

Either/Or: Using VERTEX COVER Structure in Designing FPT-algorithms — the Case of k -INTERNAL SPANNING TREE

Elena Prieto¹ Christian Sloper²

¹ School of Electrical Engineering and Computer Science,
The University of Newcastle
NSW, Australia
elena@cs.newcastle.edu.au

² Department of Informatics,
University of Bergen
Norway
sloper@ii.uib.no

Abstract. To determine if a graph has a spanning tree with few leaves is NP-hard as HAMILTONIAN PATH is a special case. In this paper we study the *parametric dual* of this problem, k -INTERNAL SPANNING TREE (Does G have a spanning tree with at least k internal vertices?). We give an algorithm running in time $O(2^{4k \log k} \cdot k^{7/2} + k^2 \cdot n^2)$. We also give a 2-approximation algorithm for the problem.

However, the main contribution of this paper is that we show the following remarkable structural bindings between k -INTERNAL SPANNING TREE and k -VERTEX COVER:

- NO for k -VERTEX COVER implies YES for k -INTERNAL SPANNING TREE.
- YES for k -VERTEX COVER implies NO for $(2k + 1)$ -INTERNAL SPANNING TREE.

We give a polynomial-time algorithm that produces *either* a vertex cover of size k or a spanning tree with at least k internal vertices. We show how to use this inherent *vertex cover structure* to design algorithms for FPT problems, here illustrated mainly by k -INTERNAL SPANNING TREE. We also briefly discuss the application of this vertex cover methodology to the parametric dual of the DOMINATING SET problem. This design technique seems to apply to many other FPT problems.

Keywords: spanning trees, fixed-parameter tractability, vertex cover, kernelization.

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 behavior. These include the well-known problems MIN CUT LINEAR ARRANGEMENT, BANDWIDTH, VERTEX COVER, and INDEPENDENT SET (for definitions the reader may refer to [GJ79]).

Definition 1. (*Fixed Parameter Tractability*) A parameterized problem $L \subseteq \Sigma^* \times \Sigma^*$ is fixed-parameter tractable if there is an algorithm that correctly decides, in time $f(k) n^\alpha$, for input $(x, y) \in \Sigma^* \times \Sigma^*$ whether or not $(x, y) \in L$, where n is the size of the input x , $|x| = n$, k is the parameter, α is a constant (independent of k) and f is an arbitrary function.

The analog of NP is the parameterized complexity class $W[1]$ [DF95b]. The fact that the k -NDTM problem is complete for $W[1]$ was proved by Cai, Chen, Downey and Fellows in [CCDF97]. Since BANDWIDTH and INDEPENDENT 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.

Further background on parameterized complexity can be found in [DF98].

The problems we address in this paper concern spanning trees, namely k -(MIN)LEAF SPANNING TREE (Does G have a spanning tree with at most k leaves?) and its parametric dual, k -INTERNAL SPANNING TREE (Does G have a spanning tree with at most $n - k$ leaves?). In the classical complexity theory these two problems are indistinguishable from each other. In the following section we show that the problems are intrinsically different when analyzed from a parameterized point of view. We prove that while k -(MIN)LEAF SPANNING TREE is $W[P]$ -hard, k -INTERNAL SPANNING TREE is in FPT. In Section 3 we describe how to use the bounded *vertex cover structure* to design an FPT algorithm for k -INTERNAL SPANNING TREE. We give an analysis of the running time of the algorithm generated by the method in Section 4, we show how the methodology created in Section 3 can be applied to other FPT problems and we conclude with some remarks about future research. Also, as a consequence of the preprocessing of the graph necessary to create our fixed-parameter algorithm, we easily obtain a polynomial time 2-approximation algorithm for k -INTERNAL SPANNING TREE.

