



Subset Feedback Vertex Set in Tournaments as Fast as Without the Subset

Satyabrata Jana  

University of Warwick, Coventry, UK

Lawqueen Kanesh  

Indian Institute of Technology, Jodhpur, India

Madhumita Kundu  

University of Bergen, Norway

Saket Saurabh  

The Institute of Mathematical Sciences, HBNI, Chennai, India

University of Bergen, Norway

Abstract

In the FEEDBACK VERTEX SET IN TOURNAMENTS (FVST) problem, we are given a tournament T and a positive integer k . The objective is to determine whether there exists a vertex set $X \subseteq V(T)$ of size at most k such that $T - X$ is a directed acyclic graph. This problem is known to be equivalent to the problem of hitting all directed triangles, thereby using the best-known algorithm for the 3-HITTING SET problem results in an algorithm for FVST with a running time of $2.076^k \cdot n^{\mathcal{O}(1)}$ [Wahlström, Ph.D. Thesis]. Kumar and Lokshtanov [STACS 2016] designed a more efficient algorithm with a running time of $1.6181^k \cdot n^{\mathcal{O}(1)}$. A generalization of FVST, called SUBSET-FVST, includes an additional subset $S \subseteq V(T)$ in the input. The goal for SUBSET-FVST is to find a vertex set $X \subseteq V(T)$ of size at most k such that $T - X$ contains no directed cycles that pass through any vertex in S . This generalized problem can also be represented as a 3-HITTING SET problem, leading to a running time of $2.076^k \cdot n^{\mathcal{O}(1)}$. Bai and Xiao [Theoretical Computer Science 2023] improved this and obtained an algorithm with running time $2^{k+o(k)} \cdot n^{\mathcal{O}(1)}$. In our work, we extend the algorithm of Kumar and Lokshtanov [STACS 2016] to solve SUBSET-FVST, obtaining an algorithm with a running time $\mathcal{O}(1.6181^k + n^{\mathcal{O}(1)})$, matching the running time for FVST.

2012 ACM Subject Classification Theory of computation \rightarrow Fixed parameter tractability

Keywords and phrases Parameterized algorithms, Feedback vertex set, Tournaments, Fixed parameter tractable, Graph partitions

Digital Object Identifier 10.4230/LIPIcs.IPEC.2024.17

Funding *Satyabrata Jana*: Supported by the Engineering and Physical Sciences Research Council (EPSRC) via the project MULTIPROCESS (grant no. EP/V044621/1)

Saket Saurabh: Supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 819416); and he also acknowledges the support of Swarnajayanti Fellowship grant DST/SJF/MSA-01/2017-18.

1 Introduction

In the d -HITTING SET problem, given a set family \mathcal{F} over a universe U of sets of size at most d and an integer k , the goal is to find a set $S \subseteq U$ of size at most k that intersects every set in \mathcal{F} . The importance of the d -HITTING SET problem stems from the number of other problems that can be re-cast in terms of it. For example, in the FEEDBACK VERTEX SET IN TOURNAMENTS (FVST) problem, the input is a tournament T together with an integer k . The task is to determine whether there exists a subset S of vertices of size at most k such that the sub-tournament $T - S$ obtained from T by removing S is acyclic. It turns out that FVST is a d -HITTING SET problem, where the vertices of T are the universe, and the family



© Satyabrata Jana, Lawqueen Kanesh, Madhumita Kundu, and Saket Saurabh; licensed under Creative Commons License CC-BY 4.0

19th International Symposium on Parameterized and Exact Computation (IPEC 2024).

