Algorithmica (2014) 69:216–231
DOI 10.1007/s00453-012-9731-6

Enumerating Minimal Subset Feedback Vertex Sets

Fedor V. Fomin · Pinar Heggernes · Dieter Kratsch · Charis Papadopoulos · Yngve Villanger

Received: 30 November 2011 / Accepted: 11 December 2012 / Published online: 15 December 2012

Abstract The Subset Feedback Vertex Set problem takes as input a pair (G, S), where $G = (V, E)$ is a graph with weights on its vertices, and $S \subseteq V$. The task is to find a set of vertices of total minimum weight to be removed from G, such that in the remaining graph no cycle contains a vertex of S. We show that this problem can be solved in time $O(1.8638^n)$, where $n = |V|$. This is a consequence of the main result of this paper, namely that all minimal subset feedback vertex sets of a graph can be enumerated in time $O(1.8638^n)$.

Keywords Exact exponential algorithms · NP-hard problems · Subset feedback vertex set

This work is supported by the Research Council of Norway. A preliminary version of this work was presented at WADS 2011 [8].

F.V. Fomin · P. Heggernes · Y. Villanger
Department of Informatics, University of Bergen, Bergen, Norway

F.V. Fomin
e-mail: fedor.fomin@ii.uib.no

P. Heggernes
e-mail: pinar.heggernes@ii.uib.no

Y. Villanger
e-mail: yngve.villanger@ii.uib.no

D. Kratsch
LITA, Université Paul Verlaine, Metz, France
e-mail: kratsch@univ-metz.fr

C. Papadopoulos (✉)
Department of Mathematics, University of Ioannina, Ioannina, Greece
e-mail: charis@cs.uoi.gr

1 Introduction

Given a graph $G = (V, E)$ and a set $S \subseteq V$, a *subset feedback vertex set* of (G, S) is a set $X \subseteq V$ such that no cycle in $G[V \setminus X]$ contains a vertex of S. A subset feedback vertex set is *minimal* if no proper subset of it is a subset feedback vertex set. Given a *weighted* graph G with positive real weights on its vertices and S as input, the SUBSET FEEDBACK VERTEX SET problem is the problem of finding a subset feedback vertex set X of (G, S) such that the sum of weights of the vertices in X is minimized.

SUBSET FEEDBACK VERTEX is a generalization of several well-known problems. When $S = V$, it is equivalent to the classical NP-hard FEEDBACK VERTEX SET problem [13]. When $|S| = 1$, it generalizes the MULTIWAY CUT problem. Given a set $T \subseteq V$, called *terminals*, a *multiway cut* of (G, T) is a set of vertices whose removal from G disconnects every pair of terminals. Given a graph $G = (V, E)$, with weights on its vertices, and $T \subseteq V$, the MULTIWAY CUT problem is the problem of computing a multiway cut of total minimum weight. We will see in the last section that this is a special case of SUBSET FEEDBACK VERTEX SET. The *unweighted* versions of the three above mentioned problems are obtained when the weight of every vertex of the input graph is 1. For further results on variants of MULTIWAY CUT see [1, 14], and for connections between variants of SUBSET FEEDBACK VERTEX SET and variants of MULTIWAY CUT see also [6].

SUBSET FEEDBACK VERTEX SET was first studied by Even et al. who obtained a constant factor approximation algorithm [7]. In this paper we are interested in the exact solution of SUBSET FEEDBACK VERTEX SET. This does not seem to have been studied before, whereas there are a series of exact results on FEEDBACK VERTEX SET. Razgon [17] gave the first non-trivial exact algorithm for unweighted FEEDBACK VERTEX SET, which was later improved by Fomin et. al. [9, 12]. Currently, a minimum feedback vertex set in an unweighted graph can be computed in time $O(1.7347^n)$ [12]. Furthermore, all minimal feedback vertex sets can be enumerated in $O(1.8638^n)$ time [9], which implies that a minimum weight feedback vertex set can be computed in time $O(1.8638^n)$. So far, this is the best known algorithm for FEEDBACK VERTEX SET. We would also like to remark that the unweighted version of SUBSET FEEDBACK VERTEX SET was only recently shown to be fixed parameter tractable [6], whereas the unweighted version of FEEDBACK VERTEX SET has long been known to be fixed parameter tractable [2, 3, 5, 16, 20].

In this paper, we show that SUBSET FEEDBACK VERTEX SET can be solved in time $O(1.8638^n)$. Prior to our result, no algorithm breaking the trivial $2^n n^{O(1)}$-time barrier has been known, even for the unweighted version of the problem. As our main result, we give an algorithm that enumerates all minimal subset feedback vertex sets of (G, S) and runs in time $O(1.8638^n)$. Thus our running time matches the best known algorithm for enumerating all minimal feedback vertex sets [9]. While the general branching approach for enumerating the subset feedback vertex sets is similar to the one enumerating the feedback vertex sets [9], we introduce and use here new ideas that are needed for the subset variant of the problem. Our enumeration algorithm can be trivially adapted to give an algorithm computing a minimum weight subset feedback vertex set in time $O(1.8638^n)$. Furthermore, as we explain in

Sect. 5, our algorithm can be used to enumerate all minimal multiway cuts within the same running time. As a consequence, we are also able to solve MULTIWAY CUT in time $O(1.8638^n)$. To our knowledge, this is the first non-trivial exact algorithm for the solution of this latter problem, even for its unweighted version.

Before we present the principal results of the paper in Sects. 3 and 4, we show in the next section that SUBSET FEEDBACK VERTEX SET is NP-hard when the input graph G is restricted to be a split graph, even for the unweighted version of the problem. As split graphs form a subclass of chordal graphs, this implies the hardness of the problem also on chordal graphs. Hence we get an interesting contrast to the fact that FEEDBACK VERTEX SET can be solved in polynomial time on chordal graphs [4, 19].

2 Preliminaries

All graphs in this paper are undirected and with weights on their vertices. All graphs are simple unless explicitly mentioned; in particular input graphs are always simple, but during the course of our algorithm, multiple edges are introduced due to contraction of edges.