2 Parametric duality

Koth and Raman were the first to notice a remarkable empirical pattern in the parameterized complexity of familiar NP-complete problems, having to do with *parametric duality* [KR00]. Koth and Raman observed that if a problem is fixed parameter tractable then its parametric dual usually is not tractable. The idea is perhaps best presented by the dual pair of problems INDEPENDENT SET and VERTEX COVER. The duality between the two problems consists in the fact that a graph G has a vertex cover of size k if and only if it has an independent set of size $n - k$. Thus, if we consider the two naturally parameterized problems, for input G and parameter k , where in the one case we are “parameterizing upward” and in the other case we are “parameterizing downward”. As we have mentioned VERTEX COVER is fixed-parameter tractable, whereas INDEPENDENT SET is complete for the class $W[1]$ and therefore very unlikely to be in FPT unless both classes are proven to be equal, which is as unlikely as proving $P = NP$. This is also the case with DOMINATING SET and its parametric dual, NONBLOCKER, the former being $W[2]$ -complete and the latter being in FPT.

We prove that Koth and Raman’s observation holds for k -(MIN)LEAF SPANNING TREE and its parametric dual k -INTERNAL SPANNING TREE. We start with the following easy lemma:

Lemma 1. *The k -(MIN)LEAF SPANNING TREE problem is hard for $W[P]$.*

Proof: It is trivially true that the HAMILTONIAN PATH problem is a special case of k -(MIN)LEAF SPANNING TREE when we make the parameter $k = 2$. Since HAMILTONIAN PATH is NP -complete [GJ79] we can see immediately that k -(MIN)LEAF SPANNING TREE is not FPT unless $FPT = W[P]$. \square

Using Robertson and Seymour’s Graph Minor Theorem it is also quite straightforward to prove the following membership in FPT.

Lemma 2. *The k -INTERNAL SPANNING TREE problem is in FPT.*

Proof: Let \mathcal{F}_k denote the family of graphs that do not have spanning trees with at least k internal vertices. It is easy to observe that for each k this family is a lower ideal in the minor order. Less formally, let (G, k) be a NO-instance of k -INTERNAL SPANNING TREE, that is a graph G for which there is not a spanning tree with k internal vertices. The local operations which configure the minor order (i.e. edge contractions, edge deletions and vertex deletions) will always transform this NO-instance into another NO-instance. By the Graph Minor Theorem of Robertson and Seymour and its companion result that order testing in the minor order is FPT [RS99] we can claim that k -INTERNAL SPANNING TREE is also FPT. (An exposition of well-quasiordering as a method of FPT algorithm design can be found in [DF98].) \square

Unfortunately, this FPT proof technique suffers from being nonuniform and nonconstructive, and gives an $O(f(k)n^3)$ algorithm with a very fast-growing parameter function compared to the one we obtain in Section 3.

We remark that it can be shown that all fixed graphs with a vertex cover of size k are well-quasi ordered by ordinary subgraphs and have linear time order tests [FFLR03]. The proof of this is substantially shorter than the Graph Minor Project and could be used to simplify Lemma 2.

3 Either/Or: We Win

Currently, the main practical methods of FPT algorithm design are based on *kernelization* and *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 $v \in V(G)$ of degree 1 connected to the vertex $u \in V(G)$, 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, NR99b, Ste00, CKJ01]). 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 .

To find the function g in the case of k -INTERNAL SPANNING TREE we are going to make use of the intrinsic relationship between this problem and the thoroughly studied VERTEX COVER. A description of this relationship and its application to design FPT-algorithms will be shown later on in this section.

Lemma 3. *Any graph G has a spanning tree T such that all the leaves of T are independent vertices in G or G has a spanning tree T' with only two leaves.*

Proof: Given a spanning tree T of a graph G we say that two leaves $u, v \in T$ are in conflict if $uv \in E(G)$. We now show that given a spanning tree with i conflicts it is possible to obtain a spanning tree with $< i$ conflicts using one of the tree rules below:

1. If x and y are in conflict and they share a common parent z then a new spanning tree T' could be constructed using xy as an edge instead of yz . The number of conflicts in T' is decreased by one.
2. If x and y are in conflict and z , the parent of x , has another leaf then a new spanning tree T' could be constructed using the edge xy instead of xz . The number of conflicts in T' is decreased by one.

