A Fixed-Parameter Algorithm for the Directed Feedback Vertex Set Problem

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

Barry O'Sullivan and Igor Razgon
Cork Constraint Computation Centre
Computer Science Department
University College Cork, Ireland
{b.osullivan,i.razgon}@cs.ucc.ie

ABSTRACT

The (parameterized) FEEDBACK VERTEX SET problem on directed graphs, which we refer to as the DFVS problem, is defined as follows: given a directed graph G and a parameter k, either construct a feedback vertex set of at most k vertices in G or report that no such set exists. Whether or not the DFVS problem is fixed-parameter tractable has been a well-known open problem in parameterized computation and complexity, i.e., whether the problem can be solved in time $f(k)n^{O(1)}$ for some function f. In this paper we develop new algorithmic techniques that result in an algorithm with running time $4^k k! n^{O(1)}$ for the DFVS problem, thus showing that this problem is fixed-parameter tractable.

Categories and Subject Descriptors

F.2 [**Analysis of Algorithms and Problem Complexity**]: General; G.2 [**Discrete Mathematics**]: Graph Theory.

General Terms

Algorithms.

1. INTRODUCTION

Let G be a directed graph. A *feedback vertex set* (FVS) F for G is a set of vertices in G such that every directed cycle in G contains at least one vertex in F, or equivalently, that the removal of F from the graph G leaves a directed acyclic graph (i.e., a DAG). The (parameterized) FEEDBACK VERTEX SET problem on directed graphs, which we refer to as the DFVS problem, is defined as follows: given a directed graph G and a parameter k, either construct an FVS of at most k vertices for G or report that no such a set exists.

The DFVS problem is a classical NP-complete problem that appeared in the first list of NP-complete problems in Karp's seminal paper [21], and has a variety of applications in areas such as operating systems [30], database systems [14], and circuit testing [23]. In particular, the DFVS problem has played an essential role in the study of *deadlock recovery* in database systems and in operating systems [30, 14]. In such a system, the status of system resource allocations can be represented as a directed graph G (i.e., the *system resource-allocation graph*), and a directed cycle in G represents a deadlock in the system. 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 graph G, so that all directed cycles in G are broken. Equivalently, we need to find a FVS in the graph G. In practice, one may expect and desire that the number of vertices removed from the graph G, which is the number of processes to be aborted in the system, be small. This motivates the study of *parameterized algorithms* for the DFVS problem that find a FVS of k vertices in a directed graph of n vertices and run in time $f(k)n^{O(1)}$ for a fixed function f, thus the algorithms become efficient in practice when the value k is small.

This study has been part of a systematic investigation of the *theory of fixed-parameter tractability* [12], which has received considerable attention in recent years. A problem Q is a *parameterized problem* if each instance of Q contains a specific integral parameter k. A parameterized problem is *fixed-parameter tractable* if it can be solved in time $f(k)n^{O(1)}$ for a function $f(k)$ that is independent of the instance size n. A large number of NP-hard parameterized problems, such as the VERTEX COVER problem [5] and the ML TYPE-CHECKING problem [24], have been shown to be fixed-parameter tractable. On the other hand, strong evidence has been given that another group of well-known parameterized problems, including the INDEPENDENT SET problem and the DOMINATING SET problem, are not fixed-parameter tractable [12]. The study of fixed-parameter tractability of parameterized problems has become increasingly interesting, for both theoretical research and practical computation.

The fixed-parameter tractability of the DFVS problem was posted as an open question in the very first papers on fixed-parameter tractability [9, 10, 11]. After numerous efforts, however, the problem remained open. In the past 15 years, the problem has been constantly and explicitly posted as an open problem in a large number of publications in the literature (see, for example, [7, 8, 9, 10, 11, 16, 17, 18, 20, 26, 27, 28]). The problem has become a well-known and outstanding open problem in parameterized computation and complexity.

In this paper we propose an algorithm solving the DFVS problem in time $O(4^k k! n^{O(1)})$. Thus we show that this problem is fixed-parameter tractable. Intuitively, this algorithm comprises four stages that are outlined below.

STOC'08, May 17–20, 2008, Victoria, British Columbia, Canada.

In the first stage the DFVS problem is transformed into a problem called the DFVS REDUCTION problem where the input is a directed graph G, a parameter k and a FVS F of G of size $k+1$. The task is to find a FVS of G of size smaller than the size of F. In particular, we show that the DFVS problem can be solved by $O(n)$ calls to the algorithm solving the DFVS REDUCTION problem. The transformation we use is known in the parameterized complexity community under the name *iterative compression*. The power of this transformation is that computing a solution of size k *in the presence* of a solution of size $k+1$ frequently reveals very useful structural information that is not easy to observe when the solution is computed with respect to the original problem. In the last three years, the use of iterative compression facilitated the resolution of the status of fixed parameter tractability, and the design of faster parameterized algorithms, for a number of problems. For more information related to the iterative compression method we refer the reader to a book [25] and to a survey article [19].

In the second stage the DFVS REDUCTION problem is transformed into the DISJOINT DFVS REDUCTION problem whose only difference from the former one is that the FVS in the output is required to be *disjoint* from a given FVS F. In particular, the DFVS REDUCTION problem is solved by guessing the intersection I of F and the FVS S in the output and for each of 2^{k+1} possible guesses, solving the DISJOINT DFVS REDUCTION problem with respect to the input $(G-I, F-I, k-|I|)$. Clearly the desired FVS S exists if and only if there is an I such that $S-I$ is returned by the respective application of the DISJOINT DFVS REDUCTION problem.

Solving the DISJOINT DFVS REDUCTION problem, we observe that due to F being a FVS of G, any directed cycle of G can be represented as a union of paths between the vertices of F. In other words, for any FVS S of G disjoint with F, one can specify a set of pairs (u,v) of vertices of F so that $G-S$ has no path from u to v. Thus, we note that the DISJOINT DFVS REDUCTION problem can be solved by iteratively guessing a set of pairs of vertices of V and solving the *directed multicut* problem with respect to this set of pairs. Moreover, we specify a *template* that this set of pairs of vertices has to satisfy. To do that we first observe that in $G-S$, any vertex of $u \in F$ is *separated from itself*, i.e. there is no path from u to u just because this path would contain a directed cycle. Moreover, we notice that there exists $u \in F$ such that for any $v \in F$, $G-S$ has no path from u to v. The way to show this is similar to the way one shows that at least one vertex of a directed acyclic graph has no outgoing edges: if for each vertex of F, graph $G-S$ has an outgoing path to another vertex of F, then going so from a vertex to a vertex one will eventually return to an already visited vertex which means that $G-S$ has a cycle, hence a contradiction. Furthermore, a *repeated* application of this argument shows that there is an ordering $u_1, \ldots u_{k+1}$ of the vertices of F such that $G-S$ has no path from u_i to u_j whenever $i \geq j$. Finally, we show that this *separation template* is sufficient. In particular, we demonstrate that for any ordering $u_1, \ldots, u_{k+1}$ of vertices of F and any S' such that $G-S'$ has no path from u_i to u_j whenever $i \geq j$, S' is a FVS of G. Indeed, by the definition of F, any directed cycle of G is a path from some u_i to itself. By definition of S', such a path does not exist in $G-S'$.

It follows from the argument presented in the previous paragraph that the DISJOINT DFVS REDUCTION problem can be solved by considering all possible $(k+1)!$ orderings of vertices of F and solving the above separation problem for each ordering. If for at least one ordering the required separator S, of size at most k, is found then S is the desired FVS. Otherwise, we can safely conclude that no such FVS exists. After a slight syntactic modification we can formulate the separation problem with respect to the given order of F as the SKEW SEPARATOR problem defined as follows. Given two collections $[S_1, \ldots, S_l]$ and $[T_1, \ldots, T_l]$ of subsets of *terminal vertices* of a directed graph G, decide whether it is possible to remove at most k *non-terminal* vertices so that there is no path from S_i to T_j in the resulting graph whenever $i \geq j$.