A graph is denoted by $G = (V, E)$ with vertex set V and edge set E. We use the convention that $n = |V|$ and $m = |E|$. When a graph or subgraph G is mentioned without specifying its vertex and edge sets, we use $V(G)$ and $E(G)$ to denote these sets, respectively. Each vertex $v \in V$ has a weight that is a positive real number. For a vertex set $X \subseteq V$ the weight of X is the sum of the weights of all vertices in X, and the *subgraph of G induced by X* is denoted by $G[X]$. The *neighborhood* of a vertex v of G is $N(v) = \{x \mid \{v, x\} \in E\}$. For $X \subseteq V$, $N(X) = \bigcup_{x \in X} N(x) \setminus X$. In this paper, we distinguish between paths (cycles) and induced paths (induced cycles). A path (cycle) of G is *induced* if there are no edges in G between non-consecutive vertices of the path (cycle). An edge of G is called a *bridge* if its removal increases the number of connected components. A *forest* is a graph that contains no cycles, and a *tree* is a forest that is connected. The *contraction* of edge $\{u, v\}$ removes u and v from the graph, and replaces them with a new vertex that is incident with every edge that was incident with u or v. If we say that edge $\{u, v\}$ is *contracted to u*, then u takes the role of the new vertex after the contraction. Note that multiple edges might result from this operation.

Given a graph G and a vertex subset S of G, a *subset feedback vertex set* of (G, S) is a vertex subset of G whose removal from G ensures that no cycle in the remaining graph contains a vertex of S. Note that a *minimum weight* (or simply *minimum*) subset feedback vertex set is dependent on the weights of the vertices, whereas a *minimal* subset feedback vertex set is only dependent on the vertices and not their weights. Clearly, both in the weighted and the unweighted versions, a minimum subset feedback vertex set must be minimal.

We conclude this section by providing an NP-hardness result for SUBSET FEEDBACK VERTEX SET on split graphs. A graph $G = (V, E)$ is a *split graph* if V can be partitioned into a clique C and an independent set I, where (C, I) is called a *split partition* of G. Split graphs form a subclass of the larger and widely known graph class of *chordal graphs*, which are the graphs that do not contain induced cycles of length 4 or more as induced subgraphs. Interestingly, although FEEDBACK VERTEX SET is solvable in polynomial time on chordal graphs [4, 19], a simple reduction from VERTEX COVER shows that SUBSET FEEDBACK VERTEX SET is NP-hard on chordal graphs, even on their subclass split graphs.

The *decision* version of unweighted SUBSET FEEDBACK VERTEX SET takes as input a graph $G = (V, E)$, a set $S \subseteq V$, and an integer k, and asks whether there is a subset feedback vertex set of (G, S) of size at most k.

Theorem 2.1 *The decision version of* SUBSET FEEDBACK VERTEX SET *is NP-complete on unweighted split graphs.*

Proof Given $G = (V, E)$, $S \subseteq V$, and $X \subseteq V$, checking whether X is a subset feedback vertex set of (G, S) amounts to checking whether every edge incident to a vertex of $S \setminus X$ is a bridge in $G[V \setminus X]$. As this can easily be done in polynomial time, the problem is in NP. We will give a polynomial-time reduction to it from the classical NP-complete problem [15] VERTEX COVER: given a graph $G = (V, E)$ and an integer k, does G have a *vertex cover* of size at most k, i.e, is there a set $Y \subseteq V$ with $|Y| \leq k$, such that every edge in E has an endpoint in Y?

Let (G, k) be an instance of VERTEX COVER, where $G = (V, E)$ is an arbitrary graph with n vertices and m edges. We construct a split graph $H = (V', E')$ with split partition (C, I) as follows. $V' = C \cup I$ contains $n + m$ vertices: for each vertex $u \in V$, there is a vertex $u' \in C$, and for each edge $\{v, w\} \in E$, there is a vertex $u_{vw} \in I$. E' is defined so that vertices in C are pairwise adjacent, and each vertex u_{vw} of I has exactly two neighbors: vertices v' and w' in C. Consequently, C is a clique and I is an independent set.

We claim that G has a vertex cover of size at most k if and only if (H, I) has a subset feedback vertex set of size at most k. Assume that $Y \subseteq V$ is a vertex cover of G of size at most k. In H, let $Y' \subseteq C$ such that $u' \in Y'$ if and only if $u \in Y$. Now Y' is a subset feedback vertex set of (H, I), since every vertex of I has degree at most 1 in $H[V' \setminus Y']$ and therefore cannot be involved in any cycle. For the opposite direction, let X' be a subset feedback vertex set of (H, I) of size at most k. Suppose X' contains a vertex u_{vw} of I. If X' contains one of the two neighbors v' or w' of u_{vw} then X' is not minimal, and $X' \setminus \{u_{vw}\}$ is also a subset feedback vertex set of (H, I). If X' does not contain v' or w', then we can simply replace u_{vw} with v' and obtain another subset feedback vertex set of (H, I) of the same size. Hence we can assume that $X' \subseteq C$. Now, taking $X \subseteq V$ such that $u \in X$ if and only if $u' \in X'$, we can see that X is a vertex cover of G since X contains at least one endpoint of each edge in G. □

In the next two sections we present our principal results: a branching algorithm to enumerate all minimal subset feedback vertex sets of a given graph, and the analysis of its running time combining induction and Measure & Conquer.

3 Enumerating All Minimal Subset Feedback Vertex Sets

Let $G = (V, E)$ be an arbitrary graph and let $S \subseteq V$. In this section we give an algorithm that enumerates all minimal subset feedback vertex sets of (G, S).

We define an *S-forest* of G to be a vertex set $Y \subseteq V$ such that no cycle in $G[Y]$ contains a vertex of S. An S-forest Y is *maximal* if no proper superset of Y is an S-forest. Observe that X is a minimal subset feedback vertex set if and only if $Y = V \setminus X$ is a maximal S-forest. Thus, the problem of enumerating all minimal subset feedback vertex sets is equivalent to the problem of enumerating all maximal S-forests. Consequently, we present an algorithm for enumerating all maximal S-forests of the input graph G.

Our algorithm is a branching algorithm consisting of a sequence of reduction and branching rules. The running time of the algorithm is up to a polynomial factor proportional to the number of generated subproblems, or to the number of nodes of the branching tree. For more information on branching algorithms and Measure & Conquer analysis of such algorithms we refer to [11].

In our algorithm each subproblem corresponding to a leaf of the branching tree will define an S-forest, and every maximal S-forest will be defined by one leaf of the branching tree. Each of the reduction and branching rules will reduce the problem instance by making progress towards some S-forest.

