LCNS, Vol 3153, pp. 84–103, Springer 2004

Ubiquitous Parameterization—Invitation to Fixed-Parameter Algorithms

Rolf Niedermeier*

Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, Sand 13, D-72076 Tübingen, Germany
niedermr@informatik.uni-tuebingen.de

Abstract. Problem parameters are ubiquitous. In every area of computer science, we find all kinds of "special aspects" to the problems encountered. Hence, the study of parameterized complexity for computationally hard problems is proving highly fruitful. The purpose of this article is to stir the reader's interest in this field by providing a gentle introduction to the rewarding field of fixed-parameter algorithms.
Keywords. NP-hardness, parameterized complexity, fixed-parameter algorithms, parameterization.

1 Introduction

Computationally hard problems are ubiquitous. The systematic study of computational (in)tractability lies at the very heart of computer science. Garey and Johnson's [44] monograph on computational intractability surely is a landmark achievement in this direction, providing an in-depth treatment of the theory of NP-completeness. With the theory of NP-completeness and the like at hand, we can prove meaningful statements about the computational complexity of problems. But what happens after we have succeeded in proving that a problem is NP-hard (that is, "intractable") and, nevertheless, the problem has to be solved in practice? In other words, how does one cope with computational intractability? Several methods to deal with this problem have been developed [60]: approximation algorithms [12,57,86], average-case analysis [59], randomized algorithms [72], and heuristic methods [71,78]. All of them have their drawbacks, such as, respectively, the difficulty of approximation, lack of mathematical tools and results, limited power of the method itself, or the lack of any provable performance guarantees at all.

NP-hardness, in accordance with the state of the art of computational complexity theory, means that we have to take into account algorithms with exponential running times to solve the corresponding problems exactly.[1] Clearly,

* Supported by the Deutsche Forschungsgemeinschaft (DFG), Emmy Noether research group PIAF (fixed-parameter algorithms), NI 369/4.

[1] Clearly, whenever we obtain a fast enough optimal solution there is no need to search for approximate solutions. The point is to better understand to what extent we can afford to do so, that is, how costly the search for optimal solutions in various cases (parameterizations) is.

however, exponential growth quickly becomes prohibitive when running algorithms in practice. Fixed-parameter algorithmics provides guidance on the feasibility of the "exact algorithm approach" to hard problems by means of a refined, two-dimensional complexity analysis. The fundamental idea is to strive for better insight into a problem's complexity by exploring (various) problem-specific parameters and to find out how they influence the problem's computational complexity. Ideally, we aim for statements such as "if some parameter k is small in problem X, then X can be solved efficiently." For instance, in case of the NP-complete graph problem Vertex Cover we know that if the solution set we are searching for is "small", then this set can be found efficiently whatever the graph looks like and however big it is—the exponential factor in the running time amounts to approximately 1.29^k [22,76], where the parameter k is the size of the solution set sought. Thus, if we search for little (size of solution), then we have to pay little (running time). Unfortunately, an analogous behavior is highly unlikely for the NP-complete graph problem Dominating Set, as the machinery of parameterized complexity theory makes clear [28].

Parameterized complexity and exact algorithms are the subject of several surveys [10,27,34,35,36,52,87], indicating the increasing importance of this field.[2] The purpose of this incomplete and personally biased expository paper is to stir the reader's interest in the field with an emphasis on algorithmic questions and methods. A distinguishing feature of this survey compared to previous ones will be its focus on the "art of parameterizing problems." Altogether, it will be shown that fixed-parameter complexity is not at all a "small area." In fact, we believe that it is destined to become a driving force in modern, algorithmically-oriented research on computationally hard problems.

2 Satisfiability or Why to Parameterize Problems?

The Satisfiability problem for boolean formulas in conjunctive normal form may be considered the "drosophila of computational complexity theory." This fundamental NP-complete problem has been the subject of research on exact algorithms for decades [23] and it continues to play a central role in algorithmic research—the annual "SAT" conference is devoted to theory and applications of satisfiability testing. We define the problem as follows.

> **Input:** A boolean formula F in conjunctive normal form.
> **Task:** Determine whether or not there exists a truth assignment for the variables in F such that F evaluates to true.

Numerous applications such as VLSI design and model checking make Satisfiability of real practical interest. Often, however, in concrete applications the

[2] Also see the December 2003 special issue of *Journal of Computer and System Sciences*, volume 67(4), on parameterized complexity and computation, edited by Jianer Chen and Michael R. Fellows. In addition, there is a newly launched conference *IWPEC 2004 (International Workshop an Parameterized and Exact Computation)*, held in Bergen, Norway, September 2004 (proceedings appear in Springer's *LNCS* series).

corresponding problem instances turn out to be much easier than one might expect given the fact of Satisfiability's NP-completeness. For instance, large Satisfiability instances arising in the validation of automotive product configuration data turn out to be efficiently solvable [65,81]. Hence, the question arises of whether we can learn more about the complexity of Satisfiability by means of studying various problem parameters. In particular, we are less interested in empirical results but more so in provable performance bounds related to various parameterizations of Satisfiability. In this way, we hope to obtain a better understanding of problem properties that might be of algorithmic use. In the rest of this section, we discuss several parameterizations of Satisfiability.

Formally, an input of Satisfiability is a conjunction of m clauses where each clause consists of a disjunction of negated or non-negated boolean variables (so-called literals).

Parameter "clause size." The maximum number k of literals in any of the given clauses is a very natural formula parameter. For $k = 3$, however, the problem remains NP-complete whereas it is polynomial-time solvable for $k = 2$ [44]. Thus, for coping with the problem's intractability, this parameterization seems of little help in cases where $k \geq 3$.

Parameter "number of variables." The number n of different variables occurring in a formula significantly influences the complexity. Since there are 2^n different truth assignments, in essentially[3] this number of steps Satisfiability can be solved. When restricting the maximum clause size by some k, for instance, $k = 3$, better bounds are known. The current best upper bound for 3-Satisfiability is 1.324^n [62].

Parameter "number of clauses." If the number of clauses in a formula can be bounded by k, then Satisfiability can be solved in 1.239^k steps [56].

Parameter "formula length." If the total length of the formula is bounded by k, then Satisfiability can be solved in 1.074^k steps [56].

The above parameterizations will not suffice to explain all the cases of good behavior of Satisfiability in many practical situations. There are application scenarios where none of them would lead to an efficient algorithm. Hence, the following way of parameterizing Satisfiability seems prospective.

Parameters exploiting "formula structure." Szeider [84] surveys several recent exact algorithms with exponential bounds depending on certain *structural* formula parameters. He discusses parameters based on structural graph decompositions (where "variable interaction graphs" etc. are considered). Similar, more empirically oriented investigations have been started in [80].

Finally, from the parameterized complexity theory [28] point of view the following parameterization is of particular relevance.

