FPT Algorithms for Connected Feedback Vertex Set

Neeldhara Misra* Geevarghese Philip* Venkatesh Raman* Saket Saurabh*
Somnath Sikdar†

Abstract

We study the recently introduced CONNECTED FEEDBACK VERTEX SET (CFVS) problem from the view-point of parameterized algorithms. CFVS is the connected variant of the classical FEEDBACK VERTEX SET problem and is defined as follows: given a graph $G = (V, E)$ and an integer k, decide whether there exists $F \subseteq V$, $|F| \leq k$, such that $G[V \setminus F]$ is a forest and $G[F]$ is connected.

We show that CONNECTED FEEDBACK VERTEX SET can be solved in time $O(2^{O(k)}n^{O(1)})$ on general graphs and in time $O(2^{O(\sqrt{k}\log k)}n^{O(1)})$ on graphs excluding a fixed graph H as a minor. Our result on general undirected graphs uses, as a subroutine, a parameterized algorithm for GROUP STEINER TREE, a well studied variant of STEINER TREE. We find the algorithm for GROUP STEINER TREE of independent interest and believe that it could be useful for obtaining parameterized algorithms for other connectivity problems.

1 Introduction

FEEDBACK VERTEX SET (FVS) is a classical NP-complete problem and has been extensively studied in all subfields of algorithms and complexity. In this problem we are given an undirected graph $G = (V, E)$ and a positive integer k as input, and the goal is to check whether there exists a subset $F \subseteq V$ of size at most k such that $G[V \setminus F]$ is a forest. This problem originated in combinatorial circuit design and found its way into diverse applications such as deadlock prevention in operating systems, constraint satisfaction and Bayesian inference in artificial intelligence. We refer to the survey by Festa, Pardalos and Resende [13] for further details on the algorithmic study of feedback set problems in a variety of areas like approximation algorithms, linear programming and polyhedral combinatorics.

In this paper we focus on the recently introduced connected variant of FEEDBACK VERTEX SET, namely, CONNECTED FEEDBACK VERTEX SET (CFVS). Here, given a graph $G = (V, E)$ and a positive integer k, the objective is to check whether there exists a vertex-subset F of size at most k such that $G[V \setminus F]$ is a forest and $G[F]$ is connected. Sitters and Grigoriev [17] recently introduced this problem and obtained a polynomial time approximation scheme (PTAS) for CFVS on planar graphs. We find it a bit surprising that the connected version of FVS has not been studied in the literature until now. This is in complete contrast to the fact that the connected variants of other problems, like VERTEX COVER—CONNECTED VERTEX COVER, and DOMINATING SET—CONNECTED DOMINATING SET are extremely well-studied in the literature (See, e.g, work by Mölle et al. [20] and Fomin et al. [15], respectively.). In this paper, we initiate the algorithmic study of CFVS from the view-point of parameterized algorithms.

Parameterized complexity is a two-dimensional generalization of "P vs. NP" where, in addition to the overall input size n, one studies how a secondary measurement (called the *parameter*), that captures additional relevant information, affects the computational complexity of the

*The Institute of Mathematical Sciences, Chennai, India. {neeldhara|gphilip|vraman|saket}@imsc.res.in
†RWTH Aachen University, Aachen, Germany. sikdar@cs.rwth-aachen.de

problem in question. Parameterized decision problems are defined by specifying the input, the parameter, and the question to be answered. The two-dimensional analogue of the class P is decidability within a time bound of $f(k)n^c$, where n is the total input size, k is the parameter, f is some computable function and c is a constant that does not depend on k or n. A parameterized problem that can be decided in such a time-bound is termed *fixed-parameter tractable* (FPT). For general background on the theory of fixed-parameter tractability, see the books by Downey and Fellows [12], Flum and Grohe [14], and Niedermeier [23].

FVS has been extensively studied in the context of parameterized algorithms. The earliest known FPT algorithms for FVS go back to the early 90's (e.g, Bodlaender's paper at WG 1991 [2]). After several rounds of improvements, the current best FPT algorithm for FVS runs in time $O(3.83^k kn^2)$ [5].

In this paper, we show that CFVS can be solved in time $O(2^{O(k)}n^{O(1)})$ on general graphs and in time $O(2^{O(\sqrt{k}\log k)}n^{O(1)})$ on graphs excluding a fixed graph H as a minor. Most of the known FPT algorithms for connectivity problems enumerate *all* minimal solutions and then try to connect each solution using an algorithm for the STEINER TREE problem. For instance, this is the case with the existing FPT algorithms for CONNECTED VERTEX COVER(e.g, in the algorithm due to Mölle et al. [20]). The crucial observation which the algorithms for CONNECTED VERTEX COVER rely on is that there are at most 2^k *minimal* vertex covers of size at most k. However, this approach fails for CFVS as the number of minimal feedback vertex sets of size at most k is $\Omega(n^k)$ (consider a graph that is a collection of k vertex-disjoint cycles each of length approximately n/k). To circumvent this problem, we make use of "compact representations" of feedback vertex sets. A compact representation is simply a collection of families of mutually disjoint sets, where each family represents a number of different feedback vertex sets. This notion was defined by Guo et al. [18] who showed that the set of all minimal feedback vertex sets of size at most k can be represented by a collection of set-families of size $O(2^{O(k)})$.

We use compact representations to obtain an FPT algorithm for CFVS in Section 3. In order to do this we need an FPT algorithm for a general version of STEINER TREE, namely GROUP STEINER TREE (GST), which is defined as follows: Given a graph $G = (V, E)$; $|V| = n, |E| = m$, subsets $T_i \subseteq V$, $1 \leq i \leq l$, and an integer p, does there exist a subgraph of G on p vertices that is a tree T and includes at least one vertex from each T_i? Observe that when the T_i's are each of size one, then GST is the STEINER TREE problem. Our FPT algorithm for GST runs in $O(2^l \cdot n^{O(1)})$ time and polynomial space. It uses a reduction to a directed version of STEINER TREE, called DIRECTED STEINER OUT-TREE, which we show to be fixed-parameter tractable. We note that GST is known to be of interest to database theorists, and that it has been studied by Ding et al. [10], where an algorithm with running time $O(3^l \cdot n + 2^l \cdot (n + m))$ (that uses exponential space) is discussed. We also show that CFVS does not admit a polynomial kernel (See Section 2) on general graphs but has a quadratic kernel on the class of graphs that exclude a fixed graph H as minor. Finally, in Section 4 we design a subexponential-time algorithm for CFVS on graphs excluding some fixed graph H as a minor using the theory of bidimensionality. This algorithm is obtained using an $O^*(w^{O(w)})$-time[1] algorithm that computes an optimal connected feedback vertex set in graphs of treewidth at most w.

