Fixed-Parameter Tractability Results for Feedback Set Problems in Tournaments

Michael Dom, Jiong Guo*, Falk Hüffner*, Rolf Niedermeier, and Anke Truß

Institut für Informatik, Friedrich-Schiller-Universität Jena,
Ernst-Abbe-Platz 2, D-07743 Jena, Germany
{dom, guo, hueffner, niedermr, tanke}@minet.uni-jena.de

Abstract. Complementing recent progress on classical complexity and polynomial-time approximability of feedback set problems in (bipartite) tournaments, we extend and partially improve fixed-parameter tractability results for these problems. We show that Feedback Vertex Set in tournaments is amenable to the novel iterative compression technique. Moreover, we provide data reductions and problem kernels for Feedback Vertex Set and Feedback Arc Set in tournaments, and a depth-bounded search tree for Feedback Arc Set in bipartite tournaments based on a new forbidden subgraph characterization.

1 Introduction

Feedback set problems deal with destroying cycles in graphs using a minimum number of vertex or edge removals [10]. Although feedback set problems usually are NP-hard for undirected as well as for directed graphs, the algorithmic treatment by means of approximation, exact, or parameterized algorithms seems to be significantly easier in the undirected case where more and better results are known. In particular, in the case of directed graphs the research so far mainly focused on a special class of graphs, so-called tournaments, since they appear in applications such as voting systems, rankings, and graph drawing.

A tournament is a directed graph where there is exactly one arc between each pair of vertices. Also due to important applications, feedback set problems in tournaments recently received considerable interest, e.g., [1, 2, 3, 4, 5, 6, 16, 20]. For instance, the NP-hardness of Feedback Arc Set in tournaments has recently been addressed by at least four independent groups of researchers [1, 2, 5, 6]. Here, we contribute new results concerning the algorithmic tractability of Feedback Arc Set (FAS) and Feedback Vertex Set (FVS) in tournaments and bipartite tournaments.

Table 1 surveys known and new complexity results for feedback set problems in (bipartite) tournaments. Concerning polynomial-time approximability, the following results are known. For FVS in tournaments (FVST), the trivial factor 3 has been improved to 2.5 [3] whereas for FVS in bipartite tournaments

* Supported by the Deutsche Forschungsgemeinschaft, Emmy Noether research group PIAF (fixed-parameter algorithms), NI 369/4.

Table 1. Complexity results for feedback set problems in tournaments. Herein, n denotes the number of vertices and k denotes the size of the desired feedback solution set.

			Approximation		Fixed-parameter tractability		
	Complexity		factor	runtime	runtime		kernel
FVST	NP-c	[18]	2.5 [3]	$O(n^3)$	$O(2^k \cdot n^2(\log n + k))$	[§3]	$O(k^3)$ [§4.1]
FVSBT	NP-c	[4]	3.5 [4]	$O(n^3)$	$O(3.12^k \cdot n^4)$	[19]	?
FAST	NP-c [2, 5, 6]		2 [20]	—	$O(2.42^k \cdot n^{2.38})$	[16]	$O(k^2)$ [§4.2]
FASBT	?		?	?	$O(3.38^k \cdot n^6)$	[§5]	?

(FVSBT) the trivial factor 4 has been improved to 3.5 [4]. For FAS in tournaments (FAST) a factor-2 approximation is known [20] whereas we are not aware of any approximation results for FAS in bipartite tournaments (FASBT).

Alternatively, it is reasonable to study feedback set problems from a parameterized point of view [8, 14]. For instance, in undirected graphs, there has been recent progress showing that a feedback vertex set of size at most k can be found in $c^k \cdot n^{O(1)}$ time for some constant c [7, 13], where n is the number of graph vertices. The corresponding question for directed graphs is famously open. Restricting the consideration to the class of tournaments, Raman and Saurabh [16] have given the first positive result by giving fixed-parameter algorithms for weighted FVST and weighted FAST running in $O(2.42^k \cdot n^{O(1)})$ time. For the unweighted case of FVST, the previously fastest algorithm is obtained by a reduction to 3-Hitting Set and runs in $O(2.18^k \cdot n^{O(1)})$ time [9]. The algorithm for FVSBT with a running time of $O(3.12^k \cdot n^4)$ is derived in a similar way [19].

