The Computer Journal Special Issue on Parameterized Complexity: Foreward by the Guest Editors

Rodney G. Downey
Victoria University
Wellington, New Zealand

Michael R. Fellows
University of Newcastle
Newcastle, Australia

Michael A. Langston University of Tennessee
Knoxville, Tennessee

November 23, 2007

1 Introductory Remarks

Parameterized complexity studies a generalization of the notion of polynomial time, where, in addition to the overall input size n, one also considers the effects on computational complexity of a secondary measurement, the *parameter*. 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 function (usually exponential) of the parameter k, and c is a constant. The subject unfolds in two basic complementary projects and associated mathematical toolkits: (1) How to design (and improve) FPT algorithms, for parameterized problems that admit them, and (2) How to gather evidence that a parameterized problem probably does not admit an FPT algorithm.

There are several things that one can say about the field, in a general way.

- It is in some sense a very old subject in computer science. Practitioners, and even theorists, have paid attention to natural problem parameters, and designed efficient algorithms that pay attention to them, "since the beginning." It can happen that one offers a practical computing scientist an FPT algorithm for an NP-hard problem having a natural small parameter and be told: "That's what I already do!"
- The field has steadily grown, since the first papers on the subject in the late 1980's and early 1990's, to the point where it now interacts importantly and substantially with many different areas of both computer science theory and applied algorithmics, and has a steady stream of topical conferences and workshops.

- The field has developed strong connections to practical computing, heuristics, and outlooks on the mission of theoretical computer science such as articulated by *algorithms engineering*.

This Special Issue of surveys of various aspects of parameterized complexity and algorithmics began on the suggestion of the Chief Editor, Fionn Murtagh, who after hearing a broad account of the field at a colloquium at Royal Holloway, University of London, declared, "This is a subject that every computer scientist should know about."

2 Roots of the Field

Historically and intellectually, there are three principal roots of the field:
• Practical algorithmics.
• Mathematical modeling and naturality.
• The richness of the mathematical opportunities.

2.1 Practical algorithmics

One can provocatively ask:

> *When is it, that there is nothing more than the overall input size n that is relevant to assessing the computational complexity of problems encountered with real-world input distributions? When is there no other measurement of relevance: not the size of the solution, not some structural aspect of the instances encountered in practice; nothing except the overall input size really matters?*

Outside of cryptography (where we make it so) the answer is basically: *never*.

Practical computing is full of motivations to the subject of parameterized complexity and algorithmics. We mention two prominent examples that have been influential in the development of the field.

Example 1. Type Checking in ML.

ML compilers work reasonably well. One of the problems the compiler must solve is the checking of the compatibility of type declarations. This problem is complete for deterministic exponential time [HM91], so the situation appears dire from the standpoint of one-dimensional classical complexity theory. The implementations work well in practice, however, using an algorithm that previously might have been called a heuristic. The compiler solves the type checking problem with an an FPT subroutine having a running time of

$O(2^k n)$, where n is the size of the program and k is the maximum nesting depth of the type declarations [LP85]. Since normally $k \leq 6$, this is clearly practical.

What this example motivates is the search for small parameters that may be highly relevant to assessing real-world computational complexity. Many more good examples of problems having small natural parameters can be found in the survey by Panos Giannopoulos, Christian Knauer and Sue Whitesides (concerning parameterized geometric problems), the survey by Jens Gramm, Arfst Nickelsen and Till Tantau (concerning applications in phylogenetics) and the survey by Georg Gottlob and Stefan Szeider (applications in artificial intelligence, constraint satisfaction and database problems).

Nesting depth is an example of a kind of *structural parameterization*. The study of parameterized algorithms has led to the identification of *general* stuctural parameters that turn up in a wide variety of settings, leading to general FPT algorithmic machineries, such as *bounded treewidth*, surveyed by Hans Bodlaender and Arie Koster. Important applications of bounded treewidth algorithmics in multiple sequence alignment are discussed in the survey by Liming Cai, Xiuzhen Huang, Chunmei Liu, Frances Rosamond and Yinglei Song. The exploration of general structural parameters of hypergraphs and matroids is surveyed by Georg Gottlob, Peter Hlineny, Sang-il Oum and Detlef Seese. Erik Demaine and MohammadTaghi Hajiaghayi survey their powerful new *bidimensionality theory* that studies how treewidth interacts with other structural parameters.

