Directed Feedback Vertex Set Problem is FPT *

Jianer Chen[1], Yang Liu[1] and Songjian Lu[1]

Department of Computer Science
Texas A&M University
College Station, TX 77843, USA
{chen,yangliu,sjlu}@cs.tamu.edu

Abstract. To decide if the PARAMETERIZED FEEDBACK VERTEX SET problem in directed graph is fixed-parameter tractable is a long standing open problem. In this paper, we prove that the PARAMETERIZED FEEDBACK VERTEX SET in directed graph is fixed-parameter tractable and give the first FPT algorithm of running time $O((1.48k)^k n^{O(1)})$ for it. As the FEEDBACK ARC SET problem in directed graph can be transformed to a FEEDBACK VERTEX SET problem in directed graph, hence we also show that the PARAMETERIZED FEEDBACK ARC SET problem can be solved in time of $O((1.48k)^k n^{O(1)})$.

Keywords. directed feedback vertex set problem, parameterized algorithm, fixed-parameter tractability, network flow

1 Introduction

The FEEDBACK VERTEX SET problem is defined as: given a graph $G = (V, E)$, find a subset F in the graph such that $G - F$ is acyclic. We usually call F a feedback vertex set of G, or an FVS of G. The graph G can be undirected or directed. If we want to find an FVS with minimum size or with a size bounded by k which we call it PARAMETERIZED FEEDBACK VERTEX SET problem, the problem is NP-complete in both directed and undirected graph [15]. The PARAMETERIZED FEEDBACK VERTEX SET problem in undirected graph has been proved to be *fixed-parameter tractable* (FPT) [3,8] long time ago, i.e. the running time of the algorithm is bounded by $f(k)n^{O(1)}$, where $f(k)$ is a function that depends only on k. But for the PARAMETERIZED FEEDBACK VERTEX SET problem in directed graph, the problem remains open.

The FEEDBACK VERTEX SET problem, especially in directed graph, has important applications in Database System [14] and Operating System [27] to solve deadlock problems, such as the *deadlock recovery* in operation systems [27], in which a deadlock is presented by a cycle in a *system resource-allocation graph G*. Therefore, in order to recover from deadlocks, we need to abort a set of processes in the system, i.e., to remove a set of vertices in the directed graph G, so that all cycles in G are broken. Equivalently, we need to find an FVS in G.

When the graph G is undirected, there are considerable results about the FEEDBACK VERTEX SET problem, such as there are exact algorithms of finding a minimum FVS in a graph on n vertices in time $O(1.9053^n)$ [26] and in time $O(1.7548^n)$ [13]. There is also a polynomial time 2-approximation algorithm for the MINIMUM FEEDBACK VERTEX SET problem [1]. Bodlaender [3], Downey and Fellows [8] gave the first parameterized algorithms of running time $O(17k^4!n^{O(1)})$ for the PARAMETERIZED FEEDBACK VERTEX SET problem in undirected graph. Since then a chain of dramatic improvements was obtained by different researchers, such as an algorithm with a time complexity of $O((2k+1)^k n^2)$ by Downey and Fellows [9], of $O(\max\{12^k, (4\log k)^k\}n^{2.376})$ by Raman et al.[21], of $O((2\log k + 2\log\log k + 18)^k n^2)$ by Kanj et al.[19], of $O((12\log k/\log\log k + 6)^k n^{2.376})$ by Raman et al.[22], of $O((37.7)^k n^2)$ by Guo et al.[16] and of $O((10.6)^k n^3)$ by Dehne et al.[6]. The current best result has a time complexity of $O(5^k n^{O(1)})$ by Chen et al. [5].

In directed graph, the FEEDBACK VERTEX SET problem becomes harder. There are only limit progresses for it, since Karp [20] proved that to find an FVS of size bounded by k in directed graph is NP-complete in 1972. No exact algorithms with running time of $O(c^n n^{O(1)})$, where $c < 2$, and no polynomial time approximation algorithms with constant ratio have been found (there is an excellent polynomial time approximation algorithm with a ratio of $O(\log \tau^* \log\log \tau^*)$ [11], where τ^* is the size of the minimum feedback vertex set). For the PARAMETERIZED FEEDBACK VERTEX SET problem in directed graph, all current achievements only fall in some special directed graphs, such as in the *tournament* graph, while the problem in general directed graph becomes a long-standing open problem [6,7,9,10,16,17,18,19,22,23,24]. The *tournament* graph is a directed graph that there is exact one arc between every pair of vertices in the graph. As there is always a cycle of size 3 in the tournament graph if there are any cycles in the graph, it is trivial to obtain an FPT algorithm of running time $O(3^k n^{O(1)})$. Some other improvements for the FEEDBACK VERTEX SET problem in tournament graph have a time complexity of $O(2.5^k n^{O(1)})$ [28], of $O(2.42^k n^{O(1)})$ [24], of $O(2.18^k n^{O(1)})$ [12] and the current best algorithm has a running time of $O(2^k n^{O(1)})$ [7]. By the way, Raman and Saurabh gave an FPT algorithm of running time $O((c\sqrt{k/e})^k n^{O(1)})$ [23] for the PARAMETER FEEDBACK ARC SET (FAS) problem in tournament graph, which removes a subset of at most k edges to make the graph acyclic.

In this paper, we solve this well-known open problem in FPT world: is the PARAMETERIZED FEEDBACK VERTEX SET problem in general directed graph fixed-parameter tractable? Our answer is "Yes", the PARAMETERIZED FEEDBACK VERTEX SET problem in directed graph is fixed-parameter tractable. And we give an FPT algorithm of running time $O((1.48k)^k n^{O(1)})$ for it. Our idea is: first, using $O(k!)$ time to transform the PARAMETERIZED FEEDBACK VERTEX SET problem in directed graph into CONSTRAINED MULTICUT problem through DAG-BIPARTITION FVS, ORDERED DAG-BIPARTITION FVS problems (see Section 3 of the paper). Then design an FPT algorithm for the CONSTRAINED MULTICUT problem. Hence, obtain the first FPT algorithm for the PARAMETERIZED

FEEDBACK VERTEX SET problem in directed graph. As solving the PARAMETER FEEDBACK ARC SET problem in directed graph $D = (V, A)$ is equivalent to solving a PARAMETER FEEDBACK VERTEX SET problem in D's line graph [11], we also solve the PARAMETER FEEDBACK ARC SET in $O((1.48k)^k n^{O(1)})$ time.

The organization of our paper is as following: In section 2, we introduce an extended form of the Menger's Theorem in directed graph, which we will use it in Section 4. In section 3, we define the DAG-BIPARTITION FVS, ORDERED DAG-BIPARTITION FVS and CONSTRAINED MULTICUT problems. In section 4, we give the FPT algorithm for the CONSTRAINED MULTICUT problem. In section 5, we give the FPT algorithms for the DAG-BIPARTITION FVS, the PARAMETERIZED FEEDBACK VERTEX SET and the PARAMETERIZED FEEDBACK ARC SET problems.

2   The minimum V-cut from subset T_1 to subset T_2 in a digraph

In this section, we introduce an extended form of Menger's theorem. Let us start with some terminology.

