Blow-Ups, Win/Win's, and Crown Rules: Some New Directions in *FPT*

Michael R. Fellows

School of Electrical Engineering and Computer Science
University of Newcastle, University Drive, Callaghan NSW 2308, Australia
mfellows@cs.newcastle.edu.au

Abstract. This survey reviews the basic notions of parameterized complexity, and describes some new approaches to designing *FPT* algorithms and problem reductions for graph problems.

1 Introduction — the Basic Ideas

The collection of methods for classifying problems as fixed-parameter tractable, for designing *FPT* algorithms, for designing *better* FPT algorithms and transfering these results to practical implementations, and for describing *FPT* problem transformations to prove lower bound and intractability results, has developed with surprising vigor — yet it still seems we are far from having the "basic vocabulary" of effective FPT algorithm design techniques worked out.

Most parameterized problems in *FPT* have seen a trajectory of improvements through many approaches. An example is the following parameterization of the Max Leaf Spanning Tree problem, defined by declaring: (1) the input to the problem, (2) the parameter, and (3) the question.

Max Leaf Spanning Tree I
Input: A graph G and a positive integer k; **Parameter:** k; **Question:** Does G have a spanning tree with at least k leaves?

The parameter can be *anything*. A different parameterization is:

Max Leaf Spanning Tree II
Input: A graph G and a positive integer r; **Parameter:** The treewidth of G; **Question:** Does G have a spanning tree with at least r leaves?

The parameter doesn't have to be small to be interesting:

Max Leaf Spanning Tree III
Input: A graph G and a positive integer r; **Parameter:** The number of vertices of G; **Question:** Does G have a spanning tree with at least r leaves?

The parameter may be geared to any kind of mathematical insight:

Max Leaf Spanning Tree IV
Input: A graph G and a positive integer k; **Parameter:** k; **Output:** A spanning tree for G having a number of leaves that is within a factor of $(1 + 1/k)$ of the maximum number possible.

Parameterizing on the number of leaves can be done in more than one way:

MAX LEAF SPANNING TREE V
Input: A graph G on n vertices and a positive integer k; **Parameter:** k; **Question:** Does G have a spanning tree with at least $n - k$ leaves?

The trajectory for MAX LEAF SPANNING TREE I includes:
• A quadratic *FPT* algorithm based on the Graph Minor Theorem [FL92] having a ridiculously fast-growing parameter function.
• The first linear time *FPT* algorithm for the problem due to Bodlaender had a parameter function of $f(k) = (17k^4)!$ or thereabouts [Bod93]. This was based on depth-first search and bounded treewidth.
• A linear time *FPT* algorithm by Downey-Fellows with $f(k) = (2k)^{4k}$, based on a quadratic (in k) problem kernelization [DF95a].
• An improved linear time *FPT* algorithm due to Fellows, McCartin, Rosamond and Stege [FMRS00] with parameter function $f(k) = (14.23)^k$, based on linear kernelization and catalytic branching.
• The current best *FPT* algorithm for the problem, due to Bonsma, Brüggemann and Woeginger, running in time $O(n^3 + 9.4815^k k^3)$, based on a reduction to a problem kernel of size bounded by $4k$, using a substantial result from extremal graph theory [BBW03].

The parameterized complexity framework is a two-dimensional framework for complexity analysis, where the first dimension is the overall imput size n. The parameter k represents some other aspect(s) of the computational problem.

Definition 1. *A parameterized language L is a subset $L \subseteq \Sigma^* \times \Sigma^*$. For $(x, k) \in L$ we refer to k as the parameter.*

Definition 2. *A parameterized language L is fixed-parameter tractable (FPT) if it can be determined in time $f(k)q(n)$ whether $(x, k) \in L$, where $n = |(x, k)|$, $q(n)$ is a polynomial in n, and f is a computable function (otherwise unrestricted).*

These central ideas recently have been reintroduced by Impagliazzo, Paturi and Zane [IPZ98,Woe03], who propose a new notation for *FPT* results. If a parameterized problem is in *FPT* via an algorithm with a running time of $f(k)q(n)$, they write this as $O^*(f(k))$, focusing attention on the exponential complexity contribution of the parameter.

