Theorem 5.1).

5.2. Examples (fpt-time).

Graph layout problems arise in many application domains such as scheduling, VLSI design, and archaeology [40]. A *linear layout* of a graph $G(V, E)$ is a one-to-one mapping σ between V and $\{1, \ldots, |V|\}$. The *bandwidth* of a layout is $\max_{uv \in E} |\sigma(u) - \sigma(v)|$, the maximum "length" of an edge in the layout. The *cutwidth* of a layout is $\max_{1 \leq i < |V|} |\{uv \in E : \sigma(u) \leq i \text{ and } \sigma(v) > i\}|$, the maximum number of edges that cross the cut formed by two neighboring vertices in the layout. The bandwidth (resp., cutwidth) of a graph is the minimum bandwidth (resp., cutwidth) over all possible linear layouts of the graph. Bandwidth is W[1]-hard [18], while cutwidth is fixed-parameter tractable [113].

Subdividing an edge with a new degree two vertex might decrease the bandwidth of the graph. *Topological bandwidth* is the smallest bandwidth that can be achieved by repeatedly subdividing edges. Topological bandwidth is known to be W[1]-hard [15]. Fellows [50] observed that if $\mathrm{tbw}(G)$ (resp., $\mathrm{cw}(G)$) is the topological bandwidth (resp., cutwidth) of graph G, then

$$\mathrm{tbw}(G) \leq \mathrm{cw}(G) + 1 \tag{1}$$

and

$$\mathrm{cw}(G) < \mathrm{tbw}(G)^2 \tag{2}$$

holds. This means that if a graph has topological bandwidth at most k, then one can obtain a layout with cutwidth less than k^2 using the algorithm of [113]. Furthermore, using (1) (whose proof is algorithmic), this layout can be turned into a layout for a subdivision of G with bandwidth at most k^2. Therefore,

THEOREM 5.5. *([50])* TOPOLOGICAL BANDWIDTH *has a standard fpt-approximation algorithm with performance ratio function k.*

The graph parameter *clique width* was introduced by Courcelle et al. [32]. Similarly to tree width, it measures the complexity of a graph with respect to certain composition operators. A k-expression describes a way of constructing the graph using these operations; the clique width of a graph is the smallest k such that it has a k-expression. Fellows et al. [52] have shown that determining clique width is NP-hard, in fact, it cannot be approximated with an absolute error guarantee, unless P = NP. Seymour and Oum [96] use the approximability of rank width to obtain an approximation algorithm for clique width. They show that if $\mathrm{cw}(G)$ is the clique width and rw(G) is the rank width of a graph G, then

$$\mathrm{rw}(G) \leq \mathrm{cw}(G) \leq 2^{\mathrm{rw}(G)+1} - 1$$

holds. The proof is algorithmic: given a rank decomposition of width k, a $(2^{k+1} - 1)$-expression can be obtained in polynomial time. There is an fpt-approximation algorithm that computes a rank

decomposition of width $3\text{rw}(G)+1$ [95]. Therefore, clique width can be approximated by first computing an approximate rank decomposition and then turning it into a k-expression:

Theorem 5.6. *([95])* Clique Width *has a standard fpt-approximation algorithm with performance ratio function* $(2^{3k+2}-1)/k$.

The significance of Theorem 5.6 comes from the fact that certain optimization problems are linear-time solvable for every k, if a k-expression of the graph is given in the input. That is, these problems are fixed-parameter tractable parameterized by clique width, but only if the corresponding k-expression is given in the input. Theorem 5.6 can be used to remove the requirement that the k-expression is given in the input: if we know that the graph has a k-expression, then an $f(k)$-expression can be obtained in fpt-time, where $f(k)=2^{3k+2}-1$. Then, instead of using the algorithm for k-expressions, we can use the algorithm for $f(k)$-expressions: the running time might be much larger, but it is still linear with a constant depending only on k. Therefore, obtaining the approximation and then solving the problem using the $f(k)$-expression can be done in fpt-time.

5.3. Negative results.

For certain problems, it is possible to show that there is no standard fpt-approximation algorithm for *any* performance ratio function. Given a graph $G(V,E)$, an independent dominating set is an independent set $S \subseteq V$ of vertices such that for every $v \in V$, either $v \in S$, or v is a neighbor of a member of S. The Minimum Independent Dominating Set problem is the corresponding optimization problem, where the goal is to minimize the size of the set S. Downey et al. [43] prove that this problem is completely inapproximable:

Theorem 5.7. *([43]) If* Minimum Independent Dominating Set *has a standard fpt-approximation algorithm with performance ratio function* $\varrho(k)$ *for some computable function* $\varrho(k)$*, then* W[2] = FPT.

Chen et al. [30] presents inapproximability results for satisfiability problems. In the Min-WSAT(CIRC) problem a Boolean circuit is given and the task is to find a satisfying assignment of minimum weight (where the weight of an assignment is the number of true variables).

Theorem 5.8. *([30]) If* Min-WSAT(CIRC) *has a standard fpt-approximation algorithm with performance ratio function* $\varrho(k)$ *for some computable function* $\varrho(k)$*, then* W[P] = FPT.

Similar results hold for other variants of the satisfiability problem and for the parameterized halting problem [30].

All the inapproximability results presented above are somewhat unsatisfying in the sense that the problems considered are not monotone. Monotone means that (in case of a minimization problem) if we extend a feasible solution with additional vertices/true variables, then it remains feasible. Clearly, this does not necessarily hold in these examples. Therefore, it can happen that the optimum is k, and every feasible solution has cost k, which makes approximation equivalent to finding an optimum solution. The inapproximability proofs in these examples tell us more about the hardness of finding exact solutions than about the hardness of approximation. It would be much more interesting (and possibly, more difficult) to have analogous inapproximability results for the monotone problems Maximum Clique and Minimum Dominating Set, for example.

6. QUALITY OF APPROXIMATION AS PARAMETER

A polynomial-time approximation scheme can produce solutions with approximation ratio arbitrary close to 1. However, we have to pay a price for that: the running time can be ridiculously large if ϵ is small. Table 1 (reproduced from [44]) presents the running time of some approximation schemes for $\epsilon = 0.2$. As we can see, running times of $O(|x|^{1,000,000})$ or worse is not uncommon if we require that the maximum error is 20%. Obviously, such algorithms are not useful in practice. Nevertheless, these results are important from the theoretical point of view, as they show that there is no lower bound on the approximation ratio that can be achieved in polynomial time.