Let $D = (V, A)$ be a directed graph and let u and v be two vertices in D. A *path from u to v* is a simple path in D that begins from u (has only outgoing arc of the path) and ends at v (has only incoming arc of the path). Let T and u be a subset and a vertex of D respectively, a *path from u to T* is a path from u to a vertex in T (a *path from T to u* is a path from a vertex in T to u). For two subsets T_1 and T_2 in D, a *path from T_1 to T_2* is a path from a vertex in T_1 to a vertex in T_2. Two paths are *internally disjoint* if there exists no vertex that is an internal vertex for both paths.

Given a directed graph $D = (V, A)$, a subset S is *a V-cut from vertex u to vertex v* (*a V-cut from subset T_1 to subset T_2*) if no path exits from u to v (from T_1 to T_2) in the subgraph $D - S$.

In a directed graph $D = (V, A)$, we use (u, v) to denote an arc from u to v. When we say *u is adjacent to v* (or *v is a neighbor of u*), we mean (u, v) is an arc in A. Let V' be a subset of V, we denote by $D(V')$ the subgraph of D that is induced by the vertex set V'

Finally, for a subset T in $D = (V, A)$, by *merging T* (*into a single vertex*), we mean the operation that first deletes all vertices in T then creates a new vertex w and makes $(w, u)/(u, w)$ be an outgoing/incoming arc of w if there is a vertex v in T such that $(v, u)/(u, v)$ is an arc in D.

The following directed vertex form of Menger's Theorem and its proof can be found in [4].

Proposition 1. [4] (The Directed Vertex Form of Menger's Theorem) *Let s and t be two distinct vertices of a directed graph D such that s is not adjacent to v, then the maximum number of internally disjoint directed paths from s to t in D equals the size of a minimum V-cut from s to t in D.*

In Lemma 1, we generalize Proposition 1 from the case of two vertices to the case of two vertex subsets.

Lemma 1. *Let T_1 and T_2 be two disjoint vertex subsets in a directed graph D such that no vertex in T_1 is adjacent to a vertex in T_2. Then the maximum number h of internally disjoint paths from T_1 to T_2 in D is equal to the size of a minimum V-cut from T_1 to T_2 in D. Moreover, for any set π of h internally disjoint paths from T_1 to T_2 in D, every minimum V-cut from T_1 to T_2 in D contains exact on vertex in each of paths in π.*

Proof. Let D' be the graph obtained from the graph D by merging the two vertex subsets T_1 and T_2 into two vertices t_1 and t_2, respectively. Note that t_1 is not adjacent to t_2 in D'.

By the definition of the merge operation, it is easy to verify that a vertex subset S is a V-cut from the vertex subset T_1 to the vertex subset T_2 in the graph D if and only if S is a V-cut from the vertex t_1 to the vertex t_2 in the graph D'. In particular, the size of a minimum V-cut from T_1 to T_2 in D is equal to the size of a minimum V-cut from t_1 to t_2 in D'. Moreover, it is also easy to verify that for any integer h', from a set of h' internally disjoint paths from T_1 to T_2 in D, we can construct a set of h' internally disjoint paths from t_1 to t_2 in D', and *vice versa*. Therefore, the maximum number of internally disjoint paths from T_1 to T_2 in D is equal to the maximum number of internal disjoint paths from t_1 to t_2 in D'. Now the first part of the lemma follows by applying Proposition 1 to the graph D'.

To prove the second part of the lemma, let S be a minimum V-cut, of size h, from T_1 to T_2 in D, and let π be a set of h internally disjoint paths from T_1 to T_2. The vertex set S must contain at least one vertex from each of the paths in π: otherwise there would be a path from T_1 to T_2 in $D - S$, contradicting the assumption that S is a V-cut from T_1 to T_2. Moreover, the set S cannot contain more than one vertex in any path in π: otherwise S would not be able to contain at least one vertex for each of the paths in π (note that the paths in π are internally disjoint). □

Lemma 1 provides an efficient algorithm that constructs the maximum number of internally disjoint paths and a minimum-size V-cut from a vertex subset to another vertex subset.

Lemma 2. *Let T_1 and T_2 be two disjoint vertex subsets in a directed graph $D = (V, A)$ such that no vertex in T_1 is adjacent to a vertex in T_2. Then in time $O((|V| + |A|)k)$, we can decide if the size h of a minimum V-cut from T_1 to T_2 is bounded by k, and in case $h \leq k$, construct h internally disjoint paths from T_1 to T_2.*

Proof. Let D' be the graph obtained from the graph D by merging the two vertex subsets T_1 and T_2 into two vertices t_1 and t_2, respectively. As discussed in the proof of Lemma 1, it suffices to show how to decide if the size h of a minimum V-cut from t_1 to t_2 in D' is bounded by k, and in case $h \leq k$, how to construct h internally disjoint paths from t_1 to t_2.

This can be done based on the standard approach to the MAXIMUM t_1-t_2 FLOW problem [4]. For this, we modify the new directed graph D' by replacing

each vertex u (except the vertices t_1 and t_2) by two vertices u_1 and u_2 with an arc from u_1 to u_2, connecting all u's incoming arcs to the vertex u_1 and connecting all u's outgoing arcs to the vertex u_2. Finally we set all edges to have capacity 1. Let the resulting flow graph be D''.

Applying Ford-Fulkerson's standard approach using augmenting paths, in time $O((|V| + |A|)k)$, we can either construct a t_1-t_2 flow of value larger than k in D'', or end up with a maximum t_1-t_2 flow of value h bounded by k. In the former case, we conclude that the size of a minimum V-cut from t_1 to t_2 in D' is larger than k, which implies that the size of a minimum V-cut from T_1 to T_2 in D is larger than k. In the latter case, h internally disjoint paths from t_1 and t_2 in D' can be easily constructed from the maximum t_1-t_2 flow of value h in D'', from which h internally disjoint paths from T_1 to T_2 in D can be constructed.   □

3   DAG-BIPARTITION FVS, ORDERED DAG-BIPARTITION FVS **and** CONSTRAINED MULTICUT **problems**

Before we introduce our algorithm for the FVS problem in directed graph, we introduce several problems that are closely related to the FVS problem in directed graphs. We also start with some terminology. Given a directed graph $D = (V, A)$ and two subsets V_1, V_2 of V, if $V_1 \cup V_2 = V$, $V_1 \cap V_2 = \emptyset$ and both induced subgraphs $D(V_1)$ and $D(V_2)$ are directed acyclic graphs, then we say that the pair (V_1, V_2) is a *DAG bipartition* of the directed graph $D = (V, A)$. In the DAG bipartition (V_1, V_2) of the directed graph $D = (V, A)$, if $f_{order} : V_2 \to \{1, 2, \ldots, |V_2|\}$ is a topological order of V_2 such that no arcs from u to v for any $f_{order}(u) \geq f_{order}(v)$, then we say (V_1, V_2, f_{order}) is an *ordered DAG bipartition* of the directed graph $D = (V, A)$. Now we define these problems:

DAG-BIPARTITION FVS: given a directed graph $D = (V, A)$, a DAG bipartition (V_1, V_2) of D, and an integer k, either find an FVS of size bounded by k in V_1 for the directed graph D, or report that no such an FVS exists.

ORDERED DAG-BIPARTITION FVS: given a directed graph $D = (V, A)$, an ordered DAG bipartition (V_1, V_2, f_{order}) of D, and an integer k, either find a subset F (also called FVS) of size bounded by k in V_1 such that there exist no paths from u to v for any u, v in V_2 and $f_{order}(u) \geq f_{order}(v)$ in the induced subgraph $D(V - F)$, or report that no such an F exists.