To incorporate all information needed in the algorithm we use so-called red-blue S-forests. Given a set $B \subseteq V$ of *blue* vertices with $B \cap S = \emptyset$ and a set $R \subseteq V$ of *red* vertices with $R \subseteq S$, a maximal *red-blue S-forest* of G is a maximal S-forest Y of G such that $R \cup B \subseteq Y$. If the set $R \cup B$ of vertices has the property that no two red vertices, or no two blue vertices, are adjacent, then we say that the red-blue coloring of these vertices is a *proper 2-coloring*. Let $RBF(G, S, R, B)$ be the set of all maximal red-blue S-forests in G. Hence a maximal S-forest Y is an *element* of $RBF(G, S, R, B)$ if $R \cup B \subseteq Y$. Observe that the problem of enumerating all maximal S-forests of G is equivalent to enumerating all elements of $RBF(G, S, \emptyset, \emptyset)$. We refer to the vertices of $V \setminus (R \cup B)$ as *non-colored*. Before proceeding with the description of the algorithm, we need the following observations concerning the set $RBF(G, S, R, B)$.

Observation 3.1 *Let $Y = R \cup B$ be an S-forest of G that is an element of $RBF(G, S, R, B)$. Let G' be the graph obtained from $G[Y]$ by contracting every edge whose endpoints have the same color, giving the resulting vertex that same color, and removing self loops and multiple edges. Then G' is a forest. Moreover, red and blue vertices form a proper 2-coloring of G'.*

Proof Since Y is an S-forest in G and $Y = R \cup B$, we have that any cycle in $G[Y]$ contains only blue vertices. Thus, each cycle is contracted to a blue vertex in G'. Since no cycles remain in G', G' is a forest. If there is an edge between two vertices of the same color, then this edge would have been contracted, and thus the red-blue coloring of G' is a proper 2-coloring. □

Let Y be an S-forest of G and let $u \in V \setminus Y$. If $G[Y \cup \{u\}]$ contains an induced cycle C_u that contains u and some vertex of S, then we say that C_u is a *witness cycle* of u.

Observation 3.2 *Let Y be a maximal S-forest of G. Then every vertex $u \in V \setminus Y$ has a witness cycle C_u.*

Proof Let Y be a maximal S-forest, and thus $G[Y \cup \{u\}]$ has a cycle C containing u and some vertex $v \in S$. Note that u and v might be the same vertex. Since C contains v, we have that v has at least two neighbors, x and y that belong to C. Let P be a shortest x, y-path in $G[V(C) \setminus \{v\}]$. Then $G[V(P) \cup \{v\}]$ contains an induced cycle C'. This cycle contains v since P is a shortest x, y-path. It also contains u since no such cycle exists in $G[Y]$. Thus $C' = C_u$ is a witness cycle. □

We are ready to proceed with the description of the enumeration algorithm, which is given by a sequence of reduction and branching rules. We always assume that the rules are performed in the order in which they are given (numbered), such that a rule is only applied if none of the previous rules can be applied.

Initially all vertices of G are non-colored. Vertices that are colored red or blue have already been decided to be in every maximal S-forest that is an element of $RBF(G, S, R, B)$. For a non-colored vertex v, we branch on two subproblems, and the cardinality of $RBF(G, S, R, B)$ is the sum of cardinalities of the sets of maximal S-forests that contain v and those that do not. The first set is represented by coloring vertex v red or blue, and second set is obtained by deleting v. This partitioning defines a naive branching, where a leaf is reached when there is at most one maximal S-forest in the set. We define the following two procedures, which take as input vertex v and $RBF(G, S, R, B)$.

Coloring of vertex v:
  if $v \in S$ *then* proceed with $RBF(G, S, R \cup \{v\}, B)$;
  if $v \notin S$ *then* proceed with $RBF(G, S, R, B \cup \{v\})$.
Deletion of vertex v:
  proceed with $RBF(G[V \setminus \{v\}], S \setminus \{v\}, R, B)$.

After the description of each of the Rules 1–12, we argue that the rule is *sound*, which means that there is a one-to-one correspondence between the maximal S-forests in the problem instance and the maximal S-forests in the instances of the subproblem(s). We start to apply the rules on instance $RBF(G, S, \emptyset, \emptyset)$.

Rule 1 If G has a vertex of degree at most 1 then remove this vertex from the graph.

Rule 1 is sound because a vertex of degree zero or one does not belong to any cycle. Furthermore, when a vertex of degree zero or one is removed, every vertex that previously belonged to a cycle still belongs to a cycle and maintains degree at least two.

Note that removal of vertex v means v belongs to every element of $RBF(G, S, R, B)$. We emphasize that there is a crucial difference to *Deletion* of vertex v which means that the non-colored vertex v belongs to no element of $RBF(G, S, R, B)$. Such a removal of a vertex belonging to every element of $RBF(G, S, R, B)$ is done in Rules 1, 4 and 5 and it necessitates the backtracking part of our algorithm to be explained later.

Rule 2 If $R = \emptyset$, and $S \neq \emptyset$ then select an arbitrary non-colored vertex $v \in S$, and branch into two subproblems. One subproblem is obtained by applying *Deletion* of v and the other by *Coloring* of v.

Rule 2 is sound for the following reason. Only vertices of S are colored red. Thus if $R = \emptyset$, all vertices of S are non-colored vertices. For every maximal S-forest Y, we have that either $v \in Y$ (corresponding to *Coloring* of v), or $v \notin Y$ (corresponding to *Deletion* of v).

After the application of Rule 2 there always exists a red vertex, unless $S = \emptyset$ when we reached a leaf of the branching tree. For many of the following rules we need to fix a particular vertex t of the S-forest $R \cup B$. We call it *pivot vertex* t. If no pivot vertex exists (at some step a pivot vertex might be deleted), we apply the following rule to select a new one.

Rule 3 If there is no pivot vertex then select a red vertex as new pivot vertex t.

The following reduction rule is to ensure (by making use of Observation 3.1) that the graph $G[R \cup B]$ induces a forest and that the current red-blue coloring is a proper 2-coloring of this forest.

Rule 4 If there are two adjacent red vertices u, v, then contract edge $\{u, v\}$ to u to obtain a new graph G'. Let Z be the set of non-colored vertices that are adjacent to u via multiple edges in G'. If v was the pivot then use u as new pivot t. Proceed with problem instance $RBF(G' \setminus Z, S \setminus (\{v\} \cup Z), R \setminus \{v\}, B)$.