The notion of *efficient polynomial-time approximation scheme (EPTAS)* tries to formalize a restricted class of approximation schemes, where the algorithm might have a chance of being useful. If the running time of the algorithm is of the form $c^{1/\epsilon}\cdot n$, and c is not too large, then the algorithm can be efficient for, say, $\epsilon = 0.2$. In many cases, the first approximation scheme obtained for a problem was not an EPTAS, but later it was improved to an EPTAS. For example, Arora [6] presented an $|x|^{O(1/\epsilon)}$ time PTAS for Euclidean TSP, which is not an EPTAS. However, in the journal version of the paper [7], the running-time of the algorithm is improved to $|x|\cdot\log^{O(1/\epsilon)}|x| = 2^{O(1/\epsilon^2)}\cdot|x|^2$, hence the problem admits an EPTAS. Arora et al. [8] presented an $|x|^{O(1/\epsilon)}$ time approximation scheme for the Euclidean k-median problem. Later this was improved to an EPTAS with running time $2^{O(1/\epsilon\cdot\log 1/\epsilon)}\cdot n$ [8, 66]. In certain cases, the effort to obtain an EPTAS was only partially successful. For example, Hunt et al. [71] show that Maximum Independent Set for unit disk graphs admits an $|x|^{O(1/\epsilon)}$ time PTAS. They are unable to present an EPTAS for the problem in general, but for the special case of λ-precision unit disk graphs (where the centers of the disk are not closer than λ from each other) they give a $2^{O(1/(\epsilon\lambda)^2)}\cdot n$ time EPTAS.

TABLE 1. The running time of some PTASs with 20% error.

Problem	Reference	Running time for 20% error
EUCLIDEAN TSP	Arora [6]	$O(\|x\|^{15,000})$
MULTIPLE KNAPSACK	Chekuri and Khanna [26]	$O(\|x\|^{9,375,000})$
MAXIMUM SUBFOREST	Shamir and Tsur [109]	$O(\|x\|^{958,267,391})$
MAXIMUM INDEPENDENT SET for disk graphs	Erlebach et al. [46]	$O(\|x\|^{523,804})$

If a problem resists every attempt to obtain an EPTAS, then we should consider looking for some negative evidence showing that no EPTAS is possible, i.e., that it is not possible to get $1/\epsilon$ out of the exponent of the input size (modulo some complexity-theoretic assumption). It is not very surprising that parameterized complexity can provide such evidence, since getting out certain parameters from the exponent is the central issue of the theory. The basic tool that can be used to prove negative results for the existence of EPTASs is the following connection, observed independently by Bazgan [13] and Cesati and Trevisan [25]:

PROPOSITION 2. *If an optimization problem X admits an EPTAS, then the standard parameterization of X is fixed-parameter tractable.*

Proof. Assume that we have an algorithm that produces a $(1+\epsilon)$-approximate solution in time $f(1/\epsilon) \cdot |x|^{O(1)}$. Given an instance (x,k) of the standard parameterization of X, we set $\epsilon := 1/(2k)$, and use the EPTAS to find a $(1+\epsilon)$-approximate solution in time $f(1/\epsilon) \cdot |x|^{O(1)} = f(2k) \cdot |x|^{O(1)}$. Assume first that X is a minimization problem. If the optimum is at most k, then the cost of the approximate solution is at most $(1+\epsilon)k \leq k + 1/2 < k+1$. As the cost is integer, the cost of the approximate solution is at most k. If the optimum is greater than k, then the cost of the approximate solution is also greater than k. Therefore, by checking whether the cost of the approximate solution is at most k, we can decide whether the optimum is at most k. The proof is similar in the case of maximization problems: if the optimum is at least k, then the cost of an $(1+\epsilon)$-approximate solution is at least $k/(1+\epsilon) = k-1/(2+1/k) > k-1$. □

We can use the contrapositive of Prop. 2 to show that it is unlikely that a particular problem admits an EPTAS:

COROLLARY 1. *If the standard parameterization of an optimization problem is* W[1]*-hard, then the optimization problem does not have an EPTAS (unless* FPT = W[1]*).* □

It has to be remarked that the converse of Prop. 2 is not true. For example, MINIMUM VERTEX COVER is fixed-parameter tractable, but it is APX-hard, hence it does not even have a PTAS. Therefore, Prop. 2 has limited applicability.

6.1. Examples.

Table 2 lists problems where the existence of an EPTAS was ruled out using Prop. 2. The first four problems are geometric problems. For a set V of geometric objects, the *intersection graph* of V is a graph with vertex set V where two vertices are connected if and only if the two objects have non-empty intersection. Intersection graphs of disks, rectangles, line segments and other objects arise in applications such as facility location [116], frequency assignment [80], and map labeling [2]. The first four lines in Table 2 show that if we restrict MAXIMUM INDEPENDENT SET and MINIMUM DOMINATING SET to the intersection graphs of unit-radius disks or unit-size squares, then the problem admits a PTAS [71, 94, 93], but does not have an EPTAS [83, 87]. COVERING POINTS WITH SQUARES is also a geometric problem, but it is not defined in terms of intersection graphs. In this problem n points are given in the plane, and the task is to cover all of them with at most k (axis-parallel) unit squares. The squares can be at arbitrary locations in the plane.

DISTINGUISHING SUBSTRING SELECTION and CLOSEST SUBSTRING are two pattern matching problems that are motivated by applications in computational biology. In the CLOSEST SUBSTRING problem the task is to find a string of length L that approximately appears in each of the k input strings, where "approximately appears" means that it appears with at most d mismatches. The goal is to minimize this number d. In the DISTINGUISHING SUBSTRING SELECTION problem the input strings are divided into "good" strings and "bad" strings, and we have to find a string of length L that approximately appears in each of the good strings, but does not appear (even approximately) in any of the bad strings. We do not give a more detailed description of these problems, as many variants, parameterizations, and optimization goals are defined in the literature, see [77, 59, 58, 82, 51].

Khanna and Motwani [75] defined classes of optimization problems that have polynomial-time approximation schemes. The key to approximability in these problems is the underlying planar structure, which allows us to use Baker's layering technique [11]

TABLE 2. Problems where a W[1]-hardness result rules out the possibility of an EPTAS.

Problem	PTAS result	W[1]-hardness
Maximum Independent Set for unit disks	$\|x\|^{O(1/\epsilon)}$ [71]	[83]
Maximum Independent Set for unit squares	$\|x\|^{O(1/\epsilon)}$ [71]	[83]
Minimum Dominating Set for unit disks	$\|x\|^{O(1/\epsilon)}$ [71]	[87]
Minimum Dominating Set for unit squares	$\|x\|^{O(1/\epsilon)}$ [71]	[87]
Covering Points with Squares	$\|x\|^{O(1/\epsilon^2)}$ [69]	[83]
Distinguishing Substring Selection	$\|x\|^{O(1/\epsilon)}$ [39]	[57, 28]
Closest Substring	$\|x\|^{O(1/\epsilon^4)}$ [77]	[82]
Planar MPSAT	$\|x\|^{O(1/\epsilon)}$ [74]	[23]
Planar TMIN	$\|x\|^{O(1/\epsilon)}$ [74]	[23]
Planar TMAX	$\|x\|^{O(1/\epsilon)}$ [74]	[23]
Planar Multivalued Max 3CSP	$\|x\|^{O(1/\epsilon)}$ [74]	[25]

and algorithms for bounded tree width. The problems are formulated using Boolean logical expressions. Recall that a formula in *disjunctive normal form* (DNF) is a disjunction of terms, e.g., $(x_1 \wedge \bar{x}_3) \vee (\bar{x}_1 \wedge \bar{x}_2 \wedge \bar{x}_4) \vee \bar{x}_3$. A DNF is *positive* (resp., negative), if every literal is positive (resp., negated). The weight of an assignment is the number of variables that are set to true.

