An $\mathcal{O}^*(2^{O(k)})$ FPT Algorithm for the Undirected Feedback Vertex Set Problem*

Frank Dehne†, Michael R. Fellows‡, Michael A. Langston§,
Frances A. Rosamond‡ and Kim Stevens¶

Abstract

We describe an algorithm for the FEEDBACK VERTEX SET problem on undirected graphs, parameterized by the size k of the feedback vertex set, that runs in time $\mathcal{O}(c^k n^3)$ where $c = 10.567$ and n is the number of vertices in the graph. This improves on the best previous FPT algorithms due to [RSS02, KPS04] and [RS05] that have running times of the form $\mathcal{O}^*(2^{\mathcal{O}(k \lg \lg k)})$ and $\mathcal{O}^*(2^{\mathcal{O}(k \frac{\lg k}{\lg \lg k})})$, respectively. These previous algorithms were based on the method of bounded search trees, branching on short cycles. Our algorithm is based on the relatively new FPT technique of iterative compression, and we prove our main result for a more general "annotated" form of the problem, where a subset of the vertices may be marked as *not* to belong to the feedback vertex set. We also establish "exponential optimality" for our algorithm; we prove that no FPT algorithm with a running time of the form $\mathcal{O}^*(2^{o(k)})$ is possible, unless there is an unlikely collapse of parameterized complexity classes, FPT $= M[1]$.

1 Introduction

Our focus in this paper is on the following parameterized problem that generalizes the familiar FEEDBACK VERTEX SET problem by allowing some annotation of a problem instance:

FEEDBACK VERTEX SET (FVS)

Instance:	An undirected graph $G = (V, E)$ (loops and multiple edges are allowed), an annotated subset $U \subseteq V$ of vertices, and a positive integer k.
Parameter:	k
Question:	Is there a subset S of the vertices not in U, $S \subseteq V - U$, of size at most k, $\lvert S\rvert \leq k$, such that $G - S$ is acyclic?

The FEEDBACK VERTEX SET problem is $\mathcal{NP}$-complete for both directed and undirected graphs [GJ79]. There are numerous applications of the problem in areas such as circuit testing, deadlock resolution, analyzing manufacturing processes and computational biology [BGNR98, ENSS98, FHPSS04, FHS03, KW90]. The minimization version of the problem is approximable within a factor of 2 in polynomial time [BBF99].

*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, and by the U.S. Department of Energy under contract DE–AC05–00OR22725.

†School of Information and Communication Technologies, Griffith University, Brisbane QLD 4111, Australia; frank@dehne.net

‡School of Electrical Engineering and Computer Science, University of Newcastle, Callaghan NSW 2308, Australia; {mfellows,fran}@cs.newcastle.edu.au

§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.; langston@cs.utk.edu

¶The Mechanics Institute, Bob's Farm NSW 2316, Australia; wonganellawines@telstra.com

The FVS problem has been extensively studied from the parameterized point of view [BBG00, Bod94, DF92, DF99, KPS04, RSS02, RS05]. A parameterized problem is said to be *fixed-parameter tractable* (FPT) if it can be solved in time $f(k)n^c$ for some function f (unrestricted), where n is the total input size, k is the declared parameter and c is a constant independent of k and n. This running time may be written as $\mathcal{O}^*(f(k))$ in the notation introduced by Woeginger [Woe03] that focuses attention on the exponential time costs due to the parameter and ignores the polynomial time costs due to the overall input size. Highlights of previous research on the FVS problem in the parameterized framework include:

• A randomized FPT algorithm due to Becker et al. [BBG00] running in time $\mathcal{O}(4^k kn)$ finds a minimum feedback vertex set of size k with probability at least $1-(1-4^{-k})^{c4^k}$ for an arbitrary constant c.

• After several rounds of improvement, the best previous deterministic FPT algorithm, due to Kanj et al. [KPS04], has a running time of $\mathcal{O}((2\lg k + 2\lg\lg k + 18)^k n^2)$, or expressed qualitatively $\mathcal{O}^*(2^{\mathcal{O}(k \lg\lg k)})$. (This algorithm depends substantially on a previous algorithm due to Raman et al. [RSS02] having a similar qualitative running time.) The basic idea for these algorithms is to branch on short cycles in a bounded search tree approach (see [Nie02] for a survey of FPT techniques). A recent algorithm in [RS05] has a running time of $\mathcal{O}^*(2^{\mathcal{O}(k\frac{\lg k}{\lg\lg k})})$,

A number of problems concerning FVS have notably remained open:

(1) Is there an $\mathcal{O}^*(2^{\mathcal{O}(k)})$ FPT algorithm for FVS on undirected graphs?