Definition 3. *A parameterized language L is many:1 parametrically reducible to a parameterized language L' if there is an FPT algorithm that transforms (x, k) into (x', k') so that: (1) $(x, k) \in L$ if and only if $(x', k') \in L'$, and (2) $k' = g(k)$ (for some recursive function g).*

The main parameterized classes are organized in the hierarchy:

$$FPT \subseteq M[1] \subseteq W[1] \subseteq W[2] \subseteq \cdots \subseteq W[P] \subseteq AW[P] \subseteq XP$$

XP is the class of parameterized problems solvable in time $f(k)n^{g(k)}$, where f and g are unrestricted functions.

The CLIQUE problem (parameterized by the size of the clique) is $W[1]$-complete [DF99], and is a frequent source problem for $W[1]$-hardness reductions. $W[1]$ is strongly analogous to NP because the k-STEP HALTING PROBLEM FOR NONDETERMINISTIC TURING MACHINES (with unlimited nondeterminism) is complete for $W[1]$ ([DF95b] + [CCDF97]).

What is known about the five different parameterizations of the classic MAX LEAF SPANNING TREE problem?

About Max Leaf I and Related Problems. This has an *FPT* algorithm with running time $O^*(9.4815^k)$ [BBW03]. Where will the cascade of progress on this problem stop?

There are *two main games* being played in *FPT* algorithms research:
(1) **The $f(k)$ game.** This is the game emphasized by the $O^*(f(k))$ notation. Improvements generally depend on three things, in some mix:
• More powerful preprocessing/kernelization data reduction rules.
• Better concrete branching strategies for the exhaustive analysis of the kernel.
• More sophisticated methods of analysis.
(2) **The kernelization game.** The goal is to produce smaller problem kernels. Success depends on a mix of the first and third of the above.

The PLANAR DOMINATING SET problem provides some examples. An $O^*(11^k)$ *FPT* algorithm was described in [DF99]. This was improved to $O^*(8^k)$ by Alber, et al. [AFFFNRS01]. The algorithm did not change, only the analysis. By more powerful reduction rules, a linear problem kernel was proved in [AFN02] — but this does not yield any improvement in the $f(k)$ game over the $O^*(8^k)$ algorithm.

Both games are worth playing — both lead to better understanding of the *problem structure* afforded by the fixed parameter value — both may lead to new strategies for practical implementations [ALSS03,DRST02].

About Max Leaf Spanning Tree II and related problems. Here the parameter is the treewidth of G. We know by Bodlaender's $O^*(2^{35k^3})$ algorithm for structural parsing of graphs of treewidth at most k [Bod96], together with dynamic programming, that this parameterization is *FPT*. Treewidth is an almost universal parameter leading to *FPT* algorithms.

About Max Leaf Spanning Tree III and related problems. Here the parameter k is the number of vertices of G. This is clearly in *FPT* by brute force! and might not seem interesting. But it is, because there are new "lower bound" techniques for proving qualitative optimality results for *FPT* algorithms. Combining results of [CJ01] and [DFPR03], it is now known that while VERTEX COVER admits an $O^*(2^{O(k)})$ *FPT* algorithm (where the parameter k, as here, is the number of vertices), this cannot be improved to $O^*(2^{o(k)})$ unless $FPT = M[1]$. A similar result holds for MAX LEAF SPANNING TREE III. By parameterizing on such things as the number of vertices, these new methods allow us to prove lower bound results in the area of "worst-case exponential" algorithms.

About Max Leaf Spanning Tree IV and related problems. Here we are parameterizing by the goodness of approximation. If we had an *FPT* result for this problem, then we would have a special kind of PTAS (an EPTAS [CT97])

where the exponent of the polynomial does not depend on the goodness of the approximation.

Recent PTAS results include:

- The PTAS for the EUCLIDEAN TSP due to Arora [Ar96] has a running time of around $O(n^{300/\epsilon})$. Thus for a 20% error, we have a "polynomial-time" algorithm that runs in time $O(n^{1500})$.
- The PTAS for the MULTIPLE KNAPSACK problem due to Chekuri and Khanna [CK00] has a running time of $O(n^{12(\log(1/\epsilon)/\epsilon^8)})$. Thus for a 20% error we have a "polynomial-time" algorithm that runs in time $O(n^{9375000})$.
- The PTAS for the MAXIMUM SUBFOREST PROBLEM due to Shamir and Tsur [ST98] has a running time of $O(n^{2^{2^{1/\epsilon}}-1})$. For a 20% error we thus have a "polynomial" running time of $O(n^{958267391})$.
- The PTAS for the MAXIMUM INDENDENT SET problem on geometric graphs due to Erlebach, Jansen and Seidel [EJS01] has a running time of $O(n^{(4/\pi)(1/\epsilon^2+2)^2(1/\epsilon^2+1)^2})$. Thus for a 20% error, $O(n^{532804})$.

