

The Undirected Feedback Vertex Set Problem Has a Poly(k) Kernel [★]

Kevin Burrage¹ and Vladimir Estivill-Castro² and Michael Fellows³ and
Michael Langston⁴ and Shev Mac¹ and Frances Rosamond⁵

¹ Department of Mathematics, University of Queensland, Brisbane, QLD 4072

² Griffith University, Brisbane QLD 4111, Australia

³ School of EE & CS, University of Newcastle, Callaghan NSW 2308, Australia

⁴ Department of Computer Science, University of Tennessee, Knoxville TN
37996-3450 and Computer Science and Mathematics Division, Oak Ridge National
Laboratory, Oak Ridge, TN 37831-6164 U.S.A.

⁵ The Retreat for the Arts and Sciences, Newcastle, Australia

Abstract. Resolving a noted open problem, we show that the UNDIRECTED FEEDBACK VERTEX SET problem, parameterized by the size of the solution set of vertices, is in the parameterized complexity class $Poly(k)$, that is, polynomial-time pre-processing is sufficient to reduce an initial problem instance (G, k) to a decision-equivalent simplified instance (G', k') where $k' \leq k$, and the number of vertices of G' is bounded by a polynomial function of k . Our main result shows an $O(k^{11})$ kernelization bound.

1 Introduction

One of the most important concrete problems yet to be analyzed from the parameterized perspective is the FEEDBACK VERTEX SET problem, in both its undirected and directed forms. For a survey of many applications of feedback sets, see [FPR99]. The problem asks whether there is a set S of at most k vertices of the input graph (digraph) G , such that every cycle (directed cycle) in G contains at least one vertex in S . The directed form of the problem is notoriously open as to whether it is fixed-parameter tractable (FPT); the problem remains open even for the restriction to planar digraphs.

Previous results on FPT algorithms for the UNDIRECTED FEEDBACK VERTEX SET problem have followed a trajectory of steady improvements in the run times [DF92, Bod94, DF99, BBG00, RSS02, KPS04, RSS05, GGHNW05, DFLRS05].

The current best results are:

[★] This research has been supported in part by the U.S. National Science Foundation under grant CCR-0075792, by the U.S. Office of Naval Research under grant N00014-01-1-0608, by the U.S. Department of Energy under contract DE-AC05-00OR22725, and by the Australian Research Council under the auspices of the Australian Centre for Bioinformatics, through Federation Fellowship support of the first author, and through Discovery Project support of the second and third authors.

- A practical randomized FPT algorithm due to Becker, et al., runs in time $\mathcal{O}(4^k kn)$ and finds a feedback vertex set of size k (assuming one exists) with probability at least $1 - (1 - 4^{-k})^{c4^k}$ for an arbitrary constant c [BBG00].
- A deterministic FPT algorithm due independently to Guo, et al., and to Dehne, et al., solves the problem in time $\mathcal{O}^*(10.567^k)$ [GGHNW05,DFLRS05] (the runtime analysis is in the latter reference).

Here we address a noted open problem:

Is there a polynomial-time algorithm that kernelizes FVS on undirected graphs to a kernel of size polynomial in k ?

Kernelization bounds for FPT parameterized problems are an area of increasing interest, because of the strong connection between effective kernelization and practical algorithmics [Wei98,Wei00,Nie02]. The issue of efficient kernelization is “completely general” for parameterized complexity because a parameterized problem Π is in FPT if and only if there is a transformation from Π to itself, and a function g , that reduces an instance (x, k) to (x', k') such that:

- (1) the transformation runs in time polynomial in $|x, k|$,
- (2) (x, k) is a yes-instance of Π if and only if (x', k') is a yes-instance of Π ,
- (3) $k' \leq k$, and
- (4) $|x'| \leq g(k)$.