(2) Is there a polynomial-time algorithm that kernelizes FVS on undirected graphs to a kernel of size polynomial in k? (See [Nie02] for a discussion of kernelization and FPT.)

(3) Is the FVS problem in FPT for directed graphs?

In this paper we answer the first of these by an approach based on the relatively new technique of *iterative compression* [RSV03, DFRS04].

In the next section we provide a brief discussion of this approach and its application to the FVS problem. In §3 we describe our FPT algorithm for the solution-compression form of the FVS problem. In §4 we prove an "optimality" result for our algorithm (giving a lower bound on the possibility of improvements). In §5 we conclude with a review of open problems.

2 Iterative Compression Applied to FVS

The FPT technique of *iterative compression* seems first to have appeared in an FPT algorithm devised by Reed, Smith and Vetta for the problem of deleting k vertices to render a graph bipartite [RSV03]. The approach was articulated as a general FPT design technique in [DFRS04]. Some applications of the method can be found in [RSV03, DFRS04, Ma04].

Here we use this approach to solve the FVS decision problem by recursively solving the following constructive *solution-compression* form of the problem:

SOLUTION COMPRESSION FOR FEEDBACK VERTEX SET

Instance:	An undirected graph $G = (V, E)$ (loops and multiple edges are allowed), an annotated subset $U \subseteq V$ of vertices, a *solution set* $S \subseteq V - U$ such that $G - S$ is acyclic, where $\lvert S\rvert = k + 1$.
Parameter:	k
Output:	Either: (1) a solution set S' of size k, or (2) NO (i.e., no solution of size k is possible).

We employ an FPT algorithm for the above compression form of the FVS problem in the following way. We recursively solve a constructive form of the problem of deciding whether a graph $G = (V, E)$ admits a

feedback vertex set of size k with vertices to be chosen from $V-U$. In this constructive form of the decision problem we are required either to produce a solution of size k, if one exists, or to return NO otherwise.

Given an instance $(G=(V,E), U \subseteq V, k)$, we recursively address the constructive decision problem for the instance $(G-v, U, k)$ where v is an arbitrarily chosen vertex in $V-U$. If this recursive call on $G-v$ returns NO, that is, no k-vertex solution for $G-v$ is possible, then clearly the correct answer for G is NO as well.

Alternatively, if the recursive call on the instance $(G-v, U, k)$ returns a k-element solution $S \subseteq V-U$, then $S \cup \{v\}$ is a solution of size $k+1$ for G. We now employ as a subroutine the FPT algorithm for the solution compression problem. If $f(k)n^c$ is the running time we have achieved for SOLUTION COMPRESSION FOR FVS, then our recursive solution to the constructive decision problem runs in time $f(k)n^{c+1}$, where n is the number of vertices in the graph G.

In the next section we describe our FPT algorithm for the problem of SOLUTION COMPRESSION FOR FVS.

3 An FPT Algorithm for FVS Solution Compression

We will use the following *reduction rules* that can be easily applied to simplify (or summarily decide) an instance of the problem. Recall that some vertices (the vertices in U in the problem definition) may be annotated as *not* to belong to a solution set.

Rule 1: The Degree One Rule. If v is a vertex (annotated or not) of degree 1 in G, then delete v and adjust the rest of the input data accordingly.

Rule 2: The Degree Two Rule. If v is a vertex (annotated or not) 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 $a=b$) and adjust the rest of the input data accordingly.

Rule 3: Annotation Contraction. If u and v are adjacent annotated vertices (that is, $u,v \in U$) then contract one of the edges between u and v and adjust the rest of the input data accordingly.

Rule 4: The Loop Rules. If there is a loop on an annotated vertex v then answer NO. If there is a loop on an unannotated vertex $v \in V-U$ then take v into the solution set, that is, reduce to the instance $(G-v, U, k-1)$.

Rule 5: Multiedge Reduction. If there are more than two edges between u and v (annotated or not) then delete all but two of these.