Therefore, in the third stage of the algorithm the DISJOINT DFVS REDUCTION problem is transformed into the SKEW SEPARATOR problem. In this context, it is worth noting that the relationship between the DFVS problem and multi-cut problems has been studied in the research of approximation algorithms for the FEEDBACK VERTEX SET problem [13, 22]. The originality of our approach is that we specify the template for the set of pairs of the multicut problem and use a transformation whose runtime exponentially depends on k; the latter seems inappropriate for the design of an approximation algorithm.

In the last stage, the algorithm solves the SKEW SEPARATOR problem using a bounded search tree method [25], which uses the following main branching rule: pick an edge (s_l, u) where $s_l \in S_l$ and u is a non-terminal vertex, on the first branch remove u (i.e include u into the separator), on the second branch join u into S_l, proceed recursively on each of these branches while decreasing the parameter on the branch where u is removed. Thus, it is clear how the search tree is bounded if u is removed but it is non-trivial to see the bounding factor when u is joined to S_l. To handle this, we consider two cases. In the first case, joining u into S_l *increases* the size of the smallest vertex cut from S_l to $[T_1, \ldots, T_l]$, which is clearly a lower bound on the size of the smallest skew separator from $[S_1, \ldots, S_l]$ to $[T_1, \ldots, T_l]$. Thus, in this case, on any branch the *gap between the size of the parameter and the lower bound decreases*, which implies that along each path from the root to a leaf there are at most $2k$ branching nodes of this type. The most important point of the algorithm is that these are *the only branching nodes that occur*! More specifically, we prove that if joining u to S_l does not increase the size of the smallest vertex cut from S_l to $[T_1, \ldots, T_l]$, then it does not increase the size of the smallest skew separator from $[S_1, \ldots, S_l]$ to $[T_1, \ldots, T_l]$. Consequently, in this case there is no need to branch on u and it can be safely joined to S_l. The resulting algorithm for the SKEW SEPARATOR problem takes $O(4^k n^{O(1)})$ time.

The straightforward multiplication of the runtimes for the above four stages results in an upper bound of $O(8^k k! n^{O(1)})$ for the runtime of the algorithm. A slightly more careful analysis results in an upper bound of $O(4^k k! n^{O(1)})$.

Before we move to the technical discussion of our algorithms, we remark that the FEEDBACK VERTEX SET problem on undirected graphs (the UFVS problem) has also been an interesting and active research topic in parameterized computation and complexity. Since the first fixed-parameter tractable algorithm for the UFVS was published almost 20 years ago [2], there has been an impressive list of improved algorithms for the problem. Currently the best algorithm for the UFVS problem runs in time $O(5^k kn^2)$ [4]. The FEED-

BACK VERTEX SET problem on directed graphs (i.e., the DFVS problem) seems very different from the problem on undirected graphs (i.e., the UFVS problem). This fact has also been reflected in the study of approximation algorithms for these problems. The FEEDBACK VERTEX SET problem on undirected graphs is polynomial-time approximable with a ratio 2. This holds true even for weighted graphs [1]. On the other hand, it still remains open whether the FEEDBACK VERTEX SET problem on directed graphs has a constant-ratio polynomial-time approximation algorithm. The current best such algorithm for the problem on directed graphs has a ratio of $O(\log n \log\log n)$ [13].

The remainder of the paper is organized as follows. Section 2 introduces the necessary terminology. In Section 3 an algorithm for the SKEW SEPARATOR problem is presented. Section 4 and 5 present a two-stage transformation from the DFVS problem to the SKEW SEPARATOR problem, and Section 6 describes an extension of the results and outlines some possible future research.

2. PRELIMINARIES

Let $G = (V, E)$ be a directed graph. Without loss of generality, we assume that G contains no self-loops. We say that the edge $e = [u, v]$ *goes out* from the vertex u and *comes into* the vertex v. The edge e is called an *outgoing edge* of the vertex u, and an *incoming edge* of the vertex v. These concepts can be extended from single vertices to general vertex sets. Thus, for two vertex sets S_1 and S_2, we can say that an edge goes out from S_1 and comes into S_2 if the edge goes out from a vertex in S_1 and comes into a vertex in S_2. Moreover, we say that an edge *goes out from* S_1 if the edge goes out from a vertex in S_1 and comes into a vertex not in S_1. An edge *comes into* S_2 if the edge goes out from a vertex not in S_2 and comes into a vertex in S_2.

A *path* P from a vertex v_1 to a vertex v_h in the graph G is a sequence $\{v_1, v_2, \ldots, v_h\}$ of vertices in G such that $[v_i, v_{i+1}]$ is an edge in G for all $1 \leq i \leq h-1$. The path P is *simple* if no vertex is repeated in P. The path P is a *cycle* if $v_1 = v_h$, and the cycle is *simple* if no other vertices are repeated. We say that a path is from a vertex set S_1 to a vertex set S_2 if the path is from a vertex in S_1 to a vertex in S_2. The graph G is a *DAG* (i.e., directed acyclic graph) if it contains no cycles.

For a vertex subset $V' \subseteq V$ in the directed graph $G = (V, E)$, we denote by $G[V']$ the subgraph of G that is induced by the vertex subset V'. Without ambiguity, we will denote by $G - V'$ the induced subgraph $G[V - V']$, and by $G - w$, where w is a vertex in G, the induced subgraph $G[V - \{w\}]$. A vertex subset F in the directed graph G is a *feedback vertex set* (FVS) if the graph $G - F$ is a DAG.

DEFINITION 2.1. *(***Skew Separator***) Let $[S_1, \ldots, S_l]$ and $[T_1, \ldots, T_l]$ be two collections of l vertex subsets in a directed graph $G = (V, E)$. A skew separator X for the pair $([S_1, \ldots, S_l], [T_1, \ldots, T_l])$ is a vertex subset in $V - \bigcup_{i=1}^{l}(S_i \cup T_i)$ such that for any pair of indices i and j satisfying $l \geq i \geq j \geq 1$, there is no path from S_i to T_j in the graph $G - X$.*

The subsets $S_1, \ldots, S_l$ will be called the *source sets* and the subsets $T_1, \ldots, T_l$ will be called the *sink sets*. A vertex is a *non-terminal vertex* if it is not in $\bigcup_{i=1}^{l}(S_i \cup T_i)$. Note that by definition, all vertices in a skew separator must be non-terminal vertices. Moreover, a skew separator X is asymmetric to the source sets and the sink sets: a path from S_i to T_j with $i < j$ may exist in the graph $G - X$.

When there is only one source set S_1 and one sink set T_1, a skew separator for the pair $([S_1], [T_1])$ becomes a regular cut for S_1 and T_1, i.e., a vertex set whose removal leaves a graph in which there is no path from S_1 to T_1. Therefore, a skew separator for the pair $([S_1], [T_1])$ is also called a *cut from S_1 to T_1*. A cut from S_1 to T_1 is a *min-cut* (i.e., a minimum cut) if it has the smallest cardinality over all cuts from S_1 to T_1.

LEMMA 2.2. *Let S and T be two vertex subsets in a directed graph G of n vertices, and let k be a parameter. There is an $O(kn^2)$ time algorithm that, on input (G, S, T, k), either constructs a min-cut of size bounded by k from S to T in the graph G, or reports that the size of a min-cut from S to T in G is larger than k.*

PROOF. The lemma can be proved by first shrinking the sets S and T into two vertices s and t then constructing a min-cut of size k from s to t in the resulting graph. This can be done in time $O(kn^2)$ using the standard maximum flow technique. □

The algorithm for the DFVS problem is obtained through a development of algorithms for a series of problems. In the following, we give the formal definitions of these problems.

SKEW SEPARATOR: Given $(G, [S_1, \ldots, S_l], [T_1, \ldots, T_l], k)$, where G is a directed graph, $S_1, \ldots, S_l$ are l source sets and $T_1, \ldots, T_l$ are l sink sets in G, and a parameter k, such that:
(1) all sets S_1, ..., S_l, T_1, ..., T_l are pairwise disjoint;
(2) for each i, $1 \leq i \leq l-1$, there is no edge coming into the source set S_i; and
(3) for each j, $1 \leq j \leq l$, there is no edge going out from the sink set T_j,
either construct a skew separator of at most k vertices for the pair $([S_1, \ldots, S_l], [T_1, \ldots, T_l])$, or report that no such a separator exists.