Parameter "weight of assignment." Here it is asked whether a formula has a satisfying assignment with exactly (!) k variables being set true. Although

[3] We neglect polynomial-time factors in the running time throughout the paper whenever they play a minor role in our considerations.

the parameterization with this "weight parameter" seems rather artificial, it plays a major role in characterizing parameterized intractability [28]. It serves as a building block in the parameterized hardness program similar to the role Satisfiability plays in classical NP-completeness theory.

To summarize, different ways of parameterizing Satisfiability lead to a better understanding of the problem's inherent complexity facets. In particular, it might be hoped that structural parameterizations as discussed and surveyed by Szeider [84] will yield new insights on tractable cases of Satisfiability. But there surely is *no "best parameterization."* As a rule, the nature of the complexity of hard computational problems will probably almost always require such a multi-perspective view in order to win better insight into practically relevant, efficiently solvable special cases. To cope with intractability in a mathematically sound way, it seems that we have to pay the price by shifting our view from considering the input just through one pair of glasses (mostly these glasses are called "complexity measurement relative to input size") to a multitude of pairs of glasses, each of them with a different focus (that is, parameter). Note, however, that the above indicated, and in the rest of this paper further exhibited, methodology of parameterizing problems still relies on worst-case analysis. Parameterized complexity theory, as chiefly developed by Downey and Fellows [28], puts the related issues into a mathematically sound, formal framework.

3 Parameterized Complexity Theory in a Nutshell

Parameterized complexity theory [28] offers a two-dimensional framework for studying the computational complexity of problems. A *parameterized language* (problem) L is a subset $L \subseteq \Sigma^* \times \Sigma^*$ for some finite alphabet Σ. For $(x, k) \in L$, by convention, the second component denotes the *parameter*. The two dimensions of parameterized complexity analysis are constituted by the input size n, i.e., $n := |(x, k)|$ and the parameter value k (usually a nonnegative integer). A parameterized language is *fixed-parameter tractable* if it can be determined in $f(k) \cdot n^{O(1)}$ time whether $(x, k) \in L$, where f is a computable function only depending on k. The corresponding complexity class is called FPT.

A core tool in the development of fixed-parameter algorithms is polynomial-time preprocessing by *data reduction rules*, often yielding a *reduction to a problem kernel*. Here, the goal is, given any problem instance I with parameter k, to transform it into a new instance I' with parameter k' such that the size of I' is bounded by some function only depending on k', $k' \leq k$, and (I, k) has a solution iff (I', k') has a solution—see [1] for a recent thorough investigation of reduction to a problem kernel (also called *kernelization*) for the Vertex Cover problem (the problem parameter there being the size of the vertex cover set).

Downey and Fellows developed a completeness program as a formal framework to show fixed-parameter intractability [28]. Let $L, L' \subseteq \Sigma^* \times \mathbb{N}$ be two

parameterized languages.[4] We say that L *reduces to* L' *by a standard parameterized m-reduction* if there are functions $k \mapsto k'$ and $k \mapsto k''$ from $\mathbf{N}$ to $\mathbf{N}$ and a function $(x,k) \mapsto x'$ from $\Sigma^* \times \mathbf{N}$ to Σ^* such that

1. $(x,k) \mapsto x'$ is computable in time $k''|x|^c$ for some constant c and
2. $(x,k) \in L$ iff $(x',k') \in L'$.

Notably, most reductions from classical complexity turn out *not* to be parameterized ones [28]. The basic complexity class for fixed-parameter intractability, W[1], can be defined as the class of parameterized languages that are equivalent to the Short Turing Machine Acceptance problem (also known as the k-Step Halting problem). Here, we want to determine, for an input consisting of a nondeterministic Turing machine M (with unbounded nondeterminism and alphabet size), and a string x, whether or not M has a computation path accepting x in at most k steps. This can trivially be solved in $O(n^{k+1})$ time (where n denotes a bound on the total input size) and we would be surprised if this can be significantly improved. Therefore, this is the parameterized analogue of the Turing Machine Acceptance problem that is the basic generic NP-complete problem in classical complexity theory, and the conjecture that FPT $\neq$ W[1] is very much analogous to the conjecture that P $\neq$ NP. Other problems that are W[1]-*hard* (and also W[1]-*complete*) include the graph problems Clique and Independent Set, where the parameter is the size of the relevant vertex set [28]. Also, for constant maximum clause size, the aforementioned parameterization of Satisfiability by the weight of an assignment (see Sect. 2) gives a W[1]-complete problem. W[1]-hardness gives a concrete indication that a parameterized problem with parameter k is unlikely to allow for a solving algorithm with $f(k) \cdot n^{O(1)}$ running time, that is, restricting the combinatorial explosion to the parameter seems illusory.

Since the parameter k represents some aspect(s) of the input or the solution, there usually are many meaningful ways to *parameterize* a problem. An important issue herein is whether a hard problem is fixed-parameter tractable with respect to a chosen parameter or not, and, in case of fixed-parameter tractability, how small the usually exponential growth of the function f can be kept. Hence, investigating different parameterizations gives insight into what causes the computational (in)tractability of a problem and in which qualitative and quantitative senses this happens and how it can be (potentially) coped with.

4 Fixed-Parameter Tractability—Three Case Studies

How can one design fixed-parameter algorithms? Three core concepts in this context are

– preprocessing by data reduction rules and problem kernels,

[4] Generally, the second component (representing the parameter) is drawn from Σ^*; for most cases (particularly, within this paper), assuming the parameter to be a nonnegative integer (or a tuple of nonnegative integers) is sufficient.

– bounded search trees, and
– dynamic programming.

Clearly, there would be more to say about further techniques for developing fixed-parameter tractability results but the above three issues definitely are among the most important ones. Other algorithmic strategies in the derivation of exact algorithms are discussed in the surveys [35,36,87]. To illustrate the above three techniques and, in this way, some central topics in fixed-parameter complexity analysis, we discuss three important, easy to grasp, graph problems.

4.1 Case Study Multicut in Trees

We start with an optimization problem on graphs that already is NP-complete when restricted to trees [45]—(the unweighted version of) Multicut in Trees.

> **Input:** An undirected tree $T = (V, E)$, $n := |V|$, and a collection H of m pairs of nodes in V, $H = \{(u_i, v_i) \mid u_i, v_i \in V, 1 \leq i \leq m\}$.
> **Task:** Find a minimum size subset E' of E such that the removal of the edges in E' separates each pair of nodes in H.