In the situation described above, we say that we have a *kernelization bound* of $g(k)$. The proof of the above “point of view” on FPT that focuses on P-time kernelization is completely trivial, giving a kernelization bound of $g(k) = f(k)$ for an FPT problem solvable in time $f(k)n^c$. But for many important FPT problems, we can do *much* better, and the “pre-processing” routines that produce small kernels seem to have great practical value [ACFLSS04,Nie02,Nie06]. For example, the VERTEX COVER problem can be kernelized in polynomial time to a graph on at most $2k$ vertices [NT75,ACFLSS04,CFJ04]. PLANAR DOMINATING SET also has a problem kernel of linear size [AFN04].

For typical problems that have been classified as fixed-parameter tractable, we see steady improvements both in the $f(k)$ in the best known FPT algorithms solving the problem in time $f(k)n^c$, and also (as an independent issue) in the best known kernelization bounds $g(k)$. In fact, polynomial-time kernelization/pre-processing seems to be both a deeper subject than one might have thought and also one of the most universally relevant ways of dealing with hard computational problems for realistic input distributions [Nie02,Nie06]. The practical importance of efficient kernelization has focused attention on subclasses of FPT, such as the class $Lin(k)$ consisting of the parameterized problems that admit linear problem kernels, and the class $Poly(k)$ of FPT problems with polynomial-sized kernels:

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

Little is yet known about these natural subclasses of FPT. In fact, there are many famous FPT problems for which membership in $Poly(k)$ is unknown. It is easy to point to examples of problems in FPT that are unlikely to be in $Poly(k)$,

but mathematical methods to substantiate such intuitive judgements are currently lacking. (For a concrete example, we think it is unlikely that MIN CUT LINEAR ARRANGEMENT is in $Poly(k)$.) It also seems that devising polynomial-time data reduction algorithms to show membership in $Lin(k)$ and $Poly(k)$ may be fruitful combinatorial ground for novel algorithmic strategies. All of this area seems well-worth investigating because of the close connections to practical computing. The recent Ph.D. dissertations of Prieto [Pr05] and Guo [Guo06] present some of the first systematic investigations of these challenges.

In the next section, we prove our main result, an $O(k^{11})$ kernelization for the UNDIRECTED FEEDBACK VERTEX SET problem. In the concluding section we point to some open problems.

2 UFVS is in $Poly(k)$

The problem we address is formally defined as follows.

UNDIRECTED FEEDBACK VERTEX SET

Instance: An undirected multigraph $G = (V, E)$ (that is, a graph with loops and multiple edges allowed), and a positive integer k .

Parameter: k

Question: Is there a subset $V' \subseteq V$, $|V'| \leq k$, such that every cycle in G contains at least one vertex of V' ?

The basic approach of our algorithm is to compute in polynomial time a specific *structural map* of the problem instance. Some of our polynomial-time data reduction rules are defined *relative to this structural map*. We study the situation by advancing “structural claims” that hold concerning a reduced instance. The reduction rules that are defined relative to the structural map either decide the instance, or reduce the size of the instance, or result in an “improvement” in the quality of the structural map (where the quality is defined so that only polynomially many improvements are possible). (This approach to kernelization is explicated with a number of examples in [Pr05].)

Additionally, we employ the following simple reduction rules that also play a role in the best known FPT algorithms for the problem, and are adapted from that context [GGHNW05,DFLRS05]:

Rule 1: The Degree One Rule. If v is a vertex of degree 1 in G , then delete v . The parameter k is unchanged.

Rule 2: The Degree Two Rule. If v is a vertex of degree 2 in G , with neighbors a and b (allowing possibly $a = b$), then modify G by replacing v and its two incident edges with a single edge between a and b (or a loop on $a = b$). The parameter k is unchanged.

Rule 3: The Loop Rule. If there is a loop on a vertex v then take v into the solution set, and reduce to the instance $(G - v, k - 1)$.

Rule 4: Multiedge Reduction. If there are more than two edges between u and v then delete all but two of these. The parameter k is unchanged.

The soundness of these reduction rules is trivial. In linear time we can determine if any of the above reduction rules can be applied to a problem instance. An instance (G, k) is *reduced* if none of the reduction rules can be applied.

