Coordinatized Kernels and Catalytic Reductions: An Improved FPT Algorithm for Max Leaf Spanning Tree and Other Problems

Michael R. Fellows[1], Catherine McCartin[1], Frances A. Rosamond[1], and Ulrike Stege[2⋆]

[1] School of Mathematical and Computing Sciences, Victoria University
Wellington, New Zealand
{Mike.Fellows,Catherine.Mccartin,Fran.Rosamond}@mcs.vuw.ac.nz
[2] Department of Computer Science, University of Victoria
Victoria, B.C., Canada
stege@csr.uvic.ca

Abstract. We describe some new, simple and apparently general methods for designing *FPT* algorithms, and illustrate how these can be used to obtain a significantly improved *FPT* algorithm for the Maximum Leaf Spanning Tree problem. Furthermore, we sketch how the methods can be applied to a number of other well-known problems, including the parametric dual of Dominating Set (also known as Nonblocker), Matrix Domination, Edge Dominating Set, and Feedback Vertex Set for Undirected Graphs. The main payoffs of these new methods are in improved functions $f(k)$ in the *FPT* running times, and in general systematic approaches that seem to apply to a wide variety of problems.

1 Introduction

The investigations on which we report here are carried out in the framework of parameterized complexity, so we will begin by making a few general remarks about this context of our research. The subject is concretely motivated by an abundance of natural examples of two different kinds of complexity behaviour. These include the well-known problems Min Cut Linear Arrangement, Bandwidth, Vertex Cover, and Minimum Dominating Set (for definitions the reader may refer to [GJ79]).

All four of these problems are *NP*-complete, an outcome that is now so routine that we are almost never surprised. In the classical complexity framework that pits polynomial-time solvability against the ubiquitous phenomena of *NP*-hardness, they are therefore indistinguishable. All four of these decision problems take as input a pair consisting of a graph G and a positive integer k. The positive integer k is the *natural parameter* for all four problems, although one might also wish to consider eventually other problem parameterizations, such as treewidth. We have the following contrasting facts:

⋆ Ulrike Stege is supported by the Pacific Institute for the Mathematical Sciences (PIMS), where she is a postdoctoral fellow.

S. Kapoor and S. Prasad (Eds.): FST TCS 2000, LNCS 1974, pp. 240–251, 2000.

1. Min Cut Linear Arrangement and Vertex Cover are solvable in linear time for any fixed k.
2. The best known algorithms for Bandwidth and Minimum Dominating Set are respectively $O(n^k)$ and $O(n^{k+1})$.

In fact, we now have very strong evidence, in the framework of parameterized complexity, that probably Bandwidth and Minimum Dominating Set do not admit algorithms of the qualitative type that Min Cut Linear Arrangement and Vertex Cover have.

1.1 What Is the Nature of This Evidence?

We offer a new view of the study of computability and of its sequel concerns with efficient computability. We motivate how this naturally divides into three main zones of discussion, anchored by variations on the Halting Problem.

In the first zone, we have the unsolvability of the Halting Problem, the unsolvability of other problems following by recursive reductions, and Gödel's Theorem as a corollary.

In the second zone, we have "classical complexity" where the reference problem is the Halting Problem for Nondeterministic P-time Turing Machines. This problem is trivially NP-complete and essentially defines the class NP. Another way to look at this problem is as a generic embodiment of computation that is potentially exponential. If at each nondeterministic step there were two possible choices of transition, then it is a reasonable conjecture that, in general, we will not be able to analyze Turing machines of size n for the possibility of halting in n steps in much less than the $O(2^n)$ time.[1]

The third zone of negotiation with intractability is anchored by the k-step Halting Problem for Nondeterministic Turing Machines (k-NDTM), where in this case we mean Turing machines with an unrestricted alphabet size, and with unrestricted nondeterminism at each step. This is a generic embodiment of n^k computational complexity. For the same reasons as in the second zone of negotiation, we would not expect any method of solving this problem that greatly improves on exhaustively exploring the n-branching depth-k tree of possible computation paths (for a Turing machine of size n).

This leads to the following three basic definitions of the parameterized complexity framework.