2 Preliminaries

In this section we state some basic definitions related to parameterized complexity and graph theory, and give an overview of the notation used in this paper. To describe running times of algorithms we sometimes use the O^* notation. Given $f : \mathbb{N} \to \mathbb{N}$, we define $O^*(f(n))$ to be

[1]We use the O^* notation to ignore polynomial factors.

$O(f(n) \cdot p(n))$, where $p(\cdot)$ is some polynomial function. That is, the O^* notation suppresses polynomial factors in the running-time expression.

A parameterized problem Π is a subset of $\Gamma^* \times \mathbb{N}$, where Γ is a finite alphabet. An instance of a parameterized problem is a tuple (x, k), where k is called the parameter. A central notion in parameterized complexity is *fixed-parameter tractability (FPT)* which means, for a given instance (x, k), decidability in time $f(k) \cdot p(|x|)$, where f is an arbitrary function of k and p is a polynomial in the input size. The notion of *kernelization* is formally defined as follows.

Definition 1. [Kernelization] [14, 23]
A kernelization algorithm for a parameterized problem $\Pi \subseteq \Gamma^* \times \mathbb{N}$ is an algorithm that, given $(x, k) \in \Gamma^* \times \mathbb{N}$, outputs, in time polynomial in $|x| + k$, a pair $(x', k') \in \Gamma^* \times \mathbb{N}$ such that (a) $(x, k) \in \Pi$ if and only if $(x', k') \in \Pi$ and (b) $|x'|, k' \leq g(k)$, where g is some computable function. The output instance x' is called the kernel, and the function g is referred to as the size of the kernel. If $g(k) = k^{O(1)}$ (resp. $g(k) = O(k)$) then we say that Π admits a polynomial (resp. linear) kernel.

We say that a graph G (undirected or directed) *contains* a graph H if H is a subgraph of G. Given a directed graph (digraph) $D = (V, A)$, we let $V(D)$ and $A(D)$ denote the vertex and arc set of D, respectively. A vertex $u \in V(D)$ is an *in-neighbor* (*out-neighbor*) of $v \in V(D)$ if $uv \in A$ ($vu \in A$, respectively). The in- and out-neighborhood of a vertex v are denoted by $N^-(v)$ and $N^+(v)$, respectively. The *in-degree* $d^-(v)$ (resp. *out-degree* $d^+(v)$) of a vertex v is $|N^-(v)|$ (resp. $|N^+(v)|$). An *oriented tree* is a digraph obtained from a tree by converting each edge to an arc with one of the two possible directions [1]. We say that a subdigraph T of D with vertex set $V_T \subseteq V(D)$ is an *out-tree* if T is an oriented tree with only one vertex r of in-degree zero (called the *root*). The vertices of T of out-degree zero are called *leaves* and every other vertex is called an *internal vertex*.

In this paper we look at the following parameterized problem:

Connected Feedback Vertex Set

Input:	An undirected graph $G = (V, E)$ and an integer k.
Parameter:	The integer k.
Question:	Does G contain a set F of at most k vertices such that

1. the graph obtained by deleting the vertices of F from G is a forest, and
2. the subgraph of G induced on the set F is a connected graph?

3 Connected Feedback Vertex in General Graphs

In this section we give an FPT algorithm for CFVS on general graphs. We start by describing an FPT algorithm for the Group Steiner Tree problem which is crucially used in our algorithm for CFVS.

3.1 Group Steiner Tree

The Group Steiner Tree (GST) problem is defined as follows:

Input:	An undirected graph $G = (V, E)$; vertex-disjoint subsets $S_1, \ldots, S_l \subseteq V$; and an integer p.

Parameter:	The integer l.
Question:	Does G contain a tree on at most p vertices that includes at least one vertex from each S_i?

Our fixed-parameter algorithm for GST first reduces it to DIRECTED STEINER OUT-TREE (defined below) which we then show to be fixed-parameter tractable.

Input:	A directed graph $D = (V, A)$; a distinguished vertex $r \in V$; a set of terminals $S \subseteq V$; and an integer p.
Parameter:	The integer $l = \|S\|$.
Question:	Does D contain an out-tree on at most p vertices that is rooted at r and that contains all the vertices of S?

Lemma 1. *The GST problem reduces to the* DIRECTED STEINER OUT-TREE *problem.*

Proof. Given an instance $(G = (V, E), S_1, \ldots, S_l, p)$ of GST, construct an instance of DIRECTED STEINER OUT-TREE as follows. Let $S = \{r, s_1, s_2, \ldots, s_l\}$ be a set of $(l + 1)$ new vertices, that is, $r \notin V$ and $s_i \notin V$ for $1 \leq i \leq l$. Let $X = V \cup S$ and $A = \{uv, vu : \{u, v\} \in E\} \cup \bigcup_{i=1}^{l} \{xs_i : x \in S_i\}$. Now, for every $u \in V$, introduce a directed path P_u from r to u on $n = |V|$ vertices. More precisely, let $V_u = \{u_1, u_2, \cdots, u_n\}$. We let $V' = X \bigcup \cup_{u \in V} V_u$ and let $A' = A \cup \{ru_1, u_n u \mid u \in V\} \cup \{u_i u_{i+1} \mid 1 \leq i < n, u \in V\}$. Let $D = (V', A')$. The instance of DIRECTED STEINER OUT-TREE is $(D, r, S, p + n + 1 + l)$ (see Figure 1).

