FPT Algorithms for Connected Feedback Vertex Set

Neeldhara Misra, Geevarghese Philip, Venkatesh Raman, Saket Saurabh*, and Somnath Sikdar

The Institute of Mathematical Sciences, India
{neeldhara,gphilip,vraman,saket,somnath}@imsc.res.in

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 [12] 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 [21] recently introduced this problem and obtained a polynomial time approximation scheme (PTAS) for

* This work was done while the author was at the University of Bergen, Norway.

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, [17], [14], 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 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, e.g, the textbook by Flum and Grohe [13].

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, [2]). After several rounds of improvements, the current best FPT algorithm for FVS runs in time $O(5^k kn^2)$ [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, [17]). 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. [16] 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 polynomial space and uses a Turing-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 in [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 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 $O(f(n) \cdot p(n))$, 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**] *[13,20]*
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. $|N^+(v)|$). 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 (see [1]) 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.*

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 Turing-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 $S = \{s_1, s_2, \ldots, s_l\}$ be a set of l new vertices, that is, $s_i \notin V$ for $1 \leq i \leq l$. Let $V' = V \cup S$ and $A = \{uv, vu : \{u, v\} \in E\} \cup \bigcup_{i=1}^{l} \{xs_i : x \in S_i\}$. Finally, let $D = (V', A)$. It is easy to see that 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 a vertex $r \in V'$ and an out-tree in D rooted at r on at most $p + l$ vertices containing all vertices of S. $\square$

Lemma 2. Directed Steiner Out-Tree *can be solved in $O(2^l \cdot n^{O(1)})$ time using polynomial space.*

Nederlof [19] uses the Inclusion-Exclusion Principle and a notion of *branching walks* to give an algorithm for the Steiner Tree problem that runs in $O(2^l \cdot n^{O(1)})$ time using polynomial space, where l is the number of terminals. Essentially the same algorithm works for Directed Steiner Out-Tree, with the same resource bounds; we omit the details due to space constraints.

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. [16]) 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 $O(c^k \cdot m)$ [16], 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* $O(c^k \cdot n^{O(1)})$*, for some constant* c*.*

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 [22].

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'. It is easy to see that G has a connected vertex cover of size at most k if and only if G' has a connected feedback vertex set of size at most k. This completes the proof of the theorem. □

Interestingly, the results from [15] imply that CFVS has polynomial kernel on a graph class $\mathcal{C}$ which excludes a fixed graph H as a minor(See Section 4.1).

We note in passing that the algorithm for enumerating compact representations can be improved using results from [6]. The authors of [6] 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 $5k + 1$.

FOREST BIPARTITION

Input:	An undirected graph $G = (V, E)$, possibly with multiple edges and loops and a set $S \subseteq V$ such that $\lvert S\rvert = k + 1$ 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 rules in [6], we have $|V \setminus S| \leq 4k$. Using this bound in the algorithm described by Guo et al. [16], one obtains a $O^*(c^k)$-time algorithm for enumerating compact representations of minimal feedback vertex sets of size at most k, where $c = 52$. The constant c in [16] is more than 160.

Theorem 3. [6,16] *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 $O(52^k \cdot |E|)$.*

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 [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:	Does there exist $S \subseteq V$ such that $G \setminus S$ is acyclic, $G[S]$ is connected, and $|S| \leq k$?

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 [18] for a detailed exposition of this paradigm; in particular, our algorithm is similar in spirit to the algorithm given in [18] for the Connected Vertex Cover problem.

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 $\mathbf{tw}(G) \le c_H\sqrt{k}$, where c_H is a constant that depends only on the graph H.*

Proof. By [7], 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 $\mathbf{tw}(G) \le c'_H w \le c'_H \cdot (\sqrt{k/c_1} + 1) \le c_H\sqrt{k}$, 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 $O(2^{O(\sqrt{k}\log k)} \cdot n^{O(1)})$. To obtain the claimed running time bound we first apply the results from [15] and obtain an $O(k^2)$ kernel for the problem in polynomial time and then apply the algorithm described. □

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 the approximability of CFVS in general graphs is unknown. Is there a constant-factor approximation algorithm for CFVS? If not, what is the limit of approximation? Is there an $O^*(c^w)$ algorithm for CFVS, for a constant c, for graphs of treewidth at most w? Note that this question is open even in the context of finding a (unconnected) feedback vertex set in graphs of treewidth at most w.