Example 2. Karsten Weihe's Train Problem.

Karsten Weihe has described a basic optimization problem concerning the train systems of Europe. Consider a bipartite graph $G = (V, E)$ where V is bipartitioned into two sets S (stations) and T (trains), and where an edge represents that a train t stops at a station s. The relevant graphs are huge, on the order of 10,000 vertices. The problem is to compute a minimum number of stations $S' \subseteq S$ such that every train stops at a station in S'. It is easy to see that this is a special case of the Hitting Set problem, and is therefore *NP*-complete.

However, the following two reduction rules can be applied to simplify (pre-process) the input to the problem. In describing these rules, let $N(s)$ denote the set of trains that stop at station s, and let $N(t)$ denote the set of stations at which the train t stops.

1. If $N(s) \subseteq N(s')$ then delete s.
2. If $N(t) \subseteq N(t')$ then delete t'.

Applications of these reduction rules cascade, preserving at each step enough information to obtain an optimal solution. Weihe found that, remarkably, these two simple reduction rules were strong enough to "digest" the original, huge input graph into a *problem kernel* consisting of disjoint components of size at most 50 — small enough to allow the problem to then be solved optimally.

What this example motivates is the systematic, mathematical study of pre-processing (kernelization) algorithms, a subject that is central to parameterized complexity theory and parameterized algorithmics. Kernelization and other key practical FPT algorithmic techniques are surveyed by Falk Hüffner, Rolf Niedermeier and Sebastian Wernicke. A case study of the implementation of FPT techniques in an algorithms engineering context is described in the article by Michael Langston, Andy Perkins, Arnold Saxton, Jon Scharff and Brynn Voy.

2.2 Mathematical naturality and modeling

What makes a good theory of computational complexity? One can roughly say that such an enterprise revolves around a battle between two function classes. In the case of classical complexity, these are the class P of polynomial functions $p(n)$ of the single variable n, and the class of functions $2^{p(n)}$. There is more to it, of course, but at the basic level of modeling *concrete matters of concern for people who employ algorithms*, having an algorithm with a running time in the first class, rather than the second, makes a big difference. A function class battle is interesting for theoretical computer science if it:

- Models issues of interest in applied computer science.
- Supports a theory that is mathematically doable and productive.

Classical computational complexity theory achieves both of these. The only hesitation one might have is that almost all interesting computational problems turn out to be NP-hard in the one-dimensional classical framework. So what then?

Parameterized complexity can be viewed as a "two-dimensional" sequel. In addition to the overall input size n, a secondary measurement, the *parameter* is introduced. The function class battle at the heart of the subject is between the two-variable class of functions: those that are $f(k)n^c$ for some function f and constant c (the FPT class), and those that are $n^{g(k)}$ for some function g. The criteria concerning what makes a function class battle interesting are unchanged, because they capture the essential relationship between theoretical and applied science. The papers in this volume make abundantly clear that these criteria are met. Parameterization also allows a single classical problem to be parameterized in a variety of different ways, relevant to a variety of applications in the real world. The quest for FPT algorithms has led to a wide range of new techniques, discussed in the survey by Niedermeier *et al.*, and systematized in the survey by Christian Sloper and Jan Arne Telle.

Will there be further theories, based on different function class battles that go beyond both the one-dimensional classical and two-dimensional parameterized frameworks? Why not? The possibilities are to date largely unexplored. Parameterized complexity is most likely just the opening chapter of a broader exploration of *multivariate complexity analysis*, motivated by the search for more effective mathematical frameworks for realistic complexity assessment

and algorithm design. For comparison, certainly, theoretical physics has not limited itself to one-dimensional theories.