We improve the time bound of exactly solving unweighted FVST to $O(2^k \cdot n^{O(1)})$, demonstrating the applicability of an elegant technique—so-called iterative compression—in contrast to the more standard depth-bounded search tree methodology employed by Raman and Saurabh [16] and Fernau [9]. Moreover, we present a data reduction providing a size-$O(k^3)$ problem kernel for FVST. As we show, this is only one instance of a problem kernel for a larger class of vertex deletion problems. Furthermore, complementing the $O(2.42^k \cdot n^{O(1)})$-time fixed-parameter algorithm for FAST, we develop an $O(3.38^k \cdot n^{O(1)})$-time algorithm for FASBT which is based on a novel characterization by forbidden subgraphs. Finally, we also demonstrate a size-$O(k^2)$ problem kernel for FAST, complementing the search tree result of Raman and Saurabh [16]. Table 1 summarizes all results.

We feel that an important contribution of this paper—besides improving known upper bounds—is to show the applicability of innovative and practically relevant techniques such as data reduction and iterative compression to feedback set problems in tournaments. In particular, to the best of our knowledge, here we demonstrate for the first time the applicability of iterative compression to directed feedback set problems—previous applications only addressed the undirected case [7, 13, 17].

2 Preliminaries

In this paper we deal with fixed-parameter algorithms that emerge from the field of parameterized complexity analysis [8, 11, 14]. An instance of a parameterized problem consists of a problem instance I and a parameter k. A parameterized problem is *fixed-parameter tractable* if it can be solved in $f(k) \cdot |I|^{O(1)}$ time, where f is a computable function solely depending on the parameter k, not on the input size $|I|$.

A directed graph or digraph D consists of a vertex set V and an arc set E with $n := |V|$ and $m := |E|$. Each arc is an ordered pair of vertices. We consider only digraphs without loops, that is, $(v, v) \notin E$ for all $v \in V$. We call a digraph $D' = (V', E')$ an *induced subgraph* of $D = (V, E)$ if $V' \subseteq V$ and $E' = \{(u, v) \mid u, v \in V' \text{ and } (u, v) \in E\}$. The subgraph of D induced by a vertex subset V' is denoted by $D[V']$. With *reversing* an arc (u, v) we mean that we delete the arc (u, v) from E and insert (v, u) into E. A *tournament* $T = (V, E)$ is a digraph where there is exactly one arc between each pair of vertices. A digraph is a *bipartite tournament* if its vertex set is the union of two disjoint sets V_1 and V_2 such that each arc consists of one vertex from each of V_1 and V_2 and between each vertex from V_1 and each vertex from V_2 there is exactly one arc. A *cycle* is a sequence of distinct vertices $v_1, \ldots, v_s$ with $(v_i, v_{i+1}) \in E$ for all $1 \leq i < s$ and $(v_s, v_1) \in E$. A *triangle* is a cycle of length 3. A *topological sort* of a digraph $D = (V, E)$ is a sequence $v_1, v_2, \ldots, v_n$ of the vertices in V in which each vertex appears exactly once and $i < j$ for each arc $(v_i, v_j) \in E$. Clearly, a digraph has a topological sort iff it is acyclic, that is, it does not contain a cycle.

The Feedback Vertex (Arc) Set in tournaments (FV(A)ST) problem is defined as follows:

Input: A tournament T and a nonnegative integer k.
Task: Find a set F of at most k vertices (arcs) whose removal results in an acyclic digraph.

The set F is called a *feedback vertex (arc) set*. When the input digraph is restricted to bipartite tournaments, we have the Feedback Vertex (Arc) Set in bipartite tournaments (FV(A)SBT) problem.

The following property with respect to acyclicity of tournaments is well-known.

Lemma 1. *A tournament is acyclic iff it contains no triangles.*

For the purpose of showing a problem kernel for FVST in Sect. 4.1, we reduce FVST to the 3-Hitting Set (3HS) problem defined as follows:

Input: A finite set S, a collection C of size-3 subsets of S, and a nonnegative integer k.
Task: Find a subset S' of S with $|S'| \leq k$ such that S' contains at least one element from each subset in C.

Due to the following lemma shown by Raman and Saurabh [16], we can reverse arcs instead of deleting them when dealing with FAST and FASBT. This is useful because it allows us to apply feedback arc sets without leaving the class of (bipartite) tournaments.