Editors: Édouard Bonnet and Paweł Rzażewski; Article No. 17; pp. 17:1–17:17



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

\mathcal{F} is the family containing the vertex set of every directed cycle on three vertices (triangle) of T . Indeed, it can easily be shown that for every vertex set S , $T - S$ is acyclic if and only if S is a hitting set for \mathcal{F} . Another example is the CLUSTER VERTEX DELETION (CVD) problem. Here, the input is a graph G and an integer k , and the task is to determine whether there exists a subset S of at most k vertices such that every connected component of $G - S$ is a clique (such graphs are called *cluster* graphs). Also, this problem can be formulated as a d -HITTING SET problem where the family \mathcal{F} contains the vertex sets of all *induced* P_3 's of G . An induced P_3 is a path on three vertices where the first and last vertex are non-adjacent in G . Other examples include SUBSET FEEDBACK VERTEX SET (SFVS) on chordal graphs, TRIANGLE PACKING IN TOURNAMENTS, INDUCED P_3 -PACKING, etc.

The best-known fixed-parameter tractable (FPT) algorithm for d -HITTING SET runs in time $\mathcal{O}^*((d - 0.7262)^k)^1$ [21]. It is also known that d -HITTING SET admits $\mathcal{O}((2d - 1)k^{d-1} + k)$ kernel [1]. This implies an algorithm with running time $\mathcal{O}^*(2.270^k)$ and a $\mathcal{O}(k^2)$ elements kernel for the 3-HITTING SET problem. However, 3-HITTING SET can also be solved in time $\mathcal{O}^*(2.076^k)$ [23]. For a long time, advancements in its kernel size and FPT algorithms have stagnated. In response, researchers have shifted focus towards designing algorithms and kernels for implicit 3-HITTING SET problems such as FVST, CVD.

The design of algorithms with a running time better than $\mathcal{O}^*(2^k)$ and kernels with subquadratic elements for implicit 3-HITTING SET problems has emerged as a highly active and significant area of research. In pursuit of this objective, several important problems have been investigated, leading to notable successes. Fomin et al. in [9] broke the kernel barrier of some of the implicit 3-HITTING SET problems and obtained subquadratic kernels for several problems such as $\mathcal{O}(k^{3/2})$ vertex kernel for FVST, $\mathcal{O}(k^{5/3})$ vertex kernel for CVD, $\mathcal{O}(k^{3/2})$ vertex kernel for TRIANGLE PACKING IN TOURNAMENTS (TPT), and $\mathcal{O}(k^{5/3})$ vertex kernel for INDUCED P_3 -PACKING. Recently, Bessy et al. [3] introduced a novel technique called rainbow matching to design kernels for implicit 3-HITTING SET problems. They demonstrated that TPT and FVST admit (almost linear) kernels of $\mathcal{O}(k^{1 + \frac{\mathcal{O}(1)}{\sqrt{\log k}}})$ vertices. Utilizing the same technique, they showed that INDUCED 2-PATH-PACKING and INDUCED 2-PATH HITTING SET admit kernels of $\mathcal{O}(k)$ vertices.

Another well-studied implicit 3-HITTING SET problem is SUBSET FEEDBACK VERTEX SET (SFVS) on chordal and split graphs. A feedback vertex set in a graph G is a vertex set whose removal makes the remaining graph acyclic. The FEEDBACK VERTEX SET problem (FVS) is to decide whether a graph has a feedback vertex set of size at most k . In the more general SFVS problem, an additional subset S of vertices is given, and we want to find a vertex set of size at most k that hits all cycles passing through a vertex in S . SFVS has been shown to admit an algorithm with a running time of $\mathcal{O}^*(4^k)$ [15, 16]. However, on chordal and split graphs, Philip et al. [22] showed that this problem could be solved in $\mathcal{O}^*(2^k)$ time.

In this paper, we focus on SUBSET FEEDBACK VERTEX SET IN TOURNAMENTS (SUBSET-FVST). Below, we formally define the problem.

SUBSET FEEDBACK VERTEX SET IN TOURNAMENTS (SUBSET-FVST) **Parameter:** k
Input: A tournament $T = (V, A)$, a vertex set $S \subseteq V$, and a positive integer k .
Task: Find $X \subseteq V$ with $|X| \leq k$ such that $T - X$ has no cycle containing a vertex of S .