Rule 6: Multiedge Selection. If there is an annotated vertex u that is connected by two edges to an unannotated vertex v, then take v into the solution set, that is, reduce to the instance $(G-v, U, k-1)$.

The soundness of all these reduction rules is self-evident. In time $\mathcal{O}(n)$ we can determine if any of the above reduction rules can be applied to a problem instance. Note that applications of the rules may cascade. We say that an instance is *reduced* if none of the reduction rules can be applied.

Note that if we reduce an instance (G,U,k) to an instance (G',U',k') by a series of applications of the above reduction rules, then given a solution S' of size k' for G', we can in time $\mathcal{O}(n)$ recover a solution S of size k for G. We will always harmlessly assume that the instance we are working with is reduced.

Algorithm for SOLUTION COMPRESSION FOR FVS

Input: A reduced instance $(G=(V,E), U \subseteq V, k)$, and a solution $S \subseteq V-U$ of size $k+1$.
Output: Either a solution of size at most k, or NO if none exists.

Step 1: Branch on all 2^{k+1} subsets of S. The branch corresponding to a subset $A \subseteq S$ represents the search for a size k solution S' that includes the vertices of A, that is, $A \subseteq S'$, and that does not include any of the vertices of $S-A=A'$.

Thus, in the instance (G', U', k') that represents this branch:
(1) the vertices of A are deleted,
(2) the vertices of A' are annotated,
(3) $k' = k - |A|$, and
(4) the instance is further reduced according to Reduction Rules (1-6).

We will argue below that for the reduced instance $(G' = (V', E'), U', k')$ considered on any of these 2^{k+1} branches of Step 1, we have either:
(i) $|V' - U'| \leq 4k$, or
(ii) we can immediately determine that the answer is NO..

Step 2: On each branch of Step 1, exhaustively analyze the resulting reduced instance by checking each k'-element subset of the unannotated vertices to see if any provides a solution.

Step 2 requires checking at most $\binom{4k}{k}$ subsets. A simple bound on the running time of our algorithm is $\mathcal{O}(c^k n^2)$ where $c = 18.963$, since

$$\binom{4k}{k} \approx (9.4815)^k$$

A more refined version of our algorithm, detailed in §3.3, runs in time $\mathcal{O}^*(10.567^k)$.

3.1 The Reduced Instance Bound for Step 1.

The correctness of the algorithm is obvious because of its extreme simplicity. What is less obvious is the claimed $4k$ bound on the number of unannotated vertices on the reduced instance branches generated in Step 1 that need to be considered further.

Let $A \subseteq S$ and $A' = S - A$ as in the description of Step 1. The immediate instance graph G' on the A-branch of Step 1 consists of two sets of vertices:
(1) The (now) annotated vertices of A', where we have the bound $|A'| \leq k + 1$.
(2) The other vertices, which we denote F. Some of these may be annotated.

This immediate branch instance is further reduced, and this reduction process may result in some modification of the above picture. For example, connected components of the subgraph generated by A would be contracted to a single vertex, by repeated applications of Rule 3. To simplify the argument, we will assume that the immediate branch instance is already reduced so that our description of the vertices of G' as partitioned into A' and F is accurate (these sets would be modified by further reduction, but a bipartition with the same properties we make use of below would result in any case). The following structural claims hold.

Lemma 1 *The subgraph $\langle F \rangle$ induced by F is acyclic.*

Proof. Otherwise S would not be a solution for G. ■

Henceforth we may use F (for convenience) to denote also the forest induced by the vertices in the vertex set F.

Lemma 2 *Each leaf l of the forest F is adjacent to at least two distinct vertices in A'.*

Proof. In view of Lemma 1 and Reduction Rules 1 and 2, there must be at least two edges connecting l to vertices in A'. Reduction Rule 6 would apply if l were connected to only one vertex of A'. ■

The vertices in the forest F can be partitioned into three sets. Let L denote the leaves of F, let J be the vertices that have degree 2 in the forest subgraph $\langle F \rangle$. We will refer to the vertices of J as the *subdivision vertices* of F. Let B, the *branch vertices* of F, be the vertices of degree at least 3 in the subgraph $\langle F \rangle$.