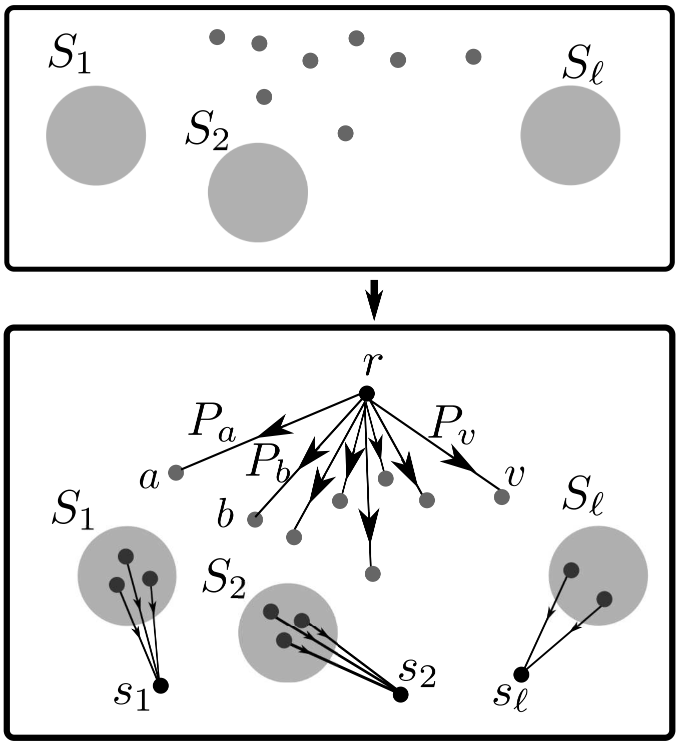

Figure 1: Reduction from GST to Directed Steiner Out-Tree. $r, s_1, \ldots, s_\ell$ are new vertices. $P_a, P_b, \ldots, P_v$ are paths of length n from r to each original vertex. Every original edge $\{u, v\}$ is replaced by the two arcs $(uv), (vu)$ (These arcs are not show in the picture.). For $1 \leq i \leq \ell$, there is an outgoing arc from each vertex in the set S_i to the vertex s_i.

Suppose G contains a tree T on at most p vertices that includes at least one vertex from each S_i. Notice that this tree is also contained in D, and we may access it from r using one of the

(r,u) paths for $u \in V$. Further, if u_i is the vertex from S_i that participates in T, then we may use the edge (u_i, s_i) in D to extend the tree to include the vertex s_i. Thus we have a directed out-tree on $(p+n+1+l)$ vertices containing r and all vertices in S.

Conversely, let T be a directed out-tree in D on at most $(p+n+1+l)$ vertices that contains r and all of S. Given that T is on at most $(p+n+1+l)$ vertices and includes all of S, T cannot include two paths P_u and P_v for $u \neq v$ (as these two paths and S alone would require $2n+l > p+n+1+l$ vertices). Since T is a connected subgraph involving r, it must contain at least one of the paths that connects r to a vertex in V. Thus T contains exactly one of the paths $\{P_u \mid u \in U\}$. This implies that $T' = T \cap V$ is a subtree of V. Note that T' has at most p vertices — as T' involved $l+n+1$ vertices apart from those in T, because it contained S, the root, and a path on n vertices. Notice that these vertices induce a Group Steiner Tree in G. Indeed, if T' omits any group S_i altogether, then T must omit s_i, since the only in-neighbors of s_i lie in S_i, a contradiction.

Thus, G contains a tree on at most p vertices that includes at least one vertex from each S_i if and only if there exists an out-tree in D on at most $p+n+l+1$ vertices containing all vertices of S, and rooted at r. □

We now show that Directed Steiner Out-Tree is fixed-parameter tractable. The algorithm outlined here is essentially the same as that for the Steiner Tree problem due to Nederlof [22]. We give an outline for the sake of completeness. First, recall the well-known Inclusion-Exclusion (IE) formula: Let U be a finite universe and $A_1, \dots, A_l \subseteq U$. Then

$$\left|\bigcap_{i=1}^{l} A_i\right| = |U| + \sum_{\emptyset \neq X \subseteq [l]} (-1)^{|X|} \left|\bigcap_{i \in X} \bar{A}_i\right| \tag{1}$$

Now note that if for all $X \subseteq [l]$, one can evaluate $|\bigcap_{i \in X} \bar{A}_i|$ in time polynomial in the input size n, then one can evaluate $|\bigcap_{i=1}^{l} A_i|$ in time $O(2^l \cdot n^{O(1)})$ and using space polynomial in n. Given a directed graph $D = (V, A)$, define a *branching walk* B in D to be a pair $(T_B = (V_B, E_B), \phi)$, where T_B is a rooted ordered out-tree and $\phi : V_B \to V$ is a digraph homomorphism (see, e.g., the book by Hell and Nešetřil [19]) from T_B to D. The length of B, denoted by $|B|$, is $|E_B|$. For a node $s \in V$, B is from s if the root of T_B is mapped to s by ϕ. We let $\phi(V_B)$ denote $\{\phi(u) : u \in V_B\}$ and $\phi(E_B)$ denote $\{(\phi(u), \phi(b)) : (a,b) \in E_B\}$. Let (D, r, S, p) be an instance of the Directed Steiner Out-Tree problem. As done by Nederlof [22], one can show that there exists an out-tree $T = (V', E')$ of D rooted at r such that $S \subseteq V'$ and $|V'| \leq p$ if and only if there exists a branching walk $B = (T_B = (V_B, E_B), \phi)$ from r such that $S \subseteq \phi(V_B)$ and $|B| \leq p-1$. We now frame the problem as an IE-formula. Let U be the set of all branching walks from r of length $p-1$. For each $v \in S$, let A_v be the set of all elements of U that contain v. Then $|\bigcap_{v \in S} A_v|$ is the number of all branching walks that contain all the vertices of S and this number is larger than zero if and only if the instance is a yes-instance.