CONSTRAINED MULTICUT: given a directed acyclic graph $D = (V, A)$, $2l$ pairwise disjoint subsets (called terminal sets) $S_1, S_2, \ldots, S_l, T_1, T_2, \ldots, T_l$ of V and an integer k, we suppose that D satisfy: 1) any vertex in S_i for $1 \leq i < l$ has no incoming arcs; 2) any vertex in T_i for $1 \leq i \leq l$ has no outgoing arcs. Either find a subset S of V (called separator) that has no elements come from any terminal sets and includes at least one vertex of any path from S_j to T_i for $j \geq i$, or report no such an S exists.

We denote the instance of the DAG-BIPARTITION FVS problem as (D, V_1, V_2, k), the instance of the ORDERED DAG-BIPARTITION FVS problem as $(D, V_1, V_2, f_{order}, k)$ and the instance of the CONSTRAINED MULTICUT problem as $(D, (S_1, \ldots, S_l), (T_1, \ldots, T_l), k)$.

For the DAG-BIPARTITION FVS problem and the ORDERED DAG-BIPARTITION FVS problem, they have the following relation.

Theorem 1. *Given an instance (D, V_1, V_2, k) of the DAG-BIPARTITION FVS problem, then (D, V_1, V_2, k) has an FVS of size bounded by k if and only if there exists an instance $(D, V_1, V_2, f_{order}, k)$ of the ORDERED DAG-BIPARTITION FVS problem that also has an FVS of size bounded by k.*

Proof. If an instance $(D, V_1, V_2, f_{order}, k)$ of the ORDERED DAG-BIPARTITION FVS problem has an FVS F of size bounded by k, then there exist no paths from u to v in the induced subgraph $D(V_1 \cup V_2 - F)$ for any u, v in V_2 that satisfy $f_{order}(u) \geq f_{order}(v)$. So if the induced subgraph $D(V_1 \cup V_2 - F)$ has a cycle, this cycle can not include any vertex from V_2. But as the induced subgraph $D(V_1)$ is a directed acyclic graph, no cycle can include all vertices just from V_1. Thus the induced subgraph $D(V_1 \cup V_2 - F)$ has no cycle, i.e. $F \subseteq V_1$ is an FVS for the directed graph D. Therefore F is an FVS for the instance (D, V_1, V_2, k) of the DAG-BIPARTITION FVS problem.

For the other direction, if the instance (D, V_1, V_2, k) of the DAG-BIPARTITION FVS problem has an FVS F of size bound by k, then $F \subseteq V_1$ is also an FVS for the directed graph D. So $D(V_1 \cup V_2 - F)$ is a directed acyclic graph. Let $(v_1, v_2, \ldots v_{|V_1 \cup V_2 - F|})$ be a topological order of the set $V_1 \cup V_2 - F$ such that no paths from v_j to v_i for $j \geq i$. If we use this order to define the order function f_{order} for the subset V_2, then it is obvious that there exist no paths from u to v in the induced graph $D(V_1 \cup V_2 - F)$ for any u, v in V_2 that satisfy $f_{order}(u) \geq f_{order}(v)$ and there exist no arcs from u to v for any u, v in V_2 that satisfy $f_{order}(u) \geq f_{order}(v)$, i.e. F is an FVS of size bounded by k for the instance $(D, V_1, V_2, f_{order}, k)$ of the ORDERED DAG-BIPARTITION FVS problem. $\square$

Given an instance $I = (D, V_1, V_2, f_{order}, k)$ of the ORDERED DAG-BIPARTITION FVS problem, where $V_2 = \{v_1, v_2, \ldots, v_l\}$ and $f_{order}(v_i) = i$, we replace each $v_i \in V_2$ by two vertices s_i and t_i, connect all v_i's incoming arcs to the vertices t_i and connect all v_i's outgoing arcs to the vertex s_i. Then we obtain a new directed graph $D' = (V', A')$. In the new directed graph $D' = (V', A')$, we let $S_i = \{s_i\}$ and $T_i = \{t_i\}$ for all $1 \leq i \leq l$, then we have the following lemma.

Lemma 3. *The directed graph $D' = (V', A')$ is acyclic and $(D', (S_1, \ldots, S_l), (T_1, \ldots, T_l), k)$ is an instance of the CONSTRAINED MULTICUT problem.*

Proof. As both induced subgraphs $D(V_1)$ and $D(V_2)$ are acyclic, any cycle C in graph D must contain at least one vertex in V_2. Suppose $v_i \in V_2$ is in cycle C, then when we create graph D', replace v_i by s_i and t_i and connect all v_i's incoming arcs to the vertices t_i and connect all v_i's outgoing arcs to the vertex s_i. As there is no arc from t_i to s_i, so the cycle C in D is break by s_i and t_i in D'. Hence, the directed graph D' is acyclic.

In the directed acyclic graph D', from the operations we make s_i and t_i. Every s_i has no incoming arcs and every t_i has no outgoing arcs, therefore all S_i and T_i satisfy the conditions for an instance of the CONSTRAINED MULTICUT problem, i.e. $(D', (S_1, \ldots, S_l), (T_1, \ldots, T_l), k)$ is an instance of the CONSTRAINED MULTICUT problem. □

We say that the instance $I' = (D', (S_1, \ldots, S_l), (T_1, \ldots, T_l), k)$ is *generated* from the instance $I = (D, V_1, V_2, f_{order}, k)$ and for the instance I', we have the following result:

Theorem 2. *Given an instance $I = (D, V_1, V_2, f_{order}, k)$ of the* ORDERED DAG-BIPARTITION FVS *problem, then the instance I has an FVS F of size bounded by k if and only if the instance I', that is generated from I, of the* CONSTRAINED MULTICUT *problem has a separator $S = F$ of size bounded by k.*

Proof. From the operations we use the instance I to generate the instance I', it is obvious that the instance I has a path from v_j to v_i for $j \geq i$ if and only if the instance I' has a path from S_j to T_i for $j \geq i$. Therefore subset $F \subseteq V_1$ breaks all path from v_j to v_i for $j \geq i$ in the instance I if and only if $F \subseteq V_1$ breaks all paths from S_i to T_i in the instance I' for $j \geq i$. Thus the theorem is correct. □

4   Algorithm for the CONSTRAINED MULTICUT problem

Given an instance $(D, (S_1, \ldots, S_l), (T_1, \ldots, T_l), k)$ of the CONSTRAINED MULTICUT problem, let $T_{other} = \bigcup_{i=1}^{l} T_i$. Without loss of generality, we suppose the size of minimum V-cut from S_l to T_{other} is not zero; or if the size of minimum V-cut from S_l to T_{other} is zero, it is obvious that to solve the instance $(D, (S_1, \ldots, S_l), (T_1, \ldots, T_l), k)$, we only need to solve the instance $(D, (S_1, \ldots, S_{l-1}), (T_1, \ldots, T_{l-1}), k)$. Let u be *an outneighbor* of S_l (i.e. u is a neighbor of a vertex in S_l and u is not in S_l) that has no neighbor in T_{other} and is not in T_{other}. As all terminal sets $S_1, \ldots, S_l$ have no incoming arcs, hence u is not in any terminal sets of $S_1, \ldots, S_l, T_1, \ldots, T_l$. Let $S_l' = S_l \cup \{u\}$, first, we give the following lemma:

