The Parameterized Complexity of Constraint Satisfaction and Reasoning

Stefan Szeider(✉)

Vienna University of Technology, A-1040 Vienna, Austria
stefan@szeider.net

Abstract. Parameterized Complexity is a new and increasingly popular theoretical framework for the rigorous analysis of NP-hard problems and the development of algorithms for their solution. The framework provides adequate concepts for taking structural aspects of problem instances into account. We outline the basic concepts of Parameterized Complexity and survey some recent parameterized complexity results on problems arising in Constraint Satisfaction and Reasoning.

1 Introduction

Computer science has been quite successful in devising fast algorithms for important computational tasks, for instance, to sort a list of items or to match workers to machines. By means of a theoretical analysis one can guarantee that the algorithm will always find a solution quickly. Such a worst-case performance guarantee is the ultimate aim of algorithm design. The traditional theory of algorithms and complexity as developed in the 1960s and 1970s aims at performance guarantees in terms of one dimension only, the input size of the problem. However, for many important computational problems that arise from real-world applications, the traditional theory cannot give reasonable (i.e., polynomial) performance guarantees. The traditional theory considers such problems as *intractable*. Nevertheless, heuristics-based algorithms and solvers work surprisingly well on real-world instances of such problems. Take for example the satisfiability problem (Sat) of propositional reasoning. No algorithm is known that can solve a Sat instance on n variables in $2^{o(n)}$ steps (by the widely believed Exponential Time Hypothesis such an algorithm is impossible [29]). On the other hand, state-of-the-art Sat solvers solve routinely instances with hundreds of thousands of variables in a reasonable amount of time (see e.g., [23]). Hence there is an enormous gap between theoretical performance guarantees and the empirically observed performance of solvers. This gap separates theory-oriented and applications-oriented research communities. This theory-practice gap has been

Research supported by the European Research Council, grant reference 239962 (COMPLEX REASON). This survey was written in 2011, some references where updated in 2013.

H. Tompits et al. (Eds.): INAP/WLP 2011, LNAI 7773, pp. 27–37, 2013.
DOI: 10.1007/978-3-642-41524-1_2,

observed by many researchers, including Moshe Vardi, who closed an editorial letter [49] as follows:

> [...] an important role of theory is to shed light on practice, and there we have large gaps. We need, I believe, a richer and broader complexity theory, a theory that would explain both the difficulty and the easiness of problems like SAT. More theory, please!

Parameterized Complexity is a new theoretical framework that offers a great potential for reducing the theory-practice gap. The key idea is to consider—in addition to the input size—a secondary dimension, the parameter, and to design and analyse algorithms in this two-dimensional setting. Virtually in every conceivable context we know more about the input data than just its size in bytes. The second dimension (the parameter) can represent this additional information. This two-dimensional setting gives raise to a foundational theory of algorithms and complexity that can be closer to the problems as they appear in the real world. Parameterized Complexity has been introduced and pioneered by R. Downey and M. R. Fellows [8] and is receiving growing interest as reflected by hundreds of research papers (see the references in [8,15,37]). In more and more research areas such as Computational Biology, Computational Geometry, and Computational Social Choice the merits of Parameterized Complexity become apparent (see, e.g., [2,22,26]).

2 Parameterized Complexity: Basic Concepts and Definitions

In the following we outline the central concepts of Parameterized Complexity.

An instance of a parameterized problem is a pair (I, k) where I is the *main part* and k is the *parameter*; the latter is usually a non-negative integer. The central notion of the field is *fixed-parameter tractability* (FPT) which refers to solvability in time $f(k)n^c$, where f is some (possibly exponential) function of the parameter k, c is a constant, and n is the size of the instance with respect to some reasonable encoding. An algorithm that runs in time $f(k)n^c$ is called an *fpt-algorithm*. As a consequence of this definition, a fixed-parameter tractable problem can be solved in polynomial time for any fixed value of the parameter, and, importantly, the order of the polynomial does not depend on the parameter. This is significantly different from problems that can be solved in, say, time n^k, which also gives polynomial-time solvability for each fixed value of k, but since the order of the polynomial depends on k it does not scale well in k and quickly becomes inefficient for small values of k.