For $X \subseteq S$ define $X' = X \cup (V \setminus S)$, and define $b_j^X(r)$ to be the number of branching walks from r of length j in the graph $G[X']$. Then $|\bigcap_{v \in X} \bar{A}_v| = b_c^{S \setminus X}(r)$. Now $b_j^X(r)$ can be computed in polynomial time:

$$b_j^X(r) = \begin{cases} 1 & \text{if } j = 0; \\ \displaystyle\sum_{s \in N^+(r) \cap X'} \sum_{j_1 + j_2 = j-1} b_{j_1}^X(s) \cdot b_{j_2}^X(r) & \text{otherwise.} \end{cases}$$

The proof of this again follows from Nederlof [22]. Now for each $X \subseteq \{1, \dots, l\}$, we can compute the term $|\bigcap_{v \in X} \bar{A}_v|$ in polynomial time and hence by identity (1), we can solve the problem in time $O(2^l \cdot n^{O(1)})$ using polynomial space:

Lemma 2. DIRECTED STEINER OUT-TREE *can be solved in $O(2^l \cdot n^{O(1)})$ time using polynomial space.*

Lemmas 1 and 2 together imply:

Lemma 3. *The* GROUP STEINER TREE *problem can be solved in $O(2^l \cdot n^{O(1)})$ time using polynomial space.*

3.2 An FPT Algorithm for CFVS

Our FPT algorithm for CFVS uses as a subroutine an algorithm (due to Guo et al. [18]) for enumerating an efficient representation of minimal feedback vertex sets of size at most k. Strictly speaking, the subroutine enumerates all compact representations of minimal feedback sets. A *compact representation* for a set of minimal feedback sets of a graph $G = (V, E)$ is a set $\mathcal{C}$ of pairwise disjoint subsets of V such that choosing exactly one vertex from every set in $\mathcal{C}$ results in a minimal feedback set for G. Call a compact representation a *k-compact representation* if the number of sets in the representation is at most k. Clearly, any connected feedback set of size at most k must necessarily pick vertices from the sets of *some* k-compact representation. Given a graph $G = (V, E)$ and a k-compact representation $S_1, \ldots, S_r$, where $r \leq k$, the problem of deciding whether there exists a connected feedback vertex set that contains at least one vertex from each set S_i reduces to the GROUP STEINER TREE problem where the Steiner groups are the sets of the compact representation.

Our algorithm therefore cycles through all k-compact representations and for each such representation uses the algorithm for GROUP STEINER TREE to check if there is a tree on at most k vertices that includes one vertex from each set S_i of the compact representation. If the answer is NO for all k-compact representations, the algorithm reports that the given instance is a NO-instance. If the answer is YES for some compact representation, the algorithm returns the tree found. Since one can enumerate all compact representations in time $O(c^k \cdot m)$ [18], we have:

Theorem 1. *Given a graph $G = (V, E)$ and an integer k, one can decide whether G has a connected feedback set of size at most k in time $O(c^k \cdot n^{O(1)})$, where c is a constant independent of n and k.*

Although CFVS is fixed-parameter tractable, it is unlikely to admit a polynomial kernel as the following theorem shows. This is in contrast to FEEDBACK VERTEX SET which admits a quadratic kernel [24].

Theorem 2. *The* CFVS *problem does not admit a polynomial kernel unless the Polynomial Hierarchy collapses to Σ_3.*

Proof. The proof follows from a polynomial-time parameter-preserving reduction from CONNECTED VERTEX COVER, which does not admit a polynomial kernel unless the Polynomial Hierarchy collapses to the third level [11]. This would prove that CFVS too does not admit a polynomial kernel [4]. Given an instance $(G = (V, E), k)$ of the CONNECTED VERTEX COVER problem, construct a new graph G' as follows: $V(G') = V(G) \cup \{x_{uv} \notin V(G) : \{u, v\} \in E(G)\}$; if $\{u, v\} \in E(G)$ then add the edges $\{u, v\}, \{u, x_{uv}\}, \{x_{uv}, v\}$ to $E(G')$. This completes the construction of G'.

Let S be a connected vertex cover of size at most k in G. We claim that S is a connected feedback vertex set in G'. To see that S induces a connected subgraph in G' is trivial, as $E(G') \supseteq E(G)$. Further, the removal of S from G' leaves a graph with degree at most one, it is clearly impossible for such a graph to admit any cycles.

On the other hand, suppose S is a connected feedback vertex set in G'. If S contains x_{uv}, a newly introduced vertex of degree two, then S must contain at least one of u or v (since

S induces a connected graph). If S contains both u and v, then note that x_{uv} is redundant, because any cycle that passes through x_{uv} necessarily passes through u and v. On the other hand, suppose S contains exactly one of u or v (say u), then in S, we replace x_{uv} with v. Notice that S continues to induce a connected graph (as $(u, v) \in E(G')$) and remains a feedback vertex set (again because all cycles passing through x_{uv} also pass through u and v). We may therefore assume, without loss of generality, that we have a connected feedback vertex set S of G' that only involves vertices of V. Notice that this set S is a connected vertex cover of G. Indeed, S is connected in G because $G'[V] = G$ and S is connected in G'. Further, S is a vertex cover because if $G[V \setminus S]$ contains an edge (u, v), then by construction, $G'[V \setminus S]$ contains the triangle (u, v, x_{uv}), a contradiction. $\square$

Interestingly, recent results due to Fomin et al. [16] imply that CFVS has polynomial kernel on a graph class $\mathcal{C}$ which excludes a fixed apex graph H as a minor(See Section 4.1).

We note that the upper bound for the size of the compact representation can be improved (from the bound due to Guo et al. [18]) using a result from the recent paper of Cao et al. [5] where the authors present the current fastest FPT algorithm for FVS. In that paper, the authors describe a set of reduction rules such that if a YES-instance of the FOREST BIPARTITION problem (defined below) is reduced with respect to this set of rules then the instance has size at most $4k + 1$.

FOREST BIPARTITION

Input:	An undirected graph $G = (V, E)$, possibly with multiple edges and loops and a set $S \subseteq V$ such that $\|S\| = k + 1$ and $G \setminus S$ is acyclic.
Parameter:	The integer k.
Question:	Does G have a feedback vertex set of size at most k contained in $V \setminus S$?

Thus in a YES-instance of FOREST BIPARTITION that is reduced with respect to the reduction rules due to Cao et al. [5], we have $|V \setminus S| \leq 3k$. Guo et al. [18] upper bound the size of the compact representation by

$$\sum_{i=0}^{k} \binom{k+1}{i} \left(\sum_{j=0}^{k-i} \binom{X}{j} \binom{Y}{k-i-j} \right),$$

where X is the number of vertices in $V \setminus S$ that have degree at least three in G, and Y is the number of paths in $G[V \setminus S]$ whose endpoints are vertices of degree at least three, and internal vertices have degree exactly two in G.