Observe that Rule 4 corresponds to applying *Deletion* of w for every vertex w of Z. Let us argue why this rule is sound. If a vertex w belongs to Z, then because $u, v \in S$, we have that w cannot be in any S-forest of G. Thus applying *Deletion* of this vertex does not change the set of maximal S-forests. Finally, every induced cycle of length more than 3 in G corresponds to a cycle of length at least 3 in the reduced instance.

Rule 5 If there are two adjacent blue vertices u, v, then contract edge $\{u, v\}$ to u to obtain a new graph G'. Let Z be the set of non-colored vertices of S that are adjacent to u via multiple edges in G'. New problem instance is $RBF(G' \setminus Z, S \setminus Z, R, B \setminus \{v\})$.

Observe that Rule 5 corresponds to applying *Deletion* of w for every vertex w of Z. No vertex of Z can be in an S-forest containing u and v. Thus applying *Deletion* of the vertices of Z is sound. As with the previous rule, every induced cycle of length more than 3 in G corresponds to a cycle of length at least 3 in the reduced instance. We conclude that Rule 5 is sound.

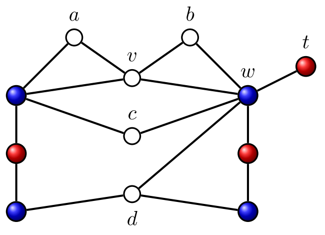

Fig. 1 Let $S \cap \{a, b, c, d, v\} = \emptyset$ and set $P(v) = \{a, b, c, d\}$. Vertex $a \in PW(v)$ by (P2), $d \in PW(v)$ by (P3). Vertices b and c do not belong to $PW(v)$

Rule 6 If a non-colored vertex v has at least two distinct neighbors w_1, w_2 in the same connected component of $G[R \cup B]$, then apply *Deletion* of v.

Let us first argue that if none of Rules 1–6 can be applied to the current instance then $R \cup B$ induces a properly 2-colored forest. If there were a cycle in $G[R \cup B]$ then the last non-colored vertex of the cycle would have two colored neighbors in the same connected component of $G[R \cup B]$ which is impossible by Rule 6. Moreover Rules 4 and 5 ensure that the red-blue coloring is a proper 2-coloring of this forest. In the following we will call such a forest (tree) a *red-blue forest* (tree). Observe that any colored path is a red-blue path.

For the soundness of Rule 6 note that the connected component of $G[R \cup B]$ that contains w_1 and w_2 is a red-blue tree T. Let $w_1, u_1, \dots, u_p = w_2$, $p \geq 1$, be the unique induced path in T between w_1 and w_2. Then either w_1 or u_1 is a red vertex, and thus belongs to S. Hence no element of $RBF(G, S, R, B)$ contains v. This shows that Rule 6 is sound.

Let T_t be the vertices of the connected component of $G[R \cup B]$ containing the pivot vertex t. Consider a non-colored vertex v adjacent to a vertex of the red-blue tree $G[T_t]$. Observe that v has exactly one neighbor w in T_t, by Rule 6. By Observation 3.2, every vertex u, which is not in a maximal S-forest Y, should have a witness cycle C_u such that all vertices of C_u except u are in Y. Hence every vertex $u \notin Y$ has at least two neighbors in Y. Since we cannot apply Rule 6 on the vertex v, this implies that if v is not in Y, at least one of the vertices from $N(v) \setminus T_t$ is in Y.

For a non-colored vertex v adjacent to a vertex of T_t, we define vertex set $P(v)$ to be the set of non-colored vertices adjacent to v or reachable from v via induced red-blue paths in $G[V \setminus T_t]$. Let w be the unique neighbor of v in T_t. We define vertex set $PW(v)$ to be the subset of $P(v)$ consisting of every vertex x of $P(v)$ for which at least one of the following conditions holds:

P1 $\{w, v, x\} \cap S \neq \emptyset$,
P2 $x \notin N(w)$, or
P3 there exists an induced red-blue path from x to v in $G[V \setminus T_t]$ containing at least one red vertex.

See Fig. 1 for an example of sets $P(v)$ and $PW(v)$. The intuition behind the definition of $PW(v)$ is the following. If a vertex v does not belong to any maximal S-forest Y of G, then there is a witness cycle C_v. This cycle C_v may pass through some connected components of $G[R \cup B]$ and some non-colored vertices. If we traverse C_v starting from v and avoiding T_t, then the first non-colored vertex we meet will be a vertex of $PW(v)$. Note that the vertex set $PW(v)$ can easily be computed in polynomial time.

Observation 3.3 *For every vertex $x \in PW(v) \cap N(T_t)$, there is an induced cycle containing x and v and at least one vertex of S. Furthermore this is a cycle in the subgraph of G induced by the union of $T_t \cup \{v, x\}$ and the vertex set of a red-blue path from v to x, and thus it contains only two non-colored vertices, namely x and v.*

Proof The fact that x and v have neighbors in T_t, implies that the subgraph of G induced by the union of $T_t \cup \{v, x\}$ and the vertex set of a red-blue path from v to x, contains an induced cycle C. This cycle C contains v and the neighbor w of v in T_t. By Rule 6, if there is an induced cycle containing non-colored vertex v and a vertex of S, then this cycle should contain another non-colored vertex. In the induced subgraph the only non-colored vertex except v is x, and thus C contains x as well. Because $x \in PW(v)$, at least one the properties P1–P3 should hold. If one of the vertices w, v, x is in S, we are done. If $x \notin N(w)$, then C contains more than one vertex from T_t, and thus at least one red vertex from S. The only remaining case is $\{v, w, x\} \cap S = \emptyset$, $x \in N(w)$, and there is a red-blue path P from v to x in $G[V \setminus T_t]$ containing a red vertex. But every red vertex is in S. □

Observation 3.4 *Let v be a non-colored vertex adjacent to a vertex of T_t. If there is an induced cycle C in G that contains v and some vertex of S, then C contains also at least one vertex of $P(v)$.*

Proof First note that C does not have to pass through the vertices of T_t. By Rule 6, C contains at least one non-colored vertex besides v. The vertex v has two neighbors in C, let x be a neighbor of v on C not equal to w, the neighbor of v in T_t. If $x \in P(v)$, we are done. If x is a red or blue vertex then x belongs to some red-blue tree T of $G[B \cup R]$. Cycle C has to leave T at some point, and thus to enter a vertex of $P(v)$. □

Lemma 3.5 *A witness cycle C_v of a vertex v contains a vertex of $PW(v)$.*

Proof Let us assume that v is not contained in a maximal S-forest Y, and let C_v be a witness cycle for v. By Observation 3.4, C_v contains at least one vertex x of $P(v)$.