A tournament is a directed graph formed by a complete graph with oriented arcs. Motivated by applications such as voting systems and rank aggregation, FVST has drawn certain interests. It is well known that a tournament has a directed cycle if and only if there

¹ We use \mathcal{O}^* notation to hide factors polynomial in the input size.

is a directed triangle [7]. The fastest algorithm for FVST runs in time $\mathcal{O}^*(1.619^k)$ [19] and by the rainbow matching technique of Bessy et al. [3] FVST admit (almost linear) kernel of $\mathcal{O}(k^{1+\frac{\mathcal{O}(1)}{\sqrt{\log k}}})$. SUBSET-FVST can also be framed as a 3-HITTING SET problem, resulting in an algorithm with running time $\mathcal{O}^*(2.076^k)$ and an $\mathcal{O}(k^2)$ kernel. Recently, in 2023, Bai and Xiao [2] improved this and obtained an algorithm that runs in time $\mathcal{O}^*(2^{k+o(k)})$; however, the question to obtain an algorithm with running time better than $\mathcal{O}^*(2^k)$ was left open. In our research, we build upon the algorithm of Kumar and Lokshtanov [19] to tackle SUBSET-FVST, achieving an algorithm with running time $\mathcal{O}^*(1.6181^k)$, that matches the running time for FVST. We obtain the following theorem.

► **Theorem 1.** SUBSET-FVST is solvable in $\mathcal{O}(1.6181^k + n^{\mathcal{O}(1)})$ time.

Ideas for Theorem 1. We closely follow the approach of Kumar and Lokshtanov [19], but due to the inherent generality of our problem, we need to deviate significantly from it while implementing the outline. Let (T, S, k) be an instance of SUBSET-FVST. The algorithm relies on a simple observation that T has no S -cycle (directed cycle containing a S vertex) if and only if T has no S -triangle. Our algorithm enumerates subexponential many sets ($2^{o(k)}$) or branches with a branching vector $(1, 2)$. The algorithm first identifies $2^{o(k)}$ subsets of S , such that for every solution H , there is at least one set, say M , that is disjoint from it. The set M allows us to discover several structures and apply reduction rules. For example, we know that $T[M]$ is a directed acyclic graph (DAG). In other words, if any vertex $v \in V(T) - M$, we have that $T[M \cup \{v\}]$ has a directed cycle, then v must be in H . Let σ be a unique topological ordering of $T[M]$. This immediately gives us the notion of M -block: the set of vertices which are common out-neighbors and in-neighbors of two consecutive vertices of M in σ (and not containing any M vertex). Now we analyze S -triangles: those that are fully contained inside a block (local triangle) or contain vertices of at least 2 blocks (shared triangle). Our main objective is to reduce the case of shared triangles to a vertex-cover like branching (either a vertex or its neighbors must be in a solution) and independently solve the problem of hitting local triangles. In the latter case, we use the fact that each block has at most $\log^{\mathcal{O}(1)} k$ many vertices from S and hence any solution must contain at most $\log^{\mathcal{O}(1)} k$ many vertices from each block. Thus, using the fact that SUBSET-FVST has a polynomial kernel of size $\mathcal{O}(k^2)$, we can solve these instances in $k^{\mathcal{O}(\log^{\mathcal{O}(1)} k)}$ time. The most interesting part of the algorithm is to reduce to this case. This requires branching on “backward arcs” between two blocks. If there is a vertex v with at least two incident backward arcs, then we branch on v , leading to a branching vector $(1, 2)$. When this is not possible, then we have that these backward arcs form a matching. In this case if no block has many backward arcs incident then we can partition these edges and decompose the problem. This leads to a divide-and-conquer step in our algorithm. This concludes a brief description of our algorithm.