Definition 1. *A parameterized language is a subset $L \subseteq \Sigma^* \times \Sigma^*$. For notational convenience, and without any loss of generality, we can also consider that $L \subseteq \Sigma^* \times IN$.*

[1] It would take $O(2^n)$ time to exhaustively explore the possible computation paths — because nondeterministic Turing machines are so unstructured and opaque, i.e., such a generic embodiment of exponential possibility.

Definition 2. *A parameterized language L is fixed-parameter tractable (FPT) if there is an algorithm to determine if $(x,k) \in L$ in time $f(k)+n^c$ where $|x|=n$, c is a constant, and f is a function (unrestricted).*

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. *$k' \leq g(k)$ (where g is an unrestricted function; k' is purely a function of k)*

The analog of *NP* in the third zone of dicussion is the parameterized complexity class $W[1]$ [DF95b]. That the k-NDTM problem is complete for $W[1]$ was proven by Cai, Chen, Downey and Fellows in [CCDF97]. Since Bandwidth and Dominating Set are hard for $W[1]$, we thus have strong natural evidence that they are not fixed-parameter tractable, as Vertex Cover and Min Cut Linear Arrangement are.[2]

1.2 What Is the Current Status of This "Third Zone" of Discussion?

As an example of how useful *FPT* algorithms can be, we know that Vertex Cover can be solved (after several rounds of improvement) in time $O((1.27)^k + n)$ for graphs of size n [CKJ99]. The problem is thus well-solved for $k \leq 100$ analytically. In practice, this worst-case analytical bound appears to be pessimistic. The current best *FPT* algorithms for this problem (which deliver an optimal solution if they terminate) appear to have completely solved the problem for input graphs *in toto*, so long as the parameter value is $k \leq 200$. These new effective algorithms for small ranges of k have applications in computational biology [Ste99].

Another promissing result is the fixed-parameter-tractable algorithm for 3-Hitting-Set presented in [NR00b]. The running time is $O(2.270^k + n)$ where k is the size of the hitting set to determine and n denotes the length of the encoding of the input. As Vertex Cover 3-Hitting Set has applications in computational biology.

The Biologist, knowing a bit about algorithms, asks for an algorithm that is "**like sorting**", i.e., a polynomial-time algorithm for her problem. Working with an *NP*-hard problem, if the fixed-parameter for the problem to solve is rather small (e.g., Protein Folding involves as the (natural) parameter the number

[2] Further background on parameterized complexity can be found in [DF98]. We remark in passing that if the fundamental mission of theoretical computer science is conceived of as *empirical and explanatory*, like theoretical physics, then a two- (or more-) dimensional theoretical framework might well be more suitable than the one-dimensional framework inherited from recursion theory, to the task of explaining the crucial differences in intrinsic problem complexity encountered in natural computational practice, even if the explanatory framework involves phenomenologically "unequal" dimensions — a situation frequently encountered in physics.

of adjacencies between hydrophobic constituents of the protein sequence, that is the parameter is less than 100 for interesting applications) the knowledgeable biologist will henceforth ask, "Can I get an algorithm **like Vertex Cover**?" There is no way to answer this question without taking the discussion into the third natural zone.

We remark, that the parameterize complexity of PROTEIN FOLDING as well as TOPLOGICAL CONTAINMENT FOR GRAPHS and Directed Feedback Vertex Set is still open. All three problems are conjectured to be in *FPT*. For *NP*-completeness of PROTEIN FOLDING and TOPLOGICAL CONTAINMENT FOR GRAPHS we refer to [CGPPY98, BL98] and [DF98], respectively. *NP*-completeness of Directed Feedback Vertex Set is shown in [GJ79].

1.3 The Substantial Open Question About Parameterized Complexity

Despite the fact that *logically, in some sense, NP*-completeness can now reasonably be considered a rather unimportant issue for problems that are, when naturally parameterized, fixed-parameter tractable, and *for which the main applications are covered by small or moderate parameter ranges*. From a practical point of view there is still an important unresolved question that motivates our work in this paper:

What are typical functions $f(k)$ for problems in *FPT*?

We make two contributions.