MPSAT
Input: A collection $\mathcal{C} = \{\phi_1, \ldots, \phi_n\}$ of DNFs.
Find: An assignment ψ.
Goal: Maximize the number of DNFs satisfied by ψ.

TMIN
Input: A collection $\mathcal{C} = \{\phi_1, \ldots, \phi_n\}$ of positive DNFs.
Find: An assignment ψ that satisfies every DNF in $\mathcal{C}$.
Goal: Minimize the weight of ψ.

TMAX
Input: A collection $\mathcal{C} = \{\phi_1, \ldots, \phi_n\}$ of negative DNFs.
Find: An assignment ψ that satisfies every DNF in $\mathcal{C}$.
Goal: Maximize the weight of ψ.

These problems generalize many of the standard optimization problems: for example, it is easy to see that Max Cut can be reduced to MPSAT; Maximum Independent Set can be reduced to TMAX; and Minimum Vertex Cover can be reduced to TMIN. (In all three reductions, we associate a variable with each vertex and a DNF with each edge.) Cesati and Trevisan [25] study a problem of similar flavor, where the variables are not necessarily Boolean, they can have arbitrary domains. Let $x_1, \ldots, x_n$ be variables and let D_i be the domain of variable x_i. A *constraint* over $x_1, \ldots, x_n$ is an arbitrary relation over $D_1, \ldots, D_n$, i.e., each possible combination of values assigned to the variables either satisfies the constraint or not. A constraint is given in the input by listing the combination of values that satisfy the constraint.

Multivalued Max kCSP
Input: A collection $\mathcal{C} = \{C_1, \ldots, C_m\}$ of constraints; each constraint is on k variables.
Find: An assignment ψ.
Goal: Maximize the number of constraints satisfied by ψ.

Given an instance of the above problems, the *incidence graph* is a bipartite graph defined by associating a vertex to each variable and each clause (constraint), and by connecting each variable to every clause (constraint) where it appears. Khanna and Motwani [74] show that MPSAT, TMIN, TMAX, Multivalued Max kCSP all admit a PTAS if the incidence graph is planar. The running time is $|x|^{O(1/\epsilon)}$, hence it is not an EPTAS. Cesati and Trevisan [25] have shown that the standard parameterization of Planar Multivalued Max 3CSP is W[1]-hard, hence by Prop 2, the PTAS cannot be improved to an EPTAS, unless W[1] = FPT. Similar results were obtained for MPSAT, TMIN, TMAX by Cai et al. [23].

Theorem 6.1. *([23, 25]) The problems* MPSAT, TMIN, TMAX, Planar Multivalued Max 3CSP *do not admit an EPTAS, unless* W[1] = FPT.

6.2. Parameterization by $1/\epsilon$.

All the negative results in Table 2 were obtained using Prop. 2. While this method is simple and apparently works, there is something unnatural about it. Our aim is to prove that $1/\epsilon$ cannot be taken out of the exponent of the input size. In order to show this, we consider the standard parameterization, and prove that k cannot be taken out of the exponent. But couldn't it be done somehow more directly, by defining $1/\epsilon$ to be the parameter? Intuitively, it is clear that what we are trying to determine is whether the problem is fixed-parameter tractable, with $1/\epsilon$ being the parameter. However, it is not entirely clear what "defining $1/\epsilon$ to be the parameter" means. The problem is that

parameterized complexity theory is not developed for optimization problems, and optimization problems are usually studied via the standard parameterization. In the decision version, however, it is not immediately clear what the meaning of ϵ is, as we do not have to produce a solution. To make the idea of parameterization by $1/\epsilon$ technically rigorous, we introduce a "gap" version of the standard parameterization: the input comes with a promise stating that there is a gap between the yes-instances and the no-instances. Parameterization by $1/\epsilon$ is parameterization by this gap size. If X is an optimization problem, then we define

GAP-X

Input: An instance x of X, an integer k, and a rational $\epsilon > 0$ such that

$$\begin{cases} \text{either } \mathrm{opt}(x) \le k \text{ or } \mathrm{opt}(x) > (1+\epsilon)k \\ \qquad \text{if goal} = \min, \\ \text{either } \mathrm{opt}(x) \ge k \text{ or } \mathrm{opt}(x) < \dfrac{k}{1+\epsilon} \\ \qquad \text{if goal} = \max. \end{cases}$$

Parameter: $\lceil 1/\epsilon \rceil$

Decide: $\begin{cases} \mathrm{opt}(x) \le k & \text{if goal} = \min, \\ \mathrm{opt}(x) \ge k & \text{if goal} = \max. \end{cases}$

An approximation scheme for optimization problem X can be used to decide GAP-X. Let X be a minimization problem (the argument for maximization problems is similar). If (x, k, ϵ) is a yes-instance of GAP-X, then $\mathrm{opt}(x) \le k$, and the approximation scheme can produce a $(1+\epsilon)$-approximate solution with cost at most $(1+\epsilon)k$. On the other hand, if (x, k, ϵ) is a no-instance, then by assumption $\mathrm{opt}(x) > (1+\epsilon)k$ holds, thus the cost of the solution produced by the approximation scheme is strictly greater than $(1+\epsilon)k$. This means that deciding an instance (x, k, ϵ) of GAP-X is not more difficult than obtaining a $(1+\epsilon)$-approximate solution. Therefore, we have

PROPOSITION 3. *If optimization problem X admits a PTAS, then* GAP-X *is in* XP. *If X admits an EPTAS, then* GAP-X *is in* FPT. □

Recall that XP is the class of parameterized problems that can be decided in polynomial time for every fixed value of the parameter. Prop. 3 can be used to show that a problem X does not have an EPTAS: if GAP-X is W[1]-hard, then X does not have an EPTAS unless FPT = W[1]. This technique is potentially more powerful than using Prop. 2: it might apply even to problems whose standard parameterizations are fixed-parameter tractable, since k is *not* a parameter of GAP-X.

To demonstrate the use of Prop. 3, we reprove a result of [23] and show that PLANAR TMIN does not admit an EPTAS. The reduction is based on the framework of matrix-type reductions introduced in [87], but the presentation here is self-contained.