Lemma 4. *Both $(D, (S_1, \ldots, S_l'), (T_1, \ldots, T_l), k)$ and $(D - u, (S_1, \ldots, S_l), (T_1, \ldots, T_l), k)$ are instances of the* CONSTRAINED MULTICUT *problem.*

Proof. As $(D, (S_1, \ldots, S_l), (T_1, \ldots, T_l), k)$ is an instance of the CONSTRAINED MULTICUT problem, no S_i has any incoming arc for $1 \leq i < l$ and no T_i has any outgoing arc for $1 \leq i \leq l$. First, it is easy to verify that adding u to S_l or deleting u from the graph D will no change the property that no S_i has any incoming arc for $1 \leq i < l$ and no T_i has any outgoing arc for $1 \leq i \leq l$. Second, it is obvious that u is not in any terminal set. Hence, terminal sets $S_1, \ldots, S_l', T_1, \ldots, T_l$ are still pairwise disjoint. Therefore the lemma is correct. □

Lemma 4 can make sure that the operations of adding u to S_l or delete u from the graph D in our algorithm **CMC**(...) will not change the instance of the

CONSTRAINED MULTICUT problem to the instance of other problems. For the instance $(D, (S_1, \ldots, S'_l), (T_1, \ldots, T_l), k)$, we have the following crucial observation for our algorithm. Figure 1 can help you to understand our proof. In the figure, the part under the diagonal dashed line is the part of graph D that its every vertex x can be reached by a path that is from a vertex in S_l to x in D.

Theorem 3. *If the size of the minimum V-cut from S_l to T_{other} is the same with the size of the minimum V-cut from $S'_l = S_l \cup \{u\}$ to T_{other}, where u is a out-neighbor of S_l, then the instance $(D, (S_1, \ldots, S_l), (T_1, \ldots, T_l), k)$ has a separator of size bounded by k if and only if the instance $(D, (S_1, \ldots, S'_l), (T_1, \ldots, T_l), k)$ has a separator of size bounded by k.*

Proof. If the instance $(D, (S'_1, \ldots, S_l), (T_1, \ldots, T_l), k)$ has a separator S of size bounded by k, then it is obvious that S is also a separator of the instance $(D, (S_1, \ldots, S_l), (T_1, \ldots, T_l), k)$.

Now we consider the other direction.

Suppose that S_m is a minimum V-cut from S'_l to T_{other}, then S_m is also a V-cut from S_l to T_{other}. In fact, by the assumption of the theorem, S_m is also a minimum V-cut from S_l to T_{other}. Let $C(S_l)$ be the set of vertices x such that either $x \in S_l$ or there is a path form S_l to x in the induced subgraph $D(V - S_m)$. In particular, $u \in C(S_l)$. Moreover, let $C(T_{other}) = V - C(S_l) - S_m$.

By Lemma 1, there exist $|S_m|$ internally disjoint paths from S_l to T_{other}, each contains exactly one vertex in the set S_m. Therefore, each of these $|S_m|$ paths is cut into two subpaths by a vertex in S_m, such that one subpath is in the induced subgraph $D(C(S_l))$ and the another subpath is in the induced subgraph $D(C(T_{other}))$. From this, we derive that there are $|S_m|$ internally disjoint paths from S_l to S_m in the induced subgraph $D(C(S_l) \cup S_m)$, each contains a distinct vertex in the set S_m.

Let S_k be a separator of the instance $(D, (S_1, \ldots, S_l), (T_1, \ldots, T_l), k)$ of size bounded by k. Define $S'_k = S_k \cap C(S_l)$, $B = S_k \cap S_m$, and $S''_k = S_k \cap C(T_{other})$. Finally, let S'_m be the set of vertices x in S_m such that there is a path from x to T_{other} in the induced subgraph $D(C(T_{other}) \cup S_m - S_k)$ (see Figure 1 for an intuitive illustration of these sets).

We first prove that $|S'_k| \geq |S'_m|$.

From the fact that there are $|S_m|$ internally disjoint paths from S_l to S_m in the induced subgraph $D(C(S_l) \cup S_m)$ in which each path contains a distinct vertex in the set S_m, we derive that there are $|S'_m|$ internally disjoint paths from S_l to S'_m in the induced subgraph $D(C(S_l) \cup S'_m)$. If $|S'_k| < |S'_m|$, then there must be a path P_1 from S_l to a vertex v' in S'_m in the subgraph $D(C(S_l) \cup S'_m - S'_k) = D(C(S_l) \cup S'_m - S_k)$. Moreover, by the definition of the set S'_m, there is also a path P_2 from v' to T_{other} in the induced subgraph $D(C(T_{other}) \cup S_m - S_k)$. The concatenation of the paths P_1 and P_2 would give a path from S_l to T_{other} in the induced subgraph $D(V - S_k)$, which contradicts the assumption that S_k is a separator of the instance $(D, (S_1, \ldots, S_l), (T_1, \ldots, T_l), k)$. Therefore, we must have $|S'_k| \geq |S'_m|$.

We then prove that the set $S_{new} = S'_m \cup B \cup S''_k$ does not include any vertex in the terminal sets of $S_1, \ldots, S'_l, T_1, \ldots, T_l$.
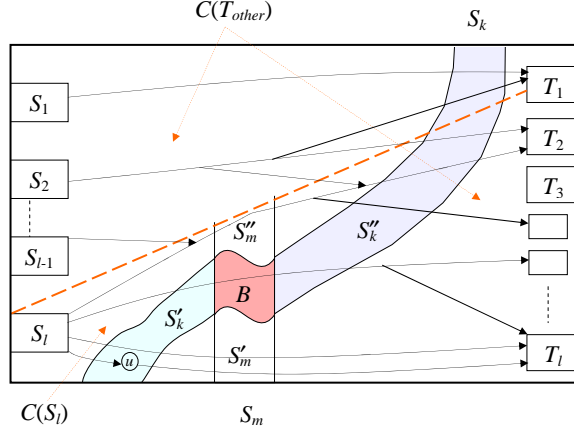
Fig. 1. Decomposition of Separators

As S_k'' is a subset of the separator S_k of the instance $(D, (S_1, \ldots, S_l), (T_1, \ldots, T_l), k)$, by the definition of the separator, S_k'' does not include any vertex from terminal sets $S_1, \ldots, S_l, T_1, \ldots, T_l$. And u is in $C(S_l)$, so $u \notin S_k''$. S_m is a minimum V-cut from S_l', that includes u, to T_{other}; hence $S_l' \cap S_m = \emptyset$ and $T_{other} \cap S_m = \emptyset$. This means S_m's subsets S_m' and B will not include any vertex in S_l and $T_1, \ldots, T_l$. Also as any vertex x in S_m can be reached by a path from a vertex in S_l' to x and all terminal sets $S_1, \ldots, S_{l-1}$ do not have incoming arcs, so S_m' and B will not include any vertex from terminal sets $S_1, \ldots, S_{l-1}$. Therefore, $S_{new} = S_m' \cup B \cup S_k''$ does not include any vertex in the terminal sets of $S_1, \ldots, S_l', T_1, \ldots, T_l$.

We now prove that the set S_{new} breaks all paths from S_l' to T_i for $l \geq i \geq 1$ and all paths from S_j to T_i for $l > j \geq i \geq 1$ in the instance $(D, (S_1, \ldots, S_l'), (T_1, \ldots, T_l), k)$. Suppose S_{new} does not break all paths that need to be break, then there are two vertices v_1 and v_2 that v_1 is in S_l' (i.e. $j = l$) or S_j, v_2 is in T_i, $j \geq i$ and there exists a path P from v_1 to v_2 in the induced subgraph $D(V - S_{new})$. We discuss this in two possible cases.