If $x \in PW(v)$, we are done with the proof. Otherwise, let $x \in P(v) \setminus PW(v)$. As a consequence, $\{v, w, x\} \cap S = \emptyset$, x is adjacent to w, and every induced red-blue path from v to x in $G[V \setminus T_t]$ contains only blue vertices.

We will first show that C_v contains a vertex $x' \neq x$ such that $x' \in P(v) \setminus PW(v)$. Let us trace the induced cycle C_v, starting from v on the path to x using only blue vertices. By definition, no vertex on the path from v to x is contained in S. Observation 3.2 and the definition of witness cycle imply that C_v contains a vertex of S. Continue now in the same direction along C_v until a vertex of S is reached. The cycle has to return to v without passing through the vertex w; otherwise the edge $\{v, w\}$ would be a chord of C_v contradicting the fact that C_v is an induced cycle. The path along the cycle C_v from x to v containing a vertex of S cannot be a red-blue path as this contradicts the definition of x. As a consequence, C_v contains a second vertex x' of $P(v) \setminus PW(v)$.

Let Y be a maximal S-forest. Since w is colored blue, Y contains w. Further we have $C_v \setminus Y = \{v\}$ due to the maximality of Y. Denote by P the path from x to x' on C_v not containing v. Now we have a contradiction since the graph $G[P \cup \{w\}]$ contains the edges $\{x, w\}$ and $\{x', w\}$ and induces a cycle containing a vertex of S. □

The following rules depend on the cardinality of the set $PW(v)$.

Rule 7 If $PW(v) = \emptyset$ then apply *Coloring* of v.

Rule 8 If $PW(v) = \{x\}$ then branch into two subproblem instances: one obtained by applying *Deletion* of v and then *Coloring* of x, and the other obtained by applying *Coloring* of v.

Rules 7 and 8 are sound due to Lemma 3.5.

Rule 9 If $|PW(v)| \geq 2$ and $PW(v) \subseteq N(T_t)$ then branch into two subproblem instances: one obtained by applying *Coloring* of v and then *Deletion* of x for all vertices $x \in PW(v)$, and the other obtained by *Deletion* of v.

To see that Rule 9 is sound, observe that vertex v is either colored, or deleted. By Observation 3.3, for each vertex $x \in PW(v) \cap N(T_t)$ the subgraph of G induced by x, v, T_t and the vertex set of a red-blue path from x to v contains a cycle with a vertex of S and x and v as its only non-colored vertices. Thus either x or v has to be deleted, for every $x \in PW(v) \cap N(T_t)$. When Rule 9 can not be applied, at least one of the vertices in $PW(v)$ is not contained in $N(T_t)$.

Rule 10 If $PW(v) = \{x_1, x_2\}$ and $x_1 \notin N(T_t)$ then branch into three subproblem instances. The first one is obtained by applying *Coloring* of v. The second by *Deletion* of v and then *Coloring* of x_1. The third one by applying *Deletion* of v and x_1 and then *Coloring* of x_2.

Let us remark that vertex v is either colored or deleted. If v is deleted then by Lemma 3.5, either x_1 or x_2 is contained in the witness cycle C_v. This shows that Rule 10 is sound.

Rule 11 If $PW(v) = \{x_1, x_2, x_3\}$ and $x_1 \notin N(T_t)$ then branch into four subproblem instances. The first instance is obtained by applying *Coloring* of v. The second by *Deletion* of v and then *Coloring* of x_1. The third by *Deletion* of v and x_1 and then *Coloring* of x_2. The fourth by *Deletion* of v, x_1, x_2 and *Coloring* of x_3.

Again, the soundness of this rule follows by Lemma 3.5.

Rule 12 If $|PW(v)| \geq 4$ then create two problem instances: one obtained by applying *Coloring* of v, and the other obtained by applying *Deletion* of v.

This rule is sound because v is either colored or deleted.

We call an instance *non-reducible* if none of Rules 1–12 can be applied to it. Such an instance corresponds to a leaf of the branching tree of our algorithm. The following property of non-reducible instances of the red-blue S-forest problem is crucial for our arguments.

Lemma 3.6 *Let (G, S, B, R) be an instance. If none of the Rules 1–12 can be applied then $RBF(G, S, R, B)$ contains at most one maximal red-blue S-forest. Moreover, this forest can be computed in polynomial time.*

Proof If $S = \emptyset$ then trivially the only maximal S-forest of G is V. Let us assume that $S \neq \emptyset$. With every rule we either remove a vertex, select a pivot vertex, color a vertex, delete a vertex, or contract an edge. Rule 2 guarantees that the set of red vertices is not empty. Rule 3 ensures that a pivot vertex t is selected. Rules 1, 2 and 4–12 can be applied as long as there are non-colored neighbors of red-blue tree T_t. When the set $N(T_t)$ becomes empty then T_t is completely removed by Rule 1. Then the algorithm selects a new pivot vertex t and component T_t by making use of Rule 3. Thus the conditions that none of the rules can be applied and $S \neq \emptyset$, yield that $V = R \cup B$. But then the only possible maximal S-forest Y of $RBF(G, S, R, B)$ is $Y = R \cup B$ which can easily be computed in polynomial time. □

Let us note that a non-reducible instance does not necessarily correspond to a *maximal* red-blue S-forest. This is mainly due to Rules 10 and 11 in which we delete set of vertices in one branch. Such a deletion does not necessarily preserve maximality. As a simple example consider the graph $G[\{w, v, x_1, x_2\}]$ that induces a clique without the edge $\{w, x_1\}$ and w colored red. The application of Rule 10 results in a branch after the deletion of both v and x_1 that contains a red-blue S-forest which is not maximal.