Lemma 2. *Let F be a minimal feedback arc set of a digraph D. Then the graph formed from D by reversing the arcs in F is acyclic.*

3 Iterative Compression for Feedback Vertex Set in Tournaments

In this section we present a fixed-parameter algorithm solving FEEDBACK VERTEX SET in tournaments in $O(2^k \cdot n^2(\log n + k))$ time. This algorithm is based on the concept of *iterative compression*, which was introduced by Reed et al. [17]. The heart of our algorithm is a *compression routine*, which computes from a tournament and a feedback vertex set of size $k+1$ a new feedback vertex set of size k, or proves that no smaller feedback vertex set exists.

Using such a compression routine, FEEDBACK VERTEX SET for a tournament T can be solved by successively considering induced subgraphs of T with increasing sizes. Let $\{v_1, \ldots, v_n\}$ be the vertex set V of T. Then the induced subgraphs $T_i := T[\{v_1, \ldots, v_i\}]$ are considered iteratively for $i = 1$ to $i = n$. The optimal feedback vertex set X_1 for the tournament T_1 is empty. For $i > 1$, assume that an optimal feedback vertex set X_{i-1} for T_{i-1} is known. Obviously, $X_{i-1} \cup \{v_i\}$ is a feedback vertex set for T_i. Using the compression routine, we can either determine that $X_{i-1} \cup \{v_i\}$ is optimal, or otherwise compute an optimal feedback vertex set for T_i. For $i = n$, we thus have computed an optimal feedback vertex set for T. It remains to describe the compression routine.

Compression Routine. To make the task of looking for a smaller feedback vertex set for a tournament $T = (V, E)$ easier, we would like to restrict our search to feedback vertex sets that are disjoint from a given one. This can be achieved by a brute-force enumeration of all $O(2^k)$ possibilities to partition the given feedback vertex set X into two vertex sets S and $X \setminus S$. For each partition, we then look only for solutions that contain all of $X \setminus S$ (they can immediately be deleted from the tournament), but none of S.

Up to this point, the algorithm is analogous to the iterative compression algorithm for undirected FEEDBACK VERTEX SET [7, 13]. The core part of the compression routine, however, is completely different; in particular, we will be able to solve the remaining task of finding a smaller feedback vertex set that is disjoint from the given one S in polynomial time, whereas in [7, 13] still exponential time is required.

The central observation is that both $T[S]$ and $T[V \setminus S]$ are acyclic ($T[S]$ because otherwise there is no feedback vertex set without vertices from S, and $T[V \setminus S]$ because S is a feedback vertex set). Then, the topological sort of a maximum acyclic subtournament of T containing all of S can be thought of as resulting from inserting a subset of $V \setminus S$ into the topological sort of S. On the one hand, the order of the inserted subset must not violate the topological sort of $T[V \setminus S]$. On the other hand, we can achieve by a data reduction rule that for every $v \in V \setminus S$, the subtournament $T[S \cup \{v\}]$ is acyclic

```
Input:  Tournament $T = (V, E)$ and a feedback vertex set $S$ for $T$.
Output: A minimum feedback vertex set $F$ for $T$ with $F \cap S = \emptyset$.
1   if $T[S]$ contains a cycle: return nil
2   $s_1, \ldots, s_{|S|} \leftarrow$ topological sort of $T[S]$
3   $R \leftarrow \emptyset$
4   while there is a triangle $u, v, w$ with $u, v \in S$ and $w \in V \setminus S$:
5       $R \leftarrow R \cup \{w\}$
6       $T \leftarrow T$ with $w$ deleted
7   for each $v \in V \setminus S$:
8       $p[v] \leftarrow \min(\{i \mid (v, s_i) \in E\} \cup \{|S| + 1\})$
9   $L \leftarrow$ topological sort of $T[V \setminus S]$
10  $P \leftarrow V \setminus S$ sorted by $p$, with position in $L$ as tie-breaker
11  $Y \leftarrow$ vertices in a longest common subsequence of $L$ and $P$
12  return $R \cup ((V \setminus S) \setminus Y)$
```

Fig. 1. A subroutine for the compression step

and therefore v has a “natural” position within the topological sort of S. We then obtain the maximum acyclic subtournament as the longest common subsequence of the topological sort of $T[V \setminus S]$ and $V \setminus S$ sorted by natural position within S.