Viewed in the parameterized framework, these results only establish that the analogs of MAX LEAF SPANNING TREE IV are in *XP*. In the case of the EUCLIDEAN TSP problem, Arora later found an *FPT* algorithm (an EPTAS) [Ar97]. Whether the other problems listed above admit *FPT* algorithms, or are $W[1]$-hard, is a significant open problem. MAX LEAF SPANNING TREE IV turns out to be hard for $W[P]$.

About Max Leaf Spanning Tree V and related problems. This is an example of a *dual parameterization*, a notion introduced by Khot and Raman [KR00]. Note that a graph has a spanning tree with at least $n - k$ leaves if and only if has a connected dominating set having at most k vertices. This problem turns out to be complete for $W[2]$. Khot and Raman noticed that quite frequently "dual" parameterized problems behave oppositely, with one being *FPT* and the other hard for $W[1]$. Exceptions are now known, but this seems to be a strong tendency for natural problems in NP. CLIQUE and VERTEX COVER (parameterized by the number of vertices in the solution), are dual, with the former being $W[1]$-complete and the latter being *FPT*. DOMINATING SET is $W[2]$-complete, while its parametric dual is in *FPT*.

2 M[1] and Blow-Ups

There is a new class of parameterized problems seemingly intermediate between *FPT* and $W[1]$:

$$FPT \subseteq M[1] \subseteq W[1]$$

$M[1]$ may turn out to be a better primary reference point for intractability than $W[1]$. There are two natural "routes" to $M[1]$.

The renormalization route to $M[1]$.
There are $O^*(2^{O(k)})$ *FPT* algorithms for many parameterized problems, such as VERTEX COVER. In view of this, we can "renormalize" and define the problem:

$k \log n$ Vertex Cover
Input: A graph G on n vertices and an integer k; **Parameter:** k; **Question:** Does G have a vertex cover of size at most $k \log n$?

The FPT algorithm for the original Vertex Cover problem, parameterized by the number of vertices in the vertex cover, allows us to place this new problem in XP. It now makes sense to ask whether the $k \log n$ Vertex Cover problem is also in FPT — or is it parametrically intractable? It turns out that $k \log n$ Vertex Cover is $M[1]$-complete.

The miniaturization route to $M[1]$.
We certainly know an algorithm to solve n-variable 3SAT in time $O(2^n)$. Consider the following parameterized problem.

Mini-3SAT
Input: Positive integers k and n in unary, and a 3SAT expression E having at most $k \log n$ variables; **Parameter:** k; **Question:** Is E satisfiable?

Using our exponential time algorithm for 3SAT, Mini-3SAT is in XP and we can wonder where it belongs — is it in FPT or is it parametrically intractable? This problem also turns out to be complete for $M[1]$.

Dozens of renormalized FPT problems and miniaturized arbitrary problems are now known to be $M[1]$-complete [DFPR03]. However, what is known is quite problem-specific. For example, one might expect Mini-Max Leaf to be $M[1]$-complete, but all that is known presently is that it is $M[1]$-hard. It is not known to be $W[1]$-hard, nor is it known to belong to $W[1]$.

The following theorem would be interpreted by most people as indicating that probably $FPT \neq M[1]$. (The theorem is essentially due to Cai and Juedes [CJ01], making use of a result of Impagliazzo, Paturi and Zane [IPZ98].)

Theorem 1. *$FPT = M[1]$ if and only if n-variable 3SAT can be solved in time $2^{o(n)}$.*

$M[1]$ supports convenient although unusual combinatorics. For example, one of the problems that is $M[1]$-complete is the miniature of the Independent Set problem defined as follows.

Mini-Independent Set
Input: Positive integers k and n in unary, a positive integer $r \leq n$, and a graph G having at most $k \log n$ vertices.
Parameter: k
Question: Does G have an independent set of size at least r?

The following example of a **blow-up reduction** (from [DEFPR03]) displays some of the combinatorial possibilities.