We will always assume that an instance we are working with is reduced with regards to Reduction Rules (1-4).

Step One.

The first step of our algorithm employs the polynomial-time 2-approximation algorithm of Bafna, et al. [BBF99] to compute an approximate solution. If the approximate solution S that is produced is too big, $|S| > 2k$, then we can decide that (G, k) is a no-instance, and we are done.

Thus we can assume as the first part of our *structural map*, a feedback vertex set S for G , where $|S| \leq 2k$. Let F denote the forest $G - S$.

Step Two.

In the second step, we greedily compute a maximal set \mathcal{P} of pairwise internally vertex-disjoint paths (really, cycles) ρ that satisfy:

- (1) ρ begins at a vertex v of S .
- (2) If ρ begins at v then ρ also ends at v (thus forming a cycle).
- (3) All internal vertices of ρ belong to $V - S$.

The collection of paths \mathcal{P} is the second part of our structural map. Note that by our assumption that (G, k) is reduced relative to Reduction Rules (1-4), every path $\rho \in \mathcal{P}$ must have at least one internal vertex, because a reduced (G, k) does not have any loops. The following reduction rule is defined relative to this structure.

Rule 5: Flower Reduction. If there is a vertex $v \in S$ such that \mathcal{P} contains at least $k + 1$ paths that begin and end at v , then reduce (G, k) to $(G - v, k - 1)$.

The soundness of Rule 5 is immediately apparent: any k -element feedback vertex set V' must contain v , since otherwise at least one of the $k + 1$ internally vertex-disjoint “loops on v ” provided by \mathcal{P} would fail to contain a vertex in V' , as there are too many of them, and any vertex in V' can only hit one of them.

Step Three (Repeated).

This is the main loop of our algorithm. Our algorithm maintains a *structural map* for the reduced instance (G, k) , developed in the first two steps, consisting of:

- (1) the feedback set S , where $|S| \leq 2k$, and
- (2) the collection of paths \mathcal{P} .

We repeatedly either:

- Improve the *quality* of this structural map, according to a list of priorities detailed below, or
- Discover an opportunity to apply a reduction rule, or
- Output a kernelized instance.

There is one more reduction rule that is defined relative to our structural map, but first we need some definitions concerning the overall structural picture provided by our structural map information.

If ρ is a path in \mathcal{P} , then let ρ' denote the (“internal”) subpath formed by the vertices of ρ in $V - S$. Let \mathcal{P}' denote the collection of such subpaths ρ' of paths ρ in \mathcal{P} . Let F' denote the subforest of F that results by deleting from F any vertex that belongs to a path in \mathcal{P}' . The vertices of F can be viewed as partitioned into:

- (1) the paths ρ' in \mathcal{P}' , and
- (2) the trees in F' .

Let \mathcal{C} denote the set of vertices that either belong to S , or belong to a path $\rho' \in \mathcal{P}'$.

In order to distinguish between different kinds of trees in F' , we consider a graph, the *F-model graph*, that models this situation. In this model graph, there is one (*red*) vertex for each tree in F' , and one (*blue*) vertex for each path ρ' in \mathcal{P}' . Each vertex x in the model represents a set of vertices $V(x)$ of F . (If x is a red vertex, then it represents the vertices in a tree of F' ; if x is a blue vertex, then it represents the vertices of a path ρ' in \mathcal{P}' .)

In the *F-model graph*, a vertex x is adjacent to a vertex x' if and only if there is a vertex $u \in V(x)$ that is adjacent to a vertex $u' \in V(x')$. Note that the *F-model graph* is acyclic, since otherwise S would fail to be a feedback vertex set in G . If T is a tree in F' , then $v(T)$ denotes the red vertex of the *F-model graph* corresponding to T . Similarly, if ρ' is a path in \mathcal{P}' , then $v(\rho')$ denotes the blue vertex of the *F-model graph* corresponding to ρ' .