The reduction rules due to Cao et al. [5] tell us that in any YES instance, that is, when there exists a FVS of size at most r, $|X| \leq 3r$. Indeed, this follows from the fact that the reduction rules leave the vertices in X unaffected, that is, they only process vertices of degree two or less. Thus, in a reduced instance, $|X| \leq 3r$. Further, by an argument of Guo et al. [18], we know that $|Y| \leq |X| + 2r$. For example, if we are looking for a FVS of size k, and we have "guessed" i vertices that belong to the FVS, $i < k$, then the number of vertices of degree at least three is bounded by $3(k - i)$. These improved bounds upper-bound the size of the compact representation by

$$\sum_{i=0}^{k} \binom{k+1}{i} \left(\sum_{j=0}^{k-i} \binom{3(k+1-i)}{j} \binom{5(k+1-i)}{k-i-j} \right) \leq$$

$$\sum_{i=0}^{k} \binom{k+1}{i} \binom{8(k+1-i)}{k-i} =$$

$$\sum_{i=0}^{k} \binom{9k-8i+9}{k} \leq$$

$$k\binom{9k+9}{k}$$

This amounts to a $O^*(c^k)$-time algorithm for enumerating compact representations of minimal feedback vertex sets of size at most k, where $c = 23.1$. In contrast, the constant c in the running time obtained by Guo et al. [18] is more than 160, because the best bound available on $|X|$ at that time was $14k$.

Theorem 3. [6, 18] *Given a graph $G = (V, E)$ and an integer k, the compact representations of all minimal feedback vertex sets of G of size at most k can be enumerated in time $O(23.1^k \cdot |E|)$.*

From Theorem 3, Lemma 3, and the procedure outlined at the beginning of this subsection, we have

Corollary 1. *Given a graph $G = (V, E)$ and an integer k, one can decide whether G has a connected feedback set of size at most k in $O(46.2^k \cdot n^{O(1)})$ time.*

4 A Subexponential FPT Algorithm for CFVS on H-Minor-Free Graphs

In the last section, we obtained an $O^*(c^k)$ algorithm for CFVS on general graphs. In this section we show that CFVS on the class of H-minor-free graphs admits a sub-exponential time algorithm with running time $O(2^{O(\sqrt{k}\log k)}n^{O(1)})$. This section is divided into three parts. In the first part we give essential definitions from topological graph theory, and in the second part we show that CFVS can be solved in time $O(w^{O(w)}n^{O(1)})$ on graphs with treewidth bounded by w. In the last part we present an algorithm with the stated running time for CFVS on H-minor-free graphs, by bounding the treewidth of the input graph using the known "grid theorems".

4.1 Definitions and Terminology

We use terminology from the book by Diestel [9]. Given an edge e in a graph G, the *contraction* of e is the result of identifying its endpoints in G and then removing all loops and duplicate edges. A *minor* of a graph G is a graph H that can be obtained from a subgraph of G by contracting edges. A graph class $\mathcal{C}$ is *minor-closed* if any minor of any graph in $\mathcal{C}$ is also an element of $\mathcal{C}$. A minor-closed graph class $\mathcal{C}$ is *H-minor-free* or simply *H-free* if $H \notin \mathcal{C}$.

A *tree decomposition* of a graph $G = (V, E)$ is a pair $(T = (V_T, E_T), \mathcal{X} = \{X_t\}_{t \in V_T})$ where T is a tree and the X_t are subsets of V such that:

1. $\bigcup_{u \in V_T} X_t = V$;
2. for each edge $e = \{u, v\} \in E$ there exists $t \in V_T$ such that $u, v \in X_t$; and
3. for each vertex $v \in V$, the subgraph $T[\{t \mid v \in X_t\}]$ is connected.

The *width* of a tree decomposition is $\max_{t \in V_T} |X_t| - 1$ and the *treewidth* of $G = (V, E)$, denoted $tw(G)$, is the minimum width over all tree decompositions of G.

A tree decomposition is called a *nice tree decomposition* [3] if the following conditions are satisfied:

- Every node of the tree T has at most two children. A node that has no children is called a *leaf* node. The non-leaf nodes are of three kinds:
  - If a node t has two children t_1 and t_2, then $X_t = X_{t_1} = X_{t_2}$, and t is called a *join* node.
  - if a node t has one child t_1, then either $|X_t| = |X_{t_1}| + 1$ and $X_{t_1} \subset X_t$ (t is called an *introduce* node), or $|X_t| = |X_{t_1}| - 1$ and $X_t \subset X_{t_1}$ (t is called a *forget* node).

It is possible to transform a given tree decomposition into a nice tree decomposition in time $O(|V| + |E|)$ [3].

4.2 Connected FVS and Treewidth

In this section we show that the CONNECTED FEEDBACK VERTEX SET problem is FPT with the treewidth of the input graph as the parameter. That is, we show that the following problem is FPT:

Input:	An undirected graph $G = (V, E)$; an integer k; and a nice tree decomposition of G of width w.
Parameter:	The treewidth w of the graph G.
Question:	Find $S \subseteq V$ such that $G \setminus S$ is acyclic, $G[S]$ is connected, and for any connected feedback vertex set R of G, $|S| \leq |R|$.

We design a dynamic programming algorithm on the nice tree decomposition with running time $O(w^{O(w)} \cdot n^{O(1)})$ for this problem. See, e.g, Moser [21] for a detailed exposition of this paradigm; in particular, our algorithm is similar in spirit to the algorithm given by Moser [21] for the CONNECTED VERTEX COVER problem. We start by giving an intuitive overview of the algorithm.

The general strategy used to obtain FPT (in the parameter treewidth) algorithms by dynamic programming on nice tree decompositions is to design a dynamic programming table (DP table) for each node of the nice tree decomposition such that:

1. The DP table for a leaf node can be computed in FPT time in the treewidth;
2. Starting at the leaves and going up to the root, given the DP table(s) of the child(ren) node(s) of a given node, the DP table for this node can be computed in time FPT in the treewidth; and,
3. The desired answer can be computed in FPT time (most often, read off directly) from the DP table for the root node.