Case 1: There is a vertex w in the path P such that $w \in C(S_l)$. Because (1) v_2 is in the set T_{other}, (2) there is a path from S_l to w in the induced subgraph $D(C(S_l))$, and (3) S_m is a V-cut from S_l to T_{other}, we conclude that there must be a vertex $s \in S_m$ that is also on the path P. We suppose that the subpath P' of P that is from s to v_2 has no vertices from $C(S_l)$ – for this we only have to pick the last vertex s in S_m when we traverse on the path P from v_1 to v_2. Then the path P' is in the induced subgraph $D(C(T_{other}) \cup S_m - S_{new})$, which is a subgraph of the induced subgraph $D(C(T_{other}) \cup S_m - S_k)$. Now by the definition of the set S_m', the vertex s is in the set S_m', thus in the set S_{new}. But this is impossible because we assumed that the path P is in the induced subgraph $G(V - S_{new})$.

Case 2: All vertices of the path P come from the induced subgraph $D(V - S_{new} - C(S_l))$. Then the vertex v_1 can not be from the set S_l. Moreover, since $D(V - S_{new} - C(S_l))$ is a subgraph of the induced subgraph $D(V - S_k)$, this implies that the path P is from S_j for $j < l$ to T_i for $j \geq i$ and contains no vertex in S_k. But this again contradicts the assumption that S_k is a separator of the instance $(D, (S_1, \ldots, S_l), (T_1, \ldots, T_l), k)$.

Combining the discussions in Case 1, Case 2 and that S_{new} having no vertex in any terminal sets of $S_1, \ldots, S'_l, T_1, \ldots, T_l$, we conclude that the set S_{new} is a separator of the instance $(D, (S_1, \ldots, S'_l), (T_1, \ldots, T_l), k)$.

Since $|S'_k| \geq |S'_m|$, $S_k = S'_k \cup B \cup S''_k$, and $S_{new} = S'_m \cup B \cup S''_k$, and S'_k does not intersect $B \cup S''_k$, we conclude that $|S_k| \geq |S_{new}|$. In particular, if the instance $(D, (S_1, \ldots, S_l), (T_1, \ldots, T_l), k)$ has the separator S_k of size bounded by k, then the instance $(D, (S_1, \ldots, S'_l), (T_1, \ldots, T_l), k)$ has the separator S_{new} of size also bounded by k.

This completes the proof of the theorem. $\qquad\square$

The proof of Theorem 3 becomes complicated partially because the vertex u may be included in a separator for the instance $(D, (S_1, \ldots, S_l), (T_1, \ldots, T_l), k)$. If we restrict that the vertex u is not in the separators for the instance $(D, (S_1, \ldots, S_l), (T_1, \ldots, T_l), k)$, then a result similar to Theorem 3 can be obtained much more easily, even without the need of the condition that the minimum V-cuts from S_l to $T_{other} = \bigcup_{j=1}^{l} T_j$ and the minimum V-cuts from S'_l to T_{other} have the same size. This is given in the following lemma. This result will also be needed in our algorithm.

Lemma 5. *Let S be a vertex subset in graph D such that S does not include the vertex u. Then S is a separator for the instance $(D, (S_1, \ldots, S_l), (T_1, \ldots, T_l), k)$ if and only if S is a separator for the instance $(D, (S_1, \ldots, S'_l), (T_1, \ldots, T_l), k)$.*

Proof. If S is a separator for the instance $(D, (S_1, \ldots, S_l), (T_1, \ldots, T'_l), k)$, then it is obvious that S is also a separator for the instance $(D, (S_1, \ldots, S_l), (T_1, \ldots, T_l), k)$.

For the other direction, suppose that the set S is a separator for the instance $(D, (S_1, \ldots, S_l), (T_1, \ldots, T_l), k)$. We show that S is also a separator for the instance $(D, (S_1, \ldots, S'_l), (T_1, \ldots, T_l), k)$. Suppose that S is not a separator for the instance $(D, (S_1, \ldots, S'_l), (T_1, \ldots, T_l), k)$. Then there is a path P in $G - S$ from S'_l to T_i for $l \geq i \geq 1$ or form S_j to T_i for $l > j \geq i \geq 1$. The path P must contain the vertex u (recall that S does not contain u) – otherwise the path P in $G - S$ would be from a S_j to T_i for $l \geq j \geq i \geq 1$, contradicting the assumption that S is a separator for $(D, (S_1, \ldots, S_l), (T_1, \ldots, T_l), k)$. However, this would imply that the path from S_l to u (recall that u is an non-terminal neighbor of S_l) then following the path P to the terminal set T_i would give a path in $G - S$ from S_l to T_i, again contradicting the assumption that S is a separator for $(D, (S_1, \ldots, S_l), (T_1, \ldots, T_l), k)$. Therefore, S is also a separator for the instance $(D, (S_1, \ldots, S'_l), (T_1, \ldots, T_l), k)$. $\qquad\square$

Now we present our FPT algorithm to solve the CONSTRAINED MULTICUT problem.

Algorithm CMC$(D, (S_1, \ldots, S_l), (T_1, \ldots, T_l), k)$
input: an instance $(D, (S_1, \ldots, S_l), (T_1, \ldots, T_l), k)$ of the
      CONSTRAINED MULTICUT problem
output: a separator of size bounded by k for $(D, (S_1, \ldots, S_l), (T_1, \ldots, T_l), k)$,
      or report "No" (i.e., no such a separator)
1. **if** S_l has a neighbor in $\bigcup_{i=1}^{l} T_i$
    then return "No";
2. **if** S_l's outneighbor w has a neighbor in $\bigcup_{i=1}^{l} T_i$
    then return $w + \textbf{CMC}(D - w, (S_1, \ldots, S_l), (T_1, \ldots, T_l), k - 1)$;
3. find the size m_1 of a minimum V-cut from S_l to $\bigcup_{i=1}^{l} T_i$;
4. **if** $m_1 > k$ **then** return "No";
5. **else if** $(m_1 = 0$ and $l = 1)$ **then** return $\emptyset$;
5.1   **if** $(m_1 = 0$ and $l > 1)$ **then**
        return $\textbf{CMC}(D, (S_1, \ldots, S_{l-1}), (T_1, \ldots, T_{l-1}), k)$;
6. **else** pick an S_l's outneighbor u ; let $S_l' = S_l \cup \{u\}$;
6.1   **if** the size of a minimum V-cut from S_l' to $\bigcup_{i=1}^{l} T_i$ is equal to m_1
        then return $\textbf{CMC}(D, (S_1, \ldots, S_l'), (T_1, \ldots, T_l), k)$;
6.2   **else** $S = u + \textbf{CMC}(D - u, (S_1, \ldots, S_l), (T_1, \ldots, T_l), k - 1)$;
        if S is not "No" **then** return S;
6.3   **else** return $\textbf{CMC}(D, (S_1, \ldots, S_l'), (T_1, \ldots, T_l), k)$.

Fig. 2. Algorithm for the CONSTRAINED MULTICUT problem

Theorem 4. *The algorithm* **CMC**$(D, (S_1, \ldots, S_l), (T_1, \ldots, T_l), k)$ *solves the* CONSTRAINED MULTICUT *problem in time* $O(n^3 k 4^k)$.