Let \mathcal{T}_0 denote the set of trees of F' whose corresponding vertices in the *F-model graph* have degree 0. Similarly, let \mathcal{T}_1 denote the set of trees of F' whose corresponding vertices in the *F-model graph* have degree 1, and let \mathcal{T}_2 denote the set of trees of F' whose corresponding vertices in the *F-model graph* have degree at least 2.

Rule 6: Tree Elimination. Suppose (G, k) is an instance with structural map (S, \mathcal{P}) produced at the end of Step Two, reduced with respect to Reduction Rules (1-5), and suppose that there is a tree T of F' such that for every pair of distinct vertices s, t in \mathcal{C} where T is adjacent to both s and t in G , there are at least $k + 2$ different trees T' in F' where each T' is also adjacent to both s and t in G . Then reduce (G, k) to $(G - T, k)$.

It is easy to see that determining whether Reduction Rule 6 applies in the situation described, can be accomplished in polynomial time. Less obvious is the soundness of the reduction rule.

Lemma 1. Reduction Rule 6 is sound.

Proof. It is obvious that if (G, k) is a yes-instance, then so is the reduced instance, since the yes-instances are hereditary under deletion. In the converse direction, suppose A is a feedback set of size k in the reduced instance. We argue that A must also be a solution for (G, k) . If not, then there must be a cycle C

in G that avoids A . Because A is a solution for the reduced graph, the cycle C must pass through T (perhaps more than once).

Any vertex v of \mathcal{C} is joined to T by at most one edge. There are two cases to consider to justify this assertion. If v belongs to S and is joined by two edges to T , then \mathcal{P} is not maximal. If v belongs to a path $\rho' \in \mathcal{P}'$ and is joined by two edges to T , then S fails to be a feedback vertex set.

It follows that the intersection of C with T consists of a number of disjoint paths ρ_i , $i = 1, \dots, m$, where for each i , a traversal of C enters ρ_i from a vertex $s_i \in H$ and exits ρ_i to a vertex $t_i \in H$ with $s_i \neq t_i$. It also follows that for $i \neq j$, the pair of vertices $\{s_i, t_i\}$ is disjoint from the pair of vertices $\{s_j, t_j\}$. Suppose that the ρ_i are indexed in the order of a traversal of C . The cycle C' that consists of (1) a path in T from s_1 to t_m , and (2) the path in C through the reduced graph from t_m to s_1 , also avoids A . In particular, s_1 and t_m do not belong to A . The pre-conditions for the reduction rule include that there are at least $k+2$ trees T' different from T in F' , each of these connected to both s_1 and t_m , and thus there are at least $k+2$ internally vertex-disjoint paths from s_1 to t_m in the reduced instance. But then, at least two of these must avoid A (since A has only k vertices) and form (together with s_1 and t_m) a cycle in the reduced graph that avoids A , a contradiction. Therefore A must also be a solution for (G, k) . \square

In the next series of lemmas we advance some claims about the situation where (G, k) is an instance with structural map (S, \mathcal{P}) (with \mathcal{P} maximal, as computed in Step Two), that is reduced with respect to Reduction Rules (1-5).

Lemma 2. \mathcal{P} contains at most $2k^2$ paths.

Proof. If there were more paths in \mathcal{P} , then the Flower Rule would apply. \square

Lemma 3. The number of trees in \mathcal{T}_2 is bounded by $2k^2 - 1$.

Proof. This follows from Lemma 2, since F is a forest. \square

Lemma 4. If T is any tree in the forest F' , then $|T| \leq 2k^2 + 2k - 2$.