One fruitful approach towards designing the DP table for solving “subset problems” (where the task is to find a subset of vertices which has some specified property) is to look at how a solution intersects the subgraph induced by all the vertices in a subtree rooted at an arbitrary node in the tree decomposition. Observe that any solution must intersect such a subgraph in *some* manner; the DP table must account for all possible ways in which this can happen, and must be designed in such a way that it is easy (in the sense listed above) to update.

For CONNECTED FEEDBACK VERTEX SET, it turns out that a correct way to design the DP table is to try to capture the *connectivity* properties of a partial solution when restricted to the subgraphs induced by subtrees of the tree decomposition. To be more precise: Let F be some fixed connected feedback vertex set of the input graph $G = (V, E)$ of size at most k, and let G_i be the subgraph induced by the vertices in the subtree rooted at an arbitrary node X_i in a nice tree decomposition of G. $G[F]$ restricted to the set X_i is, in general, a disconnected graph. So is the forest $G[V \setminus F]$ restricted to the set X_i. Our dynamic programming algorithm tries to guess the components of each of the two subgraphs $G[F \cap X_i]$ and $G[(V \setminus F) \cap X_i]$; it turns out that with this information in hand, it is not difficult to compute the DP tables of the nodes at the next higher level in the tree decomposition, be it an introduce, forget, or join node; the details follow.

Let $(T = (I, F), \{X_i | i \in I\})$ be a nice tree decomposition of the input graph G of width w and rooted at $r \in I$. We let T_i denote the subtree of T rooted at $i \in I$, and $G_i = (V_i, E_i)$ denote the subgraph of G induced on all the vertices of G in the subtree T_i, that is, $G_i = G[\bigcup_{j \in V(T_i)} X_j]$.

For each node $i \in I$ we compute a table A_i, the rows of which are 4-tuples $[S, P, Y, \mathit{val}]$. Table A_i contains one row for each combination of the first three components which denote the following:

- S is a subset of X_i.
- P is a partition of S into at most $|S|$ labelled pieces.
- Y is a partition of $X_i \setminus S$ into at most $|X_i \setminus S|$ labelled pieces.

We use $P(v)$ (resp. $Y(v)$) to denote the piece of the partition P (resp. Y) that contains the vertex v. We let $|P|$ (resp. $|Y|$) denote the number of pieces in the partition P (resp. Y). The last component *val*, also denoted as $A_i[S, P, Y]$, is the size of a smallest feedback vertex set $F_i \subseteq V(G_i)$ of G_i which satisfies the following properties:

- If $S = \emptyset$, then F_i is *connected* in G_i.
- If $S \neq \emptyset$, then
  - $F_i \cap X_i = S$.
  - All vertices of S that are in any one piece of P are in a single connected component of $G_i[F_i]$. Moreover $G_i[F_i]$ has exactly $|P|$ connected components.
  - All vertices of $X_i \setminus S$ that are in the same piece of Y are in a single connected component (a tree) of $G_i[V_i \setminus F_i]$. Moreover $G_i[V_i \setminus F_i]$ has at least $|Y|$ connected components.

If there is no such set F_i, then the last component of the row is set to ∞.

We fix an arbitrary ordering of the vertices of X_i, and compute the table A_i for each node $i \in I$ of the tree decomposition. Since there are at most $w + 1$ vertices in each bag X_i, there are no more than

$$\sum_{i=0}^{w+1} \binom{w+1}{i} i^i \cdot (w + 1 - i)^{w+1-i} \leq (2w + 2)^{2w+2}$$

rows in any table A_i. We compute the tables A_i starting from the leaf nodes of the tree decomposition and going up to the root.

Leaf Nodes. Let i be a leaf node of the tree decomposition. We compute the table A_i as follows. For each triple (S, P, Y) where S is a subset of X_i, P a partition of S, and Y a partition of $X_i \setminus S$:

- Set $A_i[S, P, Y] = \infty$ if at least one of the following holds:
  - $G_i \setminus S$ contains a cycle (i.e., S is *not* an FVS of G_i).
  - At least one piece of P is *not* connected in $G_i[S]$ or if $G_i[S]$ has less than $|S|$ connected components.
  - At least one piece of Y is *not* connected in $G_i[V_i \setminus S]$ or if $G_i[V_i \setminus S]$ has less than $|Y|$ connected components.
- In all other cases, set $A_i[S, P, Y] = |S|$.

It is easy to see that this computation correctly determines the last component of each row of A_i for a leaf node i of the tree decomposition.

Introduce Nodes. Let i be an introduce node and j its unique child. Let $x \in X_i \setminus X_j$ be the introduced vertex. For each triple (S, P, Y), we compute the entry $A_i[S, P, Y]$ as follows.

Case 1. $x \in S$. Check whether $N(x) \cap S \subseteq P(x)$; if not, set $A_i[S, P, Y] = \infty$.

- Subcase 1. $P(x) = \{x\}$. Set $A_i[S, P, Y] = A_j[S \setminus \{x\}, P \setminus P(x), Y] + 1$.
- Subcase 2: $|P(x)| \geq 2$ and $N(x) \cap P(x) = \emptyset$. Set $A_i[S, P, Y] = \infty$, as no extension of S to an fvs for G_i can make $P(x)$ connected.
- Subcase 3: $|P(x)| \geq 2$ and $N(x) \cap P(x) \neq \emptyset$. Let $\mathcal{A}$ be the set of all rows $[S', P', Y]$ of the table A_j that satisfy the following conditions:
  - $S' = S \setminus \{x\}$.
  - $P' = (P \setminus P(x)) \cup Q$, where Q is a partition of $P(x) \setminus \{x\}$ such that each piece of Q contains an element of $N(x) \cap P(x)$.

  Set $A_i[S, P, Y] = \min_{[S',P',Y] \in \mathcal{A}}\{A_j[S', P', Y]\} + 1$.

Case 2. $x \notin S$. Check whether $N(x) \cap (X_i \setminus S) \subseteq Y(x)$; if not, set $A_i[S, P, Y] = \infty$.