Take for example the Vertex Cover problem: Given a graph G and an integer k, the question is whether there is a set of k vertices such that each edge of G has at least one of its ends in this set. The problem is NP-complete, but fixed-parameter tractable for parameter k. Currently the best known fpt-algorithm for this problem runs in time of order $1.2738^k + kn$ [6]. This algorithm is practical for huge instances as long as the parameter k is below 100. The

situation is dramatically different for the INDEPENDENT SET problem, where for a given graph G and an integer k it is asked whether there is a set of k vertices of G such that no edge joints two vertices in the set. Also this problem is NP-complete, and indeed for traditional complexity the problems VERTEX COVER and INDEPENDENT SET are essentially the same, as there is a trivial polynomial-time transformation from one problem to the other (the complement set of a vertex cover is an independent set). However, no fixed-parameter algorithm for INDEPENDENT SET is known and the Parameterized Complexity of this problem appears to be very different from the complexity of VERTEX COVER. Theoretical evidence suggests that INDEPENDENT SET cannot be solved significantly faster than by trying all subsets of size k, which gives a running time of order n^k.

The subject of Parameterized Complexity splits into two complementary questions, each with its own mathematical toolkit and methods:

1. How to design and improve fixed-parameter algorithms for parameterized problems. For this question there exists a rich *toolkit of algorithmic techniques* (see, e.g., [46]).
2. How to gather evidence that a parameterized problem is not fixed-parameter tractable. For this question a *completeness theory* has been developed which is similar to the theory of NP-completeness (see, e.g., [7]) and supports the accumulation of strong theoretical evidence that a parameterized problem is *not* fixed-parameter tractable.

Every completeness theory requires a suitable notion of reduction. The classical polynomial-time reductions are not suitable for Parameterized Complexity, as they do not differentiate between problems that are fixed-parameter tractable and problems that are believed to be not (such as VERTEX COVER and INDEPENDENT SET, respectively, as discussed above). A reduction that preserves fixed-parameter tractability must ensure that the parameter of one problem maps to the parameter of the other problem. This is the case for *fpt-reductions*, the standard reductions in Parameterized Complexity. An fpt-reduction between parameterized decision problems P and Q is an fpt-algorithm that maps a problem instance (x, k) of P to a problem instance (x', k') of Q such that (i) (x, k) is a yes-instance of P if and only if (x', k') is a yes-instance of Q, and (ii) there is a computable function g such that $k' \leq g(k)$. It is easy to see that if we have an fpt-reduction from P to Q, and Q is fixed-parameter tractable, then so is P.

3 How to Parameterize?

Most research in Parameterized Complexity considers optimization problems, where the parameter is a bound on the objective function, also called solution size. For instance, the standard parameter for VERTEX COVER is the size of the vertex cover we are looking for. However, many problems that arise in Constraint Satisfaction and Reasoning are not optimization problems, and it seems more natural to consider parameters that indicate the presence of a "hidden structure" in the problem instance. It is a widely accepted view that efficient solvers exploit

the hidden structure of real-world problems. Hence such a parameter can be used to capture the presence of a hidden structure. There are several approaches to making the vague notion of a hidden structure mathematically precise in terms of a parameter.

3.1 Backdoors

If a computational problem is intractable in general, it is natural to ask for subproblems for which the problem is solvable in polynomial-time, and indeed much research has been devoted to this question. Such tractable subproblems are sometimes called "islands of tractability" or "tractable fragments." It seems unlikely that a problem instance originating from a real-world application belongs to one of the known tractable fragments, but it might be close to one. The concept of *backdoor sets* offers a generic way to gradually enlarge and extend an island of tractability and thus to solve problem instances efficiently if they are close to a tractable fragment. The size of a smallest backdoor set indicates the distance between an instance and a tractable fragment. Backdoor sets were introduced in the context of propositional and constraint-based reasoning [50] but similar notions can be defined for other reasoning problems. Roughly speaking, after eliminating the variables of a backdoor set one is left with an instance that belongs to the tractable subproblem under consideration. The "backdoor approach" to reasoning problems involves two tasks. The first task, called *backdoor detection*, is to detect a small backdoor set by an fpt-algorithm, parameterized by the size of the backdoor set. The second task, called *backdoor evaluation*, is to solve the reasoning problem efficiently using the information provided by the backdoor set.