3. If x and y are in conflict and both their parents are of degree 2, then y and x are the endpoints of paths (vertices of degree exactly 2). If it is the same path, then the spanning tree is only one long path, and has only 2 leaves as stated in the lemma. Otherwise, the path from x ends at some vertex z where z is of degree ≥ 3 . We create a new spanning tree by disconnecting the path from z and connecting x to y , repairing the conflict between x and y . Since z is now of degree at least 2 we do not create any new conflicts.

The validity of the rules are easy to verify and it is obvious that they can be executed in polynomial time. Lemma 3 then follows by recursively applying the rules until no conflicts exists. \square

For a spanning tree T we define two sets, $A(T)$ of internal vertices of T , and $B(T)$ of leaves of T . If it is obvious from the context which spanning tree is in question we will for simplicity write A and B .

Several corollaries follow easily from this Lemma. One of them gives an approximation for k -INTERNAL SPANNING TREE, the others relate the problem to the well-studied VERTEX COVER.

Corollary 1. *k -INTERNAL SPANNING TREE has a 2-approximation algorithm.*

Proof: Note that because of the semi-bipartite structure of the transformed spanning tree it is impossible to include more than $|A|$ leaves as internals in the optimal spanning tree. The maximum number of internal vertices is at most $2|A|$, and hence the spanning tree generated by the algorithm in Lemma 3 has $|A|$ internal vertices and is a 2-approximation for k -INTERNAL SPANNING TREE. \square

Corollary 2. *If a graph $G = (V, E)$ is a NO-instance for k -VERTEX COVER then G is a YES-instance for k -INTERNAL SPANNING TREE.*

Proof: If a graph does not have a vertex cover of size k then we know that it does not have an independent set of size $\geq n - k$ (see discussion in section 2). This implies that $|B| < n - k$ and $|A| \geq k$, a YES-instance of k -INTERNAL SPANNING TREE. \square

Corollary 3. *If a graph $G = (V, E)$ is a YES-instance for k -VERTEX COVER then G is a NO-instance for $(2k + 1)$ -INTERNAL SPANNING TREE.*

Proof: Again, due to the semi-bipartite structure of the transformed spanning tree we know that if G is a YES-instance for k -VERTEX COVER then there are $n - k$ vertices in the independent set B . For each vertex in the independent set B that we include as a internal in the spanning tree we must include at least one other vertex in A . Thus, at most $2k$ vertices can be internal in the spanning tree and therefore G is a NO-instance for $(2k + 1)$ -INTERNAL SPANNING TREE. \square

We now know that if a graph does not have a k -vertex cover then it is a YES-instance for k -INTERNAL SPANNING TREE. We can use the structure provided by VERTEX COVER to bound the size of the kernel.

Lemma 4. *(Kernelization Lemma) Either G is a YES-instance for k -INTERNAL SPANNING TREE or it has less than $g(k) = 2k^3 + k^2 + 2k$ vertices.*

Proof: By Corollary 2 we know that *either* a graph is a YES-instance for k -INTERNAL SPANNING TREE *or* it has a k -vertex cover. Thus, we assume that there is a k -vertex cover in G . We will use this vertex cover structure to prove the lemma.

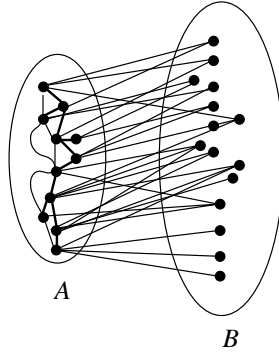


Fig. 1. Example of inherent vertex cover structure (Set A)

Note that the set A produced in Lemma 3 is a vertex cover and that the set B , its complement, is an independent set. (We assume that Lemma 3 did not produce a Hamilton path as it would be a optimal spanning tree and we could determine the result immediately.)

We define a B -bridge over a pair of vertices $w_1, w_2 \in A$ as a vertex $u \in B$ such that both uw_1 and uw_2 are in E .