Note that in an instance of the SKEW SEPARATOR, condition (2) on sources and condition (3) on sinks are not completely symmetric: the first $l-1$ source sets are not allowed to have incoming edges, but the last source set S_l is allowed to have incoming edges. On the other hand, *all* sink sets are not allowed to have outgoing edges.

We remark that conditions (1)-(3) in the definition of the SKEW SEPARATOR problem, plus the restriction that the skew separator can consist of only non-terminal vertices, may be relaxed and our techniques for the problem may still be applicable. However, the above formulation of the problem will make our discussion simpler, and also will be sufficient for our solution to the DFVS problem, which is the focus of the current paper. We leave the investigation of the separator problems of more general forms to future research.

The following two problems are special versions of the FEEDBACK VERTEX SET problem.

The DFVS REDUCTION problem: Given a triple (G, F, k), where G is a directed graph and F is a FVS of size $k+1$ for G, either construct a FVS of size bounded by k for G, or report that no such FVS exists.

The DISJOINT DFVS REDUCTION problem: Given a triple (G, F, k), where G is a directed graph and F is a FVS of size $k+1$ for G, either construct a FVS of G of size bounded by k and disjoint with F, or report that no such FVS exists.

The central problem in this paper can be stated as follows.

The DFVS problem: Given a pair (G, k), where G is a directed graph and k is the parameter, either construct a FVS of size bounded by k for G, or report that no such a FVS exists.

3. SOLVING SKEW SEPARATOR

Let $(G, [S_1, \ldots, S_l], [T_1, \ldots, T_l], k)$ be an instance of the SKEW SEPARATOR problem. Define $T_{all} = \bigcup_{1 \le i \le l} T_i$. There are some cases in which we can reduce the instance size:

Rule R1. There is no path from S_l to T_{all}: then we only need to find a skew separator of size k that separates S_i from T_j for all indices i and j satisfying $l - 1 \ge i \ge j \ge 1$, i.e., we can work instead on the instance $(G, [S_1, \ldots, S_{l-1}], [T_1, \ldots, T_{l-1}], k)$. Note that in this case, by definition, if $l = 1$, then the solution to the instance $(G, [S_1, \ldots, S_{l-1}], [T_1, \ldots, T_{l-1}], k)$ is simply the empty set $\emptyset$;

Rule R2. There is an edge from S_l to T_{all}: then there is no way to separate S_l from T_{all} – we can simply stop and report that the given instance is a "No" instance;

Rule R3. There exists a non-terminal vertex w, an edge from S_l to w, and an edge from w to T_{all}: then the vertex w must be included in the skew separator in order to separate S_l and T_{all} – we can simply work on the instance $(G - w, [S_1, \ldots, S_l], [T_1, \ldots, T_l], k - 1)$ and recursively find a skew separator of size $k - 1$.

Note that in Rule R1 and Rule R3, the two reduced instances $(G, [S_1, \ldots, S_{l-1}], [T_1, \ldots, T_{l-1}], k)$ and $(G - w, [S_1, \ldots, S_l], [T_1, \ldots, T_l], k - 1)$ are still valid instances of the SKEW SEPARATOR problem.

In the following discussion, assume that for the input instance $(G, [S_1, \ldots, S_l], [T_1, \ldots, T_l], k)$, none of the rules above is applicable. In particular, since Rule R1 is not applicable, a min-cut from S_l to T_{all} has size larger than 0. Because Rules R1-R3 are not applicable, there must be a non-terminal vertex u_0 such that (1) there is an edge from S_l to u_0; and (2) there is no edge from u_0 to T_{all}. Such a vertex u_0 will be called an *S_l-extended vertex*. Fix an S_l-extended vertex u_0, let $S'_l = S_l \cup \{u_0\}$.

LEMMA 3.1. *Let X be a subset of vertices in the graph G that does not contain the S_l-extended vertex u_0. Then X is a skew separator for the pair $([S_1, \ldots, S_l], [T_1, \ldots, T_l])$ if and only if X is a skew separator for the pair $([S_1, \ldots, S_{l-1}, S'_l], [T_1, \ldots, T_{l-1}, T_l])$.*

PROOF. ($\rightarrow$) Suppose that X is a skew separator for the pair $([S_1, \ldots, S_{l-1}, S'_l], [T_1, \ldots, T_{l-1}, T_l])$. We show that X is also a skew separator for $([S_1, \ldots, S_l], [T_1, \ldots, T_l])$. Suppose that there is a (simple) path P from S_i to T_j for some $i \ge j$ in the graph $G - X$. If P does not contain the S_l-extended vertex u_0, then either P is a path from S_i to T_j (when $i < l$), or P is a path from S'_l to T_j (when $i = l$, note $S_l \subseteq S'_l$), either case would contradict the assumption that X is a skew separator for the pair $([S_1, \ldots, S_{l-1}, S'_l], [T_1, \ldots, T_{l-1}, T_l])$. On the other hand, suppose that P contains the S_l-extended vertex u_0. Then the subpath of P from u_0 to T_j would give a path from S'_l to T_j in $G - X$, again contradicting the assumption that X is a skew separator for the pair $([S_1, \ldots, S_{l-1}, S'_l], [T_1, \ldots, T_{l-1}, T_l])$. These contradictions prove that X is also a skew separator for the pair $([S_1, \ldots, S_l], [T_1, \ldots, T_l])$.

($\leftarrow$) Suppose that X is a skew separator for the pair $([S_1, \ldots, S_l], [T_1, \ldots, T_l])$. If, by contradiction, X is not a skew separator for $([S_1, \ldots, S_{l-1}, S'_l], [T_1, \ldots, T_{l-1}, T_l])$, then there is a path P' in $G - X$ that is either from S'_l to T_j for some j, or from S_i to T_j for some $i \ge j$. The path P' must contain the S_l-extended vertex u_0 (recall that the set X does not contain u_0), otherwise the path P' in $G - X$ would be from some S_i to some T_j with $i \ge j$, contradicting the assumption that X is a skew separator for $([S_1, \ldots, S_l], [T_1, \ldots, T_l])$. However, this would imply that the path from S_l directly to u_0 (recall that there is an edge from S_l to u_0), then following the path P' to the set T_j, would give a path in $G - X$ from S_l to T_j, again contradicting the assumption that X is a skew separator for $([S_1, \ldots, S_l], [T_1, \ldots, T_l])$. Therefore, the path P' does not exist in $G - X$, and X must be a skew separator for the pair $([S_1, \ldots, S_{l-1}, S'_l], [T_1, \ldots, T_{l-1}, T_l])$. □

COROLLARY 3.2. *A skew separator for the pair $([S_1, \ldots, S_{l-1}, S'_l], [T_1, \ldots, T_{l-1}, T_l])$ is also a skew separator for the pair $([S_1, \ldots, S_l], [T_1, \ldots, T_l])$.*

COROLLARY 3.3. *The size of a min-cut from S'_l to T_{all} in the graph G is at least as large as the size of a min-cut from S_l to T_{all} in G.*

THEOREM 3.4. *If the size of a min-cut from S_l to T_{all} is equal to the size of a min-cut from S'_l to T_{all}, then the pair $([S_1, \ldots, S_l], [T_1, \ldots, T_l])$ has a skew separator of size bounded by k if and only if the pair $([S_1, \ldots, S_{l-1}, S'_l], [T_1, \ldots, T_{l-1}, T_l])$ has a skew separator of size bounded by k.*

PROOF. ($\rightarrow$) Suppose the pair $([S_1, \ldots, S_{l-1}, S'_l], [T_1, \ldots, T_{l-1}, T_l])$ has a skew separator X' of size bounded by k. By Corollary 3.2, X' is also a skew separator for $([S_1, \ldots, S_l], [T_1, \ldots, T_l])$. Thus, $([S_1, \ldots, S_l], [T_1, \ldots, T_l])$ has a skew separator of size bounded by k.