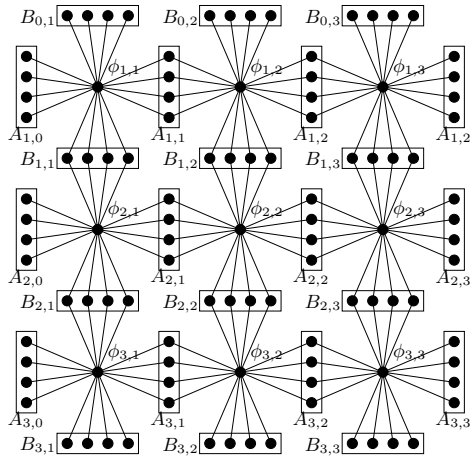

FIGURE 1. Structure of the instance constructed in Theorem 6.2 ($n = 4$, $t = 3$).

THEOREM 6.2. GAP-PLANAR-TMIN *is* W[1]*-hard. Thus* PLANAR TMIN *does not have an EPTAS, unless* FPT = W[1].

Proof. The reduction is from MAXIMUM CLIQUE. Given a graph G and an integer t, we construct an instance of GAP-PLANAR-TMIN with $k = 2t(t+1)$ and $\epsilon = 1/(2k)$ such that

- If G has a clique of size t, then the constructed instance has a solution with weight at most k.
- If G does not have a clique of size t, then the constructed instance does not have a solution with weight at most $(1+\epsilon)k$.

Assume that G has n vertices $\{1, 2, \ldots, n\}$. The constructed instance has $2t(t+1)n$ variables. The variables are arranged into blocks $A_{i,j}$ ($1 \le i \le t$, $0 \le j \le t$) and $B_{i,j}$ ($0 \le i \le t$, $1 \le j \le t$), where each block contains n variables. The variables in block $A_{i,j}$ (resp., $B_{i,j}$) will be denoted by $a_{i,j,s}$ (resp., $b_{i,j,s}$) for $s = 1, \ldots, n$.

We construct t^2 DNFs such that $\phi_{i,j}$ ($1 \le i, j \le t$) contains variables only from blocks $A_{i,j-1}$, $A_{i,j}$, $B_{i-1,j}$, $B_{i,j}$. As shown in in Figure 1, the incidence graph is planar. The formulas are defined as

$$\phi_{i,j} = \begin{cases} \bigvee_{s=1}^{n} (a_{i,j-1,s} \wedge a_{i,j,s} \wedge b_{i-1,j,s} \wedge b_{i,j,s}) \\ \qquad \text{if } i = j, \\ \bigvee_{(x,y) \in E(G)} (a_{i,j-1,x} \wedge a_{i,j,x} \wedge b_{i-1,j,y} \wedge b_{i,j,y}) \\ \qquad \text{otherwise.} \end{cases}$$

This completes the description of the instance. The optimum is either at most k, or at least $k+1 > (1+\epsilon)k = (1+1/(2k))k = k + 1/2$. Therefore, the gap size requirement is satisfied.

Assume that G has a clique $v_1, \ldots, v_t$. For every i, j, we set a_{i,j,v_i} of block $A_{i,j}$ and b_{i,j,v_j} of block $B_{i,j}$ to

true. We claim that this assignment of weight $2t(t+1)$ satisfies every formula. Indeed, if $i=j=r$, then $\phi_{i,j}$ is satisfied by the term $a_{i,j-1,v_r} \wedge a_{i,j,v_r} \wedge b_{i-1,j,v_r} \wedge b_{i,j,v_r}$; if $i \neq j$, then $\phi_{i,j}$ is satisfied by the term $a_{i,j-1,v_i} \wedge a_{i,j,v_i} \wedge b_{i-1,j,v_j} \wedge b_{i,j,v_j}$ (note that (v_i, v_j) is an edge of G, hence $\phi_{i,j}$ has such a term).

For the other direction, suppose that G has no clique of size t, but there is a satisfying assignment of weight at most $(1+\epsilon)k$. Notice that $(1+\epsilon)k < k+1$, hence the weight of the assignment is at most k. Furthermore, each block of variables has to contain at least one true variable, otherwise the adjacent formulas could not be satisfied. Therefore, the weight of the assignment is exactly k, and there is exactly one true variable in each block.

For every term of $\phi_{i,j}$, there is an s such that the term contains both $a_{i,j-1,s}$ and $a_{i,j,s}$. This means that if $\phi_{i,j}$ is satisfied, then the true variables in blocks $A_{i,j-1}$ and $A_{i,j}$ have the same index. Furthermore, this is true for all the blocks in row i, hence there is a value v_i such that a_{i,j,v_i} is true for every $1 \leq j \leq t$. Similarly, for every column j, there is a value v'_j such that variable b_{i,j,v'_j} is true. We claim that $v_i = v'_i$ for every $1 \leq i \leq t$. Indeed, if a term of $\phi_{i,i}$ is satisfied, then both $a_{i,i,s}$ and $b_{i,i,s}$ have to be true for some s, which implies $s = v_i = v'_i$. Now the vertices $v_1, \ldots, v_t$ form a clique: if v_i and v_j are not connected by an edge, then none of the terms in $\phi_{i,j}$ are satisfied. This proves the correctness of the reduction. □

6.3. Proving lower bounds.

The fact that an optimization problem does not have an EPTAS (unless FPT = W[1]) does not rule out the possibility of having a PTAS of running time $|x|^{O(\log\log\log 1/\epsilon)}$, for example. Such a running time would be almost as good as a polynomial-time algorithm. However, by a more careful analysis and by having a somewhat stronger complexity theoretic assumption, we can actually prove lower bounds on the exponent of $|x|$.

Maximum Clique can be solved in $|x|^{O(k)}$ time by brute force. It is believed that this is essentially best possible; in fact, Chen et al. [28] show that an $f(k)\cdot|x|^{o(k)}$ time algorithm would imply that 3SAT can be solved in $2^{o(n)}$ time. The fastest known algorithm for n-variable 3-SAT runs in randomized time 1.32216^n [106] (improving on the 2^n brute force algorithm). It is believed the running time of every algorithm must be exponential in n, i.e., no algorithm with running time $2^{o(n)}$ exists. This assumption is known as the Exponential Time Hypothesis (ETH) [72], and it is equivalent to the parameterized complexity conjecture FPT $\neq$ M[1] (see [42, 54]).

This lower bound on Maximum Clique can be transfered to other parameterized problems via reductions. Assume that a problem Q is proved W[1]-hard by a parameterized reduction from Maximum Clique, and the reduction constructs instances where the parameter is at most $p(k)$ (for some function p), where k is the parameter of the Maximum Clique instance. (The definition of parameterized reduction requires that such a function $p(k)$ exists.) If Q can be solved in time $f(k)\cdot|x|^{g(k)}$, then the reduction implies that Maximum Clique can be solved in time $f'(k)\cdot|x|^{g(p(k))}$. Therefore, problem Q cannot be solved in time $f''(k)\cdot|x|^{o(p^{-1}(k))}$ for any function $f''(k)$, unless ETH fails.