We describe this in more detail using the subroutine displayed in Fig. 1. First we check whether S induces a cycle in T: if so, no feedback vertex set for T disjoint from S can be found, and we abort (line 1). Then we apply data reduction to the instance: whenever there is a triangle with two vertices in S, we can only get rid of this triangle by deleting the third vertex (lines 4–6). After applying this reduction rule exhaustively, for any $v \in V \setminus S$ the subtournament $T[S \cup \{v\}]$ clearly does not contain triangles anymore and therefore is acyclic by Lemma 1. This means that we can insert v at some point in the topological sort $s_1, \ldots, s_{|S|}$ of S without introducing cycles. Since T is a tournament, there is thus some integer $p[v]$ such that for $i < p[v]$, there is an arc from s_i to v, and for $i \geq p[v]$, there is an arc from v to s_i (Fig. 2):

$$(v, s_i) \in E \iff i \geq p[v]. \tag{1}$$

We calculate p in lines 7–8: when we encounter the first s_i in the topological sort of S where $(v, s_i) \in E$, we can insert v before s_i; if there is no such s_i, we set $p[v]$ to $|S| + 1$, and (1) still holds.

We now construct a sequence P from p (line 10), where vertices from $V \setminus S$ that are positioned by p between the same two vertices of S are ordered according to their relative position in the topological sort of $T[V \setminus S]$. Clearly, any acyclic subtournament of T containing all of S must have a topological sort where the vertices from $V \setminus S$ occur in the same order as in P. The same holds for the topological sort L of $T[V \setminus S]$, which is calculated in line 9. This leads to the following lemma.

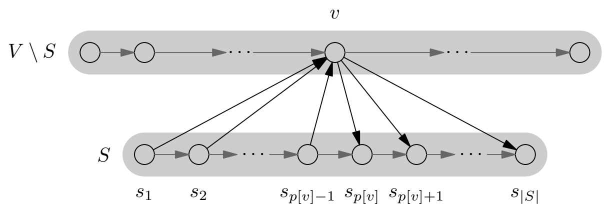

Fig. 2. Illustration of equivalence (1). For clarity, only some of the arcs are shown.

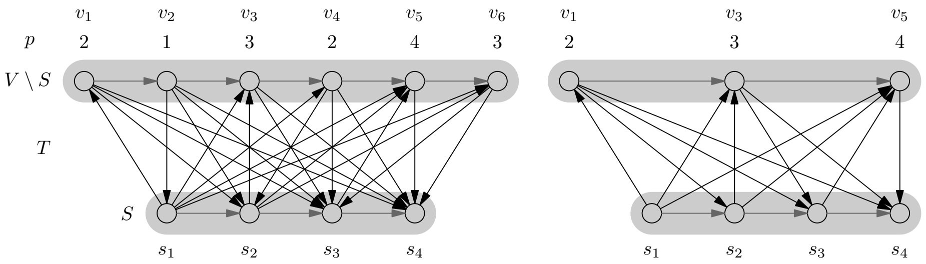

Fig. 3. Example for the subroutine in Fig 1. For clarity, only some of the arcs within the acyclic subtournaments $T[S]$ and $T[V \setminus S]$ are shown. Left: Tournament T after data reduction with $L = v_1, v_2, v_3, v_4, v_5, v_6$ and $P = v_2, v_1, v_4, v_3, v_6, v_5$. A longest common subsequence is v_1, v_3, v_5, yielding the acyclic graph shown on the right.

Lemma 3. *After line 10 of the algorithm in Fig. 1, T is acyclic iff the sequences L and P are equal.*

Proof. "$\Rightarrow$": If L and P are not equal, then there are $v, w \in V \setminus S$ with $(v, w) \in E$ but $p[v] > p[w]$. Then by (1) we have $(w, s_{p[w]}) \in E$ and $(v, s_{p[w]}) \notin E \Rightarrow (s_{p[w]}, v) \in E$, and T is not acyclic.