Proof. We first prove the correctness of the algorithm. Let $(D, (S_1, \ldots, S_l), (T_1, \ldots, T_l), k))$ be an input to the algorithm, which is an instance of the CONSTRAINED MULTICUT problem, where $D = (V, E)$ is a directed acyclic graph, $(S_1, \ldots, S_l)$, $(T_1, \ldots, T_l)$ are two groups of ordered terminal sets, and k is the upper bound of the size of the separator we are looking for.

If S_l has a neighbor in $\bigcup_{i=1}^{l} T_i$, then we have no way to separate S_l and some T_i since all vertices in a separator are supposed to be non-terminals. Step 1 handles this case correctly.

If w that is a outneighbor of S_l has a neighbor in some T_i, then w must be in the separator because otherwise we can not separate S_l from T_i (remember: w is not in any terminal sets of $S_1, \ldots, S_l, T_1, \ldots, T_l$). Thus, we can simply include the vertex w in the separator, and recursively find a separator of size bounded by $k - 1$ for the same groups of terminal sets $(S_1, \ldots, S_l)$, $(T_1, \ldots, T_l)$ in the remaining graph $D - w$ (by Lemma 4 $(D - w, (S_1, \ldots, S_l), (T_1, \ldots, T_l), k - 1))$ is

also an instance of the CONSTRAINED MULTICUT problem). This case is correctly handled by step 2.[1]

Step 3 computes the size m_1 of a minimum V-cut from the sets S_l to $\bigcup_{j=1}^{l} T_j$. By Lemma 2, m_1 can be computed in time $O((|V| + |A|)k)$. Thus, step 3 takes time $O((|V| + |A|)k)$.

If $m_1 > k$, then the size of a minimum V-cut from S_l to $\bigcup_{j=1}^{l} T_j$ is larger than k, which implies that even separating the set S_l from the other sets $\bigcup_{j=1}^{l} T_j$ requires more than k vertices. Thus, no separator of size bounded by k can exist to separate S_j from T_i for $l \geq j \geq i \geq 1$. This is handled by step 4.

In step 5 we handle the case $m_1 = 0$ and $l = 1$, this means we do not need to remove any vertex to separate S_1 and T_1, i.e. the problem is solved. Hence we just return $\emptyset$ as a separator of size 0. And in step 5.1 $m_1 = 0$ and $l > 1$ means that there is no path from S_l to any T_i for $l \geq i \geq 1$. And as all S_i for $l > i \geq 1$ have no incoming arcs and all T_i for $l \geq i \geq 1$ have no outgoing arcs, we only need to separate any path from S_j to T_i for $l - 1 \geq j \geq i \geq 1$, i.e. we need to solve the instance $(D, (S_1, \ldots, S_{l-1}), (T_1, \ldots, T_{l-1}), k))$ (note that because of step 4, here we must have $k \geq 0$. And step 5.1 repeats at most l times.).

When the algorithm reaches step 6, the following conditions hold true: (1) $k > 0$ and S_l does not have any neighbor in T_i for $l \geq i$ (because of step 1); (2) S_l does not have any no terminal neighbor w that has a neighbor in T_i for $l \geq i$ (because of step 2); (3) $0 < m_1 \leq k$ (because of steps 4-5). In particular, by condition (3), S_l must have a non-terminal neighbor u that has no neighbor in T_i for $l \geq i$.

Let m' be the size of a minimum V-cut from the sets $S_l = S_l \cup \{u\}'$ and $\bigcup_{j=1}^{l} T_j$. If $m' = m_1$, then by Theorem 3, the instance $(D, (S_1, \ldots, S_l), (T_1, \ldots, T_l), k)$ has a separator of size bounded by k if and only if the instance $(D, (S_1, \ldots, S_l'), (T_1, \ldots, T_l), k)$ has a separator of size bounded by k. In particular, as shown in the proof of Theorem 3, a separator of size bounded by k for the instance $(D, (S_1, \ldots, S_l'), (T_1, \ldots, T_l), k)$ is actually also a separator for the instance $(D, (S_1, \ldots, S_l), (T_1, \ldots, T_l), k)$. Therefore, in this case, we can recursively work on the instance $(D, (S_1, \ldots, S_l'), (T_1, \ldots, T_l), k)$, as given in step 6.1 (note that step 6.1 repeats at most n times). On the other hand, if $m' \neq m$ ($m' > m$), then we simply branch on the vertex u in two cases: (1) one case includes u in the separator (we can do it safely, for u is not in any terminal sets of $S_1, \ldots, S_l, T_1, \ldots, T_l$) and recursively works on the remaining graph for a separator of size bounded by $k - 1$, as given by step 6.2; and (2) the other case excludes u from the separator thus looks for a separator that does not include u and is of size bounded by k for the instance $(D, (S_1, \ldots, S_l), (T_1, \ldots, T_l), k)$. By Lemma 5, the second case is equivalent to finding a separator of size bounded by k for the instance $(D, (S_1, \ldots, S_l'), (T_1, \ldots, T_l), k)$. This case is thus handled by step 6.3 (**note:** by Lemma 4, both $(D - u, (S_1, \ldots, S_l), (T_1, \ldots, T_l), k - 1)$

---

[1] To simplify the expression, we suppose that "No" plus any vertex set gives a "No". Therefore, step 2 will return a "No" if $\mathbf{CMC}(D - w, (S_1, \ldots, S_l), (T_1, \ldots, T_l), k - 1)$ returns a "No".

and $(D, (S_1, \ldots, S'_l), (T_1, \ldots, T_l), k)$ are instances of the CONSTRAINED MULTI-CUT problem).

This completes the proof of the correctness of the algorithm. Now we analyze the complexity of the algorithm.

The recursive execution of the algorithm can be described as a search tree $\mathcal{T}$. We first count the number of leaves in the search tree $\mathcal{T}$. Note that only steps 6.2-6.3 of the algorithm correspond to branches in the search tree $\mathcal{T}$. Let $T(k, m)$ be the total number of leaves in the search tree $\mathcal{T}$ for the algorithm $\mathbf{CMC}(D, (S_1, \ldots, S_l), (T_1, \ldots, T_l), k)$, where m is the size of a minimum V-cut from the sets S_l to $\bigcup_{j=1}^{l} T_j$. Then steps 6.2-6.3 induce the following recurrence relation:

$$T(k, m) \leq T(k - 1, m'') + T(k, m') \tag{1}$$

where m'' is the size of a minimum V-cut from S_l to $\bigcup_{j=1}^{l} T_j$ in the graph $D - u$ as given in step 6.2, and m' is the size of a minimum V-cut from S'_l to $\bigcup_{j=1}^{l} T_j$ as given in step 6.3. Note that $m - 1 \leq m'' \leq m$ because removing the vertex u from D cannot increase the size of a minimum V-cut from S_l to $\bigcup_{j=1}^{l} T_j$, and can decrease the size of a minimum V-cut from S_l to $\bigcup_{j=1}^{l} T_j$ by at most 1. Moreover, because the minimum V-cut from S'_l to $\bigcup_{j=1}^{l} T_j$ is not less than the minimum V-cut S_l to $\bigcup_{j=1}^{l} T_j$, and because of step 6.1, the size m' of a minimum V-cut from S'_l to $\bigcup_{j=1}^{l} T_j$ in step 6.3 is at least $m + 1$, i.e. $m' \geq m + 1$. Summarizing these, we have

$$m - 1 \leq m'' \leq m \quad \text{and} \quad m' \geq m + 1 \tag{2}$$

Introduce a new function T' such that $T(k, m) = T'(2k - m)$, and let $t = 2k - m$. Then by Inequalities (1) and (2), the branch in step 6.2-6.3 in the algorithm becomes

$$T'(t) \leq T'(t_1) + T'(t_2)$$

where when $t = 2k - m$ then $t_1 = 2(k - 1) - m'' \leq t - 1$, and $t_2 = 2k - m' \leq t - 1$ (note that in step 2, 5 and 6.1, variable t will not increase). Our initial instance starts with $t = 2k - m \leq 2k$. In the case $t = 2k - m = 0$, because we also have the conditions $k \geq m \geq 0$, we can derive $m = 0$ and $k = 0$, in this case the algorithm can solve the instance without further branching. Therefore, we have $D'(0) = 1$. Combining all these, we derive

$$T(k, m) = T'(2k - m) \leq 2^{2(k+1)} = 4^{k+1},$$

and the search tree $\mathcal{T}$ has at most 4^{k+1} leaves.