Prop. 2 shows that if an optimization problem admits an approximation scheme with running time $f(1/\epsilon)\cdot|x|^{g(1/\epsilon)}$, then the standard parameterization can be solved in $f(2k)\cdot|x|^{g(2k)}$ time. Therefore, a lower bound on the standard parameterization can be used to bound the efficiency of any PTAS for the problem. The W[1]-hardness results in Table 2 are obtained by a parameterized reduction from Maximum Clique, and these reductions increase the parameter at most by some polynomial function $p(k)$. Therefore, assuming ETH, for each of these problems there is an integer c such that there is no $f(k)\cdot|x|^{o(k^{1/c})}$ time algorithm for the standard parameterization, which implies that there is no $f(1/\epsilon)\cdot|x|^{o((1/\epsilon)^{1/c})}$ time PTAS. The only exception is Closest Substring: the reduction presented in [82] increases the parameter exponentially, thus all we can prove is that there is no $f(1/\epsilon)\cdot|x|^{o(\log 1/\epsilon)}$ time PTAS.

In a similar way, we can obtain lower bounds using Prop. 3. For example, the reduction in Theorem 6.2 constructs instances where the parameter $1/\epsilon$ is $O(t^2)$. Therefore, we have

Corollary 2. *There is no $f(1/\epsilon)\cdot|x|^{o(\sqrt{1/\epsilon})}$ time PTAS for* Planar TMIN*, unless ETH fails.*

This result is somewhat better than the $f(1/\epsilon)\cdot|x|^{o((1/\epsilon)^{1/4})}$ bound that follows from the proof in [23]. However, there is still a gap between this lower bound and the $|x|^{O(1/\epsilon)}$ upper bound given by [74]. It would be interesting to close this gap, possibly by proving stronger lower bounds. Using bidimensionality theory [37, 35], one may show that the standard parameterization of Planar TMIN can be solved in time $f(k)\cdot|x|^{O(\sqrt{k})}$. This means that no bound better than $f(1/\epsilon)|x|^{o(\sqrt{1/\epsilon})}$ can be obtained for Planar TMIN by using Prop. 2. Note that if we are using Prop. 3, then there is no such apparent barrier. It seems that we have to abandon Prop. 2 and directly parameterize by $1/\epsilon$, if we want to prove tight lower bounds on the efficiency of approximation schemes.

7. CONCLUSIONS

We have investigated several ways in which parameterized complexity and the theory of approximation algorithms can benefit from each other. It is not clear at the moment which directions will lead to fruitful areas of research. It seems that there are many examples

of fpt-approximability for parameterized instances (as discussed in Section 3), and it can be reasonably expected that many such results will follow. It is much less clear to what extent standard fpt-approximability (Section 4, 5) will be important. The relative lack of examples might be the sign that we are missing some fundamental method. Evaluating the efficiency of approximation schemes (Section 6) has become a standard technique, with results of this type steadily accumulating. Besides obtaining such results for a wider range of problems, here the challenge is to prove tight results.

There are some further connections between parameterized complexity and approximation algorithms that we have not touched upon. *Kernelization* of a parameterized problem is a polynomial-time preprocessing algorithm that constructs an equivalent instance such that the size of the new instance can be bounded by a function of the parameter. For example, Nemhauser and Trotter [91] present a preprocessing algorithm for the MINIMUM VERTEX COVER problem that constructs an instance with at most $2k$ vertices (where k is the parameter of the new instance). In many cases, such a linear-size kernelization algorithm can be used to obtain a constant-factor approximation algorithm. Therefore, any lower bound on the approximation ratio (obtained by the PCP theorems) immediately translates into a lower bound on the kernel size. (A different approach for obtaining kernelization lower bounds is proposed in [27].)

Another interesting connection appears in the use of the *iterative compression* technique. The main idea of iterative compression is that instead of solving the search problem "find a solution of size k," we solve the possibly easier compression problem "given a solution of size $k+1$, find a solution of size k." It turns out that for certain minimization problems if we can solve the compression problem, then we can solve the search problem as well by iteratively solving larger and larger problems [101, 34, 85, 62, 63, 88]. Furthermore, as observed in [63], if we can solve the compression problem and have a constant-factor approximation algorithm, then the approximate solution can be compressed, and there is no need for the iterative trick. In fact, an approximation algorithm with arbitrary performance ratio function (Section 5) would be sufficient for this application.

In *counting problems* the task is not only to check if there is a solution, but we have to determine the total number of solutions to the problem. The counting problem can be hard even for those problems where the existence of a solution is polynomial-time decidable. For example, checking whether a graph has a perfect matching can be done in polynomial time, but Valiant has shown that counting the number of perfect matching is complete for the complexity class #P [115]. A similar increase in complexity appears in the case of parameterized problems: finding a path of length k is fixed-parameter tractable [3, 90, 99], but counting the number of these paths is #W[1]-hard [53]. Given the hardness of exact counting, it is very natural to investigate the possibility of approximate counting. In the parameterized complexity framework, Arvind and Raman [9] present an fpt-time randomized approximation scheme for counting the number of subgraphs isomorphic to a bounded tree width graph H (thus, in particular, the algorithm can be used to approximate the number of paths of length k). Recently, Müller [89] proved a parameterized complexity analog of a classical result of Stockmeyer by showing that approximately counting the number of solutions for a problem in W[P] can be done in randomized fpt-time with a W[P] oracle.

Finally, to motivate further research, we propose four open questions, related to the four main issues discussed in the paper:

- Investigate the approximability of geometric problems, parameterized by the dimension.
- Find a convincing example where no fpt-time exact algorithm is known for a problem, but it has a standard fpt-approximation algorithm with constant performance ratio.
- Is there a standard fpt-approximation algorithm for MAXIMUM CLIQUE or MINIMUM DOMINATING SET with *some* performance ratio function?
- Prove tight lower bounds (assuming ETH) on the efficiency of the PTAS for some natural problem (MPSAT, TMIN, geometric problems, etc.)

ACKNOWLEDGMENTS

I would like to thank Mike Fellows for asking me to write this survey, and for the many ideas, comments, and references that were essential in compiling the material. I'm grateful to Martin Grohe and Magdalena Grüber for their comments on the manuscript.

REFERENCES

[1] P. K. Agarwal and C. M. Procopiuc. Exact and approximation algorithms for clustering. *Algorithmica*, 33(2):201–226, 2002.

[2] P. K. Agarwal, M. van Kreveld, and S. Suri. Label placement by maximum independent set in rectangles. *Comput. Geom.*, 11(3-4):209–218, 1998.

[3] N. Alon, R. Yuster, and U. Zwick. Finding and counting given length cycles. *Algorithmica*, 17(3):209–223, 1997.

[4] E. Amir. Efficient approximation for triangulation of minimum treewidth. In *UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, pages 7–15, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[5] K. Appel and W. Haken. *Every planar map is four colorable*, volume 98 of *Contemporary Mathematics*. American Mathematical Society, Providence, RI, 1989. With the collaboration of J. Koch.