The survey by Demaine and Hajiaghayi explores some of the multivariate theme. Jianer Chen and Jie Meng overview the complexity classes and basic structure theory of parameterized complexity. Iris van Rooij and Todd Wareham survey uses of parameterized complexity analysis in modeling issues relevant to theory-formation in Cognitive Science. Fine-grained complexity theory is relevant to fields of science concerned with natural forms of computation, and that relevance increases by the day.

2.3 The sheer richness of the program

A third powerful source of the rapid expansion of the field of parameterized complexity and parameterized algorithmics, is the sheer richness of the mathematical opportunities to develop and apply the framework, empowering useful and interesting insights into concrete computational problems. Here we will quickly describe the main subprograms that have developed, meaning by a *subprogram*, a kind of result for which one can expect there will eventually be hundreds of concrete examples. For each subprogram, we describe a few concrete examples, a prominent concrete open problem or two, and pointers to those surveys in this collection that are relevant.

1. Is it FPT or W-hard? Of course, the most basic kind of concrete question, the one that the field began by attending to, is just the question of whether a particular parameterized problem is FPT, or whether it is hard for $W[1]$. Despite all the concrete analyses that have been accomplished so far, such as the recent breakthrough result that Directed Feedback Vertex Set, parameterized by the size of the solution, is FPT, there is a super-abundance of still unresolved natural concrete parameterized problems. For example, it is still unknown whether Graph Topological Containment is FPT or parameterized intractable. It is open whether it is FPT to delete k clauses of a 2SAT expression E, to obtain an expression E' that is satisfiable. Many simply-stated naturally parameterized problems concerning digraphs are open as to their parameterized complexity, a topic surveyed by Gregory Gutin and Anders Yeo.

2. Can the PTAS be improved to an EPTAS? An important special case of Subprogram 1, is where the parameter is $k = 1/\epsilon$, where ϵ has its usual role in the definition of a PTAS. The definition of a PTAS requires that for every fixed ϵ, the algorithm runs in polynomial time. This is insensitive to whether ϵ occurs in the exponent of the polynomial, which is frequently the case. The question of whether "epsilon can be gotten out of the exponent" is entirely natural, and matters, if PTASs are to be useful (as they are often claimed to be). An EPTAS (*efficient PTAS*) is one that runs in FPT time, parameterized by $k = 1/\epsilon$. In 1996, Arora described a PTAS for the Euclidean Traveling Salesman Problem [Ar96]. In 1997, he was able to improve this to an EPTAS [Ar97]. There are ways, both straightforward and more intricate, for deploying the framework of parameterized complexity to investigate this issue. All of this is very ably surveyed by Daniel Marx,

in this collection. For an example of a negative outcome, Marx showed in [Marx06] that the Minimum Dominating Set for Unit Disk Graphs problem, that has a PTAS, cannot have an EPTAS unless FPT = W[1]. Given the huge amount of effort to devise PTASs for various problems in recent years, there are hundreds of concrete open problems of this kind.

3. The Subprogram of "The $f(k)$ Game." Once a parameterized problem is known to be FPT, *the race begins!* There are now at least a dozen examples of spectacular trajectories in FPT algorithms for specific problems, where the $f(k)$ in the FPT running time gets better and better. These "competitions" have been useful in the development of the field, as new approaches have been invented and deployed (to gain, usually, only a temporary leadership of the pack). The best known FPT algorithm currently for the Vertex Cover problem runs in time $O(1.274^k + kn)$ [CKX06]. This can probably still be improved. The Undirected Feedback Vertex Set problem can be solved in time $O(10.567^k)n^2$. Surely this $f(k)$ can be improved. The survey by Leizhen Cai identifies a number of FPT problems where the race is still on. Other good examples can be found in the survey of Gramm *et al.* on combinatorial problems in phylogenetics.