Proof. We first argue that for any tree T on m vertices, there must be at least $m+2$ edges connecting T to the rest of G , that is, to $G - T$. First note that any leaf of T must be connected to the rest of G by at least two edges, since otherwise either the Degree One Reduction Rule or the Degree Two Reduction Rule would apply. If T consists of only one vertex, then similarly it must be connected to the rest of G by at least three edges. If u is an internal vertex of T of degree 2 relative to T , then at least one edge must join u to the rest of G , since otherwise the Degree Two Rule would apply. Let l denote the number of leaves of T , let j denote the number of internal vertices of T of degree 2, and let b denote the number of internal vertices of T of degree greater than 2. Then $m = l + j + b$. By the above observations, the number of edges joining T to the rest of G is at least $c = 2l + j$. The inequality we seek, $c \geq m + 2$, follows from the elementary fact that $l \geq b + 2$.

Lemma 4 follows, since if the bound stated in the lemma did not hold, then T would be joined by two edges either: (1) to a path $\rho' \in \mathcal{P}'$, or (2) to a vertex of S . Case (1) contradicts that S is a feedback vertex set for G , and case (2) contradicts that \mathcal{P} is maximal. \square

The Quality of the Structural Map. The *quality* of the structural map may be improved according to the following list of priorities:

- (1) The size of \mathcal{P} should be maximized.
- (2) The sum of the lengths of the paths in \mathcal{P}'

$$\sum_{\rho' \in \mathcal{P}'} |\rho'|$$

should be minimized.

Lemma 5. Suppose (G, k) is an instance with structural map (S, \mathcal{P}) (with \mathcal{P} maximal, as computed in Step Two), that is reduced with respect to Reduction Rules (1-5). Then either every path $\rho' \in \mathcal{P}'$ has length at most $6k + 2(2k^2 - 1)$, or in polynomial time we can improve the quality of the the structural map.

Proof. Consider a path $\rho' \in \mathcal{P}'$. We will argue about the length of a subpath “in the middle” of ρ' . Suppose the vertices of ρ' in the order of a traversal are:

$$\rho' = (v_1, v_2, \dots, v_l)$$

Let ρ'' denote the subpath

$$\rho'' = (v_{2k+1}, v_{2k+2}, \dots, v_{l-2k-1}, v_{l-2k})$$

In other words, ρ'' is the “middle” subpath excluding both the first $2k$ vertices of ρ' and the last $2k$ vertices of ρ' .

We argue that if $|\rho''| > 2k + 2(2k^2 - 1)$ then we can improve the quality of \mathcal{P} with respect to the second priority by replacing ρ with a shorter path. The vertices of ρ'' do not have degree 2 in G , since otherwise the Degree Two Reduction Rule would apply. Therefore each vertex x of ρ'' is adjacent to some vertex x' that is not one of its two neighbors with respect to the path ρ' . Since F is a forest, $x' \notin \rho'$. There are four possibilities: (1) $x' \in S$, (2) x' is a vertex of a tree in \mathcal{T}_2 , (3) x' is a vertex of a different path σ' in \mathcal{P}' , or (4) x' is a vertex of a tree T in \mathcal{T}_1 .

In case (4), there must be a path ρ_s from x through T to a vertex $s \in S$. Any such path is disjoint from the other paths in \mathcal{P}' . By the argument in the proof of Lemma 4, if $|T| = m$, then T must be joined by at least $m + 1$ edges to S , in view of the definition of \mathcal{T}_1 . Also, T cannot be joined to $s \in S$ by two different edges, otherwise \mathcal{P} is not maximal. It follows that $|T| \leq 2k - 1$, and that the number of internal vertices of ρ_s is bounded by $2k - 1$.

Two distinct vertices x, y of ρ'' cannot both be adjacent to a path $\sigma' \neq \rho'$, else S fails to be a feedback vertex set. Similarly, they cannot both be adjacent to the same tree in \mathcal{T}_2 . If there are distinct vertices x, y of ρ'' that both have paths to $s \in S$ (via trees in \mathcal{T}_1), then the second priority can be improved.