[6] S. Arora. Polynomial time approximation schemes for Euclidean TSP and other geometric problems. In *FOCS 1996*, pages 2–11. IEEE Comput. Soc. Press, 1996.

[7] S. Arora. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *J. ACM*, 45(5):753–782, 1998.

[8] S. Arora, P. Raghavan, and S. Rao. Approximation schemes for Euclidean k-medians and related problems. In *STOC '98 (Dallas, TX)*, pages 106–113. ACM, New York, 1999.

[9] V. Arvind and V. Raman. Approximation algorithms for some parameterized counting problems. In *ISAAC '02: Proceedings of the 13th International Symposium on Algorithms and Computation*, volume 2518 of *Lecture Notes in Comput. Sci.*, pages 453–464, Berlin, 2002. Springer.

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

[11] B. S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *J. Assoc. Comput. Mach.*, 41(1):153–180, 1994.

[12] A. Bar-Noy, M. M. Halldórsson, G. Kortsarz, R. Salman, and H. Shachnai. Sum multicoloring of graphs. *J. Algorithms*, 37(2):422–450, 2000.

[13] C. Bazgan. Schémas d'approximation et complexité paramétrée. Technical report, Université Paris Sud, 1995.

[14] H.-J. Böckenhauer, J. Hromkovič, J. Kneis, and J. Kupke. On the parameterized approximability of TSP with deadlines, 2006. To appear in *Theory of Computing Systems.*

[15] H. L. Bodlaender. Unpublished result.

[16] H. L. Bodlaender. A tourist guide through treewidth. *Acta Cybernet.*, 11(1-2):1–21, 1993.

[17] H. L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25(6):1305–1317, 1996.

[18] H. L. Bodlaender, M. R. Fellows, and M. T. Hallett. Beyond NP-completeness for problems of bounded width (extended abstract): hardness for the W hierarchy. In *STOC '94: Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 449–458, New York, NY, USA, 1994. ACM Press.

[19] H. L. Bodlaender, J. R. Gilbert, H. Hafsteinsson, and T. Kloks. Approximating treewidth, pathwidth, frontsize, and shortest elimination tree. *J. Algorithms*, 18(2):238–255, 1995.

[20] V. Bouchitté, D. Kratsch, H. Müller, and I. Todinca. On treewidth approximations. *Discrete Appl. Math.*, 136(2-3):183–196, 2004. The 1st Cologne-Twente Workshop on Graphs and Combinatorial Optimization (CTW 2001).

[21] S. Cabello, P. Giannopoulos, and C. Knauer. On the parameterized complexity of d-dimensional point set pattern matching. In *Proceedings of the International Workshop on Parameterized and Exact Computation (IWPEC 2006)*, volume 4169 of *Lecture Notes in Computer Science*, pages 175 –183. 2006.

[22] L. Cai. Parameterized complexity of vertex colouring. *Discrete Appl. Math.*, 127:415–429, 2003.

[23] L. Cai, M. Fellows, D. Juedes, and F. Rosamond. The complexity of polynomial-time approximation, 2006. To appear in *Theory of Computing Systems.*

[24] L. Cai and X. Huang. Fixed-parameter approximation: conceptual framework and approximability results. In *Proceedings of the International Workshop on Parameterized and Exact Computation (IWPEC 2006)*, volume 4169 of *Lecture Notes in Computer Science*, pages 96–108. 2006.

[25] M. Cesati and L. Trevisan. On the efficiency of polynomial time approximation schemes. *Inform. Process. Lett.*, 64(4):165–171, 1997.

[26] C. Chekuri and S. Khanna. A PTAS for the multiple knapsack problem. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms (San Francisco, CA, 2000)*, pages 213–222, New York, 2000. ACM.

[27] J. Chen, H. Fernau, I. A. Kanj, and G. Xia. Parametric duality and kernelization: lower bounds and upper bounds on kernel size. In *STACS 2005*, volume 3404 of *Lecture Notes in Comput. Sci.*, pages 269–280. Springer, Berlin, 2005.

[28] J. Chen, X. Huang, I. A. Kanj, and G. Xia. Linear FPT reductions and computational lower bounds. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, pages 212–221, New York, 2004. ACM.

[29] J. Chen, I. A. Kanj, and W. Jia. Vertex cover: further observations and further improvements. *J. Algorithms*, 41(2):280–301, 2001.

[30] Y. Chen, M. Grohe, and M. Grüber. On parameterized approximability. In *Proceedings of the International Workshop on Parameterized and Exact Computation (IWPEC 2006)*, volume 4169 of *Lecture Notes in Computer Science*, pages 109–120. 2006.

[31] N. Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical Report 388, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, 1976.

[32] B. Courcelle, J. Engelfriet, and G. Rozenberg. Context-free handle-rewriting hypergraph grammars. In *Graph grammars and their application to computer science (Bremen, 1990)*, volume 532 of *Lecture Notes in Comput. Sci.*, pages 253–268. Springer, Berlin, 1991.

[33] B. Courcelle and S.-I. Oum. Vertex-minors, monadic second-order logic, and a conjecture by Seese. *J. Combin. Theory Ser. B*, 97(1):91–126, 2007.

[34] F. Dehne, M. Fellows, M. A. Langston, F. Rosamond, and K. Stevens. An $O(2^{O(k)}n^3)$ FPT algorithm for the undirected feedback vertex set problem. In *Computing and combinatorics*, volume 3595 of *Lecture Notes in Comput. Sci.*, pages 859–869. Springer, Berlin, 2005.

[35] E. D. Demaine, F. V. Fomin, M. Hajiaghayi, and D. M. Thilikos. Subexponential parameterized algorithms on graphs of bounded-genus and H-minor-free graphs. *Journal of the ACM*, 52(6):866–893, 2005.

[36] E. D. Demaine and M. Hajiaghayi. Equivalence of local treewidth and linear local treewidth and its algorithmic applications. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2004)*, pages 833–842, New Orleans, Louisiana, January 11–13 2004.

[37] E. D. Demaine and M. Hajiaghayi. Fast algorithms for hard graph problems: Bidimensionality, minors, and local treewidth. In *Proceedings of the 12th International Symposium on Graph Drawing (GD 2004)*, volume 3383 of *Lecture Notes in Computer Science*, pages 517–533, Harlem, New York, September 29–October 2 2004.

[38] E. D. Demaine, M. Hajiaghayi, and K. Kawarabayashi. Algorithmic graph minor theory: Decomposition, approximation, and coloring. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005)*, pages 637–646, Pittsburgh, PA, October 23–25 2005.

[39] X. Deng, G. Li, Z. Li, B. Ma, and L. Wang. A PTAS for distinguishing (sub)string selection. In *ICALP '02: Proceedings of the 29th International Colloquium on Automata, Languages and Programming*, volume 2380 of *Lecture Notes in Comput. Sci.*, pages 740–751, Berlin, 2002. Springer.