There are several Parameterized Complexity results on backdoor sets for the Sat problem as described in a recent survey [21]. Backdoors have also been applied to problems beyond NP such as Model Counting and QBF-Satisfiability [38,42], and to the main reasoning problems of propositional disjunctive answer set programming (deciding whether an atom belongs to some stable model or whether it belongs to all stable models). The latter problems are located at the second level of the Polynomial Hierarchy [11] but can be solved in polynomial time for normal (disjunction-free) programs that have certain acyclicity properties. Several of these tractable classes admit a backdoor approach, with fixed-parameter tractable backdoor detection and backdoor evaluation, thus rendering the answer-set programming problems fixed-parameter tractable [13]. A similar backdoor approach has also been developed for problems of abstract argumentation [10] whose unparameterized versions are also located at the second level of the Polynomial Hierarchy.

3.2 Decompositions

A key technique for coping with hard computational problems is to decompose the problem instance into small tractable parts, and to reassemble the solutions of the parts to a solution of the entire instance. One aims at decompositions

for which the overall complexity depends on how much the parts overlap, the "width" of the decomposition. The most popular and widest studied decomposition method is *tree decomposition* with the associated parameter *treewidth*. A recent survey by Hlinený et al. covers several decomposition methods with particular focus on fixed-parameter tractability [28].

Recent results on the Parameterized Complexity of reasoning problems with respect to decomposition width include results on disjunctive logic programming and answer-set programming with weight constraints [24,41], abductive reasoning [25], satisfiability and propositional model counting [39,44], constraint satisfaction and global constraints [43,45], and abstract and value-based argumentation [9,30].

3.3 Locality

Practical algorithms for hard reasoning problems are often based on *local search* techniques. The basic idea is to start with an arbitrary candidate solution and to try to improve it step by step, at each step moving from one candidate solution to a better "neighbor" candidate solution. It would provide an enormous speed-up if one could perform k elementary steps of local search efficiently in one "giant" k-step. Such a giant k-step also decreases the probability of getting stuck at a poor local optimum. However, the obvious strategy for performing one giant k-step requires time of order N^k (assuming a candidate solution has N neighbour solutions), which is impractical already for very small values of k since typically N is related to the input size. A challenging objective is the design of fpt-algorithms (with respect to parameter k) that compute a giant k-step. Recent work on parameterized local search includes the problem of minimizing the Hamming weight of satisfying assignments for Boolean CSP [31] and for the Max Sat problem [48]. It turns out that there are interesting cases for which k-step local search is fixed-parameter tractable.

Local consistency is a further form of locality that plays an important role in constraint satisfaction and is one of the oldest and most fundamental concepts of in this area. It can be traced back to Montanari's famous 1974 paper [36]. If a constraint network is locally consistent, then consistent instantiations to a small number of variables can be consistently extended to any further variable. Hence local consistency avoids certain dead-ends in the search tree, in some cases it even guarantees backtrack-free search [1,18]. The simplest and most widely used form of local consistency is arc-consistency, introduced by Mackworth [33], and later generalized to k-consistency by Freuder [17]. A constraint network is *k-consistent* if each consistent assignment to $k-1$ variables can be consistently extended to any further k-th variable. It is a natural question to ask for the Parameterized Complexity of checking whether a constraint network is k-consistent, taking k as the parameter. This question has been subject to a recent study [20]. It turned out that in general, deciding whether a constraint network is k-consistent is complete for the parameterized complexity class co-W[2] and thus unlikely to be fixed-parameter tractable. However, if we include as secondary parameters

the maximum domain size and the maximum number of constraints in which a variable occurs, then the problem becomes fixed-parameter tractable.