Theorem 2. *There is an FPT reduction from* Mini-Independent Set *to ordinary parameterized* Independent Set *(parameterized by the number of vertices in the independent set).*

Proof. Let $G = (V, E)$ be the miniature, for which we wish to determine whether G has an independent set of size r. Here, of course, $|V| \leq k \log n$ and we may

regard the vertices of G as organized in k blocks $V_1, ..., V_k$ of size $\log n$. We now employ a simple but useful *counting trick* that can be used when reducing miniatures to "normal" parameterized problems. Our reduction is a Turing reduction, with one branch for each possible way of writing r as a sum of k terms, $r = r_1 + \cdots + r_k$, where each r_i is bounded by $\log n$. The reader can verify that $(\log n)^k$ is an FPT function, and thus that there are an allowed number of branches. A branch represents a commitment to choose r_i vertices from block V_i (for each i) to be in the independent set.

We now produce (for a given branch of the Turing reduction) a graph G' that has an independent set of size k if and only if the miniature G has an independent set of size r, distributed as indicated by the commitment made on that branch. The graph G' consists of k cliques, together with some edges between these cliques. The ith clique consists of vertices in 1:1 correspondence with the subsets of V_i of size r_i. An edge connects a vertex x in the ith clique and a vertex y in the jth clique if and only if there is a vertex u in the subset $S_x \subseteq V_i$ represented by x, and a vertex v in the subset $S_y \subseteq V_j$ represented by y, such that $uv \in E$. Verification is straightforward.

The hypothesis that $FPT \neq M[1]$ has many interesting consequences based on similar arguments. The next theorem is due to recent work of Chor, Fellows and Juedes.

Theorem 3. *If $FPT \neq M[1]$ then determining whether a graph has a k-element dominating set (k-element independent set) cannot be solved in time $O(n^{o(k)})$.*

Cai and Juedes [CJ01] proved the following path-breaking results, which established an ***FPT* optimality program** of broad applicability.

Theorem 4. *If $FPT \neq M[1]$ then there cannot be an FPT algorithm for the general* Vertex Cover *problem with a parameter function of the form $f(k) = 2^{o(k)}$, and there cannot be an FPT algorithm for the* Planar Vertex Cover *problem with a parameter function of the form $f(k) = 2^{o(\sqrt{k})}$.*

It has previously been known that Planar Dominating Set, parameterized by the number n of vertices in the graph can be solved optimally in time $O^*(2^{O(\sqrt{n})})$ by using the Lipton-Tarjan Planar Separator Theorem. Combining the lower bound theorem of Cai-Juedes with the linear kernelization result of Alber et al. [AFN02] shows that this cannot be improved to $O^*(2^{o(\sqrt{n})})$ unless $FPT = M[1]$.

3 Win/Win's and Vertex Cover Structure

Early work on parameterized complexity was motivated by the complexity consequences of the Graph Minor Theorem, that povides a powerful way to classify problems as FPT. It applies to the following problem:

MAX INTERNAL SPANNING TREE
Input: A graph G and an integer k; **Parameter:** k; **Question:** Does G have a spanning tree with at least k internal vertices?

This problem is "governed by bounded vertex cover structure" and in this setting there is a more general (in this setting) and much easier to prove classification tool.

Theorem 5. *For every fixed k, the family of graphs having a vertex cover of size at most k is well-quasiordered by ordinary subgraphs. Furthermore, restricting to this family of graphs, the* SUBGRAPH *problem (Is H a subgraph of G?), parameterized by H, is linear time FPT.*

Proof sketch. Suppose there is an infinite bad sequence of graphs of bounded vertex cover number. Then there is an infinite bad subsequence all having the same vertex cover number, which we can assume to be k. Order the k vertices of the vertex cover arbitrarily in any order for each graph. There are finitely many graphs on k vertices, so there is an infinite bad subsequence where, respecting vertex orderings, the subgraphs induced by the vertex covers are isomorphic. Every other vertex of a graph in this bad subsequence is attached to a subset of the vertex cover, and has no other incident edges. Thus each graph in the infinite bad subsequence can be described by a *census vector* of length 2^k that counts, for each subset S of the vertex cover, the number of vertices u in the complement of the vertex cover such that $N(u) = S$. By Higman's Lemma, these census vectors are well-quasi-ordered by the pointwise ordering of the census numbers. By the well-quasi-ordering of the census vectors, we reach a contradiction. Also, graphs that have a vertex cover of size k have pathwidth at most k.

As an application, we can consider the following problem, the parametric dual of GRAPH k-COLORING:

SAVING k COLORS
Input: A graph G and a positive integer k; **Parameter:** k; **Question:** Can G be colored with $n - k$ colors?