[40] J. Díaz, J. Petit, and M. Serna. A survey of graph layout problems. *ACM Comput. Surv.*, 34(3):313–356, 2002.

[41] R. Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, third edition, 2005.

[42] R. Downey, V. Estivill-Castro, M. Fellows, E. Prieto, and F. Rosamund. Cutting up is hard to do. In J. Harland, editor, *Electronic Notes in Theoretical Computer Science*, volume 78. Elsevier, 2003.

[43] R. Downey, M. Fellows, and C. McCartin. Parameterized approximation algorithms. In *Proceedings of the International Workshop on Parameterized and Exact Computation (IWPEC 2006)*, volume 4169 of *Lecture Notes in Computer Science*, pages 121–129. 2006.

[44] R. G. Downey. Parameterized complexity for the skeptic. In *Proceedings of the 18th IEEE Annual Conference on Computational Complexity*, pages 147–169, 2003.

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

[46] T. Erlebach, K. Jansen, and E. Seidel. Polynomial-time approximation schemes for geometric intersection graphs. *SIAM J. Comput.*, 34(6):1302–1323, 2005.

[47] G. Even, J. Naor, B. Schieber, and M. Sudan. Approximating minimum feedback sets and multicuts in directed graphs. *Algorithmica*, 20(2):151–174, 1998.

[48] U. Feige, M. T. Hajiaghayi, and J. R. Lee. Improved approximation algorithms for minimum-weight vertex separators. In *STOC'05: Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 563–572, New York, 2005. ACM.

[49] U. Feige and J. Kilian. Zero knowledge and the chromatic number. *J. Comput. System Sci.*, 57(2):187–199, 1998.

[50] M. R. Fellows. Unpublished result.

[51] M. R. Fellows, J. Gramm, and R. Niedermeier. On the parameterized intractability of motif search problems. *Combinatorica*, 26(2):141–167, 2006.

[52] M. R. Fellows, F. A. Rosamond, U. Rotics, and S. Szeider. Clique-width minimization is NP-hard.

[53] J. Flum and M. Grohe. The parameterized complexity of counting problems. *SIAM J. Comput.*, 33(4):892–922, 2004.

[54] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer-Verlag, Berlin, 2006.

[55] K. Giaro and M. Kubale. Edge-chromatic sum of trees and bounded cyclicity graphs. *Inform. Process. Lett.*, 75(1-2):65–69, 2000.

[56] T. F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoret. Comput. Sci.*, 38(2-3):293–306, 1985.

[57] J. Gramm, J. Guo, and R. Niedermeier. On exact and approximation algorithms for distinguishing substring selection. In *Fundamentals of computation theory*, volume 2751 of *Lecture Notes in Comput. Sci.*, pages 195–209. Springer, Berlin, 2003.

[58] J. Gramm, R. Niedermeier, and P. Rossmanith. Exact solutions for closest string and related problems. In *ISAAC '01: Proceedings of the 12th International Symposium on Algorithms and Computation*, volume 2223 of *Lecture Notes in Comput. Sci.*, pages 441–453, Berlin, 2001. Springer.

[59] J. Gramm, R. Niedermeier, and P. Rossmanith. Fixed-parameter algorithms for closest string and related problems. *Algorithmica*, 37(1):25–42, 2003.

[60] M. Grohe. Local tree-width, excluded minors, and approximation algorithms. *Combinatorica*, 23(4):613–632, 2003.

[61] M. Grohe and M. Grüber. Parameterized approximability of the disjoint cycle problem, 2007. To appear in ICALP '07.

[62] J. Guo, J. Gramm, F. Hüffner, R. Niedermeier, and S. Wernicke. Improved fixed-parameter algorithms for two feedback set problems. In *Proceedings of the 9th Workshop on Algorithms and Data Structures (WADS'05)*, volume 3608 of *LNCS*, pages 158–168. Springer-Verlag, Aug 2005.

[63] J. Guo, J. Gramm, F. Hüffner, R. Niedermeier, and S. Wernicke. Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization. *J. Comput. System Sci.*, 72(8):1386–1396, 2006.

[64] J. Guo, F. Hüffner, and R. Niedermeier. A structural view on parameterizing problems: distance from triviality. In *Proceedings of the International Workshop on Parameterized and Exact Computation (IWPEC 2004)*, volume 3162 of *Lecture Notes in Comput. Sci.*, pages 162–173, Berlin, 2004. Springer.

[65] J. Guo, R. Niedermeier, and S. Wernicke. Parameterized complexity of generalized vertex cover problems. In *Proceedings of the 9th Workshop on Algorithms and Data Structures (WADS'05)*, volume 3608 of *LNCS*, pages 36–48. Springer-Verlag, Aug 2005.

[66] S. Har-Peled and S. Mazumdar. On coresets for k-means and k-median clustering. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, pages 291–300, New York, 2004. ACM.

[67] J. Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001.

[68] P. Hliněný. A parametrized algorithm for matroid branch-width. *SIAM J. Comput.*, 35(2):259–277, 2005.

[69] D. S. Hochbaum and W. Maass. Approximation schemes for covering and packing problems in image processing and VLSI. *J. ACM*, 32(1):130–136, 1985.

[70] D. S. Hochbaum and A. Pathria. Analysis of the greedy approach in covering problems. *Naval Research Quarterly*, (45):615–627, 1998.

[71] H. B. Hunt, III, M. V. Marathe, V. Radhakrishnan, S. S. Ravi, D. J. Rosenkrantz, and R. E. Stearns. NC-approximation schemes for NP- and PSPACE-hard problems for geometric graphs. *J. Algorithms*, 26(2):238–274, 1998.

[72] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *J. Comput. System Sci.*, 63(4):512–530, 2001. Special issue on FOCS 98 (Palo Alto, CA).

[73] K. Jansen. The optimum cost chromatic partition problem. In *Algorithms and complexity (Rome, 1997)*, volume 1203 of *Lecture Notes in Comput. Sci.*, pages 25–36. Springer, Berlin, 1997.

[74] S. Khanna and R. Motwani. Towards a syntactic characterization of PTAS. In *Proceedings of the Twenty-eighth Annual ACM Symposium on the Theory of Computing (Philadelphia, PA, 1996)*, pages 329–337, New York, 1996. ACM.

[75] S. Khanna, M. Sudan, L. Trevisan, and D. P. Williamson. The approximability of constraint satisfaction problems. *SIAM J. Comput.*, 30(6):1863–1920, 2001.

[76] S. Khot and O. Regev. Vertex cover might be hard to approximate to within $2-\varepsilon$. In *18th Annual IEEE Conference on Computational Complexity (CCC'03)*, 2003.

[77] M. Li, B. Ma, and L. Wang. On the closest string and substring problems. *J. ACM*, 49(2):157–171, 2002.

[78] D. Lichtenstein. Planar formulae and their uses. *SIAM J. Comput.*, 11(2):329–343, 1982.