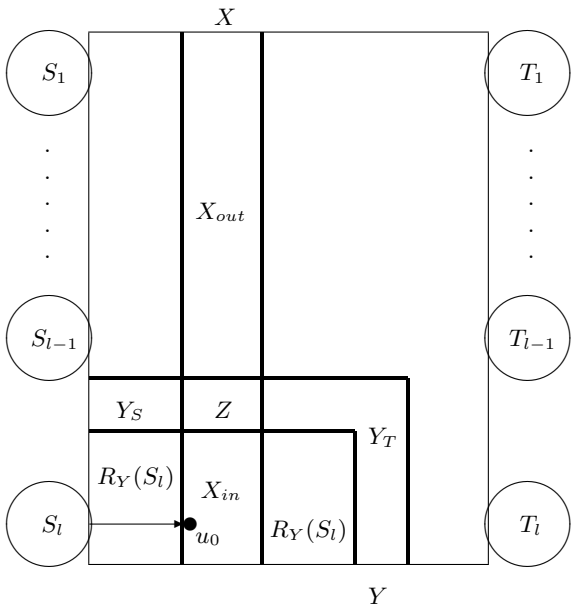
($\leftarrow$) Suppose that the pair $([S_1, \ldots, S_l], [T_1, \ldots, T_l])$ has a skew separator X of size bounded by k. If the skew separator X does not contain the S_l-extended vertex u_0, then by Lemma 3.1, X is also a skew separator of size bounded by k for the pair $([S_1, \ldots, S_{l-1}, S'_l], [T_1, \ldots, T_{l-1}, T_l])$, and the theorem is proved. Therefore, we can assume that the set X contains the S_l-extended vertex u_0.

Let Y be a min-cut from S'_l to T_{all}. Then Y does not contain the S_l-extended vertex u_0. Moreover, since there is no edge coming into S_i from outside of S_i for all $i \le l - 1$, the set Y does not contain any vertex in $\bigcup_{i=1}^{l-1} S_i$. In consequence, the set Y consists of only non-terminal vertices. By Corollary 3.2, Y is also a cut from S_l to T_{all}. Moreover, by the assumption of the theorem that the size of a min-cut from S_l to T_{all} is equal to the size of a min-cut from S'_l to T_{all}, Y is actually also a min-cut from S_l to T_{all}. Let $R_Y(S_l)$ be the set of vertices v such that either $v \in S_l$ or there is a path from S_l to v in the subgraph $G - Y$. In particular, $u_0 \in R_Y(S_l)$ because Y does not contain u_0 and there is an edge from S_l to u_0.

We introduce a number of sets as follows.

$$Z = X \cap Y; X_{in} = X \cap R_Y(S_l); X_{out} = X - (X_{in} \cup Z).$$

That is, the skew separator X for $([S_1, \ldots, S_l], [T_1, \ldots, T_l])$ is decomposed into three disjoint subsets Z, X_{in}, and X_{out}; note that by definition, $R_Y(S_l)$ and Y do not intersect.



Y: a min-cut from S'_l to T_{all}

X: a skew separator for $([S_1, \ldots, S_l], [T_1, \ldots, T_l])$

$R_Y(S_l)$: vertices reachable from S_l in $G - Y$

$Z = X \cap Y$

$X_{in} = X \cap R_Y(S_l)$

$X_{out} = X - (X_{in} \cup Z)$

Y_T: vertices in Y from which T_{all} is reachable in $G - X$

$Y_S = Y - (Y_T \cup Z)$

Figure 1: The sets in the proof of Theorem 3.4

Let Y_T be the set of vertices v in the min-cut Y such that there is a path from v to T_{all} in $G - X$. By definition, $Y_T \cap Z = \emptyset$. Let $Y_S = Y - (Y_T \cup Z)$. Thus, the min-cut Y from S_l to T_{all} is decomposed into three disjoint subsets Z, Y_T, and Y_S. See Figure 1 for an intuitive illustration of all the defined sets.

We first show that the set $Y' = Y_S \cup Z \cup X_{in}$ is also a cut from S_l to T_{all}. If by contradiction Y' is not a cut from S_l to T_{all}, then there is a path P_1 from S_l to T_{all} in the subgraph $G - Y'$. The path P_1 must contain vertices in the set Y since Y is a cut from S_l to T_{all}. Let w be the first vertex on the path P_1 that is in Y when we traverse from S_l to T_{all} along the path P_1. Then w must be in Y_T since Y' contains both Y_S and Z. Now the partial path P'_1 of P_1 from S_l to w, not including w, must be entirely contained in $R_Y(S_l)$; note that the path P_1 does not intersect $Y_S \cup Z$. Moreover, the path P'_1 contains neither vertices in $X_{in} \cup Z$, by the definition of the set Y', nor vertices in X_{out}, since the sets X_{out} and $R_Y(S_l)$ are disjoint. In summary, the subpath P'_1 from S_l to w contains no vertex in the set X. Moreover, by the definition of the set Y_T, and $w \in Y_T$, there is a path P''_1 from w to T_{all} in the subgraph $G - X$. Now the concatenation of the paths P'_1 and P''_1 would result in a path from S_l to T_{all} in the graph $G - X$, contradicting the fact that X is a skew separator for the pair $([S_1, \ldots, S_l], [T_1, \ldots, T_l])$. This contradiction shows that the set Y' must be a cut from S_l to T_{all}.

Since Y is a min-cut from S_l to T_{all}, we have $|Y| \leq |Y'|$. By definition, $Y = Y_S \cup Z \cup Y_T$ and $Y' = Y_S \cup Z \cup X_{in}$. Also note that Y_S, Z, and Y_T are pairwise disjoint, and that Y_S, Z, and X_{in} are also pairwise disjoint. Therefore, we must have $|Y_T| \leq |X_{in}|$.

Consider the set $X' = X_{out} \cup Z \cup Y_T$. The set X' has the following properties: (1) X' consists of only non-terminal vertices (because both X and Y consist of only non-terminal vertices); (2) $|X'| \leq |X|$ (because $|Y_T| \leq |X_{in}|$), so the size of X' is bounded by k; and (3) the set X' does not contain the S_l-extended vertex u_0 (this is because u_0 is in X_{in} and Y does not contain u_0). Therefore, if we can prove that X' is a skew separator for the pair $([S_1, \ldots, S_l], [T_1, \ldots, T_l])$, then by Lemma 3.1, X' is also a skew separator of size bounded by k for the pair $([S_1, \ldots, S_{l-1}, S'_l], [T_1, \ldots, T_{l-1}, T_l])$. This will complete the proof of the theorem.

Therefore, what remains is to prove that the set $X' = X_{out} \cup Z \cup Y_T$ is a skew separator for the pair $([S_1, \ldots, S_l], [T_1, \ldots, T_l])$. Let $R_Y(T_{all})$ be the set of vertices v such that either $v \in T_{all}$, or there is a path from v to T_{all} in the subgraph $G - Y$.

Suppose by contradiction that X' is not a skew separator for $([S_1, \ldots, S_l], [T_1, \ldots, T_l])$. Then there is a path P_2 in the subgraph $G - X'$ from S_i to T_j for some $i \geq j$. The path P_2 has the following properties:

1. The path P_2 contains a vertex in $R_Y(S_l)$: since X is a skew separator for the pair $([S_1, \ldots, S_l], [T_1, \ldots, T_l])$, the path P_2 from S_i to T_j with $i \geq j$ must contain at least one vertex w_1 in $X = X_{in} \cup Z \cup X_{out}$. Now since the path P_2 is in the subgraph $G - X'$, where $X' = X_{out} \cup Z \cup Y_T$, the vertex w_1 must be in X_{in}, which is a subset of $R_Y(S_l)$;

2. The path P_2 contains a vertex in Y_S: by Property 1, P_2 contains a vertex w_1 in $R_Y(S_l)$. From the vertex w_1 to T_{all} along the path P_2, there must be a vertex w_2 in $Y = Y_S \cup Z \cup Y_T$ since Y is a cut from S_l to T_{all} while w_1 is reachable from S_l in the subgraph $G - Y$. Now since $X' = X_{out} \cup Z \cup Y_T$, and the path P_2 is in the subgraph $G - X'$, the vertex w_2 on the path P_2 must be in the set Y_S;

3. The path P_2 ends at a vertex in $R_Y(T_{all})$: this is simply because P_2 is ended in T_{all}. Note that by definition, no vertex in Y_S can be in $R_Y(T_{all})$.