- Subcase 1: $Y(x) = \{x\}$. Set $A_i[S, P, Y] = A_j[S, P, Y \setminus Y(x)]$.
- Subcase 2: $|Y(x)| \geq 2$ and $N(x) \cap Y(x) = \emptyset$. Set $A_i[S, P, Y] = \infty$, as no extension of S to an fvs F_i for G_i can make $Y(x)$ a connected component in $G_i[V_i \setminus F_i]$.
- Subcase 3: $|Y(x)| \geq 2$ and $N(x) \cap Y(x) \neq \emptyset$. Let $\mathcal{A}$ be the set of all rows $[S, P, Y']$ of the table A_j where $Y' = (Y \setminus Y(x)) \cup Q$, and Q is a partition of $Y(x) \setminus \{x\}$ such that each piece of Q contains *exactly* one element of $N(x) \cap Y(x)$. Set $A_i[S, P, Y] = \min_{[S,P,Y'] \in \mathcal{A}}\{A_j[S, P, Y']\}$.

Forget Nodes. Let i be a forget node and j its unique child node. Let $x \in X_j \setminus X_i$ be the forgotten vertex. For each triple (S, P, Y) in the table A_i, let $\mathcal{A}$ be the set of all rows $[S', P', Y]$ of the table A_j that satisfy the following conditions:

- $S' = S \cup \{x\}$, and
- $P'(x) = P(y) \cup \{x\}$ for some $y \in S$.

Let $\mathcal{B}$ be the set of all rows $[S, P, Y']$ of the table A_j such that $Y'(x) = Y(z) \cup \{x\}$ for some $z \in S$. Set

$$A_i[S, P, Y] = \min\left\{\min_{[S',P',Y] \in \mathcal{A}} A_j[S', P', Y], \min_{[S,P,Y'] \in \mathcal{B}} A_j[S, P, Y']\right\}.$$

Join Nodes. Let i be a join node and j and l its children. For each triple (S, P, Y) we compute $A_i[S, P, Y]$ as follows.

- Case 1. $S = \emptyset$. If both $A_j[\emptyset, P, Y]$ and $A_l[\emptyset, P, Y]$ are positive finite, then set $A_i[\emptyset, P, Y] = \infty$. Otherwise, set $A_i[\emptyset, P, Y] = \max\{A_j[\emptyset, P, Y], A_l[\emptyset, P, Y]\}$.
- Case 2. $S \neq \emptyset$. Let $\mathcal{A}$ denote the set of all pairs of triples $\langle(S, P_1, Y_1), (S, P_2, Y_2)\rangle$, where $(S, P_1, Y_1) \in A_j$ and $(S, P_2, Y_2) \in A_l$ with the following property: Starting with the partitions $Q_p = P_1$ and $Q_y = Y_1$ and repeatedly applying the following set of operations, we reach stable partitions that are identical to P and Y. The first operation that we apply is:

  If there exist vertices $u, v \in S$ such that they are in different pieces of Q_p but are in the same piece of P_2, delete $Q_p(u)$ and $Q_p(v)$ from Q_p and add $Q_p(u) \cup Q_p(v)$.

  To describe the second set of operations, we need some notation. Let $Z = X_i \setminus S$ and let the connected components of $G_i[Z]$ be $C_1, \ldots, C_q$. First contract each connected component C_i to a vertex c_i, the *representative* of that component, and let $\mathcal{C} = \{c_1, \ldots, c_q\}$. Note that for each $1 \leq i \leq q$, the component C_i is not split across pieces in either Y_1 or Y_2. Denote by Y_1' and Y_2' the partitions obtained from Y_1 and Y_2, respectively, be replacing each connected component C_i by its representative vertex c_i. Let $Q_y = Y_1'$. Repeat until no longer possible:

  If there exist $c_a, c_b \in \mathcal{C}$ that are in different pieces of Q_y but in the same piece of Y_2 then delete $Q_y(c_a), Q_y(c_b)$ from Q_y and add $Q_y(c_a) \cup Q_y(c_b)$ provided the following condition holds: for all $c_e \in \mathcal{C} \setminus \{c_a, c_b\}$ either $Y_2(c_e) \cap Q_y(c_a) = \emptyset$ or $Y_2(c_e) \cap Q_y(c_b) = \emptyset$.

  If this latter condition does not hold, move on to the next pair of triples. Finally expand each c_i to the connected component it represents.

  Set
  $$A_i[S, P, Y] = \min_{\langle(S,P_1,Y_1),(S,P_2,Y_2)\rangle \in \mathcal{A}} \{A_j[S, P_1, Y_1] + A_l[S, P_2, Y_2] - |S|\}.$$

  The stated conditions ensure that $u, v \in S$ are in the same piece of P if and only if for each $\langle(S, P_1, Y_1), (S, P_2, Y_2)\rangle \in \mathcal{A}$, they are in the same piece of P_1 or of P_2 (or both). Similarly, the stated conditions ensure that merging solutions at join nodes do not create new cycles. Given this, it is easy to verify that the above computation correctly determines $A_i\,[S, P, Y]$.

Root Node. We compute the size of a smallest CFVS of G from the table A_r for the root node r as follows. Find the minimum of $A_r[S, P, Y]$ over all triples (S, P, Y), where $S \subseteq X_r$, P a partition of S such that P consists of a single (possibly empty) piece and Y is a partition of $X_r \setminus S$. This minimum is the size of a smallest CFVS of G.

This concludes the description of the dynamic programming algorithm for CFVS when the treewidth of the input graph is bounded by w. From the above description and the size of tables being bounded by $(2w + 2)^{2w+2}$, we obtain the following result.

Lemma 4. *Given a graph $G = (V, E)$, a tree-decomposition of G of width w, one can compute the size of an optimum connected feedback vertex set of G (if it exists) in time $O((2w + 2)^{2w+2} \cdot n^{O(1)})$.*

4.3 FPT Algorithms for H-Minor Free Graphs

We first bound the treewidth of the yes instance of input graphs by $O(\sqrt{k})$.