4. The Subprogram of "Worst Case Exponential Complexity." In the Traveling Salesman Problem (TSP) an obviously significant secondary measurement (in the sense of our provocative question), is the *number of cities* in the instance. If we let the parameter k denote this secondary measurement, then the problem is trivially in FPT (by trying all $k!$ permutations of the cities), but the "$f(k)$ game" is still very interesting. A classic result of Held and Karp [HK62] shows that under this natural parameterization, the TSP problem can be solved in time $O^*(2^k)$ by dynamic programming on subsets of the cities, where the O^* notation discards polynomial timecosts associated to the overall input size. This convenient notation for the $f(k)$ game was introduced by Impagliazzo *et al.* [IPZ98] (see also the survey by Woeginger [Woe03]). In the course of systematizing the extensive heritage of work on "worst-case exponential complexity", they inadvertently rediscovered the notion of *parameterization*, and articulated an important special case of the $f(k)$ game. Beautifully, their "framework making" efforts for this classical area, converged with research on "renormalized" FPT problems initiated by Cai and Juedes [CJ03], as codified in the current tower of the main parameterized complexity classes:

$$Lin(k) \subseteq Poly(k) \subseteq FPT \subseteq M[1] \subseteq W[1] \subseteq M[2] \subseteq W[2] \subseteq \cdots \subseteq XP$$

It is now known that FPT = M[1] (where equality of these parameterized classes is mediated by FPT reductions) if and only if the *Exponential Time Hypothesis* (ETH) fails. The ETH asserts that there is no $O^*(2^{o(k)})$ algorithm for k-variable 3SAT. The survey by Chen *et al.* discusses these developments.

5. The Subprogram of "FPT Optimality." The quest here is to show that the *form* of the $f(k)$ function achieved by an FPT algorithm is "optimal". For example, Cai and Juedes [CJ03] showed that the known FPT algorithms for Vertex Cover that run in time $O(2^{O(k)}n^c)$ are optimal in this sense: there can be no $O(2^{o(k)}n^c)$ FPT algorithm for Vertex Cover unless FPT = M[1]. They also showed optimality for the known FPT algorithm for Planar Dominating Set that runs in time $O(2^{O(\sqrt{k})}n^c)$: there can be no FPT algorithm

for PLANAR DOMINATING SET that runs in time $O(2^{o(\sqrt{k})}n^c)$ unless FPT = M[1]. For many FPT parameterized problems, FPT optimality is still an open question.

6. The Subprogram of "XP optimality." Two landmark results in this program are:
• There is no $O(n^{o(k)})$ algorithm to solve the k-INDEPENDENT SET problem for n-vertex graphs, unless FPT = M[1].
• There is no $O(n^{o(k)})$ algorithm to solve the k-DOMINATING SET problem for n-vertex graphs, unless FPT = M[2].
Obviously, both of these problems can be solved in time $O(n^{O(k)})$ by brute force, so the above results show a kind of "optimality" for the brute force algorithms. XP is the class of parameterized problems that are solvable in time $O(n^{g(k)})$. If a parameterized problem is not in FPT, we are still interested in finding the "best possible" XP algorithm, and that is what this subprogram is about: the tools that have been forged in the parameterized complexity framework are appropriate for exploring such issues. In some sense, this is a *sophisticated refinement* of the issue at play in Subprogram 1 (the origins of the field). The surprising thing is that this subprogram seems to be mathematically doable. A landmark investigation in this direction is the recent paper of Marx on the CLOSEST SUBSTRING problem [Marx05]. The foundations for this subprogram are discussed in the survey by Jianer Chen *et al.* A typical open problem for this area arises from the work of Goldschmidt and Hochbaum, who showed that the GRAPH k-CUT problem can be solved in time $O(n^{k^2})$ [GH88]. This shows that the problem is in XP. GRAPH k-CUT is hard for $W[1]$, so it is impossible to get k out of the exponent. The challenge here is either to prove that there is no $O(n^{o(k^2)})$ algorithm for GRAPH k-CUT unless FPT = M[1] (the "optimality" we seek), or else to improve the XP algorithm for this problem to one that can be shown optimal.