Lemma 3 *Each vertex $j \in J$ is connected to at least one vertex of A'.*

Proof. Otherwise, in view of Lemma 1, Reduction Rule 2 would apply. ■

Definition 1 *Let F be a forest with vertex set partitioned into the three sets: (1) the leaves L, (2) the subdivision vertices J, and (3) the branch vertices B of F. A* matching of the J-vertices of F of size r *consists of:*
(1) r mutually disjoint 2-element subsets $\{x_i, y_i\} \subseteq J$, $1 \leq i \leq r$,
(2) for each i, $1 \leq i \leq r$, a path ρ_i in F from x_i to y_i, subject to the requirement that for $i \neq j$, the paths ρ_i and ρ_j are vertex disjoint. The potential $\pi(F)$ *of the forest F is defined to be the sum of the number of leaves $|L|$ of F and the size of a maximum matching of the J-vertices. (See Figure 1 for an example.)*

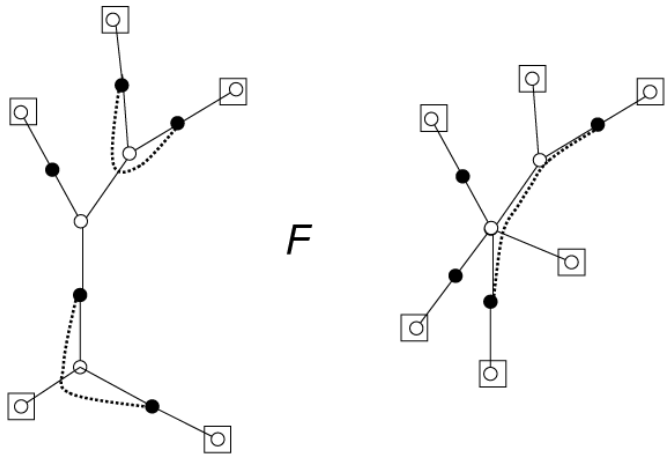

Figure 1: A maximum matching of the subdivision vertices ("J vertices") of the forest F, showing that $\pi(F) = 11 + 3 = 14$.

Lemma 4 *Suppose that for the reduced instance (G', U', k') with vertex set partitioned into A' and F as above we have $\pi(F) \geq k' + |A'|$. Then the answer for this instance is NO.*

Proof. If it were a YES-instance (for k') then there would be a feedback vertex set S' consisting of at most k' unannotated vertices. But then there would necessarily be at least $|A'|$ leaves and $J - matching$ paths ρ_i in F having empty intersection with S'. Since $S' \cap A' = \emptyset$ (because the vertices of A' are annotated), there are at least $|A'|$ *virtual edges* or *virtual loops* connecting the vertices of A' through $F - S'$. (For example, if a leaf l of F is not in S', then by Lemma 2 it is adjacent to two vertices a and b in A', which we consider here as a *virtual edge* between a and b. If the path ρ_i in F from the J-vertex x_i to the J-vertex y_i does not contain any vertices in S', then together with the connections of x_i and y_i to the set A' guaranteed by Lemma 3, we have what can be considered either a *virtual edge* between A' vertices — or a *virtual loop*, in

case the A'-adjacencies guaranteed for x_i and y_i by Lemma 3 connect these vertices to the *same* vertex of A'.) Joining the vertices of A' by $|A'|$ virtual edges or virtual loops necessarily implies that there is a cycle not including any vertices of S', that is, that S' is not a feedback vertex set, a contradiction. ■

Lemma 5 *For any forest F on m vertices, $\pi(F) \geq (m+1)/2$.*

The proof of Lemma 5 is somewhat involved, and we defer the discussion to the next subsection.

Lemma 6 *If on the A-branch of Step 1 we have a reduced instance (G', U', k') where the vertices of G' are partitioned into A' and F as in the discussion above, and where $|F| \geq 4k+1$, then this is a NO-instance.*

Proof. By Lemma 5, $\pi(F) \geq 2k+1$. The rest follows by Lemma 4, since $|A'| \leq k+1$ and $k' \leq k$. ■

3.2 The Proof of Lemma 5

Lemma 5 states that any forest F on n vertices has potential $\pi(F) \geq (n+1)/2$.

Proof. There are two parts to the argument:

(1) We prove the Lemma for trees of maximum degree 3. The proof is by structural induction.

(2) We then prove the Lemma for arbitrary trees by minimum counterexample, using (1) essentially as the base case. The Lemma for arbitrary forests follows almost trivially.