Since S contains at most $2k$ vertices, there are at most $2k^2 - 1$ trees in \mathcal{T}_2 , and there are at most $2k^2 - 1$ paths in \mathcal{P}' other than ρ' , the lemma follows, by the Pigeonhole Principle. \square

Theorem 1. There is a kernelization bound $g(k) = O(k^{11})$ such that for an instance (G, k) either:

- (1) $|G| \leq g(k)$, that is, the instance is kernelized, or
- (2) In polynomial time, we can compute a structural map (S, \mathcal{P}) for (G, k) and either discover an opportunity to apply a reduction rule, or discover an opportunity to improve the structural map according to the priorities listed above.

Proof. The forest F' consists of trees of size $O(k^2)$, by Lemma 4. The total number of vertices that either belong to S , or belong to a path in \mathcal{P}' , by Lemmas 2 and 5, is bounded by $z = O(k^4)$. Both of these statements hold under the assumption that the instance is reduced (in polynomial time) under Reduction Rules (1-5). Now consider going through the list of trees in F' , checking, for each tree encountered on the list, if there is an opportunity to apply Reduction Rule 6, the Tree Elimination Rule. We can keep a count, for each pair of vertices s, t in \mathcal{C} , of the number of trees on the list, up to that point, that are attached to both s and t . Each tree T on the list must either increase the count for at least one such pair (up to a limit of $k + 2$), otherwise T qualifies for the Tree Elimination Rule. There are at most $\binom{z}{2} = O(k^8)$ such pairs in \mathcal{C} , and therefore $O(k^9)$ trees in F' , else the Tree Elimination Rule is triggered. By Lemma 4, the total number of vertices in F' is $O(k^{11})$. \square

Corollary. In polynomial time, we can kernelize an instance of UFVS to a kernel of size $O(k^{11})$.

Proof. This follows from the fact that the priorities can be improved only a polynomial number of times. \square

3 Summary and Open Problems

In this paper, we have addressed a notable concrete open problem about FPT kernelization, and we have shown that the fixed-parameter tractable UNDIRECTED FEEDBACK VERTEX SET problem has an $O(k^{11})$ kernel. Might this problem have a linear-size kernel? We think this may well be so, despite our rather weak result here, in view of the fact that the problem admits a constant factor P-time approximation algorithm (a different, but seemingly related issue). Polynomial-time “crown-type” reductions may potentially play a role in showing that UFVS is in $Lin(k)$ [CFJ04,ACFLSS04,DFRS04,LS05,Pr05,Slo05].

The subject of parameterized complexity has been dogged from the beginning by concerns that the definition of FPT is “too lax” to capture a really meaningful framework for “coping with intractability” in the sense of [GJ79]. (The early entanglement of the subject with graph minors theory perhaps did not help in this regard.) In particular, the usual definition of FPT calls for solvability in time $f(k)n^c$ where $f(k)$ is a completely unrestricted function. Despite the fact that many of the most important and natural FPT parameterized problems seem to admit FPT algorithms with relatively (surprisingly?) benign $f(k)$, there has been motivation to explore subclasses of FPT with stronger definitional claims on practical parametric feasibility [FGW04,Wey04,FG06]. Certainly, considering FPT from the point of view of polynomial-time data reduction gives a very natural in-road to more sensitive distinctions concerning parameterized tractability.

Pre-processing is a humble strategy for coping with hard problems, almost universally employed. It has become clear, however, that far from being trivial and uninteresting, that pre-processing has unexpected practical power for real-world input distributions [Wei98,Wei00,BKV05], and is mathematically a much deeper subject than has generally been understood. It is almost impossible to talk about pre-processing in the classical complexity framework in any sensible and interesting way, and the historical neglect of this vital subject may be related to this fact.