- We substantially improve the best known *FPT* algorithm for the MAX LEAF SPANNING TREE problem. The best previous algorithm due to Downey and Fellows in [DF95a, DF98] has a running time of $O(n+(2k)^{4k})$. Our algorithm runs in time $O(n+(k+1)(14.23)^k)$. In the concluding section we discuss the fine-grained significance of this improvement.
- We introduce new methods that appear to be widely useful in designing improved *FPT* algorithms. The first new method is that of *coordinatized kernelization arguments* for establishing problem kernelizations. The second new method, *catalytic reduction*, employs a small amount of partial information about potential solutions to guide the efficient development of a search tree.

2 Prototype: An Improved FPT Algorithm for the Max Leaf Spanning Tree Problem

The history of the problem and some recent complexity developments can be found in [D74, GJ79, GMM94, GMM97, LR98]. An interesting application of the problem is described in [KKRUW95]. The flagship problem for the new techniques we introduce here is defined as follows.

MAX LEAF SPANNING TREE

Input: A graph G and a positive integer k.
Parameter: k
Question: Does G have a spanning tree with at least k leaves?

One of the remarkably nice properties of *FPT* is that the following is an equivalent definition of the tractable class of parameterized problems [DFS99].

Definition 4. *A parameterized language L is in FPT if and only if there is:*

1. *A function $g(k)$.*
2. *A 2-variable polynomial $q(n,k)$.*
3. *A many:1 parametric reduction Φ of L to itself, requiring time at most $q(n,k)$, that transforms an instance (x,k), where $|x| = n$, to an instance (x',k') with $|x'| \leq g(k)$ and $k' \leq k$, so that $(x,k) \in L$ if and only if $(x',k') \in L$.*

In other words, a problem with parameter k is in *FPT* if and only if an input to the problem can be reduced *in ordinary polynomial time* to an equivalent input whose size is bounded by a function (only) of the parameter. For most problems in *FPT*, moreover, a *natural* set of reduction rules are known that accomplish the transformation Φ by a series of "local simplifications". This process is termed *kernelization* in the terminology of [DF95a][3] and, currently, the main practical methods of *FPT*-algorithm design are based on *kernelization* and *the method of bounded search trees*.

The idea of kernelization is relatively simple and can be quickly illustrated for the VERTEX COVER problem. If the instance is (G,k) and G has a pendant vertex v of degree 1 connected to the vertex u, then it would be silly to include v in any solution (it would be better, and equally necessary, to include u), so (G,k) can be reduced to $(G',k-1)$, where G' is obtained from G by deleting u and v. Some more complicated and much less obvious reduction rules for the VERTEX COVER problem can be found in the current state-of-the-art *FPT* algorithms (see [BFR98, DFS99, CKJ99, NR99b, Ste99]). The basic schema of this method of *FPT* algorithm design is that reduction rules are applied until an *irreducible* instance (G',k') is obtained. At this point in the *FPT* algorithm, a *Kernelization Lemma* is invoked to decide all those instances where the reduced instance G' is larger than $g(k')$ for some function g. For example, in the cases of VERTEX COVER and PLANAR DOMINATING SET, if a reduced graph is *large* then (G',k') is a no-instance for a suitable linear function g. In the case of MAX LEAF SPANNING TREE and NONBLOCKER, large reduced instances are automatically yes-instances.

[3] These natural kernelization algorithms have significant applications in the design of heuristics for hard problems, since they are a reasonable preprocessing step for *any* algorithmic attack on an intractable problem [DFS99].

In first phase of our algorithm a set of reduction rules transforms an instance (G, k) of Max Leaf Spanning Tree to another instance (G', k') where $k' \leq k$ and $|G'| \leq 5.75k$. By exploring all k-subsets of the problem kernel G', this immediately implies an *FPT* algorithm with running time $O(n + k2^{5.75k}) = O(n+k(33.1)^k)$. But we will do substantially better than that, namely we present an algorithm running in time $O(n + (k+1)(14.23)^k)$.

Our algorithm has three phases:

Phase 1: Reduction to a problem kernel of size $5.75k$.
Phase 2: The introduction of catalytic vertices.
Phase 3: A search tree based on catalytic branching (section 2.2) and coordinatized reduction (section 2.1).

Our algorithm is actually based on a slight variation on Max Leaf Spanning Tree defined as follows.