We are finally in the position to describe the algorithm. The algorithm enumerates all elements of $RBF(G, S, \emptyset, \emptyset)$ by applying Rules 1–12 in priority of their numbering as long as possible. Let $\mathcal{F}$ be the set of all non-reducible instances produced by the application of the rules. These are the instances corresponding to the leaves of the branching tree. By Lemma 3.6, for each non-reducible instance of $\mathcal{F}$ there is at most one red-blue S-forest which can be computed in polynomial time. To enumerate all maximal S-forests of the input graph, we have to add to each S-forest of an instance of $\mathcal{F}$ all vertices which were possibly removed by applications of some of the rules on the unique path from the root of the branching tree to the corresponding leaf. This can be done in polynomial time by backtracking in the branching tree.

The correctness of the algorithm follows by Lemma 3.6 and the fact that each rule is sound. Thus each maximal S-forest of the input graph can be mapped to a private element of $\mathcal{F}$. In the next section we analyze the running time.

4 Running Time Analysis of the Enumeration Algorithm

With every rule we either remove a vertex, select a pivot vertex, color a vertex, delete a vertex, or contract an edge. Thus the height of the branching tree is $O(|V| + |E|)$.

Hence, for every non-reducible instance, the backtracking part of the algorithm producing the corresponding maximal S-forest in G can be performed in polynomial time. Therefore, the running time of the algorithm, up to a polynomial multiplicative factor, is proportional to the number of non-reducible instances produced by reduction and branching rules.

In what follows, we upper bound the number of maximal S-forests of the input graph enumerated by the algorithm, or equivalently, the number of leaves in the corresponding branching tree. Rules 1, 3, 4, 5, 6, and 7 are reduction rules and generate only one problem instance. Thus they do not increase the number of leaves in the branching tree. Therefore we may restrict ourselves to the analyses of the branching Rules 2, 8, 9, 10, 11, and 12.

Our proof combines induction with Measure & Conquer [10]. Let us first define a measure for any problem instance generated by the algorithm. All colored vertices have weight 0, non-colored vertices contained in $N(T_t)$ have weight 1, and non-colored vertices not contained in $N(T_t)$ have weight $1+\alpha$. A problem instance $\mathit{RBF}(G,S,R,B)$ will be defined to have weight $|N(T_t)| + (1+\alpha)|V \setminus (R \cup B \cup N(T_t))|$. Define $f(\mu)$ to be the maximum number of leaves in the branching tree for any instance $\mathit{RBF}(G,S,R,B)$ of weight μ where $\mu \geq 0$ is a real number.

The induction hypothesis is that $f(\mu) \leq x^{\mu}$ for $x = 1.49468$. Note that the number of possible measures of problem instances is finite, and thus induction is over a finite set.

For the base case, let $\mu = 0$. Since no vertex has weight greater than 0, we have that all vertices are colored, and thus V is the unique maximal S-forest, implying $f(0) = 1$. By the induction hypothesis we assume that $f(k) \leq x^k$ for $k < \mu$, and we want to prove that $f(\mu) \leq x^{\mu}$. We prove this by showing that each rule reduces a problem instance of weight μ to one or more problem instances of weight $\mu_1, \dots, \mu_r$ where $\mu_i < \mu$ such that $f(\mu) \leq \sum_{i=1}^{r} f(\mu_i) \leq x^{\mu}$ if $f(\mu_i) \leq x^{\mu_i}$ for $1 \leq i \leq r$. Before proceeding to the detailed analysis, we mention that the instance $\mathit{RBF}(G,S,\emptyset,\emptyset)$ has weight $n(1+\alpha)$, and the result will thus imply that $f(n(1+\alpha)) \leq 1.49468^{n(1+\alpha)} \leq 1.8638^n$ for $\alpha = 0.5491$.

Rule 2. Since R is empty, pivot vertex t is undefined, and $N(T_t)$ is defined to be the empty set. As a consequence all non-colored vertices have weight $1+\alpha$. In both new instances the weight of v is reduced from $1+\alpha$ to zero. In the case when v is colored (Rule 3), we use v as vertex t, and due to the minimum degree 2 property by Rule 1, there are at least two neighbors with weights reduced by α. The two subproblem instances are *Deletion* of v: $\mu_1 \leq \mu - 1 - \alpha$ and *Coloring* of v: $\mu_2 \leq \mu - 1 - 3\alpha$, and we get that

$$f(\mu) \leq f(\mu - 1 - \alpha) + f(\mu - 1 - 3\alpha) \leq x^{\mu-1-\alpha} + x^{\mu-1-3\alpha} \leq x^{\mu}.$$

Rule 8. In both cases the weight of the vertex v is reduced from 1 to zero. If x is contained in $N(T_t)$ then it has weight 1, otherwise x has weight $1+\alpha$. Consider first the case $x \in N(T_t)$. Since $x \in PW(v)$, we have by Observation 3.3 that the subgraph of G induced by x, v, T_t and the vertex set of a red-blue path from x to v contains a cycle with a vertex of S and x and v as its only non-colored vertices. Hence either v or x has to be deleted. If v is colored, then x is deleted by Rule 6 in order to break the abovementioned cycle, and if v is deleted, then x is colored since it has to be in the witness cycle. We have for *Deletion* of v and *Coloring* of x: $\mu_1 \leq \mu - 2$; for *Deletion* of x and *Coloring* of v: $\mu_2 \leq \mu - 2$. Thus

$$f(\mu) \leq f(\mu-2) + f(\mu-2) \leq x^{\mu-2} + x^{\mu-2} \leq x^{\mu}$$

If $x \notin N(T_t)$, then the weight of x is $1+\alpha$, and we have for *Deletion* of v and *Coloring* of x: $\mu_1 \leq \mu - 2 - \alpha$; for *Coloring* of v: $\mu_2 \leq \mu - 1 - \alpha$, resulting in

$$f(\mu) \leq f(\mu-2-\alpha) + f(\mu-1-\alpha) \leq x^{\mu-2-\alpha} + x^{\mu-1-\alpha} \leq x^{\mu}.$$