We bound the number of vertices in B by showing that there is a limited amount of B -bridges. To do so we need to prove the following two claims which can be seen as kernelization rules for k -INTERNAL SPANNING TREE.

Claim 1 *If there exists a vertex $u \in B$ such that for all vertex pairs v_1, v_2 where u is a B -bridge over v_1 and v_2 there exists $2k + 1$ other B -bridges over v_1, v_2 then u can be removed from G (see Figure 2).*

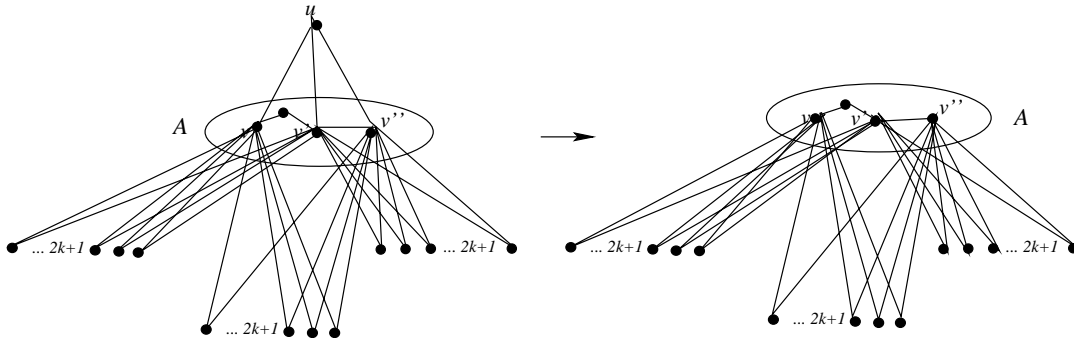


Fig. 2. Kernelization Rule 1

Proof of Claim 1. We prove this claim by showing that G has a k -internal spanning tree if and only if $G' = G \setminus u$ has a k -internal spanning tree.

If G' has a k -internal spanning tree then G obviously has a k -internal spanning tree as the introduction of u could not decrease the number of internals.

If G' has a k -internal spanning tree T then one of three cases apply:

1. u has degree 1 in the spanning tree (u is a leaf) and its neighbor, vertex z , has one or more other leaves. In this case $T \setminus u$ would be a spanning tree with k internal vertices for G' .
2. u has degree 1 in the spanning tree (u is a leaf) and its neighbor, vertex z , has no other leaves. We know that z has at least $2k+1$ other neighbors. No more than k of the $2k+1$ are internal vertices and at most k are needed as leaves elsewhere. We are left with at least one vertex of the $2k+1$ and it can be used as a leaf on z . Case 1 now applies.
3. u has degree $i \geq 2$ in the spanning tree (u is internal). In this case it is possible to change the spanning tree to a spanning tree where u has degree $< i$. Consider any two vertices $x, y \in N(u)$ in T . We know that x and y have at least $2k+1$ other B -bridges. The same argument as above applies, hence at least one vertex z of these B -bridges is an unessential leaf. Remove xu from the spanning tree and add xz, zy to obtain a spanning tree with more internals and where u is of degree $(i-1)$ in the spanning tree. Recursively apply this rule to obtain a spanning tree where u is of degree 1, where case 1 or 2 applies.

We see that we can always obtain a spanning tree where u is an unessential leaf, and can be removed without lowering the number of internal vertices of k . \square

This rule gets rid of many vertices of degree greater than or equal to two in the set B since they are part of B -bridges. We still could have countless vertices of degree 1 in the graph and therefore be unable to bound the size of B . We need the following reduction rule to eliminate those.