By trying all possibilities (and using $|E| = n - 1$) we can solve the problem in 2^{n-1} steps. But can we do better if there exists a small solution set E'? Let us define the parameter $k := |E'|$ and study how k influences the problem complexity. Note that clearly $k < n$ but in some problem cases $k \ll n$ seems to be a reasonable assumption. Here, the following simple algorithm works. Assume that the given tree is (arbitrarily) rooted. Consider a pair of nodes $(u, v) \in H$ such that the uniquely determined path p between u and v has maximum distance from the root r. Call w the node on p that is closest to r. Then, one may easily see (using the tree structure) that there is an optimal solution that contains at least one of the two path edges connected to w. But this implies that we can build a search tree of depth bounded by k: Simply search for w as specified above and then branch into the two cases (at least one of them has to yield an optimal solution) of taking one of the two neighboring path edges of w. Iterating this process, we obtain a search tree of size bounded by 2^k and,[5] thus, Multicut in Trees is fixed-parameter tractable with respect to parameter k. There are also simple data reduction rules for Multicut in Trees, but to prove a problem kernel as discussed in the following for Vertex Cover is quite hard for Multicut in Trees [55]. Hence, we prefer to switch to the simpler problem Vertex Cover in graphs.[6]

[5] In a straightforward way, by taking both edges, one also obtains a polynomial-time factor-2 approximation algorithm. Nothing better is known [45].

[6] Vertex Cover can be directly reduced to Multicut in Trees restricted to star graphs (that is, trees of height one) [45]. Thus, approximation and exact algorithms for Multicut in Trees easily transfer to Vertex Cover.

4.2 Case Study Vertex Cover

Vertex Cover has been one of the most important problems in fixed-parameter tractability studies:

> **Input:** An undirected graph $G = (V, E)$.
> **Task:** Find a minimum size subset V' of V such that each edge in E has at least one of its endpoints in V'.

By way of contrast to Multicut, Vertex Cover is trivial when restricted to trees. Assuming an arbitrarily rooted tree, in a bottom-up fashion from the leaves to the root one can determine an optimal solution in linear time by a straightforward dynamic programming. For general graphs, defining the parameter $k := |V'|$, one easily obtains a size 2^k search tree—one of the two endpoints of an edge *has* to be part of a vertex cover. Opposite to Multicut in Trees, however, the search tree size can easily be shrunk below 2^k. We describe a very simple possibility based on *"degree-branching"* here:

1. If there is a vertex of degree one, then put its neighbor into V'.
2. If there is a vertex v of degree two, then either put v into V' together with all neighbors of its neighbors or put both neighbors of v into V'.[7]
3. If there is a vertex v of degree three, then either put v or all its neighbors into V'.

It is not hard to see that the above search strategy always leads to an optimal solution. Moreover, in steps 2 and 3 the search branches into two cases each time. In step 2, each branch puts at least two vertices into V' and in step 3 the first branch puts one vertex into V' and the second branch puts at least three vertices into V'. This branching process is recursively repeated until an optimal solution is found. If the solution has size k, the corresponding search tree has size bounded by 1.47^k (which can be determined by solving the corresponding recurrences using standard mathematical tools; see, for example, [66]). Much more refined branching strategies (significantly increasing the number of case distinctions) led to search tree sizes smaller than 1.29^k [22,76].

Other than search tree strategies which are inherently exponential, *data reduction by preprocessing* is a polynomial-time methodology to shrink the input size as much as possible. Ideally, in some rare cases this might even solve the complete problem. But, if no solution is obtained solely by preprocessing (as the name already indicates), how do we measure the quality of the data reduction process besides purely empirical statements? To this end, parameterized complexity offers the concept of *problem kernels*. A simple preprocessing strategy for Vertex Cover is based on the following observation. Since we are looking for a vertex set V' of size at most k, for a vertex v of degree greater than k we have no choice rather than putting v into V'. Otherwise, no V' with $|V'| \leq k$ can exist. In this way, we can get rid of all high-degree vertices. The remaining instance consists of vertices of maximum degree k. But then one vertex can

[7] Chen et al. [22] present a "folding trick" that makes it possible to avoid branching for degree-two vertices.

cover at most k edges and, thus, there only can exist a vertex cover set of size at most k if the remaining instance has at most k^2 edges. In summary, either we can decide in this way that there is no solution or the solution has to be found in a graph of size $O(k^2)$—the problem kernel. The decisive point is that the size of the problem kernel exclusively depends on the size of the parameter k.

There is one drawback in the just described preprocessing—it assumes that we know the value of k in advance. By way of contrast, the much more sophisticated data reduction due to Nemhauser and Trotter [73,64] does not require the value of k in advance and, furthermore, it provides a problem kernel graph of at most $2k$ vertices (see [22] for its application to a fixed-parameter algorithm for Vertex Cover). Hence, after this polynomial-time preprocessing the exponential-time (search tree) algorithm can concentrate on a size-$2k$ graph. Further data reduction techniques such as "crown reduction rules" [63] (which can be interpreted as a generalization of the just described "degree-one rule") and their benefits in practice are discussed in a new study [1]. Observe that it is worth keeping an eye on the fact whether or not the problem kernelization rules make explicit use of the parameter value k. Clearly, "*parameter-independence*" of data reduction rules is preferable. So far, this distinction seemingly has been neglected in formal fixed-parameter complexity studies. Finally, we briefly remark that it is (provably) beneficial to employ kernelization techniques over and over again, for example, by interleaving search tree branching with data reduction [75].

4.3 Case Study Dominating Set

Dominating Set is a "parameterized antagonist" of Vertex Cover. Both are minimization problems but Dominating Set appears to be intractable with respect to the parameterization "solution size k." This also reflects in approximation results. Whereas Vertex Cover has a trivial factor-2 polynomial-time approximation algorithm, for Dominating Set we only have a factor-$\Theta(\log n)$ approximation [33]. Dominating Set is defined as follows.

> **Input:** An undirected graph $G = (V, E)$.
> **Task:** Find a minimum size subset V' of V such that for every vertex $v \in V$ either $v \in V'$ or there is at least one vertex in V' that is a neighbor of v, or both.

With respect to parameter $k := |V'|$, Dominating Set is W[2]-complete [28] (thus, W[1]-hard). For instance, the bounded search tree approach based on degree-branching (cf. Sect. 4.2) appears to fail: In analogy to Vertex Cover, one might consider the neighborhood of a vertex v. To dominate v, either v or *one*[8] of its neighboring vertices has to be in the desired solution set. The point is that this yields a branching into a potentially *unbounded* number of cases. For bounded degree graphs with maximum vertex degree d this yields a search tree of size upper-bounded by d^k. However, there seems to be no way to overcome

[8] Compare this with Vertex Cover where we use that either v or *all* of its neighbors have to be in an optimal solution.

the dependence on d and to restrict the combinatorial explosion exclusively to k. Hence, concerning the parameterization "solution size," it is reasonable to study special graph classes.