Here is the difficulty. If my problem Π is NP-hard, then probably there is no P-time algorithm to *solve* the problem, that is, to completely dispose of the input. If you suggested that perhaps I should settle for a P-time algorithm that instead of completely disposing of the input, at least simplifies it by getting rid of, or reducing away, the easy parts — then this would seem a highly compelling suggestion. But how can this be formalized? The obvious first shot is to ask for a P-time algorithm that reduces the input I to an input I' where $|I'| < |I|$ in a way that loses no essential information (i.e., trades the original input for smaller input, which can be called *data reduction*). The difficulty with this “obvious” formalization of the compelling suggestion is that if you had such a P-time data reduction algorithm, then by repeatedly applying it, you could dispose of the entire input in polynomial time, and this is impossible, since Π is NP-hard. Thus, in the classical framework, an effort to formulate a mathematically interesting program to explore polynomial-time preprocessing immediately crashes.

In the parameterized complexity framework, however, such a program can be formulated in an absolutely interesting and productive way. The effectiveness of P-time processing is measured against the structure represented by the parameter. This is precisely the intellectual location of this paper. You might reasonably call the subject of FPT kernelization the *Lost Continent of Polynomial Time*.

Kernelization as we have considered it here is a polynomial time many:1 transformation of a parameterized problem Π to itself. One can also consider Turing kernelizations. A nontrivial example of a $2k$ Turing kernelization, based on iterative compression, for the VERTEX COVER problem is described by Dehne, et al. in [DFRS04]. Of course, in the case of VERTEX COVER, this is matched by a $2k$ many:1 kernelization. Nevertheless, Turing kernelization may in general have greater power than many:1 kernelization. It makes sense to ask if the UNDIRECTED FEEDBACK VERTEX SET problem might admit a linear-size Turing kernelization.

There seems to be much interesting work to do in exploring kernelizability, and its limits, for the many parameterized problems now known to be in FPT.

References

- [ACFLSS04] F. N. Abu-Khzam, R. L. Collins, M. R. Fellows, M. A. Langston, W. H. Suters and C. T. Symons. Kernelization algorithms for the vertex cover problem: theory and experiments. *Proceedings of the 6th Workshop on Algorithm Engineering and Experiments (ALENEX)*, New Orleans, January, 2004, ACM/SIAM, *Proc. Applied Mathematics 115*, L. Arge, G. Italiano and R. Sedgewick, eds.

- [AFN04] J. Alber, M. Fellows and R. Niedermeier. Polynomial time data reduction for dominating set. *Journal of the ACM* 51 (2004), 363–384.
- [BBF99] V. Bafna, P. Berman and T. Fujito. A 2-approximation algorithm for the undirected feedback vertex set problem. *SIAM Journal on Discrete Mathematics* 12 (1999), 289–297.
- [BBG00] A. Becker, R. Bar-Yehuda and D. Geiger. Random algorithms for the loop cutset problem. *Journal of Artificial Intelligence Research* 12 (2000), 219–234.
- [BKV05] H. Bodlaender, A. Koster and F. van den Eijkhof. Preprocessing rules for triangulation of probabilistic networks. *Computational Intelligence* 21 (2005), 286–305.
- [Bod94] H. Bodlaender. On disjoint cycles. *International Journal of Foundations of Computer Science* 5 (1994), 59–68.
- [CFJ04] B. Chor, M. Fellows and D. Juedes. Linear kernels in linear time, or how to save k colors in $O(n^2)$ steps. *Proceedings WG 2004*, Springer-Verlag, *Lecture Notes in Computer Science* 3353 (2004), 257–269.
- [DF92] R. Downey and M. Fellows. Fixed-parameter tractability and completeness. *Congressus Numerantium* 87 (1992), 161–187.
- [DF99] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer-Verlag, 1999.
- [DFLRS05] F. Dehne, M. Fellows, M. Langston, F. Rosamond and K. Stevens. An $O^*(2^{O(k)})$ FPT algorithm for the undirected feedback vertex set problem. In: *Proceedings COCOON 2005*, Springer-Verlag, *Lecture Notes in Computer Science* 3595 (2005), 859–869.
- [DFRS04] F. Dehne, M. Fellows, F. Rosamond and P. Shaw. Greedy localization, iterative compression and modeled crown reductions: new FPT techniques, an improved algorithm for set splitting and a novel $2k$ kernelization for vertex cover. *Proceedings of the First International Workshop on Parameterized and Exact Computation*, Springer-Verlag, *Lecture Notes in Computer Science* vol. 3162 (2004), 271–280.
- [FG06] *Parameterized Complexity Theory*, J. Flum and M. Grohe, Springer-Verlag, 2006.
- [FGW04] J. Flum, M. Grohe and M. Weyer. Bounded fixed-parameter tractability and $\log^2 n$ nondeterministic bits. *Proceedings of the 31st ICALP*, Springer-Verlag, *Lecture Notes in Computer Science* 3142 (2004), 555–567.
- [FPR99] P. Festa, P. M. Pardalos and M. G. C. Resende. Feedback set problems. In: D. Z. Du, P. M. Pardalos (Eds.), *Handbook of Combinatorial Optimization*, Vol. A, Kluwer (1999), 209–258.
- [GGHNW05] J. Guo, J. Gramm, F. Hueffner, R. Niedermeier, S. Wernicke. Improved fixed-parameter algorithms for two feedback set problems. *Proceedings of WADS 2005*, Springer-Verlag, *Lecture Notes in Computer Science* 3608 (2005), 158–169.
- [GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman, 1979.
- [Guo06] J. Guo. *Algorithm design techniques for parameterized problems*. Ph.D. Thesis, Friedrich-Schiller-Universität, Jena, 2006.
- [KPS04] I. Kanj, M. Pelsmajer and M. Schaefer. Parameterized algorithms for feedback vertex set. *Proceedings of the First International Workshop on Parameterized and Exact Computation*, Springer-Verlag, *Lecture Notes in Computer Science* vol. 3162 (2004), 235–247.
- [LS05] D. Lokshtanov and C. Sloper. Fixed-parameter set-splitting, linear kernel and improved running time. In: *Algorithms and Complexity in Durham 2005: Proceedings of the First ACiD Workshop*, King’s College Press, *Texts in Algorithmics* 4 (2005), 105–113.