Finally, it is easy to verify that along each root-leaf path in the search tree $\mathcal{T}$, the running time of the algorithm is bounded by $O(n^3 k)$, where n is the number of vertices in the graph. In conclusion, the running time of the algorithm $\mathbf{CMC}(D, (S_1, \ldots, S_l), (T_1, \ldots, T_l), k)$ is bounded by $O(n^3 k 4^k)$.

This completes the proof of the theorem.   □

5   Directed FVS problem is FPT

After a long and hard section for the CONSTRAINED MULTICUT problem, we come to an easy and important part of our paper. In this section, we will give the first FPT algorithm for the directed FVS problem. But before we do it, let us give an FPT algorithm for the DAG-BIPARTITION FVS problem.

Theorem 5. *Given an instance* (D, V_1, V_2, k) *of the* DAG-BIPARTITION FVS *problem, let* $|V_2| = l$. *Then in time of* $O(l! n^3 k 4^k)$, *we can either find an FVS* F *of size bounded by* k *for the instance, or report no such an* F *exists.*

Proof. By Theorem 1, the instance (D, V_1, V_2, k) of the DAG-BIPARTITION FVS problem has an FVS F of size bounded by k if and only if there exists at least one instance $(D, V_1, V_2, f_{order}, k)$ of the ORDERED DAG-BIPARTITION FVS problem that has an FVS of size bounded by k. As $|V_2| = l$, we have at most $l!$ possible orders for the elements in V_2. Hence, we need only test at most $l!$ instances of the ORDERED DAG-BIPARTITION FVS problem to know if there is one instance of the ORDERED DAG-BIPARTITION FVS problem that has an FVS of size bounded by k.

By Theorem 2, an instance I of the ORDERED DAG-BIPARTITION FVS problem has an FVS F of size bounded by k if and only if the instance I', that is generated from the instance I, of the CONSTRAINED MULTICUT problem has a separator $S = F$ of size bounded by k. Generating I' from I takes only $O(|A|)$ time, where A is the arc set of the directed graph D and by Theorem 4, finding a separator in I' takes $O(n^3 k 4^k)$ time. Therefore the total time to solve the DAG-BIPARTITION FVS problem is $O(l!(|A| + n^3 k 4^k)) = O(l! n^3 k 4^k)$.   □

Now, we introduce our FPT algorithm for the directed FVS problem.

Theorem 6. *Given a directed graph* $D = (V, A)$ *and an integer* k, *then in time of* $O(n^4 (1.48k)^k)$, *we can either find an FVS* F *of size bounded by* k *in the graph* D, *or report no such an* F *exists.*

Proof. To solve the FVS problem in a directed graph D, we apply the iterative compression technique. First, we use a greedy algorithm to find an FVS F'. If $|F'| \leq k$, then the problem is solved. Else in case of $|F'| > k$, we remove $|F'| - k$ vertices from F' to obtain a new graph D'; then the new graph D' has an FVS of size k. Second, step by step, we add the vertices, that have been removed from the graph, back. In each step, we add one vertex back to D' to obtain a new graph D'', then the new graph $D'' = (V'', A'')$ has an FVS of size $k + 1$. If we can find an FVS of size bounded by k in D'', we continue to add another vertex back and find an FVS of size bounded by k in the new graph, and so on. If we can not find an FVS of size bounded by k in the graph D'' in some step, we can conclude that the graph D does not have an FVS F of size bounded by k. If we add all vertices back and still find an FVS F of size bounded by k, then this F is just the FVS we need.

In a directed graph D'' that has an FVS F'' of size $k + 1$, if D'' has an FVS F of size bounded by k, then there is a subset F_1 such that $F \cap F'' = F_1$

and $0 \le |F_1| = i \le k$. It is obvious that both induced grpahs $D''(V'' - F'')$ and $D''(F'' - F_1)$ are acyclic. We remove F_1 form D'' (we need to try at most $\binom{k+1}{i}$ possible cases, then we can make sure that in one case, F_1 is removed from D''). In the new graph $D'' - F_1 = D''' = (V'', A''')$, if we find an FVS $F_2 \subset (V''' - (F'' - F_1)) = (V'' - F'')$ of size bounded by $j = k - i$, then $F_1 \cup F_2$ is an FVS of size bounded by k in the directed graph D''. In the new directed graph D''', as $D'''(V''' - (F'' - F_1)) = D''(V'' - F'')$ and $D'''(F'' - F_1) = D''(F'' - F_1)$ are acyclic, hence $(D''', V''' - F'', F'' - F_1, j)$ is an instance of the DAG-BIPARTITION FVS problem, where $|F'' - F_1| = k + 1 - i = j + 1$. By Theorem 5, this instance can be solved in $O((j + 1)!n^3 j 4^j)$ time.

From above proof, the total time complexity to solve the FVS problem in directed graph is:

$$O(n \sum_{j=0}^{k} \binom{k+1}{i}(j+1)!n^3 j 4^j) = O(n \sum_{j=0}^{k} \binom{k+1}{j+1}(j+1)!n^3 j 4^j)$$

$$= O(n^4 \sum_{j=0}^{k} \frac{(k+1)!}{(k-j-1)!} j 4^j)$$

$$\le O(n^4 (k+1)!k \sum_{j=0}^{k} 4^j)$$

$$\le O(n^4 (k+1)!k 4^{k+1})$$

$$\le O(n^4 k^{2.5} (1.48k)^k)$$

This completes the proof of the theorem.   □

Remark As there exists a polynomial time approximation algorithm of ratio $O(\log \tau^* \log \log \tau^*)$ for the MINIMUM FEEDBACK VERTEX SET problem in directed graph [11], where τ^* is the size of the minimum feedback vertex set, we can apply this approximation algorithm at first. If the size of FVS F' we found by the approximation algorithm is larger than $O(k \log k \log \log k)$, then we conclude that the instance of the FVS problem has no FVS of size bounded by k. In case of $|F'| \le O(k \log k \log \log k)$, we only remove $O(k \log k \log \log k)$ vertices from F' to make the new graph D' have an FVS of size bounded by k. Hence, we need to add at most $O(k \log k \log \log k)$ vertices (not $O(n)$ vertices as before) back to graph D'. Therefore the algorithm to solve FVS problem in directed graph can be improved to $O(n^3 k^{3.5} \log k \log \log k (1.48k)^k)$.