Theorem 6. SAVING k COLORS *is in FPT.*

Proof. For each fixed k, the graph complements of the no-instances constitute a lower ideal in the subgraph order. Theorem 5 therefore applies.

The essence of the **win/win** strategy in *FPT* algorithm design is to play off one kind of structure against another. An old example from [FL92]:

Theorem 7. *In time $O(n)$ we can find either:*
(1) A cycle of length at least k, or,
(2) A tree decomposition of width at most k.

This is just a plain linear time algorithm, there is nothing exponential in k hiding in the big O. Thus if we are interested in determining whether a graph has

a cycle of length at least k, either we are done, or "for free" we have a bounded width tree decomposition.

A Win/Win for Max Internal.
A win/win approach to designing an *FPT* algorithm for MAX INTERNAL SPANNING TREE has recently been described by Prieto and Sloper [PS03]. They describe an algorithm that in time $O(n^2)$ produces either: (1) a vertex cover of size at most k, or (2) a k-internal spanning tree. The rest of their *FPT* algorithm is based on a kernelization rule that can be applied in the situation where there is a small vertex cover.

A Win/Win for The Dual of Dominating Set.
The complement of a dominating set of size $n - k$ is termed a *nonblocking* set of vertices (of size k). Faisal Abu-Khazm has described a simple algorithm that in time $O(n^2)$ produces either: (1) a vertex cover of size at most k, or (2) a nonblocking set of size k. If the outcome is (2), then we are done. If the outcome is (1), then in $O(n)$ time we can read off a path decomposition of width k as follows. Let C denote the vertex cover, and let v_1, v_2,..., be the remaining vertices in any order. Then we can take the sequence of bags of the decomposition to be: $C \cup \{v_1\}, C \cup \{v_2\}, ...$ Given this path decomposition, we can use the efficient dynamic programming algorithm of Proskurowski and Telle [PT93] for the MINIMUM DOMINATING SET problem, further improved by Alber and Niedermeier [AN02], to get an $O^*(4^k)$ *FPT* algorithm for the problem. (This matches the running time of McCartin's algorithm [McC03], obtained by a completely different route.)

3.1 Crown Rules: A Surprising New Technique for Kernelization

Consider the VERTEX COVER problem, and the reduction rule for this problem for vertices of degree 1. If (G, k) is an instance of the VERTEX COVER problem where G has a vertex v of degree 1 with neighbor $u \neq v$, then we can reduce to the instance (G', k') where $G' = G - u - v$ and $k' = k - 1$. This simple *local* reduction rule admits a global structural generalization.

Definition 4. *A* crown decomposition *of G is a partition of the vertex set of G into three sets, H, C and J such that the following conditions hold:*
(1) C is an independent set,
(2) H is matched into C, and
(3) H is a cutset, i.e., there are no edges between C and J.

If (G, k) is an instance of the VERTEX COVER problem, and G admits a crown decomposition, then we can reduce to (G', k') where $G' = G - H - C$ and $k' = k - |H|$, as a generalization of the degree 1 reduction rule. Presently, it is an open problem whether determining if a graph has a crown decomposition is in P, or is NP-complete. However, if we are given: (1) a graph $G = (V, E)$ without isolated vertices, and (2) a vertex cover V' for G that satifies the inequality $2|V'| < |V|$ (in other words, the vertex cover is less than half the size of G), then we can compute a crown decomposition in polynomial time as follows. Let

$V'' = V - V'$, and compute a maximum matching between V' and V''. For $x \in V'$, let $m(x) \in V''$ denote the vertex of V'' (if any) to which x is matched. For $U \subseteq V'$, define $m(U) = \{v \in V'' : \exists x \in U \text{ with } m(x) = v\}$. For $W \subseteq V''$ let $N(W) = \{x \in V' : \exists y \in W, xy \in E\}$. Let $C_0 = V'' - m(V')$. The size condition on the vertex cover V' insures that C_0 is nonempty. Note that since V' is a vertex cover, V'' is an independent set in G.

Now we iteratively compute:
$H_1 = N(C_0)$, $C_1 = m(H_1) \cup C_0$,
$H_2 = N(C_1)$, $C_2 = m(H_2) \cup C_1$,
...
until a fixed point is reached in two final sets H and C. Now take $J = V - C - H$. It is easy to verify (using the fact that m is a *maximum* matching) that we have a nontrivial crown decomposition of G — and thus any instance (G, k) of the VERTEX COVER problem can be reduced by our *crown reduction rule*. It seems that many parameterized problems that are "subject to vertex cover structure" have reduction rules that are flavors of this approach.