By Properties 2 and 3, the path P_2 contains a vertex not in $R_Y(T_{all})$ and ends at a vertex in $R_Y(T_{all})$. Thus, there must be an internal vertex w in the path such that w is not in $R_Y(T_{all})$ but all vertices after w along the path P_2 (from S_i to T_j) are in $R_Y(T_{all})$. Note that no vertex w' after the vertex w along the path P_2 can be in the set X: w' in X would imply w' in X_{in} (since P_2 is a path in the subgraph $G - X'$), which would imply that there is another vertex after w' that is in Y thus is not in $R_Y(T_{all})$. Moreover, the vertex w must be in the set Y (otherwise, w would be in $R_Y(T_{all})$). Since P_2 is a path in $G - X'$ and $X' = X_{out} \cup Z \cup Y_T$, the vertex w must be in the set Y_S. However, this derives a contradiction: the subpath of P_2 from w to T_{all} shows that the vertex w should belong to the set Y_T (note that all vertices after w on the path are not in X), and the sets Y_S and Y_T are disjoint. This contradiction proves that the set X' must be a skew separator for the pair $([S_1, \ldots, S_l], [T_1, \ldots, T_l])$. Since the size of the set X' is bounded by k and X' does not contain the S_l-extended vertex u_0, by Lemma 3.1, the set X' is also a skew separator for the pair $([S_1, \ldots, S_{l-1}, S'_l], [T_1, \ldots, T_{l-1}, T_l])$, and the size of X' is bounded by k. □

Theorem 3.4 leads to a parameterized algorithm for the SKEW SEPARATOR problem, as given in Figure 2.

Algorithm SMC$(G, [S_1, \ldots, S_l], [T_1, \ldots, T_l], k)$
Input: An instance $(G, [S_1, \ldots, S_l], [T_1, \ldots, T_l], k)$ of the SKEW SEPARATOR problem.
Output: A skew separator of size bounded by k for the pair $([S_1, \ldots, S_l], [T_1, \ldots, T_l])$, or report "No" (i.e., no such a separator exists).

1. **if** $l = 1$ **then** solve the problem in time $O(kn^2)$;
2. **if** Rule R2 applies or $k < 0$ **then** return "No";
3. **if** Rule R1 applies **then**
   return **SMC**$(G, [S_1, \ldots, S_{l-1}], [T_1, \ldots, T_{l-1}], k)$;
4. **if** Rule R3 applies on a vertex w **then**
   return $\{w\} \cup$ **SMC**$(G \setminus w, [S_1, \ldots, S_l], [T_1, \ldots, T_l], k-1)$; §
5. pick an S_l-extended vertex u_0; let $S'_l = S_l \cup \{u_0\}$;
6. let m be the size of a min-cut from S_l to $T_{all} = \bigcup_{i=1}^{l} T_i$;
7. **if** $m > k$ **then** return "No";
8. let m' be the size of a min-cut from S'_l to T_{all};
9. **if** $(m = m')$ **then**
9.1. return **SMC**$(G, [S_1, \ldots, S_{l-1}, S'_l], [T_1, \ldots, T_{l-1}, T_l], k)$;
9.2. **else**
   $X = \{u_0\} \cup$ **SMC**$(G \setminus u_0, [S_1, \ldots, S_l], [T_1, \ldots, T_l], k-1)$;
   if $X \neq$ "No" **then** return X;
9.3. **else** return **SMC**$(G, [S_1, \ldots, S_{l-1}, S'_l], [T_1, \ldots, T_{l-1}, T_l], k)$.

§ We assume that a "No" plus anything gives a "No".

Figure 2: Algorithm for the SKEW SEPARATOR problem

THEOREM 3.5. *The algorithm* **SMC** *solves the* SKEW SEPARATOR *problem in time* $O(4^k kn^3)$*, where* n *is the number of vertices in the input graph* G.

PROOF. We first prove the correctness of the algorithm. Let $(G, [S_1, \ldots, S_l], [T_1, \ldots, T_l], k)$ be an input to the algorithm, which is an instance of the SKEW SEPARATOR problem, where $G = (V, E)$ is a directed graph, $[S_1, \ldots, S_l]$ and $[T_1, \ldots, T_l]$ are the source sets and the sink sets, respectively, and k is the upper bound of the size of the skew separator we are looking for.

If $l = 1$, then the problem becomes the construction of a min-cut of size bounded by k from S_1 to T_1, which can be solved in time $O(kn^2)$ by Lemma 2.2. Steps 2-4 were justified in the discussions of Rules 2, 1, 3, respectively, at the beginning of this section (note that we have also consistently defined that an instance is a "No" instance if the parameter k has a negative value). Therefore, if the algorithm reaches Step 5, then none of the Rules 1-3 is applicable. In particular, since Rule 1 is not applicable and the sets S_l and T_{all} are disjoint, there must be an edge $[v, w]$, where $v \in S_l$ and $w \notin S_l$. Since Rule 2 is not applicable, the vertex w is not in the set T_{all}. The vertex w also cannot be in any source set S_i for $i < l$ because there is no edge coming into S_i from outside of S_i. Therefore, the vertex w is a non-terminal vertex. Finally, since Rule 3 is not applicable, there is no edge from w to T_{all}. Thus, w must be an S_l-extended vertex. This proves that at Step 5, the algorithm can always find an S_l-extended vertex u_0.

In the case $m > k$ in Step 7, i.e., the size m of a min-cut from S_l to T_{all} is larger than the parameter k, then even separating a single source set S_l from the sink sets $T_{all} = \bigcup_{j=1}^{l} T_j$ requires more than k vertices. Thus, no skew separator of size bounded by k can exist to separate S_i from T_j for all $l \geq i \geq j \geq 1$. Step 7 correctly handles this case by returning "No".

In the case $m = m'$ in Step 9, i.e., the size m of a min-cut from S_l to T_{all} is equal to the size m' of a min-cut from S'_l to T_{all}, by Theorem 3.4, the pair $([S_1, \ldots, S_l], [T_1, \ldots, T_l])$ has a skew separator of size bounded by k if and only if the pair $([S_1, \ldots, S_{l-1}, S'_l], [T_1, \ldots, T_{l-1}, T_l])$ has a skew separator of size bounded by k. Moreover, by Corollary 3.2, a skew separator for the pair $([S_1, \ldots, S_{l-1}, S'_l], [T_1, \ldots, T_{l-1}, T_l])$, of size bounded by k, is also a skew separator for the pair $([S_1, \ldots, S_l], [T_1, \ldots, T_l])$. Therefore, in this case we can recursively call **SMC**$(G, [S_1, \ldots, S_{l-1}, S'_l], [T_1, \ldots, T_{l-1}, T_l], k)$, and look instead for a skew separator of size bounded by k for the pair $([S_1, \ldots, S_{l-1}, S'_l], [T_1, \ldots, T_{l-1}, T_l])$, as handled by Step 9.1.

In the case $m \neq m'$, then the algorithm branches into two subcases: Step 9.2 includes the S_l-extended vertex u_0 in the skew separator and recursively looks for a skew separator of size bounded by $k-1$ in the remaining graph $G - u_0$ for the pair $([S_1, \ldots, S_l], [T_1, \ldots, T_l])$; and Step 9.3 excludes the S_l-extended vertex u_0 from the skew separator and recursively looks for a skew separator that does not contain u_0 and is of size bounded by k in the graph G for the pair $([S_1, \ldots, S_l], [T_1, \ldots, T_l])$ (which, by Lemma 3.1, is a skew separator for the pair $([S_1, \ldots, S_{l-1}, S'_l], [T_1, \ldots, T_{l-1}, T_l])$ of size bounded by k).

This completes the verification of the correctness of the algorithm. Now we analyze its complexity.