Catalytic Max Leaf Spanning Tree

Input: A graph $G = (V, E)$ with a distinguished *catalytic* vertex $t \in V$, $k \in Z^+$.
Parameter: k
Question: Does G have a spanning tree T having at least k leaves, such that t is an internal vertex of T?

In the following subsection we prove that this variant of the originl problem has a kernel of size $5.75k$. Because the reduction rules used are almost identical to the reduction rules used in the proof that Max Leaf Spanning Tree has a kernel of size $5.75k$, we will concentrate on the kernelization of Catalytic Max Leaf Spanning Tree only.

2.1 The Kernelization Lemma and the Method of Coordinatized Kernels

How does one proceed to discover an adequate set of reduction rules, or elucidate (and prove) a bounding function $g(k)$ that insures for instances larger than this bound, that the question can be answered simply?

The technique of *coordinatized kernels* is aimed at these difficulties, and we will illustrate it by example with the Max Leaf Spanning Tree problem. We seek a Lemma of the following form:

Lemma 1. *If $(G = (V, E), k)$ is a reduced instance of* Catalytic Max Leaf Spanning Tree *with catalytic vertex $t \in V$, and G has more than $g(k)$ vertices, then (G, k) with catalytic vertex t is a yes-instance.*

Proof. Suppose that:

(1) G has more than $g(k)$ vertices. (We will eventually determine $g(k)$, cf. page 247.)

(2) G is connected and reduced. (As we make the argument, we will see how to define the reduction rules.)
(3) G is a yes-instance for k, witnessed by a subtree T (with t internal; not necessarily spanning) having k leaves.
(4) G is a no-instance for $k+1$.
(5) Among all such G satisfying (1-4), the witnessing tree T has a minimum possible number of vertices.
(6) Among all such G and T satisfying (1-5), the quantity $\sum_{l \in L} d(t,l)$ is minimized, where L is the set of leaves of T and $d(t,l)$ is the distance in T to the root vertex t.

Then we argue for a contradiction.

Comment. The point of all this is to set up a framework for argument that will allow us to see what reduction rules are needed, and what $g(k)$ can be achieved. In essence we are setting up a (possibly elaborate, in the spirit of extremal graph theory) argument by minimum counterexample — and using this as a discovery process for the *FPT* algorithm design. Condition (3) gives us a way of "coordinatizing" the situation by giving us the structure of a solution to refer to (how this is used will become clear as we proceed).

Since G is connected, any tree subgraph T of G with k leaves extends to a spanning tree with k leaves. This witnessing subgraph given by condition (3) is minimized by condition (5). Refer to the vertices of $V - T$ as *outsiders*. The following claims are easily established. The first five claims are enforced by condition (4).

Claim 1: No outsider is adjacent to an internal vertex of T.
Claim 2: No leaf of T can be adjacent to two outsiders.
Claim 3: No outsider has three or more outsider neighbors.
Claim 4: No outsider with 2 outsider neighbors is connected to a leaf of T.
Claim 5: The graph induced by the outsider vertices has no cycles.

It follows from Claims (1-5) that the subgraph induced by the outsiders consists of a collection of paths, where the internal vertices of the paths have degree 2 in G. Since we are ultimately attempting to bound the size of G, this suggests (as a discovery process) the following reduction rule for kernelization.

Kernelization Rule 1: If (G,k) has two adjacent vertices u and v of degree 2, neither of which is the catalyst t, then:
(Rule 1.1) If uv is a bridge, then contract uv to obtain G' and let $k' = k$.
(Rule 1.2) If uv is not a bridge, then delete the edge uv to obtain G' and let $k' = k$.

The soundness of this reduction rule is not completely obvious, although not difficult. Having now partly clarified condition (2), we can continue the argument. The components of the subgraph induced by the outsiders must consist of paths having either 1,2 or 3 vertices.

The first possibility leads to another reduction rule which eliminates pendant vertices. This leads to a situation where the only possibilities for a component C of the outsider graph are:

1. The component C consists of a single vertex and C has at least 2 leaf neighbors in T.
2. The component C consists of two vertices, and C has at least 3 leaf neighbors in T.
3. The component C has three vertices, and has at least four leaf neighbors in T.