Besides bounded-degree graphs as indicated above, fixed-parameter tractability results are also known for planar graphs[9] and slight generalizations thereof. For instance, here also the bounded search tree works. But things are more complicated than might be expected. Although Euler's formula tells us that there always exists a degree at most five vertex in a planar graph, this does not yield a size 6^k search tree by the degree-branching method. The difficulty comes from the fact that we have to distinguish between three vertex states in the course of the search tree processing: a vertex can either be "dominating" (that is, it is in V') or "dominated" (that is, at least one of its neighbors is in V') or "non-dominated." Then, the difficulty is that we cannot delete dominated vertices because an optimal solution may require that such a vertex is still part of it. An already dominated vertex, however, is not a suitable candidate for the degree-branching technique because it is *not* true that either itself or one of its neighbors *has* to be in the dominating set. Thus, using the Euler formula one has to make sure that the selected vertex is a non-dominated one. With significant technical expenditure one can show that there always exists a non-dominated degree-seven vertex, directly giving a size 8^k search tree [6]. Notably, comparing the search tree algorithms for Vertex Cover and Dominating Set (restricted to planar graphs), in the Vertex Cover case one has a trivial branching (leading to a size 2^k search tree) and the algorithm becomes more and more complicated in order to shrink the size below 1.29^k. By way of contrast, in the Dominating Set case, the search tree itself is simple (and it is not known how to improve on that) whereas to prove its correctness (that is, the existence of a non-dominated degree-seven vertex during the course of the whole search tree algorithm) is difficult.

A completely different approach to solve Dominating Set on planar and related graphs is due to the concept of *tree decompositions*. Intuitively, a graph has a tree decomposition of small width if it is "tree-like"[10] and the smaller this so-called *treewidth* is, the better many otherwise hard graph problems can be solved (see [13,14] for surveys). Note that, similar to Vertex Cover, Dominating Set is easily linear-time solvable on trees. By a much more complicated dynamic programming, this can be generalized to graphs of bounded treewidth [85]. Using the parameter treewidth t together with the corresponding tree decomposition of the underlying graph, one can show that Dominating Set can be solved in $O(4^t \cdot n)$ time on n-vertex graphs of treewidth t [4]. Thus, we have a completely different parameterization that yields fixed-parameter tractability for Dominating Set—the *structural parameterization* by treewidth. Note that, in principle, in this way the size of the solution set does not matter anymore.

[9] A graph is planar if it can be drawn in the plane without edge crossings. A planar graph with n vertices can have at most $3n - 6$ edges (Euler formula).

[10] A tree has treewidth 1.

Can the two parameterizations "solution size k" and "treewidth t" be related? In case of planar graphs, the answer is yes. That is, it was shown that for Dominating Set—analogous results hold for Vertex Cover and many other domination-like problems [4]—on planar graphs, assuming that k denotes the size of an optimal dominating set, it holds that

$$t = O(\sqrt{k}).$$

This implies that Dominating Set on planar graphs can be solved in $2^{O(\sqrt{k})} \cdot n$ steps [4]. (See [41,42] for improvements in the constants of the O-notation.) We mention in passing that bounding the exponential growth by a function as $2^{O(\sqrt{k})}$ is asymptotically the best what can be hoped for [18]. The involved constants of the worst-case analysis are fairly large. Recently, further fixed-parameter tractability results for Dominating Set and related problems were achieved for generalizations of planar graphs such as, for example, graphs of bounded genus [31,25]. Moreover, a general study of the relationship between treewidth and solution size has been undertaken, yielding interesting characterization results concerning the corresponding graph classes [24].

There is also a second route to $2^{O(\sqrt{k})} \cdot n$ time algorithms for Dominating Set and related problems on planar graphs. This is based on Lipton and Tarjan's planar separator theorem [68]. As noted and further discussed in [8,9], the existence of a linear-size problem kernel (as we already know for Vertex Cover by the Nemhauser-Trotter theorem, see Sect. 4.2) directly implies $2^{O(\sqrt{k})} \cdot n$ time algorithms for Vertex Cover and Dominating Set on planar graphs. And, indeed, a size-$O(k)$ problem kernel by polynomial-time preprocessing could also be proven for Dominating Set [7]. The corresponding data reduction rules, which are based on "local neighborhood considerations," are relatively simple and can be efficiently implemented. (The technically most demanding part is the proof of the linear-size bound on the problem kernel.) Besides the theoretical result itself, these parameter-independent (in the sense as discussed in Sect. 4.2) preprocessing rules turn out to be of high practical value [3]. Finally, we remark that the "planar separator theorem approach" seems to be less practical (due to the high constant factors involved) than the "bounded treewidth approach" discussed before.

In summary, given the W[2]-completeness of Dominating Set on general graphs, a reasonable way out of this quandary is to look at special graph classes and different parameterizations. A promising route of future investigations is to study graph classes where Dominating Set is polynomial-time solvable or fixed-parameter tractable and then to introduce useful parameterizations "away from" these classes [16,53]—one example for that is the above mentioned parameter genus of a graph that "parameterizes away from planarity." As also noted by Leizhen Cai [16], more possibilities are around. On a broader perspective, this leads to the "art of parameterization" that will be discussed next.

5 The Art of Parameterization

The direct way to parameterize an optimization problem is to take the size of the desired solution set as the parameter. This adheres to a "pay for what you get approach"—the smaller the desired solution is, the smaller the combinatorial explosion is one has to take into account. It also turns out that this is *the* parameterization to study when considering preprocessing rules to yield a reduction to problem kernel. By way of contrast, in Sect. 4.3 we additionally considered the structural parameter treewidth (relating to the structure of the input graph, that is, more precisely, its "tree-likeness") that also allowed for efficient solutions in case of small width values.

As a concrete example for the fruitfulness of different parameterizations, recall Multicut in Trees. In the previous section we considered the parameter "number of edges to delete," which again measures the solution size. An alternative parameterization is to consider the maximum number of paths passing through a node or an edge of the given tree. This measures an input property more or less independent of the solution size. Such a parameterization is of particular interest in case of the (edge-)weighted version of Multicut in Trees: By means of dynamic programming, it can be shown that Weighted Multicut in Trees is fixed-parameter tractable with respect to the parameter "maximum path number" [54] whereas this is open concerning the parameterization "solution size."

5.1 Standard Ways of Parameterization

The above discussion shall indicate that parameterizing problems itself is an interesting and important game to play. For instance, Fellows [35] discusses how to parameterize the Max Leaf Spanning Tree problem in at least five different ways. Since the leitmotif of parameterized complexity theory is to gain a better understanding of problem hardness through a refined complexity analysis that uses a two-dimensional view on problems, the choice of various reasonable problem parameters deserves special attention. In what follows, we try to illustrate various facets of problem parameterization and argue that it may become a task of its own.

First, let us consider the NP-complete Closest String problem.

> **Input:** Strings $s_1, s_2, \ldots, s_k$ over alphabet Σ of length L each.
> **Task:** Find a string s of length L that minimizes the maximum Hamming distance to the k input strings.