References

1. Bang-Jensen, J., Gutin, G.Z.: Digraphs: Theory, Algorithms and Applications, 2nd edn. Springer, Heidelberg (2009)
2. Bodlaender, H.L.: On disjoint cycles. In: Schmidt, G., Berghammer, R. (eds.) WG 1991. LNCS, vol. 570, pp. 230–238. Springer, Heidelberg (1992)
3. Bodlaender, H.L.: A linear-time algorithm for finding tree-decompositions of small treewidth. SIAM Journal on Computing 25(6), 1305–1317 (1996)

4. Bodlaender, H.L., Downey, R.G., Fellows, M.R., Hermelin, D.: On problems without polynomial kernels (extended abstract). In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part I. LNCS, vol. 5125, pp. 563–574. Springer, Heidelberg (2008)
5. Chen, J., Fomin, F.V., Liu, Y., Lu, S., Villanger, Y.: Improved algorithms for the feedback vertex set problems. In: Dehne, F., Sack, J.-R., Zeh, N. (eds.) WADS 2007. LNCS, vol. 4619, pp. 422–433. Springer, Heidelberg (2007)
6. Dehne, F., Fellows, M., Langston, M.A., Rosamond, F., Stevens, K.: An $O(2^{O(k)}n^3)$ FPT-Algorithm for the Undirected Feedback Vertex Set problem. In: Wang, L. (ed.) COCOON 2005. LNCS, vol. 3595, pp. 859–869. Springer, Heidelberg (2005)
7. Demaine, E.D., Hajiaghayi, M.: Linearity of grid minors in treewidth with applications through bidimensionality. Combinatorica 28(1), 19–36 (2008)
8. Demaine, E.D., Hajiaghayi, M., ichi Kawarabayashi, K.: Algorithmic graph minor theory: Decomposition, approximation, and coloring. In: Proceedings of FOCS 2005, pp. 637–646. IEEE Computer Society, Los Alamitos (2005)
9. Diestel, R.: Graph Theory, 3rd edn. Springer, Heidelberg (2005)
10. Ding, B., Yu, J.X., Wang, S., Qin, L., Zhang, X., Lin, X.: Finding top-k min-cost connected trees in databases. In: ICDE, pp. 836–845. IEEE, Los Alamitos (2007)
11. Dom, M., Lokshtanov, D., Saurabh, S.: Incompressibility through Colors and IDs. In: Albers, S., et al. (eds.) ICALP 2009. LNCS, vol. 5555, pp. 378–389. Springer, Heidelberg (2009)
12. Festa, P., Pardalos, P.M., Resende, M.G.: Feedback set problems. In: Handbook of Combinatorial Optimization, pp. 209–258. Kluwer Academic Publishers, Dordrecht (1999)
13. Flum, J., Grohe, M.: Parameterized Complexity Theory. Texts in Theoretical Computer Science. An EATCS Series. Springer, Berlin (2006)
14. Fomin, F.V., Grandoni, F., Kratsch, D.: Solving connected dominating set faster than 2^n. Algorithmica 52(2), 153–166 (2008)
15. Fomin, F.V., Lokshtanov, D., Saurabh, S., Thilikos, D.M.: Bidimensionality and kernels. In: Proceedings of SODA 2010 (2010) (to appear)
16. Guo, J., Gramm, J., Hüffner, F., Niedermeier, R., Wernicke, S.: Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization. Journal of Computer and System Sciences 72(8), 1386–1396 (2006)
17. Mölle, D., Richter, S., Rossmanith, P.: Enumerate and expand: Improved algorithms for connected vertex cover and tree cover. Theory of Computing Systems 43(2), 234–253 (2008)
18. Moser, H.: Exact algorithms for generalizations of vertex cover. Master's thesis, Institut für Informatik, Friedrich-Schiller-Universität (2005)
19. Nederlof, J.: Fast polynomial-space algorithms using möbius inversion: Improving on steiner tree and related problems. In: Albers, S., et al. (eds.) ICALP 2009, pp. 713–725. Springer, Heidelberg (2009)
20. Niedermeier, R.: Invitation to Fixed-Parameter Algorithms. Oxford Lecture Series in Mathematics and its Applications, vol. 31. Oxford University Press, Oxford (2006)
21. Sitters, R., Grigoriev, A.: Connected feedback vertex set in planar graphs. In: Paul, C., Habib, M. (eds.) WG 2009. LNCS, vol. 5911. Springer, Heidelberg (2009)
22. Thomassé, S.: A quadratic kernel for feedback vertex set. In: Proceedings of SODA 2009, pp. 115–119. Society for Industrial and Applied Mathematics (2009)