"$\Leftarrow$": By Lemma 1, it suffices to look for triangles to decide whether T is acyclic. Since $T[S]$ and $T[V \setminus S]$ are acyclic and we destroyed all triangles with two vertices in S, there can only be triangles with exactly two vertices in $V \setminus S$. If L and P are equal, then for all $v, w \in V \setminus S$ with $(v, w) \in E$ we have $p[v] \leq p[w]$. Then by (1) there cannot be any s_i with $(w, s_i) \in E$ and $(s_i, v) \in E$, and there can be no triangle in T. $\square$

With the same justification, Lemma 3 holds for induced subgraphs of T and the corresponding sequences L and P. Clearly, deleting a vertex $v \in V \setminus S$ from T affects L and P only insofar as v disappears from L and P. Therefore, the cheapest way to make T acyclic by vertex deletions can be found by finding the cheapest way to make L and P equal by vertex deletions; this is exactly the complement of the longest common subsequence of L and P. We then obtain the desired feedback vertex set for T by adding the vertices of this complement to those of R, which were determined to be in any feedback vertex set in the reduction step (lines 11–12). Figure 3 shows an example for the execution of the subroutine from Fig. 1.

In summary, the subroutine from Fig. 1 is correct and can be used to solve Feedback Vertex Set in tournaments by iterative compression as described at the beginning of this section.

Theorem 1. Feedback Vertex Set *in tournaments of* n *vertices with* k *vertex deletions can be solved in* $O(2^k \cdot n^2(\log n + k))$ *time.*

Proof. We have shown how to solve Feedback Vertex Set in tournaments using iterative compression. It remains to analyze the runtime. First we examine the subroutine from Fig. 1. Lines 1–2 can be easily done in $O(|S|) = O(k)$ time. Finding triangles in line 4 can be done in $O(nk)$ time: for every $v \in V \setminus S$, we iterate over the topological sort of S; if we encounter a vertex s_i with $(v, s_i) \in E$ and later a vertex s_j with $(s_j, v) \in E$, we have a triangle as desired. Line 9 can be done in $O(n)$ time and line 10 in $O(n \log n)$ time. Since L and P are permutations of each other, finding a longest common subsequence reduces to finding a longest increasing subsequence, which can be done in $O(n \log n)$ time [12]. In summary, the subroutine can be executed in $O(n(\log n + k))$ time. In the compression routine, the subroutine is called $O(2^k)$ times, once for each partition of X into two subsets. The compression routine itself is called n times when inductively building up the graph structure. In total, we have a runtime of $O(2^k \cdot n^2(\log n + k))$. □

4 Problem Kernels by Data Reduction

Developing good kernelizations is among the most important contributions of fixed-parameter algorithmics for hard problems [8, 14]. A *data reduction rule* replaces, in polynomial time, a given problem instance (I, k) by a "simpler" instance (I', k') such that (I, k) is a yes-instance iff (I', k') is a yes-instance. An instance to which none of a given set of reduction rules applies is called *reduced* with respect to these rules. A parameterized problem is said to have a *problem kernel* if, after the application of the reduction rules, the resulting reduced instance has size $f(k)$ for a function f depending only on k.

4.1 Feedback Vertex Set in Tournaments

With Lemma 1, it is easy to observe that Feedback Vertex Set in tournaments (FVST) is a special case of 3-Hitting Set (3HS). Based on the kernelization method for 3HS [15], we show that FVST admits a kernel.

Theorem 2. Feedback Vertex Set *in tournaments admits a problem kernel with an* $O(k^3)$*-vertex tournament, and it can be found in* $O(n^3)$ *time.*

Proof. The basic idea of the kernelization process is to do a trivial transformation from a given FVST instance to a 3HS instance and to perform the known kernelization process [15] on this constructed 3HS instance. The kernel of the FVST instance is then constructed from the reduced 3HS instance—this is the core contribution. In the following, we first describe these three steps and give an estimation of the runtime. Then, we prove the size bound of the kernel and the correctness of the kernelization process.

The transformation from a given FVST instance $(T = (V, E), k)$ to a 3HS instance (S, C, k) with $S := V$ is easy: By Lemma 1, it suffices to enumerate all triangles in T and, for each triangle, add its three vertices as a three-element subset into the subset collection C. This transformation can be done in $O(n^3)$ time. Note that $|C| \leq n^3$.

Then, we apply the data reduction rules for 3HS given in [15] to the generated 3HS instance. Herein, the second rule removes some elements from S, which have to be contained in every size-k solution of the 3HS instance. We use a set H to store these elements; H is initialized as an empty set.

Rule 1. If there is a pair of elements x and y appearing together in more than k three-element subsets, then delete all these subsets from C and add a two-element subset $\{x, y\}$ to C.