The weakest of the population ratios for our purposes in bounding the kernel size is given by case (3). We can conclude, using Claim 2, that the number of outsiders is bounded by $3k/4$.

The next step is to study the tree T. Since it has k leaves it has at most $k-2$ branch vertices. Using conditions (5) and (6), it is not hard to see that:

1. Any path in T between a leaf and its parental branch vertex has no subdivisions.
2. Any other path in T between branch vertices has at most 3 subdivisions (with respect to T).

Consequently T has at most $5k$ vertices, unless there is a contradiction. This yields our $g(k)$ of $5.75k$. We believe that this bound can be improved by a more detailed structural analysis in this same framework.

2.2 Catalytic Reduction in Search Tree Branching

The catalytic branching technique is described as follows. Let $c = 5.75$ for convenience. Assume that any instance G for parameter k can be reduced in linear time to an instance G' of size at most ck. Suppose we are considering an instance (G, k) with catalytic vertex t. We can assume that G is connected. Consider a neighbor u of t in G.

Catalytic Branching. We have the following basic branching procedure: (G, k) with catalytic vertex t is a *yes*-instance if and only if one of the following two branch instances is a *yes*-instance. The first branch is developed on the assumption that u is also an internal vertex of a k-leaf spanning tree T (for which t is internal). The second branch is developed on the assumption that u is a leaf for such a tree T.

First Branch: Here we have (G', k), where G' is obtained from G by contracting the edge between t and u. The resulting combined vertex is the catalytic vertex for G'.

Second Branch: Here we begin with $(G', k-1)$, where G' is obtained by deleting u. But now, since the parameter has been decreased, we may *re-kernelize* so that the resulting graph has size at most $c(k-1)$. Depending on the size of G', the size of the instance that we reduce to on this branch is somewhere between $n-1$ and $n-c-1$, when G has size n, in the worst case.

The key to the efficiency of this technique is in the re-kernelization on the second branch. Because the amount of re-kernelization varies, this leads to a somewhat complicated recurrence. Our bound on the running time is based on

a simpler recurrence that provides an upper bound that is probably not particularly tight.

We have thus described Phase 3 of our algorithm. We must still describe Phase 2.

Introducing Catalytic Vertices. A simple way to accomplish this task is to simply choose a set of $k+1$ vertices in G. If (G, k) is a yes-instance for MAX LEAF SPANNING TREE then one of these vertices can be assumed to be an internal vertex of a solution k-leaf spanning tree. The $k+1$ branches must all be explored.

2.3 Analysis of the Running Time

We define an abstract *value* $v(n, k)$ for a node (G, k) in the search tree, where G is a graph on n vertices. Choosing an appropriate *abstract weighting* w (by computational experiment) for the parameter k, in order to capture some of the information about the efficiency of catalytic branching, we define $v(n, k) = 8k+n$ (that is, $w = 8$ seems to work best for our current kernelization bound). Our kernelization bound of $n \leq 5.75k$ means that we require an upper bound on the size of a search tree with root value $v(n, k)$ of at most $13.75k$. The catalytic branching gives the recurrence

$$f(v) \leq f(v-1) + f(v-9)$$

which yields a positive real root of $\alpha = 1.2132$. Evaluating α^v at $v = 13.75k$, and noting that the nodes of the search tree require $O(k)$ time to process, we immediately obtain a parameter function of $k(14.23)^k$ (for each of the $k+1$ search trees initiated in Phase 2 by the introduction of a catalytic vertex). By the speedup technique of Niedermeier and Rossmanith [NR00a], we get a running time of $O(n + (k+1)(14.23)^k)$ for our algorithm.

3 Catalytic Branching as General *FPT* Technique

The catalytic branching strategy can easily be adapted to a number of other *FPT* problems. The following are some sketches of further applications. (Note, however, that to make use of catalytic branching, it is first necessary to prove a kernelization procedure that respects the presence of a catalytic vertex.)

*Example 1 (*FEEDBACK VERTEX SET FOR UNDIRECTED GRAPHS). The catalytic vertex t is required *not* to be in the feedback vertex set. If the neighbor u is also *not* in the feedback vertex set, then the edge tu can be contracted. If u is in the fvs, then (G', k') is obtained by deleting u, setting $k' = k - 1$ and re-kernelizing.