3.4 Above or Below Guaranteed Bounds

For some optimization problems that arise in constraint satisfaction and reasoning, the standard parameter (solution size) is not a very useful one. Take for instance the problem Max Sat. The standard parameter is the number of satisfied clauses. However, it is well-known that one can always satisfy at least half of the clauses. Hence, if we are given m clauses, and if we want to satisfy at least k of them, then the answer is clearly *yes* if $k \leq m/2$. On the other hand, if $k > m/2$ then $m < 2k$, hence the size of the given formula is bounded in terms of the parameter k, and thus can be trivially solved by brute force in time that only depends on k. Less trivial is the question of whether we can satisfy at least $m/2 + k$ clauses, where k is the parameter. Such a problem is called *parameterized above a guaranteed value* [34,35]. Over the last few years, several variants of Max Sat but also optimization problems regarding ordering constraints have been studied, parameterized above a guaranteed value. A recent survey by Gutin and Yeo covers these results [27].

4 Kernelization: Preprocessing with Guarantee

Preprocessing and data reduction are powerful ingredients of virtually every practical solver. Before performing a computationally expensive case distinction, it seems always better to seek for a "safe step" that simplifies the instance, and to preprocess. Indeed, the success of practical solvers relies often on powerful preprocessing techniques. However, preprocessing has been neglected by traditional complexity theory: if we measure the complexity of a problem just in terms of the input size n, then reducing the size from n to $n - 1$ in polynomial time yields a polynomial-time algorithm for the problem as we can iterate the reduction [12]. Hence it does not make much sense to study preprocessing for NP-hard problems in the traditional one-dimensional framework. However, the notion of "kernelization", a key concept of Parameterized Complexity provides the means for studying preprocessing, since the impact of preprocessing can measured in terms of the parameter, not the size of the input. When a problem is fixed-parameter tractable then each instance (I, k) can be reduced in polynomial time to an equivalent instance (I', k'), the *problem kernel*, where $k' \leq k$ and the size of I' is bounded by a function of k. The smaller the kernel, the more efficient the fixed-parameter algorithm. For a parameterized problem it is therefore interesting to know whether it admits a *polynomial kernel* or not. Over the last few years, this question has received a lot of attention in the Parameterized Complexity community [32].

Several optimization problems, such as Vertex Cover and Feedback Vertex Set admit polynomial kernels with respect to the standard parameter solution size [5,6]. However, it turns out that many fixed-parameter tractable

problems in the areas of constraint satisfaction, global constraints, satisfiability, nonmonotonic and Bayesian reasoning do not have polynomial kernels unless the Polynomial Hierarchy collapses to its third level [47]. Such super-polynomial kernel lower bounds can be obtained by means of recent tools [4,16]. A positive exception is the consistency problem for certain global constraint, which admits a polynomial kernel for an interesting parameter [19].

5 Breaking Complexity Barriers with FPT-Reductions

Many important problems in constraint satisfaction and reasoning are located above the first level of the Polynomial Hierarchy or are even PSPACE-complete, thus considered "harder" than the SAT problem. Above we have discussed some results that establish fixed-parameter tractability for such problems (including ASP problems and QBF satisfiability, parameterized by backdoor size). However, for such hard problems, asking for fixed-parameter tractability is asking for a lot and requires the parameters to be quite restrictive. Therefore, it seems to be an even more interesting approach to exploit some structural properties of the instance in terms of a parameter, not to solve the instance, but to reduce it to an equivalent instance of a problem of lower classical complexity. The parameter can thus be less restrictive and can therefore be small for larger classes of inputs. The reduction cannot run in polynomial time, unless the Polynomial Hierarchy collapses, but the enhanced power of *fpt-reductions* (see Sect. 2) can break the barriers between classical complexity classes. The SAT problem is well-suited as a target problem (say, with the constant parameter 0), since by means of fpt-reductions to SAT we can make today's extremely powerful SAT solvers applicable to problems on higher levels of the Polynomial Hierarchy. In fact, there are some known reductions that, in retrospect, can be seen as fpt-reductions to SAT. A prominent example is *Bounded Model Checking* [3], a technique of immense practical significance for hardware and software verification, which can be seen as an fpt-reduction from the PSPACE-complete model checking problem for linear temporal logic to SAT. The parameter is an upper bound on the size of a counterexample (or the diameter of the instance).