Lemma 5. *If (G, k) is a yes-instance of CFVS where G excludes a fixed graph H as a minor, then $\mathbf{tw}(G) \leq c_H\sqrt{k}$, where c_H is a constant that depends only on the graph H.*

Proof. Demaine and Hajiaghayi [7] showed that for any fixed graph H, every H-minor-free graph G that does not contain a $(w \times w)$-grid as a minor has treewidth at most $c'_H w$, where c'_H is a constant that depends only on the graph H. Clearly a $(w \times w)$-grid has a feedback vertex set of size at least $c_1 w^2$, where c_1 is a constant independent of w. Therefore if G has a connected feedback vertex set of size at most k, it cannot have a $(w \times w)$-grid minor, where $w > \sqrt{k/c_1}$. Therefore $\mathbf{tw}(G) \leq c'_H w \leq c'_H \cdot (\sqrt{k/c_1} + 1) \leq c_H\sqrt{k}$, where $c_H = (c'_H + 1)/\sqrt{c_1}$. □

Theorem 4. *CFVS can be solved in time $O(2^{O(\sqrt{k}\log k)} n^{O(1)})$ on H-minor-free graphs.*

Proof. Given an instance (G, k) of CFVS, we first find a tree-decomposition of G using the polynomial-time constant-factor approximation algorithm of Demaine et al. [8]. If $\mathbf{tw}(G) > c_H\sqrt{k}$, then the given instance is a no-instance; else, use Lemma 4 to find an optimal CFVS for G. All this can be done in $O(2^{O(\sqrt{k}\log k)} \cdot n^{O(1)})$. □

5 Conclusion

We conclude with some open problems. The obvious question is to obtain an $O^*(c^k)$ algorithm for CFVS in general graphs with a smaller value of c. Also, it is not known if CFVS admits any non-trivial approximation algorithm. Another question which we find interesting is whether CFVS admits a polynomial kernel on graphs excluding a fixed graph H as a minor, where H belongs to a class of graphs that is distinct from the class of apex graphs. It will also be interesting to find other applications of GROUP STEINER TREE.

References

[1] J. Bang-Jensen and G. Z. Gutin. *Digraphs: Theory, Algorithms and Applications*. Springer, second edition, 2009.

[2] H. L. Bodlaender. On disjoint cycles. In G. Schmidt and R. Berghammer, editors, *Proceedings on Graph–Theoretic Concepts in Computer Science (WG '91)*, volume 570 of *LNCS*, pages 230–238. Springer, 1992.

[3] H. L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25(6):1305–1317, 1996.

[4] H. L. Bodlaender, R. G. Downey, M. R. Fellows, and D. Hermelin. On problems without polynomial kernels (extended abstract). In *Proceedings of ICALP 2008*, LNCS, pages 563–574. Springer, 2008.

[5] Y. Cao, J. Chen, and Y. Liu. On feedback vertex set, new measure and new structures. *12th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT)*, to appear, 2010.

[6] F. Dehne, M. Fellows, M. A. Langston, F. Rosamond, and K. Stevens. An $O(2^{O(k)}n^3)$ FPT-Algorithm for the Undirected Feedback Vertex Set problem. In *Proceedings of COCOON 2005*, volume 3595 of *LNCS*, pages 859–869. Springer, 2005.

[7] E. D. Demaine and M. Hajiaghayi. Linearity of grid minors in treewidth with applications through bidimensionality. *Combinatorica*, 28(1):19–36, 2008.

[8] E. D. Demaine, M. Hajiaghayi, and K. ichi Kawarabayashi. Algorithmic graph minor theory: Decomposition, approximation, and coloring. In *Proceedings of FOCS 2005*, pages 637–646. IEEE Computer Society, 2005.

[9] R. Diestel. *Graph Theory*. Springer-Verlag, Heidelberg, third edition, 2005.

[10] B. Ding, J. X. Yu, S. Wang, L. Qin, X. Zhang, and X. Lin. Finding top-k min-cost connected trees in databases. In *ICDE*, pages 836–845. IEEE, 2007.

[11] M. Dom, D. Lokshtanov, and S. Saurabh. Incompressibility through Colors and IDs. In *Proceedings of ICALP 2009*, volume 5555 of *LNCS*, pages 378–389. Springer, 2009.

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

[13] P. Festa, P. M. Pardalos, and M. G. Resende. Feedback set problems. In *Handbook of Combinatorial Optimization*, pages 209–258. Kluwer Academic Publishers, 1999.

[14] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer-Verlag, Berlin, 2006.

[15] F. V. Fomin, F. Grandoni, and D. Kratsch. Solving connected dominating set faster than 2^n. *Algorithmica*, 52(2):153–166, 2008.

[16] F. V. Fomin, D. Lokshtanov, S. Saurabh, and D. M. Thilikos. Bidimensionality and kernels. In *Proceedings of the 21th ACM-SIAM Symposium on Discrete Algorithms (SODA 2010)*, pages 503–510. ACM and SIAM, 2010.

[17] A. Grigoriev and R. Sitters. Connected feedback vertex set in planar graphs. In *Graph-Theoretic Concepts in Computer Science, 35th International Workshop, WG 2009, Montpellier, France, June 24-26, 2009. Revised Papers*, volume 5911 of *Lecture Notes in Computer Science*, pages 143–153, 2009.

[18] J. Guo, J. Gramm, F. Hüffner, R. Niedermeier, and S. Wernicke. Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization. *Journal of Computer and System Sciences*, 72(8):1386–1396, 2006.

[19] P. Hell and J. Nešetřil. *Graphs and Homomorphisms*. Oxford University Press, 2004.

[20] D. Mölle, S. Richter, and P. Rossmanith. Enumerate and expand: Improved algorithms for connected vertex cover and tree cover. *Theory of Computing Systems*, 43(2):234–253, 2008.

[21] H. Moser. Exact algorithms for generalizations of vertex cover. Master's thesis, Institut für Informatik, Friedrich-Schiller-Universität, 2005.

[22] J. Nederlof. Fast polynomial-space algorithms using möbius inversion: Improving on steiner tree and related problems. In *Proceedings of ICALP 2009*, pages 713–725, 2009.

[23] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*, volume 31 of *Oxford Lecture Series in Mathematics and its Applications*. Oxford University Press, Oxford, 2006.

[24] S. Thomassé. A quadratic kernel for feedback vertex set. In *Proceedings of SODA 2009*, pages 115–119. Society for Industrial and Applied Mathematics, 2009.