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 9.2–9.3 of the algorithm correspond to branches in the search tree $\mathcal{T}$. Let $D(k, m)$ be the total number of leaves in the search tree $\mathcal{T}$ for the algorithm **SMC**$(G, [S_1, \ldots, S_l], [T_1, \ldots, T_l], k)$, where m is the size of a min-cut from S_l to T_{all}. Then Steps 9.2–9.3 induce the following recurrence relation:

$$D(k, m) \leq D(k-1, m_1) + D(k, m_2) \tag{1}$$

where m_1 is the size of a min-cut from S_l to T_{all} in the graph $G - u_0$ as given in Step 9.2, and m_2 is the size of a min-cut from S'_l to T_{all} in the graph G as given in Step 9.3. Note that $m - 1 \leq m_1 \leq m$ because removing the vertex u_0 from the graph G cannot increase the size of a min-cut from S_l to T_{all}, and can decrease the size of a min-cut for the two

sets by at most 1. Moreover, by Corollary 3.3, in Step 9.3 we must have $m_2 \geq m+1$. Summarizing these, we have

$$m-1 \leq m_1 \leq m \quad \text{and} \quad m_2 \geq m+1. \tag{2}$$

Introduce a new function D' such that $D'(2k-m) = D(k,m)$, and let $t = 2k-m$. Then by Inequalities (1) and (2), the branch in Steps 9.2–9.3 in the algorithm becomes

$$D'(t) \leq D'(t_1) + D'(t_2) \tag{3}$$

where when $t = 2k-m$ we have

$$t_1 = 2(k-1)-m_1 \leq t-1, \quad \text{and} \quad t_2 = 2k-m_2 \leq t-1. \tag{4}$$

We also point out that certain non-branching steps (i.e., Steps 3, 4, and 9.1) may also change the values of k and m, thus changing the value of $t = 2k-m$. However, none of these steps increases the value of $t = 2k-m$: (i) Step 3 keeps the value of k unchanged and does not decrease the value of m (because in this case the size of a min-cut from S_l to T_{all} is 0 that cannot be larger than the size of a min-cut from S_{l-1} to $\bigcup_{j=1}^{l-1} T_j$); (ii) Step 4 decreases the value of k by 1 and the value of m by at most 1 (because removing a vertex from G can reduce the size of a min-cut from S_l to T_{all} by at most 1), which as a total will decrease the value of $t = 2k-m$ by at least 1; (iii) by the condition assumed, Step 9.1 keeps both the values of k and m unchanged, thus unchanging the value of $t = 2k-m$. In summary, the value of $t = 2k-m$ after a branching step to the next branching step can never be increased.

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 must have $m = 0$ and $k = 0$, in this case the algorithm can solve the instance without further branching. Therefore, we have $D'(0) = 1$. Combining this with Inequalities (3) and (4), we get $D'(k) \leq 2^k$. Therefore,

$$D(k,m) = D'(2k-m) \leq 2^{2k-m} \leq 2^{2k} = 4^k,$$

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

The running time of each execution of the algorithm **SMC**, not counting the time for the recursive calls in the execution, is bounded by $O(kn^2)$, where n is the number of vertices in the input graph. In particular, by Lemma 2.2, Step 1 that looks for a min-cut of size bounded by k from S_1 to T_1, Steps 6-7 that determine if the size m of a min-cut from S_l to T_{all} is bounded by k, and Steps 8-9 that determine if the size of a min-cut from S'_l to T_{all} is equal to m ($m \leq k$ at this point), all have their running time bounded by $O(kn^2)$.

Observe that for each recursive call in an execution of the algorithm **SMC**, either the number of source-sink pairs in the instance is decreased by 1 (Step 3), or the number of non-terminal vertices in the instance is decreased by 1 (Steps 4, 9.1, 9.2, and 9.3). When the number of source-sink pairs is equal to 1, the problem is solved in time $O(kn^2)$ by Step 1, and when the number of non-terminal vertices is equal to 0, one of the Steps 2 and 3 can be applied directly. In conclusion, along each root-leaf path in the search tree $\mathcal{T}$, there are at most $O(n)$ recursive calls to the algorithm **SMC**. Therefore, the running time along each root-leaf path in the search tree $\mathcal{T}$ is bounded by $O(kn^3)$.

Summarizing the above discussions, we conclude that the running time of the algorithm **SMC** is bounded by $O(4^k kn^3)$. This completes the proof of the theorem. □

4. SOLVING DISJOINT DFVS REDUCTION

In this section we prove the fixed-parameter tractability of the DISJOINT DFVS REDUCTION problem. Recall that the problem gets as input a triple (G, F, k) where G is a directed graph, k is a parameter and F is a FVS of G of size $k+1$. The output is a FVS of G disjoint from F and having size at most k or a report that no such FVS exists.

THEOREM 4.1. *The* DISJOINT DFVS REDUCTION *problem can be solved in time $O(4^k k! k^2 n^3)$ where n is the number of vertices of G.*

We will prove the theorem by showing that the DISJOINT DFVS REDUCTION problem can be solved by solving at most $(k+1)!$ instances of the SKEWED SEPARATOR problem. Then the result will immediately follow from Theorem 3.5. Before the proof of Theorem 4.1, we prove a number of auxiliary statements.

LEMMA 4.2. *Let (G, F, k) be an instance of the* DISJOINT DFVS *problem and let S be a FVS of G which is disjoint with F. Let $F' \subseteq F$. Then there is $u \in F'$ such that for any $v \in F'$, $G - S$ has no path from u to v.*

PROOF. Suppose the lemma does not hold, then for any vertex u in F' there is at least one vertex v in F' such that $G-S$ contains a path from u to v. Thus, starting from any vertex u_1 in F', we would be able to find an infinite sequence $u_1, u_2, \ldots$, of vertices in F' such that there is a path from u_i to u_{i+1} in $G-S$ for all $i \geq 1$. Since F' is a finite set, there is at least one vertex that repeats in the sequence, which corresponds to a cycle in the subgraph $G-S$. But this would contradict our assumption that S is a FVS. □

COROLLARY 4.3. *Let (G, F, k) and S be as in Lemma 4.2. Then there is an ordering $u_1, \ldots, u_{k+1}$ of the vertices of F such $G - S$ does not have a path from u_i to u_j whenever $i \geq j$.*

PROOF. According to Lemma 4.2 there is a vertex $u_{k+1} \in F$ such that $G-S$ has no path from u_{k+1} to any $v \in F$. Proceeding inductively, consider i such that $1 \leq i < k+1$ and assume that there are vertices $u_{i+1}, \ldots, u_{k+1}$ such that for any j, $i+1 \leq j \leq k+1$, $G-S$ has no path from u_j to any other vertex of F, in case $j = k+1$ or to any other vertex of $F - \{u_{j+1}, \ldots, u_{k+1}\}$ otherwise. Apply Lemma 4.2 once again and choose a vertex u_i such that $G-S$ has no path from u_i to any other vertex of $F - \{u_{i+1}, \ldots, u_{k+1}\}$. Continuing so until $i = 1$, we establish the desired ordering. □

LEMMA 4.4. *Let (G, F, k) be an instance of the* DISJOINT DFVS REDUCTION *problem and let $u_1, \ldots, u_{k+1}$ be an arbitrary ordering of vertices of F. Let S be a subset of vertices of G disjoint with F and such that $G - S$ has no path from u_i to u_j whenever $i \geq j$. Then S is a FVS of G.*

PROOF. Since F is a FVS of G any cycle of $G-S$ is a path of $G-S$ from u_i to u_i. Such path is impossible by definition of S. □

PROOF. (Theorem 4.1) It follows from the combination of Corollary 4.3 and Lemma 4.4 that the DISJOINT DFVS REDUCTION problem can be solved as follows. Consider all possible orderings of vertices of F. For the given ordering $u_1, \ldots, u_{k+1}$, find out wether there is a set S of at most k vertices not contained in F so that $G-S$ has no path from

u_i to u_j whenever $i \geq j$. If such a set S is found for at least one ordering, it is returned, otherwise the algorithm reports that no FVS with the required properties exists.

Observe that the above separation problem with respect to the given ordering $u_1, \ldots, u_{k+1}$ of vertices of F can be solved as follows. Split each u_i into two vertices s_i and t_i such that s_i is incident to all the outgoing edges of u_i and t_i is incident to all the incoming edges of u_i. Let G' be the resulting graph. Then solve the SKEW SEPARATOR problem with respect to G' and the collections $[S_1, \ldots, S_{k+1}]$ and $[T_1, \ldots, T_{k+1}]$ where $S_i = \{s_i\}$ and $T_i = \{t_i\}$ for $i = 1, \ldots, k+1$. Indeed, let S be a skew separator with respect to this instance. Then S separates $u_1, \ldots, u_{k+1}$ in the sense that $G - S$ has no path from u_i to u_j whenever $i \geq j$ because otherwise such a path can be transformed into a path from s_i to t_j by just renaming the first and last vertices. Similarly, any S such that $G - S$ has no path from u_i to u_j whenever $i \geq j$ is a skew separator with respect to G', $[S_1, \ldots, S_{k+1}]$, and $[T_1, \ldots, T_{k+1}]$ because any path from s_i to t_j, $i \geq j$ can be transformed into a path from u_i to u_j by renaming its first and last vertices.