7. The Subprogram of "The Kernelization Game." There are two different ways of exploring the issue of improved FPT algorithms. The most obvious one, and the one that has so far garnered the most attention, is "The $f(k)$ Game" (Subprogram 3). However, *there is a completely different way* of considering the fundamental notion of fixed-parameter tractability. As discussed in the survey by Niedermeier *et al.*, in this collection, a parameterized problem is in FPT if and only if there is a *polynomial time* (that is, running in time polynomial in both the input size n and the parameter k) that transforms the original input (x, k) to (x', k') such that: (1) (x, k) is a yes-instance if and only if (x', k') is a yes-instance, (2) $k' \leq k$, and (3) $|(x', k')| \leq g(k)$. That is, a problem is in FPT if and only if, we can in polynomial time *pre-process* or *kernelize* the input to something of size bounded by a function of k only. The proof is quite trivial; take $g(k) = f(k)$ for an FPT problem solvable in time $f(k)n^c$. However, this perspective leads to the quest for polynomial time kernelization algorithms yielding the best possible bounding function $g(k)$, a completely different game from the "$f(k)$ game" of subprogram 3. In recent years it has become clear that there are unexpectedly rich mathematical opportunities and strong practical payoffs from research in this direction.

The payoffs come from the fact that this subprogram is turning up sophisticated preprocessing algorithms (that run in polynomial time!), and usually these routines *lose no information*, and hence are of practical value even when the parameter k is *not* small. The

example discussed above concerning Karsten Weihe's Train Problem shows how for real-world instances, reduction rules can be very effective in reducing the complexity of a problem, because kernelization rules can cascade. Again, *the race is on!* In 2006 there was a major breakthrough showing a kernelization for FEEDBACK VERTEX SET to a problem kernel on $O(k^{11})$ vertices (a $Poly(k)$ kernelization). This was quickly improved by Bodlaender and van Dijk, who gave an algorithm kernelizing to $O(k^3)$ vertices, and demonstrated with an implementation its practical utility [BvD07]. Can this be further improved? Does the DIRECTED FEEDBACK VERTEX SET problem (recently shown to be FPT) have a $Poly(k)$ kernelization? A number of concrete challenges of this kind are discussed in the survey of Hüffner, Niedermeier and Wernicke.

8. Kernelization Lower Bounds. One of the big surprises about FPT is that so many natural problems fall into what one would expect to be highly restrictive subclasses of FPT, having, respectively, P-time kernelization algorithms to linear-in-k and polynomial-in-k kernels:

$$Lin(k) \subseteq Poly(k) \subseteq FPT$$

On the other hand, there are still many FPT problems that are not known to have $Poly(k)$ kernelization, such as the CLIQUE COVER problem (covering the edges of a graph with at most k cliques). This subprogram is concerned with lower bound techniques for kernel sizes. Recently, there have been major developments that have opened up this direction for concrete exploration. Combining recent work of Bodlaender, Downey, Fellows and Hermelin [BDFH07] and Fortnow and Santhanam [FS07], it is now known that, for example, the FPT problem of determining whether a graph has a path of length at most k, cannot be in $Poly(k)$ unless the polynomial hierarchy collapses to the third level. Whether CLIQUE COVER is in $Poly(k)$ is still unknown, as are many other concrete issues in this new direction. These frontiers are discussed in the survey by Hüffner *et al.*

9. FPT Approximation. The study of specifically parameterized forms of approximation has recently opened up in a productive way. For example, Grohe and Grüber [GG07] have recently shown that while determining if a directed graph has at least k vertex-disjoint directed cycles is hard for $W[1]$, one can in FPT time either determine that k are impossible, or find at least k' ($<< k$) disjoint cycles, where $k' = h(k)$ for some univariate function h. In the case of the $W[2]$-hard INDEPENDENT DOMINATING SET problem, Downey *et al.* [DFMc06] showed a contrasting outcome: no such FPT approximation algorithm is possible for INDEPENDENT DOMINATING SET unless FPT = W[2]. For many W-hard problems the issue is unresolved. This new area of research is discussed in the survey by Daniel Marx.