Rule 2. If there is an element x appearing in more than k^2 three-element subsets or in more than k two-element subsets, then delete all subsets containing x from C, add x to H, and decrease the parameter k by one.

A 3HS instance can be transformed in $O(\max\{|S|, |C|\}) = O(n^3)$ time into a reduced instance [15].

Finally, from the reduced 3HS instance (S', C', k'), we construct an FVST instance (T', k') with $k' = k - |H|$. First, we replace the two-element subsets in C' by some three-element subsets. Note that any two-element subset $\{x, y\}$ was added to C' by an application of Rule 1; this application did remove a set A of three-element subsets from C with $|A| > k$. We partially "reverse" this application, that is, we delete $\{x, y\}$ from C', choose exactly $k' + 1$ three-element subsets from A and add them to C'. We choose the $k' + 1$ subsets such that they do not contain any element from H; because $k = k' + |H|$, this is always possible. After replacing all two-element subsets in C', we define S'' as the set containing all elements of S appearing in at least one subset in C'. Then the tournament $T' = (V', E')$ is constructed by setting $T' := T[S'']$. Due to Rule 2, the subset collection of the reduced 3HS instance contains $O(k^2)$ two-element subsets; otherwise, there is no solution. We can construct T' from C' in $O(k^3)$ time.

Summarizing the runtimes of the three steps, the runtime of the kernelization process for FVST is $O(n^3)$.

In the construction of T', we add for each two-element subset exactly $k' + 1$ three-element subsets. There are at most $(k')^2$ two-element subsets in the subset collection of the reduced 3HS instance. Together with the problem kernel of 3HS shown in [15] with $|C'| = O(k^3)$, we have $O((k')^3)$ elements in S''. Therefore, $|V'| = O(k^3)$.

It remains to show the correctness of the kernelization process: tournament T has a feedback vertex set of size at most k iff T' has a feedback vertex set of size at most k'.

Given a feedback vertex set F for T with $|F| \leq k$, $F' := V' \cap F$ is a feedback vertex set for T': with the transformation from the FVST instance to the 3HS instance and the kernelization process for 3HS, the elements in H generated by Rule 2 correspond to vertices v in T that are in more than k triangles that, except for v, are vertex-disjoint. Thus, the vertices corresponding to the elements in H are clearly in every feedback vertex set of T, and $H \subseteq F$. Moreover, since T' is an induced subgraph of T, F' is a feedback vertex set of T'. From $H \cap V' = \emptyset$, we have $|F'| \leq |F| - |H| = k'$, that is, F' is a feedback vertex set of T' with at most k' vertices.

Given a feedback vertex set F' of T' with at most k' vertices, $F' \cup H$ is a feedback vertex set of T: Every triangle in T corresponds to a three-element subset in C. If such a three-element subset contains no element from H, then either it is not changed during the kernelization process or it is removed since it contains two elements x and y which appear together in more than k three-element subsets in C. For the former case we have a triangle in T' due to the construction of T' and, thus, at least one vertex of this triangle is in F'. Considering the latter case, after the kernelization process of 3HS, there is a two-element subset $\{x, y\} \in C'$. While constructing T', we have added $k' + 1$ three-elements subsets containing x and y to C'; this results in at least $k' + 1$ triangles in T' containing x and y. Thus, $\{x, y\} \cap F' \neq \emptyset$. Summarizing both cases, $F' \cup H$ is a feedback vertex set of T with at most k vertices. □

The basic idea for the kernelization of FVST can be generalized to any vertex deletion problem whose goal graph can be characterized by a finite set of forbidden subgraphs consisting of three vertices; this results in the following theorem.

Theorem 3. *If a vertex deletion problem on directed or undirected graphs has a goal graph that can be characterized by a finite set of forbidden subgraphs consisting of three vertices, then this problem admits a problem kernel consisting of a graph with $O(k^3)$ vertices, where k denotes the number of allowed vertex deletions.*

4.2 Feedback Arc Set in Tournaments

We present a simple data reduction rule for Feedback Arc Set in tournaments (FAST), which leads to a kernel for this problem consisting of a tournament with $O(k^2)$ vertices. Without loss of generality, we assume that each vertex of the input tournament $(T = (V, E), k)$ is in at least one triangle.

Data reduction rule. If there is an arc in more than k triangles, then reverse this arc, add this arc to the solution, and decrease the parameter k by one.