As it is simpler, we treat the second step first, assuming (1) for the moment. Let T be a counterexample tree having a minimum number of vertices, $|T| = m$. By (1), T must have at least one vertex v of degree 4 or more. We consider *breaking* T into two trees T_1 and T_2 as illustrated in Figure 2. The vertex v is "broken" into two copies by choosing an incident edge e and "detaching" T_1 as the subtree joined to the rest of T at v by the edge e, and by making one of the copies of v a leaf in T_1. The tree T_2 consists of T with e and the subtree (T_1) attached by e removed. Thus in T_2, the degree of (the other copy of) v is decreased by 1, and we have $|T_1| + |T_2| = |T| + 1$.

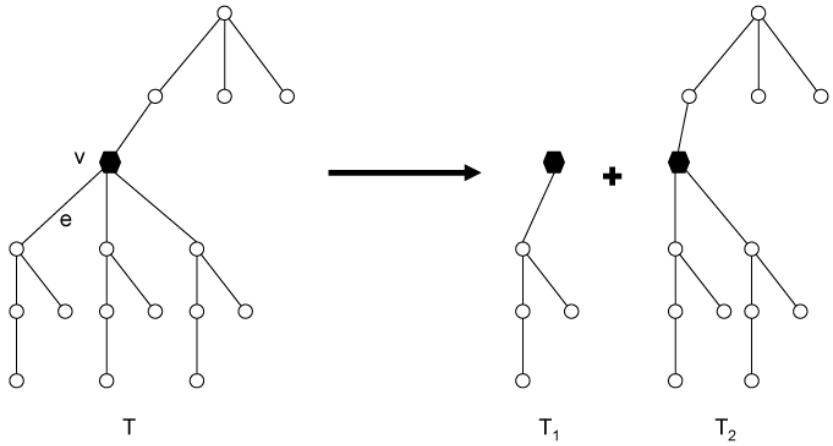

Figure 2: Breaking T into $T1$ and $T2$ at v.

Let $m_i = |T_i|$ for $i = 1, 2$. Thus $m_1 + m_2 = m + 1$. The Lemma must hold for each of the trees T_i, since T is presumed to be a minimum counterexample. Therefore $\pi(T_i) = (m_i + 1)/2$ for $i = 1, 2$. Choose suitable J_i-matchings in the T_i that witness this. Combining these witness structures in T ("putting T back together") gives

$$\pi(T) \geq (m_1 + 1)/2 + (m_2 + 1)/2 - 1$$

with the -1 term because a leaf is lost when the two copies of v are fused back together. (Note that the copy of v in T_2 has degree at least 3 in T_2 and therefore does not belong to J_2, so that there are no other losses in combining the two witness structures.) This gives:

$$\pi(T) \geq (m + 3)/2 - 1 = (m + 1)/2$$

and the Lemma is proved, assuming (1).

To prove the Lemma for trees of maximum degree 3, we induct on the structure of such trees. Each such tree T is considered to be rooted at a vertex r, where either: (1) r is a leaf of T, or (2) r has degree 2 in T. We will refer to (1) and (2) as the *types* of the rooted trees we discuss.

Trees of maximum degree 3 are generated by two operations on these rooted trees:
(i) A unary operation $x(T)$ (*extension* of T) that can be applied to rooted trees of type either (1) or (2) and that consists in adding a new vertex r' connected to r, with r' becoming the root of the resulting "extended" tree.
(ii) A binary operation $T_1 \oplus T_2$ (*join* of T_1 and T_2) that applies only when both T_1 and T_2 are of type (1), that is, have roots of degree 1. In this operation, the roots of the two trees are identified, resulting in a rooted tree of type (2).

The two operations are illustrated in Figure 3.

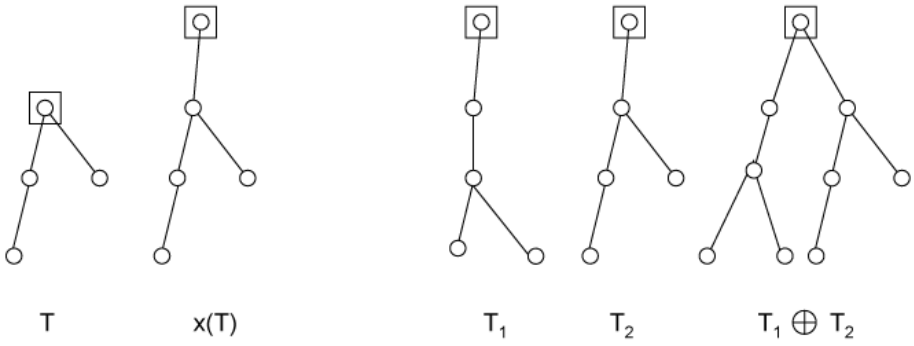

Figure 3: The parsing operations for trees of maximum degree 3.