10. k-Step Local Search. Local search heuristics for hard problems utilize a polynomial-sized *neighborhood* $N(S)$ of a solution S, and proceed by exhaustively searching among $S' \in N(S)$, and "moving" to S' if S' is a better solution than S. The parameter can model the possibility of *speeding up* local search, by collapsing k steps into a single giant step. If $|N(S)| = n^c$, then a giant-step neighborhood of S of size $O(n^{ck})$ must be searched. This can always be done exhaustively in XP time, but when it can be done in FPT time there may be a substantial practical payoff. This subprogram is relevant to any problem amenable to local search (i.e., just about everything). Initial results, both positive, such as the results

of Khuller *et al.* [KBP03] giving an FPT k-local search algorithm for an important network problem, and negative, such as the future-classic result of Marx showing that searching the Traveling Salesman Problem k-change neighborhood for an improved solution (even assuming the Triangle Inequality) is hard for $W[1]$ [Marx08], make this an exciting frontier for potentially very useful applications. Is k-local search FPT for the Euclidean TSP problem?

3 The Future

One can discern a number of themes emerging from recent research in parameterized complexity and algorithmics. One of these is *creative deployments of parameterization*. In the end, what the theory offers is mathematical machinery that "mediates" between the bivariate function classes FPT and XP. Put simply: the parameter can be put to work wherever the contrast between these two classes can model something interesting. From this theme has emerged subprograms 2 (where the parameter models goodness of approximation) and 10 (where the parameter models the size of a giant step for local search). Other new deployments are on the horizon. Liming Cai has recently introduced the notion of an *operational parameterization* of the internal functioning of an algorithm, a subject discussed in the survey by Cai *et al.* in this collection.

Another theme is the development of new lower bound methods, such as in subprogram 5 (FPT optimality), subprogram 6 (XP optimality) and subprogram 9 (FPT approximation). There is a need for further new tools of this sort. For example, it can be shown that the "$f(k)$ game" for Vertex Cover cannot go on *forever* : unless FPT = M[1], there exists some positive constant $\epsilon_{vc} > 0$ such that there is no FPT algorithm for Vertex Cover with a running time of $O^*((1 + \epsilon_{vc})^k)$. Currently we have no quantitative information at all about what this "barrier constant" might be, we only know that it exists. Similarly, Vertex Cover can be kernelized in polynomial time to a graph of minimum degree 4 (in fact, "almost" minimum degree 5). There must be some bound on the minimum degree to which the problem can be kernelized in polynomial time, but we currently lack any methods for establishing such *structural lower bounds* for kernelization.

Another theme that has begun to emerge in parameterized complexity research is the point of view that regards an FPT parameter as synonymous with an *effective structure theory* that can be exploited in a variety of ways. In this regard, the theory of graph minors (which played a major role in motivating the field of parameterized complexity) is viewed as the structure theory associated with the FPT problem of Minor Testing.

Finally, there are still important empirical beacons to research in the field that are simply not well understood. The Train Problem mentioned above is a nice example. Clearly, the train system graph has some kind of structure, that the two kernelization rules are able to exploit quite powerfully. But the problem itself is both NP-hard and W-hard. But *what is* that structure, that allows the problem to be solved exactly in practice for these huge

graphs? Adding interest to this question is the fact the same structure turns up elsewhere, such as in the FEATURE SELECTION problem for biomedical microarray datasets. We still do not know what the relevant structural parameter is, and we do not have a satisfying theoretical picture of this general practical-algorithmics phenomenon.

The field seems to have a very big future. Please enjoy these surveys of its basic ideas and concepts, significant applications and many frontiers.

References

[Ar96] S. Arora. Polynomial time approximation schemes for euclidean TSP and other geometric problems. In: *Proceedings of the 37th IEEE Symposium on Foundations of Computer Science*, 1996, pp. 2–12.