Thus the DISJOINT DFVS REDUCTION problem can be solved by $O(k!k)$ applications of the algorithm for the SKEWED SEPARATOR problem on a graph with $O(n)$ vertices. Multiplying the runtime of solving the SKEW SEPARATOR problem obtained in Theorem 3.5 into $O(k!k)$ yields the desired runtime for solving the DISJOINT DFVS REDUCTION problem. □

5. SOLVING DFVS

Based on the previous sections, we can present our algorithm for the DFVS problem. We start with a more restricted version of the problem, the DFVS REDUCTION problem, which has been defined in Section 2.

LEMMA 5.1. *The* DFVS REDUCTION *problem on a triple* (G, F, k) *is solvable in time* $O(n^3 4^k k^3 k!)$, *where* n *is the number of vertices in the input graph* G.

PROOF. Assume that we are given a set $I \subseteq F$, $|I| \leq k$ and the requirement is to find a FVS S of G of size at most k such that $F \cap S = I$ or to report that there is no such FVS. Clearly this can be done by solving the instance $(G - I, F - I, k - |I|)$ of the DISJOINT DFVS REDUCTION problem. If the output is a FVS S' of $G - I$ then $S' \cup I$ is the required FVS of G. Otherwise, we may safely conclude that G has no required FVS.

Thus the DFVS REDUCTION problem can be solved by guessing all possible intersections i.e. by considering all $2^{k+1} - 1$ subsets of F of size at most k and for each subset I applying the DISJOINT DFVS REDUCTION problem as shown in the previous paragraph. A straightforward evaluation yields the $O(n^3 8^k k^3 k!)$ upper bound on the runtime of the resulting algorithm. We improve the upper bound through a more careful analysis.

To do this we calculate the runtime spent on processing the sets I of the given size i, $0 \leq i \leq k$. If $i = 0$ then the only possibility is $I = \emptyset$ hence by Theorem 4.1, the runtime spent is $O(4^k k! k^2 n^3)$. For $1 \leq i \leq k$, there are $\binom{k+1}{i}$ possible guesses and the runtime spent for each guess is $O(4^{k-i}(k-i)!(k-i)^2 n^3)$ by Theorem 4.1. Hence the general time spent is proportional to $\binom{k+1}{i} 4^{k-i}(k-i)!(k-i)^2 n^3$ which is at most $\frac{(k+1)!}{(k+1-i)!} 4^{k-i}(k-i)!(k-i)^2 n^3$. By simple transformation, the latter expression can be shown equal to

$$\frac{(k+1)!}{k+1} 4^{k-i}(k-i)^2 n^3 = k! 4^{k-i}(k-i)^2 n^3 \leq 4^k k! k^2 n^3.$$

Since the guessed sets may have at most $(k+1)$ distinct sizes the resulting time of solving the DFVS REDUCTION problem is $O(4^k k! k^3 n^3)$, as required. □

The rest of our process for solving the DFVS problem is to apply the *iterative compression* method. The method was proposed by Reed, Smith, and Vetta [29] and has been used for solving the FEEDBACK VERTEX SET problem on undirected graphs [7, 17]. Here we extend the method and apply it to solve the DFVS problem.

THEOREM 5.2. *The* DFVS *problem is solvable in time* $O(n^4 4^k k^3 k!)$.

PROOF. Let (G, k) be an instance of the DFVS problem, where $G = (V, E)$ is a directed graph with $n = |V|$ vertices, and k is the parameter. Pick any subset V_0 of $k+1$ vertices in G, and let F_0 be any subset of k vertices in V_0. Note that the set F_0 is an FVS of k vertices for the induced subgraph $G_0 = G[V_0]$ since the graph $G_0 - F_0$ consists of a single vertex (note that by our assumption, the graph G contains no self-loops).

Let $V - V_0 = \{v_1, v_2, \ldots, v_{n-k-1}\}$. Let $V_i = V_0 \cup \{v_1, \ldots, v_i\}$, and let $G_i = G[V_i]$ be the subgraph induced by V_i, for $i = 0, 1, \ldots, n-k-1$. Inductively, suppose that for an integer i, $0 \leq i < n-k-1$, we have constructed an FVS F_i of size bounded by k for the induced subgraph G_i (this has been the case for $i = 0$). Without loss of generality, we can assume that the set F_i consists of exactly k vertices – otherwise we simply pick $k - |F_i|$ vertices (arbitrarily) from $G_i - F_i$ and add them to the set F_i. Now consider the set $F'_{i+1} = F_i + v_{i+1}$. Since $G_{i+1} - F'_{i+1} = G_i - F_i$ and F_i is an FVS for G_i, the set F'_{i+1} is an FVS of size $k+1$ for the induced subgraph G_{i+1}. In particular, the triple (G_{i+1}, F'_{i+1}, k) is a valid instance for the DFVS REDUCTION problem.

Apply Lemma 5.1 to the instance (G_{i+1}, F'_{i+1}, k), which either returns an FVS F_{i+1} of size bounded by k for the graph G_{i+1}, or claims that no such an FVS exists. It is easy to see that if the induced subgraph $G_{i+1} = G[V_{i+1}]$ does not have an FVS of size bounded by k, then the original graph G cannot have an FVS of size bounded by k. Therefore, in this case, we can simply stop and conclude that there is no FVS of size bounded by k for the original input graph G. On the other hand, suppose that an FVS F_{i+1} of size bounded by k is constructed for the graph G_{i+1} in the above process, then the induction successfully proceeds from i to $i+1$ with a new pair (G_{i+1}, F_{i+1}).

In conclusion, the above process either stops at some point and correctly reports that the input graph G has no FVS of size bounded by k, or eventually ends with an FVS F_{n-k-1} of size bounded by k for the graph $G_{n-k-1} = G[V_{n-k-1}] = G$.

This process is involved in solving at most $n-k-1$ instances (G_i, F_i, k) of the DFVS REDUCTION problem, for $0 \leq i \leq n-k-2$. By Theorem 5.1, the running time of the process is bounded by $O(n^3 4^k k^3 k!(n-k-1)) = O(n^4 4^k k^3 k!)$, and the process correctly solves the DFVS problem. □

Remark. The running time of the algorithm in Theorem 5.2 can be further improved by taking advantage of

existing approximation algorithms for the FEEDBACK VERTEX SET problem on directed graphs. Even, Naor, Schieber, and Sudan [13] have developed a polynomial time approximation algorithm for the FEEDBACK VERTEX SET problem that for a given directed graph G, produces a FVS F of size bounded by $c \cdot \tau \log \tau \log \log \tau$, where c is a constant and τ is the size of a minimum FVS for the graph G. Therefore, for a given instance (G, k) of the DFVS problem, we can first apply the approximation algorithm in [13] to construct a FVS F for the graph G. If $|F| > c \cdot k \log k \log \log k$, then we know that the graph G has no FVS of size bounded by k. On the other hand, suppose that $|F| \leq c \cdot k \log k \log \log k$. Then we pick a subset F_0 of arbitrary k vertices in F, and let $G_0 = G - (F - F_0)$. The set F_0 is a FVS of size k for the graph G_0. Now we can proceed exactly the same way as we did in the theorem: let $F - F_0 = \{v_1, v_2, \ldots, v_h\}$, where $h \leq c \cdot k \log k \log \log k - k$, and let $V_i = V_0 \cup \{v_1, \ldots, v_i\}$, and $G_i = G[V_i]$, for $i = 0, 1, \ldots, h$. By repeatedly applying the algorithm in Lemma 5.1, we can either stop with certain index i where the induced subgraph G_{i+1} has no FVS of size bounded by k (thus the original input graph G has no FVS of size bounded by k), or eventually construct a FVS F_h of size bounded by k for the graph $G_h = G[V_h] = G$. This process calls for the execution of the algorithm in Lemma 5.1 at most $h = O(k \log k \log \log k)$ times, and each execution takes time $O(n^3 4^k k^3 k!)$. In conclusion, the DFVS problem can be solved in time $O(n^3 4^k k^4 k! \log k \log \log k + t(n))$, where $t(n)$ is a polynomial of n, independent of k, that is the running time of the approximation algorithm given in [13].