In recent work [14] we have developed fpt-reductions that break complexity barriers for the main reasoning problems of disjunctive answer-set programming. These problems are located at the second level of the Polynomial Hierarchy in general, but drop back to the first level if restricted to normal (i.e., disjunction-free) programs. Thus, it is natural to consider as a parameter the *distance of a disjunctive program form being normal*; the backdoor size with respect to the base class of normal programs provides such a distance measure. And indeed, there is an fpt-reduction with respect to this parameter, that takes as input a disjunctive program P and an atom x, and outputs a CNF formula that is satisfiable if and only if x is in some answer set of P (a similar fpt-reduction outputs a CNF formula that is unsatisfiable if and only if x is in all answer sets of P). In terms of parameterized complexity, this shows that the brave reasoning problem for disjunctive ASP is paraNP-complete; paraNP is the class of all parameterized decision problems that can be solved in time $f(k)n^c$ by a *nondeterministic*

algorithm [15]. For parameterizations of NP-problems, a paraNP-completeness result is considered as very negative. For a problem that is harder than NP, however, a paraNP-completeness result is a positive one, as it shows that the structure represented by the parameter can be exploited to break the complexity barrier. Very recently, we developed similar fpt-reductions for problems arising in propositional abductive reasoning which are also located on the second level of the Polynomial Hierarchy, taking as parameters the distance of the input theory form being Horn or being Krom [40].

6 Conclusion

Over the last decade, Parameterized Complexity has become an important field of research in Algorithms and Complexity. It provides a more fine-grained complexity analysis than the traditional theory taking structural aspects of problem instances into account. In this brief survey we have outlined the basic concepts of Parameterized Complexity and indicated some recent results on the Parameterized Complexity of problems arising in Constraint Satisfaction an Reasoning.

References

1. Atserias, A., Bulatov, A., Dalmau, V.: On the power of k-consistency. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 279–290. Springer, Heidelberg (2007)
2. Betzler, N., Bredereck, R., Chen, J., Niedermeier, R.: Studies in computational aspects of voting- a parameterized complexity perspective. In: Bodlaender, H.L., Downey, R., Fomin, F.V., Marx, D. (eds.) Fellows Festschrift 2012. LNCS, vol. 7370, pp. 318–363. Springer, Heidelberg (2012)
3. Biere, A., Cimatti, A., Clarke, E., Zhu, Y.: Symbolic model checking without BDDs. TACAS 1999. LNCS, vol. 1579, pp. 193–207. Springer, Heidelberg (1999)
4. Bodlaender, H.L., Downey, R.G., Fellows, M.R., Hermelin, D.: On problems without polynomial kernels (extended abstract). In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part I. LNCS, vol. 5125, pp. 563–574. Springer, Heidelberg (2008)
5. Cao, Y., Chen, J., Liu, Y.: On feedback vertex set new measure and new structures. In: Kaplan, H. (ed.) SWAT 2010. LNCS, vol. 6139, pp. 93–104. Springer, Heidelberg (2010)
6. Chen, J., Kanj, I.A., Xia, G.: Improved upper bounds for vertex cover. Theoret. Comput. Sci. **411**(40–42), 3736–3756 (2010)
7. Chen, J., Meng, J.: On parameterized intractability: hardness and completeness. Comput. J. **51**(1), 39–59 (2008)
8. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Monographs in Computer Science. Springer, New York (1999)
9. Dunne, P.E.: Computational properties of argument systems satisfying graph-theoretic constraints. Artif. Intell. **171**(10–15), 701–729 (2007)
10. Dvorák, W., Ordyniak, S., Szeider, S.: Augmenting tractable fragments of abstract argumentation. Artif. Intell. **186**, 157–173 (2012)