An elementary induction shows that all trees of maximum degree 3 can be parsed in terms of these two operations on (smaller) rooted trees. For a rooted tree of type (1) or type (2) our induction hypothesis is as follows. Here we consider that the vertices of T are partitioned into the four sets: $\{r\}$, L, J and B, of the root, the leaves, the subdivision vertices and the branch vertices, respectively, as in earlier discussions, but with the exception of the root. In particular, here we do not consider that the root belongs to J, even for rooted trees of type (2).

Induction Hypothesis. One of the following claims holds:
(1) $|J|$ is even and the J-vertices of T admit a perfect matching in the sense defining $\pi(T)$, or
(2) $|J|$ is odd and the J-vertices can be matched in T with the exception of one vertex $u \in J$, and furthermore

the matching can be accomplished so that there is a path from u to the root r that is disjoint from the paths in T that realize the J-matching.

The induction hypothesis is illustrated in Figure 4.

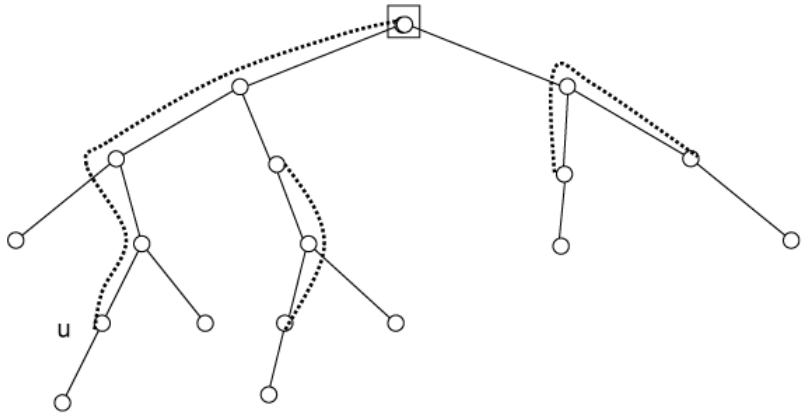

Figure 4: An example of the induction hypothesis for $|J|$ odd.

It is straightforward to verify the several cases of the induction step for the two parsing operations. For example, we can verify that for the operation $T_1 \oplus T_2$ where both T_1 and T_2 satisfy case (2) of the induction hypothesis with unmatched J-vertices, respectively, u_1 and u_2, the outcome T of the operation satisfies case (1) of the induction hypothesis. In this outcome, the paths from the unmatched vertices u_i to the joined roots r_i are combined to form a path matching u_1 to u_2 in T. We leave the other cases to the reader.

By the above inductive argument, it follows that there can be at most one unmatched J-vertex in a maximum J-matching in an (unrooted) tree T of maximum degree 3, where the vertices of T are partitioned into the three sets L, J and B. Since $|B| = |L| - 2$ and therefore

$$m = |L| + |J| + |B| = 2|L| + |J| - 2$$

we have

$$\pi(T) \geq |L| + (|J| - 1)/2 = (m + 1)/2$$

which proves the Lemma for trees of maximum degree at most 3. ■

3.3 A More Efficient Version

Lemma 4 shows that there is a simple way to improve the efficiency of our algorithm. On the branch of Step 1 corresponding to a subset A of the $(k+1)$-sized solution S, we can answer NO if for the reduced instance we have $\pi(F) \geq k' + |A'|$. Since $k' = k - |A|$ and $|A'| = k + 1 - |A|$, and using Lemma 5, the total bound on the number of possible solutions explored in Steps 1 and 2 is

$$\sum_{i=0}^{k} \binom{k+1}{i} \binom{2((k+1-i)+(k-i)-1)-1}{k-i} = \sum_{i=0}^{k} \binom{k+1}{i} \binom{4k-4i-1}{k-i}$$

Define

$$f(x,k) = \binom{k}{x} \binom{4(k-x)}{k-x}$$

and suppose $f(x,k)$ is maximized for $x^* = x(k)$. Then our sum above is bounded by $(k+1) \cdot f(x^*, k+1)$.