Claim 2 *If a graph G is a YES-instance for k -INTERNAL SPANNING TREE and there exist two vertices u, v such that the degree of u and v is 1 and u and v have the same neighbor, then $G' = G \setminus v$ is a YES-instance for k -INTERNAL SPANNING TREE*

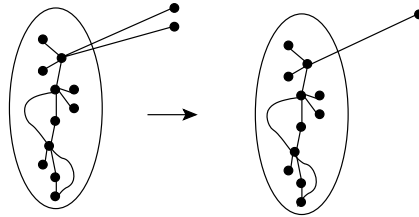


Fig. 3. Kernelization Rule 2

Proof of Claim 2. We prove this by showing that G has a k -internal spanning tree if and only if G' has a k -internal spanning tree.

If G' has a k -internal spanning tree then obviously G has a k -internal spanning tree since adding a leaf cannot decrease the number of internal vertices.

If G has a k -internal spanning tree we know that z is one of the internal vertices (because of x and y) and that x and y are leaves. In G' the vertex z is still an internal vertex (because of y) and y is a leaf. Thus, the number of internals in a spanning tree in G' is not affected by the missing leaf x . \square

We recursively apply Claims 1 and 2 to obtain a reduced instance where the claims no longer can be applied.

There are no more than k^2 pairs p_1, \dots, p_{k^2} of elements in A . Claim 1 implies that at least one of these pairs p_i has no more than $2k + 1$ unmarked B -bridges. Mark all these B -bridges and the pair p_i . Again by Claim 1 we have that at least one unmarked pair p_j with less than $2k + 1$ B -bridges. Mark these B -bridges and the pair p_j . Repeat this operation until all pairs are marked. This can go on for at most k^2 steps, thus no more than a total of $(2k + 1) \cdot k^2 = 2k^3 + k^2$ vertices of degree greater than or equal to 2 can exist in set B .

Now, due to Claim 2 we know that each vertex in A can have at most one “pendant” vertex of degree 1 in B , which gives us total of k vertices of degree 1 in any reduced instance of G . This, together with the $2k^3 + k^2$ vertices of degree greater than 1 in B and another k vertices in set A give us a maximum size of $2k^3 + k^2 + 2k$.

If a reduced instance has more than $2k^3 + k^2 + 2k$ we have reached a contradiction and the assumption that there was a vertex cover of size k must be wrong. Hence, by Corollary 2 we can conclude that G has a k -internal spanning tree.

This concludes the proof of Lemma 4. □

4 Analysis of the running time

Our algorithm works in several stages. It first runs a regular spanning tree algorithm and then modifies it to make the leaves independent. Then, if the spanning tree doesn't contain enough internals, we run our reduction rules to reduce the instance in size to $O(k^3)$. We would like to note that a limited number of experiments indicate that this algorithm is a very good heuristic. Then, we finally employ a brute-force spanning tree algorithm to find an optimal solution for the reduced instance.

Since we do not require a minimum spanning tree, we can use a simple breadth-first search to obtain a spanning tree. This can be done in time $O(V + E)$ [CLR90]. The conflicts can be detected in time $O(E)$ and repaired in time $O(V)$. Note that we could obtain the Vertex Cover structure by running one of the celebrated VERTEX COVER-algorithms instead, but for this particular problem our heuristic is sufficient.

To reduce the number of B -bridges we can apply the following algorithm. For each vertex $u \in B$ we count the number of vertices that are B -bridges for each pair of neighbors of u . If all pairs have more than $2k + 1$ B -bridges we can remove u (Claim 1). As there is less than k^2 such pairs this can be done in time $O(k^2 \cdot |V|)$ for each vertex in B .

We can easily remove superfluous leaves (Claim 1) in time $O(|V|)$. Thus reducing the instance to a cubic kernel requires in total $O(k^2 \cdot |V^2|)$ time.

To determine if there is a k -internal spanning tree in the reduced kernel we test every possible k -set of the kernel, there are less than k^{3k} such k -sets. Note that this can be rewritten as $2^{3k \log k}$. We now have to verify if these k vertices can be used as the internal vertices of a spanning tree. To do this we try every possible construction of a tree T with these k vertices, by Cayley's formula there are no more than k^{k-2} such trees. This, again, can be rewritten as $(2^{k \log k})/k^2$. Then we test whether or not each leaf in T can be assigned at least one vertex in the remaining kernel as its leaf. This is equivalent to testing if the leaves and the remaining kernel have a perfect bipartite matching, which can be done in time $O(\sqrt{V} * E)$. In this particular bipartite subset there are not more than k^4 edges giving us a total of $k^{11/2}$ for the matching. Thus for each k -set we can verify if it is a valid solution in $2^{k \log k} \cdot k^{7/2}$ time.