*Example 2 (*PLANAR DOMINATING SET). The catalytic vertex t is required to belong to the dominating set. If the neighbor u is not in the dominating set then it can be deleted (first branch). On the second branch, the edge tu can be contracted, and the resulting graph can be re-kernelized for $k - 1$.

*Example 3 (*Edge Dominating Set). (The *FPT* algorithms for Matrix Domination are currently based on a reduction to this problem.) Very similar to Example 2.

*Example 4 (*Nonblocker *(Also called enclaveless sets [HHS98]).* The parametric dual of Minimum Dominating Set.)

Input: A graph $G = (V, E)$ where $|V| = n$ and a positive integer k.
Parameter: k
Question: Does G admit a dominating set of size at most $n-k$? Equivalently, is there a set $N \subseteq V$ of size k with the property that for every element $x \in N$, there is a neighbor y of x in $V - N$?

For this problem, a kernelization respecting a catalytic vertex is known. We require that the catalytic vertex t be a member of $V - N$. On the first branch of the search tree, we can contract tu if the neighbor u of t is also in $V - N$. On the second branch, we delete u and re-kernelize for $k-1$. The detailed algorithm is going to be published elsewhere [FMRS00].

Catalytic branching is a general idea that might be applied in other settings besides graph problems — what it really amounts to is a search tree branching method based on retaining a small amount of partial information about potential solutions.

4 Concluding Remarks

How does one evaluate the goodness of an *FPT* algorithm? Since every problem in *FPT* can be solved in time $f(k) + n^c$ where c is a fixed constant (usually $c \leq 3$), and there are no hidden constants, we can measure the success of an *FPT* algorithm by its *klam value*, defined to be the maximum k such that the parameter function $f(k)$ for the algorithm (where $c \leq 3$) is bounded by some universal limit U on the number of basic operations any computation in our practical universe can perform. We will (perhaps too optimistically) take $U = 10^{20}$. This might appear a bit strange at first, but parameterized complexity in many ways represents a welding of engineering sensibilities (with the attendant sensitivity to particular finite ranges of magnitudes), and mathematical complexity analysis. Engineers have never been too keen on asymptotic analysis for practical situations.

Max Leaf Spanning Tree was first observed to be in *FPT* nonconstructively via the Robertson-Seymour graph minors machinery by Fellows and Langston [FL88]. This approach had a klam value of zero! Bodlaender subsequently gave a constructive *FPT* algorithm based on depth-first search methods with a parameter function of around $17k^4$! which has a klam value of 1 [Bod89]. This was improved by Downey and Fellows [DF95a, DF98] to $f(k) = (2k)^{4k}$ which

has a klam value of 5. Our algorithm here has a klam value of 16, according to our current analysis, which is probably not very tight.[4]

At this point in time, there are many examples of trajectories of this sort in the design of *FPT* algorithms. VERTEX COVER is another classic example of such a trajectory of (eventually) striking improvements (see [DF98]). What these algorithm design trajectories really show is that we are still discovering the basic elements, tricks and habits of mind required to devise efficient *FPT* algorithms. It is a *new game* and it is a rich game. After many rounds of improvements the best known algorithm for VERTEX COVER runs in time $O((1.27)^k + n)$ [CKJ99] and has a klam value of 192. Will MAX LEAF SPANNING TREE admit klam values of more than 50? How much more improvement is possible? Can any plausible mathematical limits to such improvements be established?

References

[Bod89] H. L. Bodlaender. "On linear time minor tests and depth-first search." In F. Dehne et al. (eds.), *Proc. First Workshop on Algorithms and Data Structures*, LNCS 382, pp. 577–590, 1989.

[BFR98] R. Balasubramanian, M. R. Fellows, and V. Raman. "An Improved Fixed-Parameter Algorithm for Vertex Cover." *Information Processing Letters* 65:3, pp. 163–168, 1998.

[BL98] B. Berger, T. Leighton. "Protein Folding in the Hydrophobic-Hydrophilic (HP) Model is NP-Complete." In S. Istrail, P. Pevzner, and M. Waterman (eds.), *Proceedings of the Second Annual International Conference on Computational Molecular Biology (RECOMB98)*, pp. 30–39, 1998.