Rule 9. All vertices in $PW(v)$ have weight 1. Thus we have for *Coloring* of v and *Deletion* of $PW(v)$: $\mu_1 \leq \mu - 1 - |PW(v)|$; for *Deletion* of v: $\mu_2 \leq \mu - 1$. Since $|PW(v)| \geq 2$, we have that

$$f(\mu) \leq f(\mu-3) + f(\mu-1) \leq x^{\mu-3} + x^{\mu-1} \leq x^{\mu}.$$

Rule 10. The vertex v has weight 1, x_1 has weight $1+\alpha$, and x_2 has weight 1 or $1+\alpha$. Consider first the case where x_2 has weight 1, meaning that $x_2 \in N(T_t)$. If v is colored, then x_2 is deleted by Rule 6 and Observation 3.3. We have for *Coloring* of v: $\mu_1 \leq \mu - 2 - \alpha$; *Deletion* of v and *Coloring* of x_1: $\mu_2 \leq \mu - 2 - \alpha$; and *Deletion* of v, x_1 and *Coloring* of x_2: $\mu_3 \leq \mu - 3 - \alpha$. Thus

$$f(\mu) \leq 2f(\mu-2-\alpha) + f(\mu-3-\alpha) \leq 2x^{\mu-2-\alpha} + x^{\mu-3-\alpha} \leq x^{\mu}.$$

If $x_2 \notin N(T_t)$, then it has weight $1+\alpha$. We have for *Coloring* of v: $\mu_1 \leq \mu - 1 - 2\alpha$; *Deletion* of v and *Coloring* of x_1: $\mu_2 \leq \mu - 2 - \alpha$; and *Deletion* of v, x_1 and *Coloring* of x_2: $\mu_3 \leq \mu - 3 - 2\alpha$. Therefore,

$$\begin{aligned} f(\mu) &\leq f(\mu-1-2\alpha) + f(\mu-2-\alpha) + f(\mu-3-2\alpha) \\ &\leq x^{\mu-1-2\alpha} + x^{\mu-2-\alpha} + x^{\mu-3-2\alpha} \leq x^{\mu}. \end{aligned}$$

Rule 11. Let i be the number of vertices in $PW(v) \setminus N(T_t)$ and assume that $x_j \notin N(T_t)$ for $j \leq i$. The case $i = 0$ is covered by Rule 9. For $i = 1, 2$, we have for *Coloring* of v: $\mu_1 \leq \mu - 4 + i - i\alpha$; *Deletion* of v and *Coloring* of x_1: $\mu_2 \leq \mu - 2 - \alpha$; *Deletion* of v, x_1 and *Coloring* of x_2: $\mu_3 \leq \mu - 3 - i\alpha$; and *Deletion* of v, x_1, x_2 and *Coloring* of x_3: $\mu_4 \leq \mu - 4 - i\alpha$. In total

$$\begin{aligned} f(\mu) &\leq f(\mu-4+i-i\alpha) + f(\mu-2-\alpha) + f(\mu-3-i\alpha) + f(\mu-4-i\alpha) \\ &\leq x^{\mu-4+i-i\alpha} + x^{\mu-2-\alpha} + x^{\mu-3-i\alpha} + x^{\mu-4-i\alpha} \leq x^{\mu}. \end{aligned}$$

For $i = 3$, we have for *Coloring* of v: $\mu_1 \leq \mu - 1 - 3\alpha$, for *Deletion* of v and *Coloring* of x_1: $\mu_2 \leq \mu - 2 - \alpha$, *Deletion* of v, x_1 and *Coloring* of x_2: $\mu_3 \leq \mu - 3 - 2\alpha$, and *Deletion* of v, x_1, x_2 and *Coloring* of x_3: $\mu_4 \leq \mu - 4 - 3\alpha$, and we get that

$$\begin{aligned} f(\mu) &\leq f(\mu-1-3\alpha) + f(\mu-2-\alpha) + f(\mu-3-2\alpha) + f(\mu-4-3\alpha) \\ &\leq x^{\mu-1-3\alpha} + x^{\mu-2-\alpha} + x^{\mu-3-2\alpha} + x^{\mu-4-3\alpha} \leq x^{\mu}. \end{aligned}$$

Rule 12. Let i be the number of vertices in $PW(v)\setminus N(T_t)$ and assume that $x_j \notin N(T_t)$ for $j \le i$. The case where $i = 0$ is covered by Rule 9. For $i \ge 1$, we have for *Coloring* of v: $\mu_1 \le \mu - (1 + |PW(v)|) + i - i\alpha$; and for *Deletion* of v: $\mu_2 \le \mu - 1$. Since $|PW(v)| \ge 4$, we notice that the value is minimum when $i = 4$ and we get

$$f(\mu) \le f(\mu - 1 - 4\alpha) + f(\mu - 1) \le x^{\mu-1-4\alpha} + x^{\mu-1} \le x^{\mu}.$$

We conclude the analysis of the running time of the algorithm with the following theorem, which is the main result of this paper.

Theorem 4.1 *Let G be a graph and let S be a set of vertices in G. The maximum number of maximal S-forests of G is at most 1.8638^n. All minimal subset feedback vertex sets of (G, S) can be enumerated in time $O(1.8638^n)$.*

Proof Correctness and completeness follows from the arguments above. The number of leaves in the branching tree is at most $x^{(1+\alpha)n} n^{O(1)}$ and $1.49468^{1+0.5491} < 1.8638$. □

It is worth mentioning that the running time of our algorithm enumerating all minimal subset feedback vertex sets is the same as the running time of the algorithm enumerating all minimal feedback vertex sets given in [9].

5 Conclusion

We start this section with two direct consequences of Theorem 4.1, before we give a few remarks and open questions on the number of minimal subset feedback vertex sets. Since we can check whether a given set X is a subset feedback vertex set of (G, S) and determine its total weight in polynomial time, the following result is an immediate consequence of Theorem 4.1.

Corollary 5.1 SUBSET FEEDBACK VERTEX SET *can be solved in time $O(1.8638^n)$.*

In the introductory section, we mentioned that MULTIWAY CUT is a special case of SUBSET FEEDBACK VERTEX SET. This is explained in the proof of the following result, which thus follows from Theorem 4.1 and the above corollary.