The total running time of the algorithm is $O(2^{4k \log k} \cdot k^{7/2} + k^2 \cdot n^2)$.

5 Vertex Cover Structure: Further Applications

In this section we give another example of how to use the methodology described in Section 3 to produce algorithms for other FPT problems.

Consider the parametric dual of k -DOMINATING SET, k -NONBLOCKER (Does $G = (V, E)$ have a subset of size k , V' , such that every element of V' has at least one neighbor in $V \setminus V'$?). Using the bounded vertex cover structure we can produce an algorithm for k -NONBLOCKER running in time $O(4^k + n^\alpha)$ as follows: We first compute a *maximal* independent set I in G . The complement of I , \bar{I} , is either a vertex cover of size $\leq k$ or a nonblocking set of size $\geq k + 1$. This simple algorithm, an analog of our Lemma 3, was first suggested by Faisal Abu-Khzam [A03]. This vertex cover structure allows us now to easily compute a path decomposition of the graph of width k . Let j_1, j_2, j_3, \dots be an arbitrary sequence of I . The path decomposition is a sequence of bags $B_i = \bar{I} \cup j_i$. Now, using the algorithm introduced by Telle and Proskurowski [PT93] and further improved by Alber and Niedermeier [AN02] we can compute a minimum dominating set in time $O(4^k + n^\alpha)$.

This result matches the running time of McCartin's algorithm [McC03] who used a completely different route to get the same running time.

We would also like to mention that $(n - k)$ -COLORING has recently been proven to be FPT using vertex cover structure [CFJ03].

6 Conclusions

In this paper we have given a hardness result for k -(MIN)LEAF SPANNING TREE and a fixed parameter algorithm for its *parametric dual*, k -INTERNAL SPANNING TREE. The algorithm runs in time $O(2^{4k \log k} \cdot k^{7/2} + k^2 \cdot n^2)$ which is the best currently known for this problem.

We also give a 2-approximation algorithm for the problem which can easily be further improved and the same idea used to find more approximation algorithms for other related problems.

We have shown the remarkable structural bindings between k -INTERNAL SPANNING TREE and k -VERTEX COVER in Corollaries 2 and 3. We believe that similar structural bindings exist between VERTEX COVER and other fixed-parameter tractable problems and we are confident that this inherent vertex cover structure can be used to design potent algorithms for these problems, especially when combined with constructive polynomial time algorithms that produce either a vertex cover or a solution for the problem in question. Even if such polynomial time *either/or*-algorithms do not exist, we may still use the quite practical FPT VERTEX COVER-algorithm to find the vertex cover structure.

The current state of the art algorithm for VERTEX COVER runs in time $O(1.286^k + n)$ [CKJ01] and has been proven useful in implementations by groups at the University of Carleton and University of Tennessee in Knoxville for exact solutions for values of n and k up to 2,500 [L03]. We believe that exploiting vertex cover structure may be one of the most powerful tools to designing algorithms for other fixed parameter tractable problems for which structural bindings with VERTEX COVER exist. For example, we suspect that the parameterized versions of MAX LEAF SPANNING TREE, MINIMUM INDEPENDENT DOMINATING SET and MINIMUM PERFECT CODE are very likely to fall into this class of problems.

Acknowledgements: We would like to thank Mike Fellows, Andrzej Proskurowski and Frances Rosamond for very helpful conversations and encouragement.