Two immediately arising and practically motivated parameterizations are by the "error measure" Hamming distance d (which should be minimized) and by the number of input strings k. Two completely different methods show fixed-parameter tractability with respect to both parameterizations—deep results of integer linear programming for parameter k and a search tree of size d^d for parameter d [51]. Here, parameterization by solution size as might be measured by the length of the solution string probably is of less interest. Still, however,

for constant alphabet size simply enumerating all possible solution candidates leads to fixed-parameter tractability also with respect to the parameter L.

The situation changes when moving on to generalized (and for applications in computational biology more relevant) versions of Closest String, namely Closest Substring and, further on, Distinguishing Substring Selection. In Closest Substring, the goal string now only needs to be a *substring* in every of the given strings, thus increasing the combinatorial complexity of the problem. And, indeed, with respect to the parameter k, Closest Substring is W[1]-hard [37] and its parameterized complexity is open with respect to the distance parameter d. For the still more general Distinguishing Substring Selection, W[1]-hardness can be shown for both parameterizations [48]. Notably, Closest String, Closest Substring, and Distinguishing Substring Selection all possess polynomial-time approximation schemes [67,26]. The parameterized intractability of Distinguishing Substring Selection with respect to "optimization parameter" d, however, indicates that the corresponding PTAS [26] cannot be turned into an efficient one (see [48] for more on this, [19] for the concept of efficient PTAS, and [27,34,36] for general discussions concerning the efficiency of PTAS's, connections with parameterized complexity, and some case studies). For both Closest Substring and Distinguishing Substring Selection, the only known parameterization that works (for constant-size alphabet) is with respect to solution string length (simple enumeration of all possibilities); but this has limited applicability in many cases of practical interest. New parameterizations and special cases of practical relevance are sought—the search for them can be guided by known parameterized complexity results as the above ones.

We mention in passing that similar parameterized complexity studies have been undertaken for the Longest Common Subsequence problem—the parameterized complexity investigations mostly show hardness results [15,28,79].

Summarizing, the above examples should illustrate the great wealth of possibilities and the strong need for parameterizing problems. So far, we have seen (also refer to Sect. 4) parameters "solution size" (such as size of the dominating set), "solution quality" (such as distance parameter in Closest String), "input partitioning" (such as number of input strings in Closest String), or "input structure" (such as treewidth of graphs or maximum vertex degree in Dominating Set or alphabet size in Closest String). In Sect 5.2, we will propose a further, rich set of possibilities to parameterize problems—parameterization by "distance from triviality."

5.2 New Ways of Parameterization

Leizhen Cai [16] recently initiated a study of Graph Coloring (where the task is to color the graph vertices such that all adjacent vertices have different colors) as follows. Note that Graph Coloring is already NP-complete for three colors, so the parameterization by number of colors is of no help.[11] For instance,

[11] Juedes et al. [63] show that coloring an n-vertex-graph with $n - k$ colors is fixed-parameter tractable with respect to k.

considering the class of split graphs (where Graph Coloring is known to be solvable in polynomial time) he showed that coloring is fixed-parameter tractable with respect to parameter k on graphs that are formed from split graphs by adding or deleting at most k edges. By way of contrast, he shows that it is W[1]-hard when deletion of at most k vertices leads to a split graph. Interestingly, the problem is much harder in case of (the 2-colorable) bipartite graphs instead of split graphs: Coloring becomes NP-complete for graphs that origin from bipartite graphs by adding three edges or if two vertex deletions are needed to make a graph bipartite [16]. In summary, Cai states that "this new way of parameterizing problems adds a new dimension to the applicability of parameterized complexity theory" [16]. The prospective idea brought forward here is that the distance parameter k measures a distance from a "triviality": $k = 0$ represents a special case that is solvable in polynomial time.

A second example for the parameterization "distance from triviality" is described by Hoffmann and Okamoto [58] when studying exact solutions for the Traveling Salesman problem in the two-dimensional Euclidean plane: Given a set of n points with pairwise Euclidean distances, determine a shortest round-trip through all points. Consider a set of n points in the Euclidean plane. Determine their convex hull. If all points lie on the hull, then this gives the shortest tour. Otherwise, Hoffmann and Okamoto show that the problem is solvable in $O(k! \cdot k \cdot n)$ time where k denotes the number of points inside the convex hull. Thus, the distance from triviality here is the number k of inner points.

Finally, we give an example related to Satisfiability. Assume that a formula in conjunctive normal form has a matching between variables and clauses that matches all clauses. Then, it is easy to observe that such a formula is satisfiable. For a formula F, considered as a set of m clauses over n variables, define the *deficiency* as $\delta(F) := m - n$. The maximum deficiency is $\delta^*(F) := \max_{F' \subseteq F} \delta(F')$. Szeider showed that the satisfiability of a formula F can be decided in $O(2^{\delta^*(F)} \cdot n^3)$ time [83]. Note that a formula F with $\delta^*(F) = 0$ has a matching as described above. Again, $\delta^*(F)$ is a structural parameter measuring the distance from triviality in the described sense.

Further case studies for Clique, Power Dominating Set, Set Cover, and Longest Common Subsequence exhibiting new distance from triviality parameterizations can be found in [53].

In conclusion, to parameterize a problem usually can be done in several useful ways—the discovery and treatment of new parameterizations may require new ideas and techniques.

A further parameterized view we neglected so far is the "parameterizing above guaranteed values" approach introduced by Mahajan and Raman [69]. Whereas Mahajan and Raman study parameterizations of the Maximum Satisfiability and Maximum Cut problems, here let us only give a brief example concerning the Independent Set problem—which is the dual problem of Vertex Cover—on planar graphs. Due to the famous four-color theorem, one can conclude that every planar graph with n vertices has an independent set of size at least $\lceil n/4 \rceil$. Thus, the natural task arises to determine whether a planar graph

has an independent set of size $\lceil n/4 \rceil + d$—the new parameter being d and the guaranteed value being $\lceil n/4 \rceil$. The parameterized complexity with respect to parameter d is open. Seen from a different view, this again can be interpreted as distance from triviality; now, however, triviality means a guaranteed parameter value.

In some cases, moreover, it may be of interest to try to find relationships among different parameters of one problem—an example is given by the size of the dominating set in a planar graph and the treewidth of the underlying graph [4] (see Sect. 4). Clearly, in this way a whole *tableau of parameterized complexity questions* comes up for nearly every problem. Note, however, that often the different parameterizations of a problem are incomparable to each other and there is no point in discussing what the "best" parameterization is in general. The nature of complexity (and vice versa) simply seems to bring along the need for different views (that is, parameterizations) on the considered problem. Parameterized complexity provides the formal framework and guidance to do so.

6 Concluding Remarks

Parameterized complexity with fixed-parameter algorithmics is a fast growing field. This expository paper, oriented more towards algorithms than towards structural complexity theory, did not even touch several important topics and results of current research. Our approach to fixed-parameter algorithms is further pursued in a forthcoming monograph [74]. Besides the so far sole, more complexity-theoretic monograph by Downey and Fellows [28], we refer the reader to the numerous recent surveys [10,27,34,35,36,52,87] to further broaden the perspective on the field. Our main focus here laid on questions of parameterization itself.