Corollary 5.2 *Let G be a graph and let T be a set of vertices (terminals) in G. All minimal multiway cuts of (G, T) can be enumerated in time $O(1.8638^n)$, and* MULTIWAY CUT *can be solved in time $O(1.8638^n)$.*

Proof Let (G, T) be an instance of MULTIWAY CUT. We construct a new graph G' by adding to G a new vertex s whose weight is larger than the sum of the weights of all vertices in G, and by making s adjacent to all terminals in T. Any cycle of G' containing s corresponds to a path in G connecting two terminals of T. Thus a vertex subset is a minimum weight subset feedback set of $(G', \{s\})$ if and only if it is a minimum weight multiway cut of (G, T). □

We would like to remark that the number of minimal feedback vertex sets in a graph can be exponentially larger or smaller than the number of minimal subset feedback vertex sets. For example, the graph consisting of $n/3$ disjoint triangles, has $3^{n/3}$ minimal feedback vertex sets (every triangle contains exactly one vertex from every such a set), whereas if $S = \emptyset$, the only minimal subset feedback vertex set is $\emptyset$. The example of a graph with polynomial number of minimal feedback vertex sets and exponential number of minimal subset feedback vertex sets is the following split graph G on $n = 6k$ vertices. Graph G has a clique C of size $3k$ and an independent set I of size $3k$. The vertices of C and I are partitioned into k triples; clique triples (a_i, b_i, c_i) and independent set triples (x_i, y_i, z_i), $1 \le i \le k$. For each i, we add edges between vertices of clique and independent set triples as follows: x_i is adjacent to a_i, b_i; y_i to b_i, c_i; and z_i to a_i, c_i. We let $S = I$. Every minimal subset feedback vertex set contains exactly 2 vertices from each clique triple, so there are 3 possible options for each triple, and the total number of such sets is 3^k. On the other hand, every minimal feedback vertex set should contain at least $3k - 2$ vertices from C, and thus the number of such sets is $O(k^2)$.

We close with a couple of open questions. Fomin et al. [9] show that there are graphs with 1.5926^n minimal feedback vertex sets. However, no graph with 1.5927^n or more minimal feedback vertex sets is known. Are there graphs having 1.5927^n or more minimal feedback vertex sets or minimal subset feedback vertex sets? Can it be that our enumeration algorithm overestimates the maximum number of minimal subset feedback vertex sets, and that this number is significantly smaller than 1.8638^n, say $O(1.6^n)$? It is known that all minimal feedback vertex sets of a graph can be enumerated by an output-sensitive algorithm of polynomial delay [18]. Are there output-sensitive algorithms enumerating all subset feedback vertex sets of output-polynomial running time or even of polynomial delay? As mentioned, our enumeration algorithm can be used to solve SUBSET FEEDBACK VERTEX SET in time $O(1.8638^n)$. It would be interesting to know whether a better running time can be obtained for unweighted SUBSET FEEDBACK VERTEX SET.

Acknowledgements The authors would like to thank the anonymous referees whose valuable suggestions helped improve the presentation of the paper.

References

1. Calinescu, G.: Multiway cut. In: Encyclopedia of Algorithms, vol. 12, pp. 1–99. Springer, Berlin (2008)
2. Cao, Y., Chen, J., Liu, Y.: On feedback vertex set new measure and new structures. In: Proceedings of SWAT 2010. LNCS, vol. 6139, pp. 93–104. Springer, Berlin (2010)
3. Chen, J., Fomin, F.V., Liu, Y., Lu, S., Villanger, Y.: Improved algorithms for feedback vertex set problems. J. Comput. Syst. Sci. **74**(7), 1188–1198 (2008)
4. Corneil, D.G., Fonlupt, J.: The complexity of generalized clique covering. Discrete Appl. Math. **22**(2), 109–118 (1989)
5. Cygan, M., Nederlof, J., Pilipczuk, M., Pilipczuk, M., van Rooij, J.M.M., Wojtaszczyk, J.O.: Solving connectivity problems parameterized by treewidth in single exponential time. In: Proceedings of FOCS 2011, pp. 150–159. IEEE Press, New York (2011)
6. Cygan, M., Pilipczuk, M., Pilipczuk, M., Wojtaszczyk, J.O.: Subset feedback vertex set is fixed parameter tractable. In: Proceedings of ICALP 2011. LNCS, vol. 6755, pp. 449–461. Springer, Berlin (2011)

7. Even, G., Naor, J., Zosin, L.: An 8-approximation algorithm for the subset feedback vertex set problem. SIAM J. Comput. **30**(4), 1231–1252 (2000)
8. Fomin, F.V., Heggernes, P., Kratsch, D., Papadopoulos, C., Villanger, Y.: Enumerating minimal subset feedback vertex sets. In: Proceedings of WADS 2011. LNCS, vol. 6844, pp. 399–410. Springer, Berlin (2011)
9. Fomin, F.V., Gaspers, S., Pyatkin, A.V., Razgon, I.: On the minimum feedback vertex set problem: exact and enumeration algorithms. Algorithmica **52**(2), 293–307 (2008)
10. Fomin, F.V., Grandoni, F., Kratsch, D.: A measure & conquer approach for the analysis of exact algorithms. J. ACM **56**(5), 25 (2009)
11. Fomin, F.V., Kratsch, D.: Exact Exponential Algorithms. Texts in Theoretical Computer Science. Springer, Berlin (2010)
12. Fomin, F.V., Villanger, Y.: Finding induced subgraphs via minimal triangulations. In: Proceedings of STACS 2010, vol. 5, pp. 383–394. Schloss Dagstuhl—Leibniz-Zentrum fuer Informatik, Leibniz (2010)
13. Garey, M.R., Johnson, D.S.: Computers and Intractability. Freeman, New York (1978)
14. Garg, N., Vazirani, V.V., Yannakakis, M.: Multiway cuts in node weighted graphs. J. Algorithms **50**(1), 49–61 (2004)
15. Karp, R.M.: Reducibility among combinatorial problems. In: Complexity of Computer Computations, pp. 85–103 (1972)
16. Raman, V., Saurabh, S., Subramanian, C.R.: Faster fixed parameter tractable algorithms for finding feedback vertex sets. ACM Trans. Algorithms **2**(3), 403–415 (2006)
17. Razgon, I.: Exact computation of maximum induced forest. In: Proceedings of SWAT 2006. LNCS, vol. 4059, pp. 160–171. Springer, Berlin (2006)
18. Schwikowski, B., Speckenmeyer, E.: On enumerating all minimal solutions of feedback problems. Discrete Appl. Math. **117**, 253–265 (2002)
19. Spinrad, J.P.: Efficient Graph Representations. Fields Institute Monograph Series, vol. 19. AMS, Providence (2003)
20. Thomassé, S.: A k^2 kernel for feedback vertex set. ACM Trans. Algorithms **6**(2), 32 (2010)