[CCDF97] L. Cai, J. Chen, R. Downey, and M. Fellows. "The parameterized complexity of short computation and factorization." *Archive for Mathematical Logic* 36, pp. 321–338, 1997.

[CGPPY98] P. Crescenzi, D. Goldman, C. Papadimitriou, A. Piccolboni and M. Yannakakis. "On the complexity of protein folding." In S. Istrail, P. Pevzner, and M. Waterman (eds.), *Proceedings of the Second Annual International Conference on Computational Molecular Biology (RECOMB98)*, 1998.

[CKJ99] J. Chen, I. Kanj, and W. Jia. "Vertex cover: Further Observations and Further Improvements." *25th International Workshop on Graph-Theoretic Concepts in Computer Science (WG'99)* Ascona, Switzerland, June 1999.

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

[DF95b] R. Downey and M. Fellows. "Fixed-parameter tractability and completeness II: completeness for $W[1]$." *Theoretical Computer Science A* 141, pp. 109–131, 1995.

[DF98] R. Downey and M. Fellows. *Parameterized Complexity*. Springer-Verlag, 1998.

[DFS99] R. Downey, M. Fellows, and U. Stege. "Parameterized complexity: a framework for systematically confronting computational intractability." In: *Contemporary Trends in Discrete Mathematics* (R. Graham, J. Kratochvil, J. Nesetril and

[4] Keep in mind that this is still worst-case analysis, via theoretical estimates on the sizes of search trees. In practice, some *FPT* algorithms seem to have much larger empirical klam values [Ste99].

F. Roberts, eds.), *AMS-DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 49, pp. 49–99, 1999.

[D74] E. Dijkstra. "Self-Stabilizing Systems in Spite of Distributed Control." *Communications of the ACM* 17, pp. 643–644, 1974.

[FL88] M. Fellows and M. Langston. "On Well-Partial-Order Theory and Its Applications to Combinatorial Problems of VLSI Design." *Technical Report CS-88-188*, Department of Computer Science, Washington State University, 1988.

[FMRS00] M. Fellows, C. McCartin, F. Rosamond, and U. Stege. "The parametric dual of Dominating Set is fixed-parameter tractable," 2000.

[GMM94] G. Galbiati, F. Maffioli, and A. Morzenti. "A Short Note on the Approximability of the Maximum Leaves Spanning Tree Problem." *Information Processing Letters* 52, pp. 45–49, 1994.

[GMM97] G. Galbiati, A. Morzenti and F. Maffioli. "On the Approximability of some Maximum Spanning Tree Problems." *Theoretical Computer Science* 181, pp. 107–118, 1997.

[GJ79] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman, San Francisco, 1979.

[HHS98] T. Haynes, S. Hedetniemi, and P. Slater. *Fundamentals of Domination in Graphs.* Marcel Dekker, Inc, 1998.

[KKRUW95] E. Kranakis, D. Krizanc, B. Ruf, J. Urrutia, G. Woeginger. "VC-dimensions for graphs." In M. Nagl, editor, *Graph-theoretic concepts in computer science*, LNCS 1017, pp. 1–13, 1995.

[LR98] H.-I. Lu and R. Ravi. "Approximating Maximum Leaf Spanning Trees in Almost Linear Time." *Journal of Algorithms* 29, pp. 132–141, 1998.

[NR99b] R. Niedermeier and P. Rossmanith. "Upper Bounds for Vertex Cover Further Improved." In C. Meinel and S. Tison, editors, *Proceedings of the 16th Symposium on Theoretical Aspects of Computer Science*, LNCS 1563, pp. 561–570, 1999.

[NR00a] R. Niedermeier and P. Rossmanith. "A General Method to Speed Up Fixed-Parameter-Tractable Algorithms." *Information Processing Letters*, 73, pp. 125–129, 2000.

[NR00b] R. Niedermeier and P. Rossmanith. "An efficient fixed parameter algorithm for 3-Hitting Set." accepted for publication in *Journal of Discrete Algorithms*, August 2000.

[Ste99] U. Stege. *Resolving Conflicts from Computational Biology.* Ph.D. thesis, Department of Computer Science, ETH Zürich, Switzerland, 1999.