We next work out two estimates $x_1(k)$ and $x_2(k)$ such that

$$x_1(k) \leq x^*(k) \leq x_2(k)$$

and we will therefore have a bound on our sum of

$$(k+1) \cdot \binom{k+1}{x_2(k+1)} \binom{4((k+1) - x_1(k+1))}{(k+1) - x_1(k+1)}$$

(The reason for the two estimates is that the first part of $f(x,k)$ increases with x, and the second part decreases with x.)

We study the ratio $f(x,k)/f(x+1,k)$. The maximizing value x^* is located (essentially) at the point where this ratio is equal to 1. Assuming that k is large, this ratio is approximately:

$$\frac{f(x,k)}{f(x+1,k)} \approx \left(\frac{x+1}{k-x}\right)(4)(4/3)^3$$

This yields the estimates:
$x_1(k) = (27/283)k$ and
$x_2(k) = (28/283)k$.

Using the well-known bound (based on Stirling's approximation for $n!$) that

$$\binom{ak}{bk} \leq \left(\frac{a^a}{b^b(a-b)^{a-b}}\right)^k$$

for constants $a > b$, we obtain the bound on our total cost sum of $(k+1)(10.567)^k$.

4 Optimality

Our FPT algorithm for the problem of SOLUTION COMPRESSION FOR FVS yields, by the approach of §2, an FPT algorithm for the parameterized FEEDBACK VERTEX SET problem that runs in time $\mathcal{O}(c^k n^3)$ where $c = 10.567$. In qualitative terms, we have given an algorithm with a running time of the form $\mathcal{O}^*(2^{\mathcal{O}(k)})$. We next show that this is, in a qualitative sense, "optimal" for the problem.

Theorem 1 *There can be no FPT algorithm for* FEEDBACK VERTEX SET *with a running time of the form* $\mathcal{O}^*(2^{o(k)})$ *unless FPT* $= M[1]$.

Proof. Determining whether a graph on n vertices has a vertex cover of size at most $k \log n$, where the parameter is k, termed the $k \log n$ VERTEX COVER problem, is complete for the parameterized complexity class $M[1]$ [DEFPR03, CF04]. The theorem follows because there is a simple linear-size and parameter-preserving (i.e., $k' = k$) polynomial-time reduction from VERTEX COVER to FEEDBACK VERTEX SET, by simply replacing each edge of the VERTEX COVER instance with a pair of parallel edges. Thus if there were an FPT algorithm for FEEDBACK VERTEX SET running in time $\mathcal{O}^*(2^{o(s)})$ where s is the size of the feedback vertex set, then we would have an algorithm for the $k \log n$ VERTEX COVER problem running in time $\mathcal{O}^*(2^{o(k \log n)})$, but as shown in [CJ03], this is an FPT running time. By the completeness of the $k \log n$ VERTEX COVER problem for $M[1]$ [DEFPR03], we would have FPT $= M[1]$. ■

Remark 1 *The consequence FPT* $= M[1]$ *is highly unlikely, since it is known that FPT* $= M[1]$ *if and only if satisfiability of 3SAT instances on* n *variables can be decided in time* $\mathcal{O}^*(2^{o(n)})$. *(See [DEFPR03, CF04] for further information and discussion.)*

Remark 2 *A number of other "FPT optimality" results have been shown for various problems [DFR03, CJ03]. A notable example is the parameterized* PLANAR DOMINATING SET *problem, for which there is an FPT algorithm with a running time of* $\mathcal{O}^*(2^{\mathcal{O}(\sqrt{k})})$ *[ABFKN02]. It has been shown that there can be no FPT algorithm for this problem with a running time of the form* $\mathcal{O}^*(2^{o(\sqrt{k})})$ *unless FPT* $= M[1]$ *[CJ03].*

5 Open Problems

Is there a polynomial-time kernelization algorithm for FVS on undirected graphs that reduces an instance (G, k) to (G', k') where $k' \leq k$ and the size of G' is bounded by a polynomial in k?