11. Eiter, T., Gottlob, G.: On the computational cost of disjunctive logic programming: propositional case. Ann. Math. Artif. Intell. **15**(3–4), 289–323 (1995)
12. Fellows, M.R.: The lost continent of polynomial time: preprocessing and kernelization. In: Bodlaender, H.L., Langston, M.A. (eds.) IWPEC 2006. LNCS, vol. 4169, pp. 276–277. Springer, Heidelberg (2006)
13. Fichte, J.K., Szeider, S.: Backdoors to tractable answer-set programming. Technical Report 1104.2788, Arxiv.org. Extended and updated version of a paper that appeared in the proceedings of IJCAI 2011, The 22nd International Joint Conference on Artificial Intelligence (2012)
14. Fichte, J.K., Szeider, S.: Backdoors to normality for disjunctive logic programs. In: des Jardins, M., Littman, M.L. (eds.) Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI 2013), Bellevue, Washington, USA, 14–18 July 2013, pp. 320–337. AAAI Press, California (2013)
15. Flum, J., Grohe, M.: Parameterized Complexity Theory, vol. XIV. Texts in Theoretical Computer Science. An EATCS Series. Springer, Berlin (2006)
16. Fortnow, L., Santhanam, R.: Infeasibility of instance compression and succinct PCPs for NP. In: Dwork, C. (ed.) Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, 17–20 May 2008, pp. 133–142. ACM, New York (2008)
17. Freuder, E.C.: Synthesizing constraint expressions. Commun. ACM **21**(11), 958–966 (1978)
18. Freuder, E.C.: A sufficient condition for backtrack-bounded search. J. ACM **32**(4), 755–761 (1985)
19. Gaspers, S., Szeider, S.: Kernels for global constraints. In: Walsh, T. (ed.) Proceedings of the 22nd International Joint Conference on Artificial Intelligence, IJCAI 2011, pp. 540–545. AAAI Press, California (2011)
20. Gaspers, S., Szeider, S.: The parameterized complexity of local consistency. In: Lee, J. (ed.) CP 2011. LNCS, vol. 6876, pp. 302–316. Springer, Heidelberg (2011)
21. Gaspers, S., Szeider, S.: Backdoors to satisfaction. In: Bodlaender, H.L., Downey, R., Fomin, F.V., Marx, D. (eds.) Fellows Festschrift 2012. LNCS, vol. 7370, pp. 287–317. Springer, Heidelberg (2012)
22. Giannopoulos, P., Knauer, C., Whitesides, S.: Parameterized complexity of geometric problems. Comput. J. **51**(3), 372–384 (2008)
23. Gomes, C.P., Kautz, H., Sabharwal, A., Selman, B.: Satisfiability solvers. In: van Harmelen, F., Lifschitz, V. (eds.) Handbook of Knowledge Representation, vol. 3, Foundations of Artificial Intelligence, pp. 89–134. Elsevier, Amsterdam (2008)
24. Gottlob, G., Pichler, R., Wei, F.: Bounded treewidth as a key to tractability of knowledge representation and reasoning. In: 21st National Conference on Artificial Intelligence and the 18th Innovative Applications of Artificial Intelligence Conference. AAAI Press (2006)
25. Gottlob, G., Pichler, R., Wei, F.: Abduction with bounded treewidth: from theoretical tractability to practically efficient computation. In: Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI 2008), Chicago, Illinois, USA, 13–17 July 2008, pp. 1541–1546. AAAI Press, California (2008)
26. Gramm, J., Nickelsen, A., Tantau, T.: Fixed-parameter algorithms in phylogenetics. Comput. J. **51**(1), 79–101 (2008)
27. Gutin, G., Yeo, A.: Constraint satisfaction problems parameterized above or below tight bounds: a survey. In: Bodlaender, H.L., Downey, R., Fomin, F.V., Marx, D. (eds.) Fellows Festschrift 2012. LNCS, vol. 7370, pp. 257–286. Springer, Heidelberg (2012)