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

[80] E. Malesińska. *Graph-Theoretical Models for Frequency Assignment Problems*. PhD thesis, Technical University of Berlin, 1997.

[81] D. Marx. Complexity results for minimum sum edge coloring, 2004. Manuscript.

[82] D. Marx. The closest substring problem with small distances. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*, pages 63–72, 2005.

[83] D. Marx. Efficient approximation schemes for geometric problems? In *Proceedings of 13th Annual European Symposium on Algorithms (ESA 2005)*, pages 448–459, 2005.

[84] D. Marx. Minimum sum multicoloring on the edges of planar graphs and partial k-trees. In *2nd International Workshop on Approximation and Online Algorithms (WAOA 2004)*, volume 3351 of *Lecture Notes in Computer Science*, pages 9–22. Springer, Berlin, 2005.

[85] D. Marx. Chordal deletion is fixed-parameter tractable. In *Graph-theoretic concepts in computer science (WG 2006)*, volume 4271 of *Lecture Notes in Comput. Sci.*, pages 37–48. 2006.

[86] D. Marx. Parameterized coloring problems on chordal graphs. *Theoret. Comput. Sci.*, 351(3):407–424, 2006.

[87] D. Marx. Parameterized complexity of independence and domination on geometric graphs. In *Proceedings of the International Workshop on Parameterized and Exact Computation (IWPEC 2006)*, volume 4169 of *Lecture Notes in Computer Science*, pages 154 –165. 2006.

[88] D. Marx and I. Schlotter. Obtaining a planar graph by vertex deletion, 2007. Accepted for WG 2007.

[89] M. Müller. Randomized approximations of parameterized counting problems. In *Proceedings of the International Workshop on Parameterized and Exact Computation (IWPEC 2006)*, volume 4169 of *Lecture Notes in Computer Science*, pages 50 –59. 2006.

[90] B. Monien. How to find long paths efficiently. In *Analysis and design of algorithms for combinatorial problems (Udine, 1982)*, volume 109 of *North-Holland Math. Stud.*, pages 239–254. North-Holland, Amsterdam, 1985.

[91] G. L. Nemhauser and L. E. Trotter, Jr. Vertex packings: structural properties and algorithms. *Math. Programming*, 8:232–248, 1975.

[92] S. Nicoloso, M. Sarrafzadeh, and X. Song. On the sum coloring problem on interval graphs. *Algorithmica*, 23(2):109–126, 1999.

[93] T. Nieberg and J. Hurink. A PTAS for the minimum dominating set problem in unit disk graphs. In *3rd International Workshop on Approximation and Online Algorithms (WAOA 2005)*, volume 3879 of *Lecture Notes in Computer Science*, pages 296–306. Springer, Berlin, 2006.

[94] T. Nieberg, J. Hurink, and W. Kern. New PTAS for the independent set problem in unit disk graphs. Memorandum No. 1688, Department of Applied Mathematics, Universiteit Twente, September 2003.

[95] S.-I. Oum. Approximating rank-width and clique-width quickly. In *Proceedings of the 31st International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 49–58, 2005.

[96] S.-I. Oum and P. Seymour. Approximating clique-width and branch-width. *J. Combin. Theory Ser. B*, 96(4):514–528, 2006.

[97] C. H. Papadimitriou and M. Yannakakis. The traveling salesman problem with distances one and two. *Math. Oper. Res.*, 18(1):1–11, 1993.

[98] E. Petrank. The hardness of approximation: gap location. *Comput. Complexity*, 4(2):133–157, 1994.

[99] J. Plehn and B. Voigt. Finding minimally weighted subgraphs. In *Graph-theoretic concepts in computer science (Berlin, 1990)*, volume 484 of *Lecture Notes in Comput. Sci.*, pages 18–29. Springer, Berlin, 1991.

[100] B. Reed, N. Robertson, P. Seymour, and R. Thomas. Packing directed circuits. *Combinatorica*, 16(4):535–554, 1996.

[101] B. Reed, K. Smith, and A. Vetta. Finding odd cycle transversals. *Operations Research Letters*, 32(4):299–301, 2004.

[102] B. A. Reed. Finding approximate separators and computing tree width quickly. In *STOC '92: Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 221–228, New York, NY, USA, 1992. ACM Press.

[103] B. A. Reed. Tree width and tangles: a new connectivity measure and some applications. In *Surveys in combinatorics, 1997 (London)*, volume 241

of *London Math. Soc. Lecture Note Ser.*, pages 87–162. Cambridge Univ. Press, Cambridge, 1997.

[104] N. Robertson, D. P. Sanders, P. Seymour, and R. Thomas. Efficiently four-coloring planar graphs. In *Proceedings of the Twenty-eighth Annual ACM Symposium on the Theory of Computing (Philadelphia, PA, 1996)*, pages 571–575, New York, 1996. ACM.

[105] N. Robertson and P. D. Seymour. Graph minors. XIII. The disjoint paths problem. *J. Combin. Theory Ser. B*, 63(1):65–110, 1995.

[106] D. Rolf. Improved bound for the PPSZ/Schöning-algorithm for 3-SAT. Technical Report TR05-159, Electronic Colloq. on Computational Complexity, 2005.

[107] M. R. Salavatipour. On sum coloring of graphs. *Discrete Appl. Math.*, 127(3):477–488, 2003.

[108] P. D. Seymour. Packing directed circuits fractionally. *Combinatorica*, 15(2):281–288, 1995.

[109] R. Shamir and D. Tsur. The maximum subforest problem: approximation and exact algorithms (extended abstract). In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (San Francisco, CA, 1998)*, pages 394–399, New York, 1998. ACM.

[110] A. Slivkins. Parameterized tractability of edge-disjoint paths on directed acyclic graphs. In *Algorithms – ESA 2003, 11th Annual European Symposium*, volume 2832 of *Lecture Notes in Comput. Sci.*, pages 482–493. Springer, Berlin, 2003.

[111] L. Stockmeyer. Planar 3-colorability is NP-complete. *SIGACT News*, 5:19–25, 1973.

[112] L. Sunil Chandran and F. Grandoni. Refined memorization for vertex cover. *Inform. Process. Lett.*, 93(3):125–131, 2005.

[113] D. M. Thilikos, M. J. Serna, and H. L. Bodlaender. Constructive linear time algorithms for small cutwidth and carving-width. In *Algorithms and computation (Taipei, 2000)*, volume 1969 of *Lecture Notes in Comput. Sci.*, pages 192–203. Springer, Berlin, 2000.

[114] L. Trevisan. When Hamming meets Euclid: the approximability of geometric TSP and Steiner tree. *SIAM J. Comput.*, 30(2):475–485, 2000.

[115] L. G. Valiant. The complexity of computing the permanent. *Theoret. Comput. Sci.*, 8(2):189–201, 1979.

[116] D. W. Wang and Y.-S. Kuo. A study on two geometric location problems. *Inform. Process. Lett.*, 28(6):281–286, 1988.