Theorem 4. Feedback Arc Set *in tournaments admits a problem kernel consisting of an $O(k^2)$-vertex tournament that can be found in $O(kn^3)$ time.*

Proof (sketch). Suppose that we have a reduced FAST instance (T, k) where T has a feedback arc set F with at most k arcs. Then each triangle contains at least one arc from F. Due to the data reduction rule, each arc in F can be in at most k triangles. □

5 Search Tree for Feedback Arc Set in Bipartite Tournaments

Raman and Saurabh [16] have shown that if a tournament T does not contain a particular four-vertex tournament denoted by G, then the cycles in T are pairwise vertex-disjoint. Using this, their $O(2.42^k \cdot n^{2.38})$-time algorithm solves FAST in a two-phase manner: First, it uses a depth-bounded search tree approach to get rid of all cycles contained in subtournaments G appearing in T by reversing at most k arcs; this also destroys all subtournaments G in T. In the second phase, in each tournament output by the search tree it destroys all remaining, pairwise disjoint triangles by reversing an arbitrary arc in each triangle. If after these two phases there is an acyclic tournament with at most k arcs reversed, then T has a feedback arc set with size at most k.

Following the same approach, we derive a fixed-parameter algorithm for Feedback Arc Set in bipartite tournaments (FASBT). We use the following lemma, which is easy to prove.

Lemma 4. *A bipartite tournament is acyclic iff it contains no cycle of length four.*

By Lemma 4, in order to derive a forbidden subgraph characterization for bipartite tournaments where all cycles of length four are disjoint, we consider two length-four cycles in a bipartite tournament. If they are not vertex-disjoint, then they have one, two, or three common vertices. These three possibilities lead to bipartite tournaments which contain G_1 or G_2 shown in Fig. 4 as induced subgraph. The following lemma strengthens this finding.

Lemma 5. *If a bipartite tournament B contains neither G_1 nor G_2 (shown in Fig. 4) as an induced subgraph, then the cycles in B are pairwise disjoint.*

Proof. With Lemma 4, we first consider length-four cycles. By distinguishing three cases, namely two length-four cycles sharing one, two, and three vertices, respectively, one can easily show that a $\{G_1, G_2\}$-free bipartite tournament contains no two length-four cycles having a common vertex. Moreover, observe that in a bipartite tournament B, a subgraph of B induced by the vertices lying on

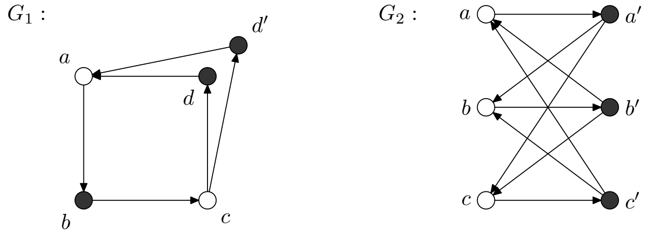

Fig. 4. Forbidden subgraphs for bipartite tournaments where all cycles of length four are disjoint. The color of the vertices describes the bipartition.

a cycle with length greater than four contains several length-four cycles which are not vertex-disjoint. Thus, a $\{G_1, G_2\}$-free bipartite tournament contains no cycle with a length greater than four. This completes the proof. $\square$

Based on Lemma 5 our algorithm solving FASBT has the same two phases as the algorithm by Raman and Saurabh [16], namely a search tree algorithm destroying all cycles contained in the induced subgraphs G_1 and G_2 from Fig. 4 and a polynomial-time second phase getting rid of the remaining, vertex-disjoint cycles. For destroying the cycles in G_1, the search tree algorithm makes a branching into six subcases, namely, reversing (a, b), reversing (b, c), reversing (c, d) and (c, d'), reversing (c, d) and (d', a), reversing (d, a) and (c, d'), and reversing (d, a) and (d', a). For each reversed arc, the parameter k is decreased by one. The size of depth-bounded search trees can be estimated using *branching vectors* [14]. The branching vector here is $(1, 1, 2, 2, 2, 2)$, corresponding to a search tree size of $O(3.24^k)$. Dealing with G_2, we make a branching into 17 subcases and, in each subcase, reverse two or three arcs. We omit the details of this branching. The worst-case runtime is determined by the branching for G_2, with a search tree size of $O(3.38^k)$. Note that finding one of G_1 and G_2 in an n-vertex bipartite tournament needs $O(n^6)$ time. When destroying vertex-disjoint cycles in the second phase, reversing arcs on cycles does not generate new cycles and, thus, we need only $O(n)$ time. The following theorem then follows.