Three brief messages we additionally want to communicate are as follows:

- Fixed-parameter algorithms are *old*—for instance, see Dreyfus and Wagner's [30] algorithm for the Steiner Tree problem in graphs.
- Fixed-parameter algorithms are *beautiful*—for instance, see Alon et al.'s [11] color coding technique to solve the Longest Path problem.
- Fixed-parameter algorithms' future is *promising*—for instance, see [2,46] for two recent examples[12] of successful young researchers' work.

We conclude with five spotlights on other important topics in current parameterized complexity research.

Implementation and experiments. Conceptual simplicity is not simply a virtue of its own but it is of high importance when it comes to implementing and testing fixed-parameter algorithms. A search tree with numerous case distinctions probably will not pay off in practice when compared to a search

[12] Selection was restricted to the Tübingen area.

tree algorithm with few case distinctions and slightly worse upper bounds on the size of the search tree. Moreover, preprocessing by data reduction rules seems almost obligatory in any practical implementation. Without experiments, it often will be hard to evaluate the concrete virtue of preprocessing, though. In the context of applications, it is also fruitful to think in "dual terms." That is, for example, to solve the W[1]-complete Clique problem one may use strongly tuned fixed-parameter algorithms for the dual problem Vertex Cover—an n-vertex graph has a Clique of size k iff its complement graph has a Vertex Cover of size $n-k$. In some cases, preprocessing, kernelization and sophisticated branching combined with branch-and-bound heuristics yield unexpectedly successful implementations even in cases where parameter values are not "small." Concerning Vertex Cover, recent implementation work is described in [1,20]. Several other results for various problems are described in [3,5,49,50,51]. To better understand and exploit the links between the design of heuristics and fixed-parameter algorithms, much remains to be done concerning implementation and experiments.

Automated algorithm development. Many fixed-parameter algorithms with small bounds on the parameter function $f(k)$ are based on intricate, extensive case distinctions. Recently, a framework for an automated generation of search tree algorithms was presented [47] that applies to many hard, not only graph (modification) problems. In some cases, even better upper bounds than achieved by algorithms designed by humans were achieved. Automated upper bound proofs for satisfiability problems can also be found in [32,77]. In this way, tedious and error-prone tasks (that is, design of lengthy case distinctions) can be mechanized, using some few basic concepts and observations on the underlying problem.

Connections to approximation algorithms. Fully polynomial-time approximation schemes (FPTAS's) as well as "efficient PTAS's" directly imply fixed-parameter tractability with respect to parameterizations by the "optimization value" [17,19]. By way of contrast, a proof of W[1]-hardness thus gives concrete indication for the non-existence of efficient PTAS's and FPTAS's, also see [27,36] for a more thorough discussion. In a broader sense, it is highly desirable (and it seems likely) to obtain further connections between approximation algorithm techniques together with inapproximability results and parameterized complexity.

Lower bounds. If a problem is fixed-parameter tractable measured by a function $f(k)$, the question arises how small the growth of f can be kept. For instance, Liming Cai and David Juedes [18] showed that under reasonable complexity-theoretic assumptions no $2^{o(\sqrt{k})} \cdot n^{O(1)}$ time algorithm for Dominating Set on planar graphs exists, asymptotically matching the upper bounds from [4]. Note that there are close connections to the so-called power indices of Stearns and Hunt III [82] and the existence of "subexponential" time algorithms as studied by Impagliazzo et al. [61]. Also see the very recent results of Chen et al. [21] for further studies in this direction. Frick and Grohe [43] showed that, unless P = NP, there is no algorithm for evaluating monadic second-order queries on trees in $f(k) \cdot n^{O(1)}$ time for any elemen-

tary function f, where k denotes the query size and n is the tree size. This strongly limits the practical usefulness of the corresponding fixed-parameter tractability results (also see [52] for a more thorough discussion). Finally, Fernau [38] presents first lower bound results (concerning constant factors) for linear size problem kernels.

New parameterized complexity classes. We only point to three recent developments in this context. In an effort to get a deeper understanding of the border between fixed-parameter tractability and intractability, Downey et al. [29] introduced the class M[1] seemingly intermediate between FPT and W[1] (also see [35] for further discussion). Motivated by the above mentioned lower bounds results by Frick and Grohe [43], Flum et al. [40] very recently initiated the study of a notion called bounded fixed-parameter tractability together with corresponding complexity classes. Finally, the parameterized complexity of counting problems also became subject of recent research [39,70].

Acknowledgments. I am very grateful to Rod Downey, Michael Fellows, Jens Gramm, Jiong Guo, Falk Hüffner, Klaus-Jörn Lange, Catherine McCartin, Naomi Nishimura, and Prabhakar Ragde for their constructive feedback on a preliminary version of this paper.

References

1. F. N. Abu-Khzam, R. L. Collins, M. R. Fellows, M. A. Langston, W. H. Suters, and C. T. Symons. Kernelization algorithms for the Vertex Cover problem: theory and experiments. In *Proc. ALENEX04*. ACM/SIAM, 2004. 4, 8, 15
2. J. Alber. *Exact Algorithms for NP-hard Problems on Networks: Design, Analysis, and Implementation*. PhD thesis, WSI für Informatik, Universität Tübingen, Germany, 2003. 14
3. J. Alber, N. Betzler, and R. Niedermeier. Experiments on data reduction for optimal domination in networks. In *Proc. International Network Optimization Conference (INOC 2003)*, pages 1–6, Evry/Paris, France, Oct. 2003. 10, 15
4. J. Alber, H. L. Bodlaender, H. Fernau, T. Kloks, and R. Niedermeier. Fixed parameter algorithms for Dominating Set and related problems on planar graphs. *Algorithmica*, 33(4):461–493, 2002. 9, 10, 14, 15
5. J. Alber, F. Dorn, and R. Niedermeier. Experimental evaluation of a tree decomposition based algorithm for Vertex Cover on planar graphs. *Discrete Applied Mathematics*, 2004. To appear. 15
6. J. Alber, H. Fan, M. R. Fellows, H. Fernau, R. Niedermeier, F. Rosamond, and U. Stege. Refined search tree technique for Dominating Set on planar graphs. In *Proc. 26th MFCS*, volume 2136 of *LNCS*, pages 111–122. Springer, 2001. Long version to appear in *Journal of Computer and System Sciences*. 9
7. J. Alber, M. R. Fellows, and R. Niedermeier. Efficient data reduction for Dominating Set: A linear problem kernel for the planar case. In *Proc. 8th SWAT*, volume 2368 of *LNCS*, pages 150–159. Springer, 2002. Long version to appear in *Journal of the ACM*. 10