In particular, we can achieve a kernel of size at most $3k$ for SAVING k COLORS problem by using crown reduction kernelization [CFJ03]. If determining whether a graph has a nontrivial crown decomposition is in P, then crown reductions provide a new way of producing a $2k$ kernel for VERTEX COVER that is different from (and orthogonal to) the Nemhauser-Trotter algorithm. (If it is NP-hard, then we can still get a $3k$ kernel from crown reductions.)

Since the WG Workshop, a way has been found to apply crown reductions together with another new and widely applicable technique called **greedy localization**, introduced by Chen et al. [JZC03] to deliver an improved FPT algorithm for the SET SPLITTING problem. Elsewhere in this volume an $O^*(72^k)$ FPT algorithm for the SET SPLITTING problem is described by Dehne, Fellows and Rosamond [DFR03]. Using greedy localization + crown reduction, this can be improved to $O^*(8^k)$. (The details will have to be described elsewhere.)

Data reduction and kernelization rules are one of the primary outcomes of research on parameterized complexity. After all, whether k is small or not — and no matter if you are going to do simulated annealing eventually — it *always* makes sense to simplify and reduce (pre-process) your input! This humble activity, because reduction rules frequently cascade, can sometimes lead to unexpected success against hard problems on real input distributions. A fine example of this has been described by Weihe [Wei98].

This survey has only scratched the surface of a few new ideas and developments in FPT algorithmic techniques. The currently best available comprehensive survey is the recent Habilitationschrift of Rolf Niedermeier, which is highly recommended [Nie02].

References

[AFN02] J. Alber, M. Fellows and R. Niedermeier. "Efficient Data Reduction for Dominating Set: A Linear Problem Kernel for the Planar Case." *Proceedings of Scandi-*

navian Workshop on Algorithms and Theory (SWAT 2002), Springer-Verlag, *Lecture Notes in Computer Science* 2368 (2002), 150–159.

[AFFFNRS01] J. Alber, H. Fan, M. Fellows, H. Fernau, R. Niedermeier, F. Rosamond and U. Stege. "Refined Search Tree Techniques for the Planar Dominating Set Problem," *Proc. 26th International Symposium on Mathematical Foundations of Computer Science (MFCS 2001)*, Springer-Verlag, *Lecture Notes in Computer Science* 2136 (2001), 111-122.

[ALSS03] F. Abu-Khzam, M. A. Langston, P. Shanbhag and C. T. Symons. "High Performance Tools for Fixed-Parameter Tractable Implementations." WADS'03 Workshop Presentation, 2003.

[AN02] J. Alber and R. Niedermeier. "Improved Tree Decomposition Based Algorithms for Domination-Like Problems," *Proceedings of the 5th Latin American Theoretical IN-formatics (LATIN 2002)*, Springer-Verlag LNCS 2286 (2002), 613–627.

[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), 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.

[BBW03] P.S. Bonsma, T. Brüggemann and G.J. Woeginger. A faster *FPT* algorithm for finding spanning trees with many leaves. Manuscript, 2003.

[Bod93] H. L. Bodlaender. "On Linear Time Minor Tests and Depth-First Search." *Journal of Algorithms* 14 (1993), 1–23.

[Bod96] H. L. Bodlaender. "A Linear Time Algorithm for Finding Tree-Decompositions of Small Treewidth." *SIAM Journal on Computing* 25 (1996), 1305–1317.

[CCDF97] L. Cai, J. Chen, R. Downey and M. Fellows. "On the Parameterized Complexity of Short Computation and Factorization." *Archive for Mathematical Logic* 36 (1997), 321–337.

[CFJ03] B. Chor, M. Fellows and D. Juedes. "An Efficient *FPT* Algorithm for Saving k Colors." Manuscript, 2003.

[CJ01] L. Cai and D. Juedes. "On the Existence of Subexponential Parameterized Algorithms," manuscript, 2001. Revised version of the paper, "Subexponential Parameterized Algorithms Collapse the W-Hierarchy," in: *Proceedings 28th ICALP*, Springer-Verlag LNCS 2076 (2001), 273–284. (The conference version contains some major flaws.)