[Ar97] S. Arora. Nearly linear time approximation schemes for euclidean TSP and other geometric problems. *Proc. 38th Annual IEEE Symposium on the Foundations of Computing* (FOCS'97), IEEE Press (1997), 554-563.

[BDFH07] H. L. Bodlaender, R. Downey, M. Fellows and D. Hermelin. On problems without polynomial kernels. Manuscript, 2007.

[BvD07] H. L. Bodlaender and T. C. van Dijk. A cubic kernel for feedback vertex set and loop cutset. Manuscript, 2007.

[CJ03] L. Cai and D. Juedes. On the existence of subexponential parameterized algorithms. *Journal of Computer and System Sciences* 64 (2003), 789-807.

[CKX06] J. Chen, I. Kanj and G. Xia. Improved parameterized upper bounds for vertex cover. In *Proceedings of the 31st International Symposium on Mathematical Foundations of Computer Science (MFCS'06)*, Springer-Verlag, *Lecture Notes in Computer Science* 4162 (2006), 238-249.

[DFMc06] R. G. Downey, M. R. Fellows, C. McCartin. Parameterized approximation problems. *Proc. IWPEC 2006* Springer-Verlag, *Lecture Notes in Computer Science* 4169 (2006) 121–129.

[FS07] L. Fortnow and R. Santhanam. Infeasibility of instance compression and succinct PCP's for NP. Report 96, *Electronic Colloquium on Computational Complexity*, 2007.

[GG07] M. Grohe and M. Grüber. Parameterized approximability of the disjoint cycle problem. *Proceedings of ICALP '07* (2007), 363–374.

[GH88] O. Goldschmidt and D. Hochbaum. Polynomial algorithm for the k-cut problem. *Proceedings of the IEEE Symposium on the Foundations of Computer Science (FOCS '88)*, IEEE Press (1988), 444–451.

[HK62] M. Held and R. Karp. A dynamic programming approach to sequencing problems. *J. SIAM* (1962), pp. 196–210.

[HM91] F. Henglein and H. G. Mairson. The complexity of type inference for higher-order typed lambda calculi. In *Proc. Symp. on Principles of Programming Languages (POPL)* (1991), 119-130.

[IPZ98] R. Impagliazzo, R. Paturi and F. Zane. Which problems have strongly exponential complexity? *Proceedings of the 39th Annual Symposium on Foundations of Computer Science (FOCS'1998)*, 653–663.

[KBP03] S. Khuller, R. Bhatia and R. Pless. Placement of meters in networks. *SIAM J. Computing* 32 (2003) 470–487.

[LP85] O. Lichtenstein and A. Pneuli. Checking that finite-state concurrent programs satisfy their linear specification. In: *Proceedings of the 12th ACM Symposium on Principles of Programming Languages* (1985), 97–107.

[Marx05] D. Marx. The closest substring problem with small distances. *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)* (2005), 63-72.

[Marx06] D. Marx. Parameterized complexity of independence and domination on geometric graphs. *Proceedings of the International Workshop on Parameterized and Exact Computation (IWPEC 2006)*, Springer-Verlag, *Lecture Notes in Computer Science* 4169 (2006), 154-165.

[Marx08] D. Marx. Searching the k-change neighborhood for TSP is W[1]-hard. To appear in *Operations Research Letters*.

[Wei98] K. Weihe. Covering trains by stations, or the power of data reduction. *Proc. ALEX'98* (1998), 1–8.

[Wei00] K. Weihe. On the differences between 'practical' and 'applied'. *Proc. WAE 2000*, Springer-Verlag, *Lecture Notes in Computer Science* 1982 (2001), 1–10.

[Woe03] G. J. Woeginger. Exact algorithms for NP-hard problems: a survey. *Proceedings of 5th International Workshop on Combinatorial Optimization-Eureka, You Shrink! Papers dedicated to Jack Edmonds,* M. Junger, G. Reinelt, and G. Rinaldi (Festschrift Eds.) Springer-Verlag, *Lecture Notes in Computer Science* 2570 (2003), 184-207.