References

- [A03] F. Abu-Khzam. Private communication.
- [AN02] J. Alber and R. Niedermeier. Improved tree decomposition based algorithms for domination-like problems. *Proceedings of the 5th Latin American Theoretical INformatics (LATIN 2002)*, number 2286 in Lecture Notes in Computer Science, pages 613–627, Springer (2002).
- [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*, Lecture Notes in Computer Science, Vol. 382, Springer (1989), 577–590.
- [BFR98] R. Balasubramanian, M. R. Fellows, and V. Raman. An Improved Fixed Parameter Algorithm for Vertex Cover. *Information Processing Letters* 65:3 (1998), 163–168.
- [CFJ03] B. Chor, M. Fellows, D. Juedes. Private communication concerning manuscript in preparation.
- [CCDF97] Liming Cai, J. Chen, R. Downey and M. Fellows. The parameterized complexity of short computation and factorization. *Archive for Mathematical Logic* 36 (1997), 321–338.
- [CKJ01] J. Chen, I. Kanj, and W. Jia. Vertex cover: Further Observations and Further Improvements. *Journal of Algorithms* Volume 41, 280–301 (2001).
- [CLR90] T.H.Cormen, C.E.Leierson, R.L.Rivest, *Introduction to Algorithms*, MIT Press.
- [DF95a] R. Downey and M. Fellows. Parameterized Computational Feasibility. *P. Clote, J. Remmel (eds.): Feasible Mathematics II* Boston: Birkhauser (1995), 219–244.
- [DF95b] R. Downey and M. Fellows. Fixed-parameter tractability and completeness II: completeness for $W[1]$. *Theoretical Computer Science A* 141 (1995), 109–131.
- [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. *Contemporary Trends in Discrete Mathematics* (R. Graham, J. Kratochvil, J. Nešetřil and F. Roberts, eds.), *AMS-DIMACS Series in Discrete Mathematics and Theoretical Computer Science* 49 (1999), 49–99.
- [FFLR03] A. Faisal, M. Fellows, M. Langston, F. Rosamond. Private communication concerning manuscript in preparation.
- [FMRS01] M. Fellows, C. McCartin. F. Rosamond and U. Stege. Spanning Trees with Few and Many Leaves. *To appear*
- [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 (1994), 45–49.
- [GMM97] G. Galbiati, A. Morzenti and F. Maffioli. On the Approximability of some Maximum Spanning Tree Problems. *Theoretical Computer Science* 181 (1997), 107–118.
- [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, 1998, p. 16.
- [KR00] Subhash Khot and Venkatesh Raman. *Parameterized Complexity of Finding Hereditary Properties*. Proceedings of COCOON. Theoretical Computer Science (COCOON 2000 special issue)
- [KKRUW] E. Kranakis, D. Kaizanc, B. Rof, J. Urrutia, G. Woeginger. VC-Dimensions for Graphs. Carleton University, School of Computer Science *Technical Report: SCS-TR-255*.
- [L03] M. Langston. Private communication.
- [LR98] H.-I. Lu and R. Ravi. Approximating Maximum Leaf Spanning Trees in Almost Linear Time. *Journal of Algorithms* 29 (1998), 132–141.
- [McC03] Catherine McCartin. Ph.D. dissertation in Computer Science, Victoria University, Wellington, New Zealand, 2003.
- [NR99a] R. Niedermeier and P. Rossmanith. A General Method to Speed Up Fixed-Parameter-Tractable Algorithms. Institute für Informatik der Technischen Universität München *Technical Report: TUM-INFO-06-19913-80/1.-FI* (1999).
- [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*, number 1563 in Lecture Notes in Computer Science, Springer-Verlag (1999), 561–570.

- [PT93] J.A.Telle and A.Proskurowski. Practical algorithms on partial k -trees with an application to domination-like problems. *Proceedings WADS'93 - Third Workshop on Algorithms and Data Structures*. Springer Verlag, Lecture Notes in Computer Science vol.709 (1993) 610-621.
- [RS99] N. Robertson, P.D. Seymour. Graph Minors. XX Wagner's conjecture. *To appear*.
- [Ste00] Ulrike Stege. Ph.D. dissertation in Computer Science, ETH, Zurich, Switzerland, 2000.