- [Nie02] R. Niedermeier. *Invitation to fixed-parameter algorithms*, Habilitationsschrift, University of Tübingen, 2002. (Electronic file available from R. Niedermeier.)
- [Nie06] R. Niedermeier. *Invitation to Fixed Parameter Algorithms*. Oxford University Press, 2006.
- [NT75] G. L. Nemhauser and L. E. Trotter. Vertex packings: structural properties and algorithms. *Mathematical Programming* 8 (1975), 232–248.
- [Pr05] E. Prieto-Rodriguez. *Systematic kernelization in FPT algorithm design*. Ph.D. Thesis, School of EE&CS, University of Newcastle, Australia, 2005.
- [RSS02] V. Raman, S. Saurabh and C. Subramanian. Faster fixed-parameter tractable algorithms for undirected feedback vertex set. In *Proceedings of the 13th Annual International Symposium on Algorithms and Computation*, Springer, *Lecture Notes in Computer Science* vol. 2518 (2002), 241–248.
- [RSS05] V. Raman, S. Saurabh and C.R. Subramanian. Faster algorithms for feedback vertex set. In: *Proceedings of the 2nd Brazilian Symposium on Graphs, Algorithms and Combinatorics*, GRACO 2005, April 27-29, 2005, Angra dos Reis (Rio de Janeiro), Brazil. Elsevier, *Electronic Notes in Discrete Mathematics* (2005).
- [Slo05] C. Sloper. *Techniques in parameterized algorithm design*. Ph.D. Thesis, Department of Informatics, University of Bergen, Norway, 2005.
- [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’ (invited paper),” *Proc. WAE 2000*, Springer-Verlag, *Lecture Notes in Computer Science* 1982 (2001), 1–10.
- [Wey04] M. Weyer. Bounded fixed-parameter tractability: the case of $2^{\text{poly}(k)}$. *Proceedings of 1st IWPEC*, Springer-Verlag, *Lecture Notes in Computer Science* 3164 (2004), 49–60.