28. Hlinený, P., Oum, S., Seese, D., Gottlob, G.: Width parameters beyond tree-width and their applications. Comput. J. **51**(3), 326–362 (2008)
29. Impagliazzo, R., Paturi, R., Zane, F.: Which problems have strongly exponential complexity? J. Comput. Syst. Sci. **63**(4), 512–530 (2001)
30. Kim, E.J., Ordyniak, S., Szeider, S.: Algorithms and complexity results for persuasive argumentation. Artif. Intell. **175**, 1722–1736 (2011)
31. Krokhin, A., Marx, D.: On the hardness of losing weight. ACM Trans. Algorithm **8**(2), 19 (2012)
32. Lokshtanov, D., Misra, N., Saurabh, S.: Kernelization – preprocessing with a guarantee. In: Bodlaender, H.L., Downey, R., Fomin, F.V., Marx, D. (eds.) Fellows Festschrift 2012. LNCS, vol. 7370, pp. 129–161. Springer, Heidelberg (2012)
33. Mackworth, A.K.: Consistency in networks of relations. Artif. Intell. **8**, 99–118 (1977)
34. Mahajan, M., Raman, V.: Parameterizing above guaranteed values: MaxSat and MaxCut. J. Algorithms **31**(2), 335–354 (1999)
35. Mahajan, M., Raman, V., Sikdar, S.: Parameterizing MAX SNP problems above guaranteed values. In: Bodlaender, H.L., Langston, M.A. (eds.) IWPEC 2006. LNCS, vol. 4169, pp. 38–49. Springer, Heidelberg (2006)
36. Montanari, U.: Networks of constraints: fundamental properties and applications to picture processing. Inf. Sci. **7**, 95–132 (1974)
37. Niedermeier, R.: Invitation to Fixed-Parameter Algorithms. Oxford Lecture Series in Mathematics and its Applications. Oxford University Press, Oxford (2006)
38. Nishimura, N., Ragde, P., Szeider, S.: Solving #SAT using vertex covers. Acta Informatica **44**(7–8), 509–523 (2007)
39. Ordyniak, S., Paulusma, D., Szeider, S.: Satisfiability of acyclic and almost acyclic cnf formulas. Theor. Comput. Sci. **481**, 85–99 (2013)
40. Pfandler, A., Rümmele, S., Szeider, S.: Backdoors to absuction. In: Proceedings of the 23th International Joint Conference on Artificial Intelligence (IJCAI 2013), Beijing, China, 3–9 August 2013 (2013) (to appear)
41. Pichler, R., Rümmele, S., Szeider, S., Woltran, S.: Tractable answer-set programming with weight constraints: bounded treewidth is not enough. In: Lin, F., Sattler, U., Truszczynski, M. (eds.) Principles of Knowledge Representation and Reasoning: Proceedings of the Twelfth International Conference (KR 2010), Toronto, Ontario, Canada, 9–13 May 2010. AAAI Press, California (2010)
42. Samer, M., Szeider, S.: Backdoor sets of quantified Boolean formulas. J. Autom. Reason. **42**(1), 77–97 (2009)
43. Samer, M., Szeider, S.: Tractable cases of the extended global cardinality constraint. Constraints **16**(1), 1–24 (2009)
44. Samer, M., Szeider, S.: Algorithms for propositional model counting. J. Discret. Algorithms **8**(1), 50–64 (2010)
45. Samer, M., Szeider, S.: Constraint satisfaction with bounded treewidth revisited. J. Comput. Syst. Sci. **76**(2), 103–114 (2010)
46. Sloper, C., Telle, J.A.: An overview of techniques for designing parameterized algorithms. Comput. J. **51**(1), 122–136 (2008)
47. Szeider, S.: Limits of preprocessing. Proceedings of the twenty-fifth conference on artificial intelligence, AAAI 2011, pp. 93–98. AAAI Press, California (2011)
48. Szeider, S.: The parameterized complexity of k-flip local search for SAT and MAX SAT. Discrete Optim. **8**(1), 139–145 (2011)
49. Vardi, M.Y.: On P, NP, and computational complexity. Commun. ACM **53**(11), 5 (Nov. 2010)

50. Williams, R., Gomes, C., Selman, B.: Backdoors to typical case complexity. In: Gottlob, G., Walsh, T. (eds.) Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, IJCAI 2003, pp. 1173–1178. Morgan Kaufmann, San Francisco (2003)

http://www.springer.com/978-3-642-41523-4

Applications of Declarative Programming and Knowledge Management
19th International Conference, INAP 2011, and 25th Workshop on Logic Programming, WLP 2011, Vienna, Austria, September 28-30, 2011, Revised Selected Papers
Tompits, H.; Abreu, S.; Oetsch, J.; Pührer, J.; Seipel, D.; Umeda, M.; Wolf, A. (Eds.)
2013, XIII, 365 p. 67 illus., Softcover
ISBN: 978-3-642-41523-4