As the FEEDBACK ARC SET problem in directed graph D has an FAS of size bounded by k if and only if the FEEDBACK VERTEX SET problem in D's line graph has an FVS of size bounded by k [11], therefore we can conclude that the PARAMETERIZED FEEDBACK ARC SET (FAS) problem can also be solved in time of $O(n^3 k^{3.5} \log k \log \log k (1.48k)^k)$.

Theorem 7. *Given a directed graph $D = (V, A)$ and an integer k, then in time of $O(n^3 k^{3.5} \log k \log \log k (1.48k)^k)$, we can either find an FAS F of size bounded by k in the graph D, or report no such an F exists.*

6   Conclusion

In this paper, we solve a long standing open problem in FPT world: if the parameterized FVS problem in directed graph is fixed-parameter tractable (FPT)? Our answer is "Yes", i.e. the parameterized FVS problem in directed graph is fixed-parameter tractable. And we give an FPT algorithm of running time $O(n^4k^{2.5}(1.48k)^k)$ for the parameterized FVS problem in the directed graph.

This is a good news for the FVS problem in directed graph which means that this problem can be solved efficiently when the size of FVS is not very large. We can even design better algorithm for FVS problem in directed graph to solved many practical problems in applications.

The parameterized FVS problem in directed graph is a very important problem and our result is still in a very prime stage. The following are some FVS problems in directed graph that are very interesting and worth of a further study. (1) Given an instance I of the FVS problem in directed graph, can we reduce the instance I into another instance I' of the FVS problem in directed graph in polynomial time, such that I is a "Yes" instance if and only if I' is a "Yes" instance, and input size of I' is bounded by a function of k? (Kernelization) (2) Is the FVS problem in directed graph has an FPT algorithm of running time $O(c^k n^{O(1)})$, where c is a constant? (3) In directed graph that each vertex has a weight of positive real number, let the weight of an FVS F be the weight sum of vertices in F. If the graph has an FVS of size bounded by k, then is there an FPT algorithm that find an FVS of size bounded by k with minimum weight?

References

1. V. Bafna, P. Berman, and T. Fujito. A 2-approximation algorithm for the undirected feedback vertex set problem. *SIAM J. Discrete Math.*, 12(1999), pp.289–297.
2. A. Becker, R. Bar-Yehuda, and D. Geiger. Randomized algorithms for the loop cutset problem. *J. Artificial Intelligence Res.*, 12(2000), pp.219–234.
3. H. Bodlaender. On disjoint cycles. *Int. J. Found. Comput. Sci.*, 5(1994), pp.59–68.
4. G. Chartrand and L. Lesniak. *Graphs & Digraphs*. The Wadsworth & Brooks/Cole Mathematics Series, (Second edition), 1986.
5. J. Chen, F. Fomin, Y. Liu, S. Lu, and Y. Villanger. Improved Algorithms for the Feedback Vertex Set Problems. *WADS* (2007) to appear.
6. F. Dehne, M. Fellows, M. Langston, F. Rosamond, and K. Stevens. An $O(2^{O(k)}n^3)$ fpt algorithm for the undirected feedback vertex set problem. *COCOON*, volume 3595 of *Lecture Notes in Computer Science*, (2005), pp.859–869.
7. M. Dom, J. Guo, F. uffner, R. Niedermeier, and A. Tru. Fixed-Parameter Tractability Results for Feedback Set Problems in Tournaments. *Proc. 6th CIAC-06*, volume 3998 of *Lecture Notes in Computer Science*, (2006), pp.320–331.
8. R. Downey and M. Fellows. Fixed Parameter Tractability and Completeness. *Complexity Theory: Current Research*, Cambridge University Press, (1992), pp.191–225.
9. R. Downey and M. Fellows. *Parameterized Complexity*. Springer-Verlag, New York, 1999.
10. R. Downey, M. Fellows. Fixed-Parameter Tractability and Completeness I: Basic Results *Theoretical Computer Science*, (1995).

11. G. Even, J. Naor, B. Schieber, and M. Sudan. Approximating minimum feedback sets and multicuts in directed graphs. *Algorithmica*, 20(1998), pp.151-174.
12. H. Fernau. A top-down approach to search-trees: Improved algorithmics for 3-hitting set. *Technical Report TR04-073*, Electronic Colloquium on Computational Complexity, 2004.
13. F. Fomin, S. Gaspers, and A. Pyatkin. Finding a minimum feedback vertex set in time $O(1.7548^n)$. *IWPEC*, volume 4169 of *Lecture Notes in Computer Science*, (2006), pp.184–191.
14. G. Gardarin, and S. Spaccapietra. Integrity of Databases: A General Lockout Algorithm with Deadlock Avoidance. *In Modeling in Data Base Management System*, G. M. Nijsssen, ed., North-Holland, Amsterdam, (1976), pp.395–411.
15. M. Garey, D. Johnson. *Computers and Intractability: A guide to the Theory of NP-Completeness.* W.H. Freeman, 1979.
16. J. Guo, J. Gramm, F. Hüffner, R. Niedermeier, and S. Wernicke. Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization. *J. Comput. Syst. Sci.*, 72(2006), pp.1386–1396.
17. J. Guo, J. Gramm, F. Huffner, R. Niedermeier, and S. Wernicke. Improved fixed-parameter algorithms for two feeback set problems. *WADS*, volume 3608 of *Lecture Notes in Computer Science*, (2004), pp.158–168.
18. G. Gutin, and A. Yeoy. Some Parameterized Problems on Digraphs, (Survey)
19. I. Kanj, M. Pelsmajer, and M. Schaefer. Parameterized algorithms for feedback vertex set. *IWPEC*, volume 3162 of *Lecture Notes in Computer Science*, (2004), pp.235–247.
20. R. Karp. Reducibility among combinatorial problems. in R. Miller and J. Thatcher(eds.), *Complexity of Computer Compuattions*, Plenum Press, New York, pp.85-103.
21. V. Raman, S. Saurabh, and C. Subramanian. Faster fixed parameter tractable algorithms for undirected feedback vertex set. *ISAAC 2002*, volume 2518 of *Lecture Notes in Computer Science*, (2002), pp.241–248.
22. V. Raman, S. Saurabh, and C. Subramanian. Faster fixed parameter tractable algorithms for finding feedback vertex sets. *ACM Trans. Algorithms*, 2(2006), pp.403–415.
23. V. Raman and S. Saurabh. Parameterized complexity of directed feedback set problems in tournaments. *WADS*, volume 2748 of *Lecture Notes in Computer Science*, (2003), pp.484–492.
24. V. Raman and S. Saurabh. Parameterized algorithms for feedback set problems and their duals in tournaments. *Theoretical Computer Science,*, 351(2006), pp.446-458.
25. V. Raman, S. Saurabh, and C. Subramanian. Faster Fixed Parameter Tractable Algorithms for Finding Feedback Vertex Sets. *ACM Transactions on Algorithms*, (2006)2, pp.403-415.
26. I. Razgon. Exact computation of maximum induced forest. *SWAT*, volume 4059 of *Lecture Notes in Computer Science*, (2006), pp.160–171.
27. A. Silberschatz and P. Galvin. *Operating System Concepts, 4th edition.* In complexity of Computer Computations, pp. 85-104, Plenum Press, N.Y., 1972.
28. E. Speckenmeyer. On feedback problems in digraphs. *WG* volume 411 of *Lecture Notes in Computer Science*, (1989), pp.218–231.