The reduction rules that we have employed here are all local and elementary in character. It would be interesting to explore if global "crown type" reduction rules for the problem might be possible, as has turned out to be usefully the case for VERTEX COVER [ACFLSS04]. Such reduction rules might also be of use in addressing the above open problem.

The potential practical significance of our algorithm should also be investigated. Our approach to the FVS problem here is a new one, and the branching in Step 1 is "not expensive". The broad parallelism of Step 1 might be usefully exploited in highly parallel implementations.

6 Recent Independent Work

Through recent email conversations we have just received a manuscript by Guo et.al. [GGHNW05] in which they independently present an improved FVS algorithm that is very similar to ours and requires time $\mathcal{O}(c^k n^{\mathcal{O}(1)})$. Their independent result is also based on the iterative compression methodology. For our method, the constant c is 10.567 whereas it's value is not determined in [GGHNW05].

References

[ABFKN02] J. Alber, H. L. Bodlaender, H. Fernau, T. Kloks and R. Niedermeier. Fixed parameter algorithms for Dominating Set and related problems on planar graphs. *Algorithmica* 33 (2002), 461–493.

[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.

[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.

[BGNR98] R. Bar-Yehuda, D. Geiger, J. Naor and R. Roth. Approximation algorithms for the feedback vertex set problem with applications to constraint satisfaction and Bayesian inference. *SIAM Journal on Computing* 27 (1998), 942–959.

[Bod94] H. Bodlaender. On disjoint cycles. *International Journal of Foundations of Computer Science* 5 (1994), 59–68.

[CF04] Y. Chen and J. Flum. On miniaturized problems in parameterized complexity theory. *Proceedings of the First International Workshop on Parameterized and Exact Computation*, Springer-Verlag, *Lecture Notes in Computer Science* vol. 3162 (2004), 108–120.

[CJ03] L. Cai and D. Juedes. On the existence of subexponential parameterized algorithms. *Journal of Computer and System Sciences* 67 (2003), 789–807.

[DEFPR03] R. Downey, V. Estivill-Castro, M. Fellows, E. Prieto-Rodriguez and F. Rosamond. Cutting up is hard to do: the complexity of k-cut and related problems. *Electronic Notes in Theoretical Computer Science* 78 (2003), 205–218.

[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.

[DFR03] F. Dehne, M. Fellows and F. Rosamond. An FPT algorithm for set splitting. *Proceedings of the 29th Workshop on Graph Theoretic Concepts in Computer Science (WG 2003)*, Springer-Verlag, *Lecture Notes in Computer Science* 2880 (2003), 180–191.

[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.

[ENSS98] G. Even, J. Naor, B. Scheiber and M. Sudan. Approximating minimum feedback sets and multicuts in directed graphs. *Algorithmica* 20 (1998), 151–174.

[FHS03] M. Fellows, M. Hallett and U. Stege. Analogs and duals of the MAST problem for sequences and trees. *Journal of Algorithms* 49 (2003), 192–216.

[FHPSS04] C. Fried, W. Hordijk, S.J. Prohaska, C.R. Stadler and P.F. Stadler. The footprint sorting problem. J. Chem. Inf. Comput. Sci., 44 (2), 332 -338, 2004

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of $\mathcal{NP}$-Completeness.* W.H. Freeman, 1979.

[GGHNW05] J. Guo, J. Gramm, F. Hueffner, R. Niedermeier, S. Wernicke. *Improved Fixed-Parameter Algorithms for Two Feedback Set Problems*. Manuscript.

[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.

[KW90] A. Kunzmann and H. Wunderlich. An analytical approach to the partial scan problem. *Journal of Electronic Testing: Theory and Applications* 1 (1990), 163–174.

[Ma04] D. Marx. Chordal deletion is fixed-parameter tractable. Manuscript, 2004.

[Nie02] R. Niedermeier. *Invitation to fixed-parameter algorithms*, Habilitationschrift, University of Tubingen, 2002. (Electronic file available from R. Niedermeier.)

[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.

[RS05] V. Raman and S. Saurabh. Faster algorithms for feedback vertex set. To appear in *Proceedings of the 2nd Brazilian Symposium on Graphs, Algorithms and Combinatorics*, (GRACO 2005), April 27-29, 2005.

[RSV03] B. Reed, K. Smith, and A. Vetta. Finding odd cycle transversals. *Operations Research Letters* 32 (2004), 299–301.

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