[CK00] C. Chekuri and S. Khanna. "A PTAS for the Multiple Knapsack Problem." *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms* (SODA 2000), pp. 213-222.

[CT97] M. Cesati and L. Trevisan. "On the Efficiency of Polynomial Time Approximation Schemes." *Information Processing Letters* 64 (1997), 165–171.

[DEFPR03] R. Downey, V. Estivill-Castro, M. Fellows, E. Prieto-Rodriguez and F. Rosamond. "Cutting Up is Hard to Do: the Parameterized Complexity of k-Cut and Related Problems." *Electronic Notes in Theoretical Computer Science* 78 (2003), 205–218.

[DF95a] R. G. Downey and M. R. Fellows. "Parameterized Computational Feasibility." In: *P. Clote and J. Remmel (eds.), Feasible Mathematics II.* Birkhauser, Boston (1995), 219–244.

[DF95b] R. G. Downey and M. R. Fellows. "Fixed Parameter Tractability and Completeness II: Completeness for $W[1]$." *Theoretical Computer Science* A 141 (1995), 109–131.

[DF99] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer-Verlag, 1999.

[DFPR03] R. Downey, M. Fellows, E. Prieto-Rodriguez and F. Rosamond. "Fixed-parameter Tractability and Completeness V: Parametric Miniatures." Manuscript, 2003.

[DFR03] F. Dehne, M. Fellows and F. Rosamond. "An *FPT* Algorithm for Set Splitting." *Proc. WG'03* (elsewhere in this volume).

[DRST02] F. Dehne, A. Rau-Chaplin, U. Stege and P. Taillon. "Solving Large *FPT* Problems on Coarse Grained Parallel Machines." Manuscript, 2002.

[EJS01] T. Erlebach, K. Jansen and E. Seidel. "Polynomial Time Approximation Schemes for Geometric Graphs." *Proc. ACM Symposium on Discrete Algorithms (SODA'01)*, ACM Press (2001), 671–679.

[FL92] M.R. Fellows and M.A. Langston. "On Well-Partial-Order Theory and its Applications to Combinatorial Problems of VLSI Design." *SIAM Journal on Discrete Mathematics* 5, 117–126.

[FMRS00] M. Fellows, C. McCartin, F. Rosamond and U. Stege. "Coordinatized Kernels and Catalytic Reductions: An Improved *FPT* Algorithm for Max Leaf Spanning Tree and Other Problems." *Proceedings of the 20th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'2000)*, Springer, Lecture Notes in Computer Science 1974 (2000), 240-251.

[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.

[JZC03] W. Jia, C. Zhang and J. Chen. "An Efficient Parameterized Algorithm for Set Splitting." Manuscript, 2003, to appear in *Journal of Algorithms*.

[KR00] S. Khot and V. Raman. "Parameterized Complexity of Finding Subgraphs with Hereditary properties." *Proceedings of the Sixth Annual International Computing and Combinatorics Conference (COCOON 2000)* July 2000, Sydney, Australia, Lecture Notes in Computer Science, Springer-Verlag 1858 (2000), 137-147.

[McC03] C. McCartin. Ph.D. dissertation in Computer Science, Victoria University, Wellington, New Zealand 2003.

[Nie02] R. Niedermeier. "Invitation to Fixed-Parameter Algorithms." Habilitationschrift, University of Tübingen, 2002.

[PS03] E. Prieto and C. Sloper. "Either/Or: Using Vertex Cover Structure in Designing *FPT* Algorithms — the Case of k-Internal Spanning Tree." *Proc. WADS'03*, Springer-Verlag LNCS 2748 (2003), 474–483.

[PT93] J.A. Telle and A. Proskurowski. "Practical Algorithms on Partial k-Trees with an Application to Domination-Like Problems." *Proceedings WADS'93 – The Third Workshop on Algorithms and Data Structures*, Springer-Verlag LNCS 709 (1993), 610–621.

[ST98] R. Shamir and D. Tzur. "The Maximum Subforest Problem: Approximation and Exact Algorithms." *Proc. ACM Symposium on Discrete Algorithms* (SODA'98), ACM Press (1998), 394–399.

[Wei98] K. Weihe, "Covering Trains by Stations, or the Power of Data Reduction," *Proc. ALEX'98* (1998), 1–8.

[Woe03] G. J. Woeginger, "Exact Algorithms for NP-hard Problems: A Survey," manuscript, 2003.