Theorem 5. FEEDBACK ARC SET *in bipartite tournaments of n vertices with k arc deletions can be solved in $O(3.38^k \cdot n^6)$ time.*

6 Outlook

Table 1 surveys and compares complexity results on feedback set problems in tournaments. As can be seen there, the class of bipartite tournaments is not yet well explored. From a parameterized view, the grand challenge is to answer the question whether FVS in general directed graphs is fixed-parameter tractable or not, a long-standing open problem. On the route to this, further studying generalizations of tournaments might be fruitful. Besides attacking problems left open in Table 1, clearly further improvements concerning the efficiency of the described algorithms are very desirable. Due to the considerable practical relevance of the considered problems in applications such as voting systems, rankings, and graph drawing, they are natural candidates for algorithm engineering.

References

1. N. Ailon, M. Charikar, and A. Newman. Aggregating inconsistent information: ranking and clustering. In *Proc. 37th STOC*, pages 684–693. ACM, 2005.
2. N. Alon. Ranking tournaments. *SIAM Journal on Discrete Mathematics*, 20(1):137–142, 2006.
3. M.-C. Cai, X. Deng, and W. Zang. An approximation algorithm for feedback vertex sets in tournaments. *SIAM Journal on Computing*, 30(6):1993–2007, 2001.

4. M.-C. Cai, X. Deng, and W. Zang. A min-max theorem on feedback vertex sets. *Mathematics of Operations Research*, 27(2):361–371, 2002.
5. P. Charbit, S. Thomassé, and A. Yeo. The minimum feedback arc set problem is NP-hard for tournaments. *Combinatorics, Probability and Computing*, 2005. To appear.
6. V. Conitzer. Computing Slater rankings using similarities among candidates. Technical Report RC23748, IBM Thomas J. Watson Research Center, Yorktown Heights, NY, 2005.
7. F. K. H. A. Dehne, M. R. Fellows, M. A. Langston, F. A. Rosamond, and K. Stevens. An $O(2^{O(k)}n^3)$ FPT algorithm for the undirected feedback vertex set problem. In *Proc. 11th COCOON*, volume 3595 of *LNCS*, pages 859–869. Springer, 2005. To appear in *Theory of Computing Systems*.
8. R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999.
9. 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.
10. P. Festa, P. M. Pardalos, and M. G. C. Resende. Feedback set problems. In D. Z. Du and P. M. Pardalos, editors, *Handbook of Combinatorial Optimization, Vol. A*, pages 209–258. Kluwer, 1999.
11. J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006.
12. M. L. Fredman. On computing the length of longest increasing subsequences. *Discrete Mathematics*, 11(1):29–35, 1975.
13. J. Guo, J. Gramm, F. Hüffner, R. Niedermeier, and S. Wernicke. Improved fixed-parameter algorithms for two feedback set problems. In *Proc. 9th WADS*, volume 3608 of *LNCS*, pages 158–168. Springer, 2005. To appear in *Journal of Computer and System Sciences*.
14. R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.
15. R. Niedermeier and P. Rossmanith. An efficient fixed parameter algorithm for 3-Hitting Set. *Journal of Discrete Algorithms*, 1(1):89–102, 2003.
16. V. Raman and S. Saurabh. Parameterized algorithms for feedback set problems and their duals in tournaments. *Theoretical Computer Science*, 351(3):446–458, 2006.
17. B. Reed, K. Smith, and A. Vetta. Finding odd cycle transversals. *Operations Research Letters*, 32(4):299–301, 2004.
18. E. Speckenmeyer. On feedback problems in digraphs. In *Proc. 15th WG*, volume 411 of *LNCS*, pages 218–231. Springer, 1989.
19. A. Truß. Parameterized algorithms for feedback set problems in tournaments (in German). Diplomarbeit, Institut für Informatik, Friedrich-Schiller-Universität Jena, Dec. 2005.
20. A. van Zuylen. Deterministic approximation algorithms for ranking and clustering problems. Technical Report 1431, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY, Sept. 2005.