8. J. Alber, H. Fernau, and R. Niedermeier. Parameterized complexity: exponential speed-up for planar graph problems. In *Proc. 28th ICALP*, volume 2076 of *LNCS*, pages 261–272. Springer, 2001. Long version to appear in *Journal of Algorithms*. 10
9. J. Alber, H. Fernau, and R. Niedermeier. Graph separators: A parameterized view. *Journal of Computer and System Sciences*, 67(4):808–832, 2003. 10
10. J. Alber, J. Gramm, and R. Niedermeier. Faster exact solutions for hard problems: a parameterized point of view. *Discrete Mathematics*, 229:3–27, 2001. 2, 14
11. N. Alon, R. Yuster, and U. Zwick. Color-coding. *Journal of the ACM*, 42(4):844–856, 1995. 14
12. G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation — Combinatorial Optimization Problems and Their Approximability Properties*. Springer, 1999. 1
13. H. L. Bodlaender. Treewidth: Algorithmic techniques and results. In *Proc. 22nd MFCS*, volume 1295 of *LNCS*, pages 19–36. Springer, 1997. 9
14. H. L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209:1–45, 1998. 9
15. H. L. Bodlaender, R. G. Downey, M. R. Fellows, and H. T. Wareham. The parameterized complexity of sequence alignment and consensus. *Theoretical Computer Science*, 147(1–2):31–54, 1995. 12
16. L. Cai. Parameterized complexity of Vertex Colouring. *Discrete Applied Mathematics*, 127(1):415–429, 2003. 10, 12, 13
17. L. Cai and J. Chen. On fixed-parameter tractability and approximability of NP optimization problems. *Journal of Computer and System Sciences*, 54(3):465–474, 1997. 15
18. L. Cai and D. Juedes. On the existence of subexponential parameterized algorithms. *Journal of Computer and System Sciences*, 67(4):789–807, 2003. 10, 15
19. M. Cesati and L. Trevisan. On the efficiency of polynomial time approximation schemes. *Information Processing Letters*, 64(4):165–171, 1997. 12, 15
20. J. Cheetham, F. Dehne, A. Rau-Chaplin, U. Stege, and P. J. Taillon. Solving large FPT problems on coarse-grained parallel machines. *Journal of Computer and System Sciences*, 67(4):691–706, 2003. 15
21. J. Chen, X. Huang, I. A. Kanj, and G. Xia. Strong computational lower bounds via parameterized complexity. Manuscript, a preliminary version appears in *36th ACM STOC 2004*, 2004. 15
22. J. Chen, I. A. Kanj, and W. Jia. Vertex Cover: Further observations and further improvements. *Journal of Algorithms*, 41:280–301, 2001. 2, 7, 8
23. E. Dantsin, E. A. Hirsch, S. Ivanov, and M. Vsemirnov. Algorithms for SAT and upper bounds on their complexity. Technical Report TR01-012, Electronic Colloquium on Computational Complexity, 2001. 2
24. E. D. Demaine, F. V. Fomin, M. T. Hajiaghayi, and D. M. Thilikos. Bidimensional parameters and local treewidth. In *Proc. 21st LATIN*, volume 2976 of *LNCS*, pages 109–118. Springer, 2004. 10
25. E. D. Demaine, F. V. Fomin, M. T. Hajiaghayi, and D. M. Thilikos. Subexponential parameterized algorithms on graphs of bounded-genus and H-minor-free graphs. In *Proc. 15th SODA*, pages 830–839. ACM/SIAM, 2004. 10
26. X. Deng, G. Li, Z. Li, B. Ma, and L. Wang. Genetic design of drugs without side-effects. *SIAM Journal on Computing*, 32(4):1073–1090, 2003. 12
27. R. G. Downey. Parameterized complexity for the skeptic. In *Proc. 18th IEEE Annual Conference on Computational Complexity*, pages 147–169, 2003. 2, 12, 14, 15

28. R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999. 2, 3, 4, 5, 8, 12, 14
29. R. G. Downey, M. R. Fellows, E. Prieto-Rodriguez, and F. A. Rosamond. Fixed-parameter tractability and completeness V: parametric miniatures. Manuscript, 2003. 16
30. S. E. Dreyfus and R. A. Wagner. The Steiner problem in graphs. *Networks*, 1:195–207, 1972. 14
31. J. Ellis, H. Fan, and M. R. Fellows. The Dominating Set problem is fixed parameter tractable for graphs of bounded genus. In *Proc. 8th SWAT*, volume 2368 of *LNCS*, pages 180–189. Springer, 2002. 10
32. S. S. Fedin and A. S. Kulikov. Automated proofs of upper bounds on the running time of splitting algorithms. Manuscript, September 2003. 15
33. U. Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998. 8
34. M. R. Fellows. Parameterized complexity: The main ideas and connections to practical computing. In *Experimental Algorithmics*, volume 2547 of *LNCS*, pages 51–77. Springer, 2002. 2, 12, 14
35. M. R. Fellows. Blow-ups, win/win's, and crown rules: Some new directions in FPT. In *Proc. 29th WG*, volume 2880 of *LNCS*, pages 1–12. Springer, 2003. 2, 6, 11, 14, 16
36. M. R. Fellows. New directions and new challenges in algorithm design and complexity, parameterized. In *Proc. 8th WADS*, volume 2748 of *LNCS*, pages 505–520. Springer, 2003. 2, 6, 12, 14, 15
37. M. R. Fellows, J. Gramm, and R. Niedermeier. On the parameterized intractability of Closest Substring and related problems. In *Proc. 19th STACS*, volume 2285 of *LNCS*, pages 262–273. Springer, 2002. 12
38. H. Fernau. Parametric duality: kernel sizes & algorithmics. Technical Report TR04-027, Electronic Colloquium on Computational Complexity, 2004. 16
39. J. Flum and M. Grohe. The parameterized complexity of counting problems. In *Proc. 43rd FOCS*, pages 538–550. IEEE Computer Society, 2002. 16
40. J. Flum, M. Grohe, and M. Weyer. Bounded fixed-parameter tractability and $\log^2 n$ nondeterministic bits. In *Proc. 31st ICALP*, LNCS. Springer, 2004. To appear. 16
41. F. V. Fomin and D. M. Thilikos. Dominating sets in planar graphs: branch-width and exponential speed-up. In *Proc. 14th SODA*, pages 168–177. ACM/SIAM, 2003. 10
42. F. V. Fomin and D. M. Thilikos. A simple and fast approach for solving problems on planar graphs. In *Proc. 21st STACS*, volume 2996 of *LNCS*, pages 56–67. Springer, 2004. 10
43. M. Frick and M. Grohe. The complexity of first-order and monadic second-order logic revisited. In *Proc. Logic in Computer Science*, pages 215–224. IEEE Computer Society, 2002. 15, 16
44. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1979. 1, 3
45. N. Garg, V. Vazirani, and M. Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica*, 18(1):3–20, 1997. 6
46. J. Gramm. *Fixed-Parameter Algorithms for the Consensus Analysis of Genomic Sequences*. PhD thesis, WSI für Informatik, Universität Tübingen, Germany, 2003. 14