6. REMARKS AND FUTURE RESEARCH

We presented a parameterized algorithm with running time $O(n^4 4^k k^3 k!)$ for the DFVS problem, which shows that the problem is fixed-parameter tractable, and resolves an outstanding open problem in parameterized computation and complexity. Before we close the paper, we give a few remarks on our results and for future research.

There is an edge version of the FEEDBACK SET problem, which is called the FEEDBACK ARC SET problem (briefly, the DFAS problem): given a directed graph G and a parameter k, either construct a set of at most k edges in G whose removal leaves a DAG, or report that no such an edge set exists. The DFAS problem is also a well-known NP-complete problem [15]. As shown by Even, Naor, Schieber, and Sudan [13], the DFAS problem and the DFVS problem can be reduced in linear time from one to the other with the same parameter. Therefore, our results also imply an $O(n^4 4^k k^3 k!)$ time algorithm for the DFAS problem.

The techniques developed in this paper for solving the SKEW SEPARATOR problem seem to be powerful and generally useful in the study of a variety of separator problems. For example, it has been used recently in developing improved algorithms for a multi-cut problem on undirected graphs in which a separator is sought to (uniformly) separate a set of given terminals [6]. It will be interesting to identify the conditions for the multi-cut problems under which these techniques, and their variations and generalizations, are applicable. In particular, it will be interesting to see if the techniques are applicable to derive the fixed-parameter tractability of the FEEDBACK VERTEX SET problem on *weighted and directed graphs*. Note that the fixed-parameter tractability of the problem on weighted and *undirected* graphs has been derived recently [4].

It will be interesting to develop new techniques that lead to faster parameterized algorithms for the DFVS problem and other related problems. For example, is it possible that the DFVS problem can be solved in time $O(c^k n^{O(1)})$ for a constant c? Another direction is to look at the kernelization of the DFVS problem, by which we refer to a polynomial-time algorithm that on an instance (G, k) of the DFVS problem, produces a (smaller) instance (G', k') of the problem, such that the size of the graph G' (the kernel) is bounded by a function of k but independent of the size of the original graph G, that $k' \leq k$, and that the graph G has a FVS of size bounded by k if and only if the graph G' has a FVS of size bounded by k'. Since now it is known that the DFVS problem is fixed-parameter tractable, by a general theorem in parameterized complexity theory [12], such a kernelization algorithm exists for the DFVS problem. However, how small can the kernel G' be? Taking into account that the UFVS problem has a kernel of a polynomial size [3], this question becomes especially intriguing.

ACKNOWLEGEMENTS

J. Chen, Y. Liu, and S. Lu were supported in part by the US National Science Foundation under the Grant CCF-0430683. B. O'Sullivan and I. Razgon were supported by Science Foundation Ireland Grant 05/IN/I886.

7. 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] H. Bodlaender, On linear time minor tests and depth-first search, *Proc. 1st Workshop on Algorithms and Data Structures* (WADS'89), (1989), pp. 577–590.

[3] H. Bodlaender, A Cubic Kernel for Feedback Vertex Set, *Proc. STACS 2007, 24th Annual Symposium on Theoretical Aspects of Computer Science, Aachen, Germany, February 22-24, 2007*, 2007,pp. 320–331.

[4] J. Chen, F. Fomin, Y. Liu, S. Lu, and Y. Villanger, Improved algorithms for the feedback vertex set problems, *Proc. 10th Workshop on Algorithms and Data Structures*, (WADS'07), *Lecture Notes in Computer Science 4619*, (2007), pp. 422–433.

[5] J. Chen, I. Kanj, and W. Jia, Vertex cover: further observations and further improvements, *Journal of Algorithms 41*, (2001), pp. 280–301.

[6] J. Chen, Y. Liu, and S. Lu, An improved parameterized algorithm for the minimum node multiway cut problem, *Proc. 10th Workshop on Algorithms and Data Structures*, (WADS'07), *Lecture Notes in Computer Science 4619*, (2007), pp. 495–506.

[7] 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, *Proc. 11th International Computing and Combinatorics Conference* (COCOON'05), *Lecture Notes in Computer Science 3595*, (2005), pp. 859–869.

[8] M. Dom, J. Guo, F. Hüffner, R. Niedermeier, and A. Truss, Fixed-parameter tractability results for feedback set problems in tournaments, *Proc. 6th*

Conference on Algorithms and Complexity (CIAC'06), *Lecture Notes in Computer Science 3998*, (2006), pp. 320–331.

[9] R. Downey and M. Fellows, Fixed parameter intractability, *Proc. 7th Annual Structural Complexity Conference*, (1992), pp. 36–49.

[10] R. Downey and M. Fellows, Fixed-parameter tractability and completeness I: Basic Results, *SIAM J. Comput. 24*, (1995), pp. 873–921.

[11] R. Downey and M. Fellows, Fixed-parameter tractability and completeness II: on completeness for $W[1]$, *Theoret. Computer Sci. 141*, (1995), pp. 109–131.

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

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

[14] G. Gardarin and S. Spaccapietra, Integrity of databases: a general lockout algorithm with deadlock avoidance, in *Modeling in Data Base Management System*, G. Nijsssen, ed., North-Holland, Amsterdam, (1976), pp. 395–411.

[15] M. Garey and 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. Hüffner, R. Niedermeier, and S. Wernicke, Improved fixed-parameter algorithms for two feeback set problems, *Proc. 9th Workshop on Algorithms and Data Structures* (WADS'05), *Lecture Notes in Computer Science 3608*, (2005), pp. 158–168.

[18] G. Gutin and A. Yeo, Some parameterized problems on digraphs, *The Computer Journal*, to appear.

[19] F. Hüffner, R. Niedermeier, S. Wernicke, Techniques for Practical Fixed-Parameter Algorithms, *The Computer Journal, 51(1)*, 2008, pp. 7–25.

[20] I. Kanj, M. Pelsmajer, and M. Schaefer, Parameterized algorithms for feedback vertex set, *Proc. 1st International Workshop on Parameterized and Exact Computation* (IWPEC'04), *Lecture Notes in Computer Science 3162*, (2004), pp. 235–247.

[21] R. Karp Reducibility among combinatorial problems. in *Complexity of Computer Computations*, R. Miller and J. Thatcher, eds., Plenum Press, New York, pp. 85–103.

[22] T. Leighton and S. Rao, An approximation max-flow min-cut theorem for uniform multi-commodity flow problems with applications to approximation algorithms, *Proc. 29th IEEE Symp. on Foundations of Computer Science* (FOCS'88), (1988), pp. 422–431.

[23] C. Leiserson and J. Saxe, Retiming synchronous circuitry, *Algorithmica 6*, (1991), pp. 5–35.

[24] O. Lichtenstein and A. Pnueli, Checking that finite state concurrent programs satisfy their linear specification. *Proc. 12th ACM Symp. Principles of Prog. Languages*, (1985), pp. 97–107.

[25] R. Niedermeier, Invitation to fixed-parameter algorithms, *Oxford Lecture Series in Mathematics and its Applications, volume 31*, 2006.

[26] 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.

[27] V. Raman and S. Saurabh, Parameterized complexity of directed feedback set problems in tournaments, *Proc. 8th Workshop on Algorithms and Data Structures* (WADS'03), *Lecture Notes in Computer Science 2748*, (2003), pp. 484–492.

[28] V. Raman and S. Saurabh, Parameterized algorithms for feedback set problems and their duals in tournaments. *Theoretical Computer Science 351*, (2006), pp. 446–458.

[29] B. Reed, K. Smith, and A. Vetta, Finding odd cycle transversals, *Oper. Res. Lett. 32*, (2004), pp. 299–301.

[30] A. Silberschatz and P. Galvin, *Operating System Concepts*, 4th ed., Addison Wesley, Reading, MA, 1994.