47. J. Gramm, J. Guo, F. Hüffner, and R. Niedermeier. Automated generation of search tree algorithms for hard graph modification problems. *Algorithmica*, 39(4):321–347, 2004. 15
48. J. Gramm, J. Guo, and R. Niedermeier. On exact and approximation algorithms for Distinguishing Substring Selection. In *Proc. 14th FCT*, volume 2751 of *LNCS*, pages 261–272. Springer, 2003. Long version to appear in *Theory of Computing Systems*. 12
49. J. Gramm and R. Niedermeier. Breakpoint medians and breakpoint phylogenies: a fixed-parameter approach. *Bioinformatics*, 18(Supplement 2):S128–S139, 2002. 15
50. J. Gramm and R. Niedermeier. A fixed-parameter algorithm for Minimum Quartet Inconsistency. *Journal of Computer and System Sciences*, 67(4):723–741, 2003. 15
51. J. Gramm, R. Niedermeier, and P. Rossmanith. Fixed-parameter algorithms for Closest String and related problems. *Algorithmica*, 37(1):25–42, 2003. 11, 15
52. M. Grohe. Parameterized complexity for the database theorist. *SIGMOD Record*, 31(4):86–96, 2002. 2, 14, 16
53. J. Guo, F. Hüffner, and R. Niedermeier. A structural view on parameterizing problems: distance from triviality. Manuscript, April 2004. 10, 13
54. J. Guo and R. Niedermeier. Exact algorithms for Tree-like Weighted Set Cover. Manuscript, April 2004. 11
55. J. Guo and R. Niedermeier. Fixed-parameter tractability results for Mulitcut in Trees. Manuscript, May 2004. 6
56. E. A. Hirsch. New worst-case upper bounds for SAT. *Journal of Automated Reasoning*, 24(4):397–420, 2000. 3
57. D. Hochbaum, editor. *Approximation Algorithms for NP-hard Problems*. PWS Publishing Company, 1997. 1
58. M. Hoffmann and Y. Okamoto. The traveling salesman problem with few inner points. In *Proc. 10th COCOON*, volume 3106 of *LNCS*. Springer, 2004. To appear. 13
59. M. Hofri. *Analysis of Algorithms: Computational Methods and Mathematical Tools*. Oxford University Press, 1995. 1
60. J. Hromkovič. *Algorithmics for Hard Problems*. Springer, 2002. 1
61. R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001. 15
62. K. Iwama and S. Tamaki. Improved upper bounds for 3-SAT. Technical Report TR03-053, Electronic Colloquium on Computational Complexity, 2003. Also appears in *Proc. ACM/SIAM SODA 2004*. 3
63. D. Juedes, B. Chor, and M. R. Fellows. Linear kernels in linear time, or how to save k colors in $o(n^2)$ steps. In *Proc. 30th WG*, LNCS. Springer, 2004. To appear. 8, 12
64. S. Khuller. The Vertex Cover problem. *SIGACT News*, 33(2):31–33, 2002. 8
65. W. Küchlin and C. Sinz. Proving consistency assertions for automotive product data management. *Journal of Automated Reasoning*, 24(1–2):145–163, 2000. 3
66. O. Kullmann. New methods for 3-SAT decision and worst-case analysis. *Theoretical Computer Science*, 223(1–2):1–72, July 1999. 7
67. M. Li, B. Ma, and L. Wang. On the Closest String and Substring problems. *Journal of the ACM*, 49(2):157–171, 2002. 12
68. R. Lipton and R. Tarjan. Applications of a planar separator theorem. *SIAM Journal on Computing*, 9(3):615–627, 1980. 10
69. M. Mahajan and V. Raman. Parameterizing above guaranteed values: MaxSat and MaxCut. *Journal of Algorithms*, 31(2):335–354, 1999. 13

70. C. McCartin. Parameterized counting problems. In *Proc. 27th MFCS*, volume 2420 of *LNCS*, pages 556–567. Springer, 2002. 16
71. Z. Michalewicz and B. F. Fogel. *How to Solve it: Modern Heuristics*. Springer, 2000. 1
72. R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995. 1
73. G. L. Nemhauser and J. L. E. Trotter. Vertex packing: structural properties and algorithms. *Mathematical Programming*, 8:232–248, 1975. 8
74. R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2005. Forthcoming. 14
75. R. Niedermeier and P. Rossmanith. A general method to speed up fixed-parameter-tractable algorithms. *Information Processing Letters*, 73:125–129, 2000. 8
76. R. Niedermeier and P. Rossmanith. On efficient fixed-parameter algorithms for Weighted Vertex Cover. *Journal of Algorithms*, 47(2):63–77, 2003. 2, 7
77. S. I. Nikolenko and A. V. Sirotkin. Worst-case upper bounds for SAT: automated proof. In *15th European Summer School in Logic Language and Information (ESSLLI 2003)*, 2003. 15
78. J. Pearl. *Heuristics*. Addison–Wesley, Reading, Massachusetts, 1984. 1
79. K. Pietrzak. On the parameterized complexity of the fixed alphabet Shortest Common Supersequence and Longest Common Subsequence problems. *Journal of Computer and System Sciences*, 67(4):757–771, 2003. 12
80. C. Sinz. Visualizing the internal structure of SAT instances (preliminary report). In *Proc. 7th Intl. Conf. on Theory and Applications of Satisfiability Testing (SAT 2004)*, Vancouver, Canada, May 2004. 3
81. C. Sinz, A. Kaiser, and W. Küchlin. Formal methods for the validation of automotive product configuration data. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 17(1):75–97, 2003. 3
82. R. E. Stearns and H. B. Hunt III. Power indices and easier hard problems. *Mathematical Systems Theory*, 23:209–225, 1990. 15
83. S. Szeider. Minimal unsatisfiable formulas with bounded clause-variable difference are fixed-parameter tractable. In *Proc. 9th COCOON*, volume 2697 of *LNCS*, pages 548–558. Springer, 2003. 13
84. S. Szeider. On fixed-parameter tractable parameterizations of SAT. In *Proc. 6th SAT*, volume 2919 of *LNCS*, pages 188–202. Springer, 2004. 3, 4
85. J. A. Telle and A. Proskurowski. Practical algorithms on partial k-trees with an application to domination-like problems. In *Proc. 3rd WADS*, volume 709 of *LNCS*, pages 610–621. Springer, 1993. 9
86. V. V. Vazirani. *Approximation Algorithms*. Springer, 2001. 1
87. G. J. Woeginger. Exact algorithms for NP-hard problems: A survey. In *Proc. 5th International Workshop on Combinatorial Optimization – Eureka, You Shrink!*, volume 2570 of *LNCS*, pages 185–208. Springer, 2003. 2, 6, 14