Related Work on Feedback Vertex Set

The FEEDBACK VERTEX SET problem (FVS) is one of the earliest known NP-complete problems shown in the influential paper by Karp [18] and has been thoroughly explored in the realm of parameterized complexity. The earliest known FPT algorithms for FVS date back to the late 1980s and early 1990s. These algorithms relied on the groundbreaking Graph Minor Theory by Robertson and Seymour. Over time, there have been multiple advancements and refinements, leading to the current best deterministic FPT algorithm for FVS, which runs in $\mathcal{O}^*(3.460^k)$ time [14]. The fastest known randomized algorithm for this problem given by Li and Nederlof [20] running in time $\mathcal{O}^*(2.7^k)$. Recently a factor

$(1 + \epsilon)$ approximation algorithm for FVS, which has better running time than the best-known (randomized) FPT algorithm for every $\epsilon \in (0, 1)$ is given by Jana et al. [17]. In directed graphs, FVS becomes harder in terms of parameterized algorithms. Whether FVS in directed graphs is FPT has been a long-standing open problem. Finally, Chen et al. [4] gave a FPT algorithm running in time $\mathcal{O}^*(4^k k!)$.

SUBSET FEEDBACK VERTEX SET problem (SFVS) was first systematically studied by Even et al. [8] where they showed that SFVS in undirected graphs admits an 8-approximation. Cygan et al. [6] showed that SFVS in undirected graphs admits an FPT algorithm running in time $\mathcal{O}^*(2^{\mathcal{O}(k \log k)})$. Recently, Iwata et al. [15, 16] improved this to single-exponential algorithm with running time $\mathcal{O}^*(4^k)$. This problem is also studied in special graph classes. Philip et al. [22] showed that in split and chordal graphs this problem can be solved in time $\mathcal{O}^*(2^k)$. Regarding kernelization, Hols and Kratsch [13] obtained a randomized kernel with $\mathcal{O}(k^9)$ vertices. Chitnis et al. [5] considered this problem in directed graph and provide a FPT algorithm running in time giving a $\mathcal{O}^*(2^{\mathcal{O}(k^3)})$.

2 Preliminaries

Let $[n]$ be the set of integers $\{1, \dots, n\}$. For a pair a, b of integers with $a < b$, we denote the set $\{a, a + 1, \dots, b\}$ by $[a, b]$. For a directed graph D , we denote the set of vertices of D by $V(D)$ and the set of arcs by $A(D)$. For a subset X of vertices $X \subseteq V(D)$, we use the notation $D - X$ to mean the graph $D[V(D) \setminus X]$. Given a digraph D , a vertex set $X \subseteq V(D)$ is called a *feedback vertex set* (in short, fvs) of D if there is no directed cycle in the graph $D - X$. Given a directed graph D and a vertex set S , a vertex $v \in V(D)$ is called an *S-vertex* if $v \in S$. A directed cycle in D is called an *S-cycle* if the cycle contains at least one *S-vertex*. A *S-cycle* is called an *S-triangle* if it is a cycle of three vertices. D is called *S-acyclic* if D has no *S-cycle*. A vertex set $X \subseteq V(D)$ is called an *S-feedback vertex set* of D (in short, *S-fvs*) if $D - X$ is *S-acyclic*. Let σ be an ordering of $V(D)$. For a pair of adjacent vertices u, v with an arc (u, v) , we say the arc is *backward* (resp, *forward*) with respect to the ordering σ if $v \leq_\sigma u$ (resp, $u \leq_\sigma v$). It is called an *S-backward* arc (resp, *S-forward* arc) if there exists some *S-vertex* s such that $v \leq_\sigma s \leq_\sigma u$ (resp, $u \leq_\sigma s \leq_\sigma v$). We call an ordering without *S-backward* arcs an *S-topological ordering*. A directed graph is called a *tournament* if there is an arc between every pair of vertices. Unless specified, we use cycle to mean directed cycle.

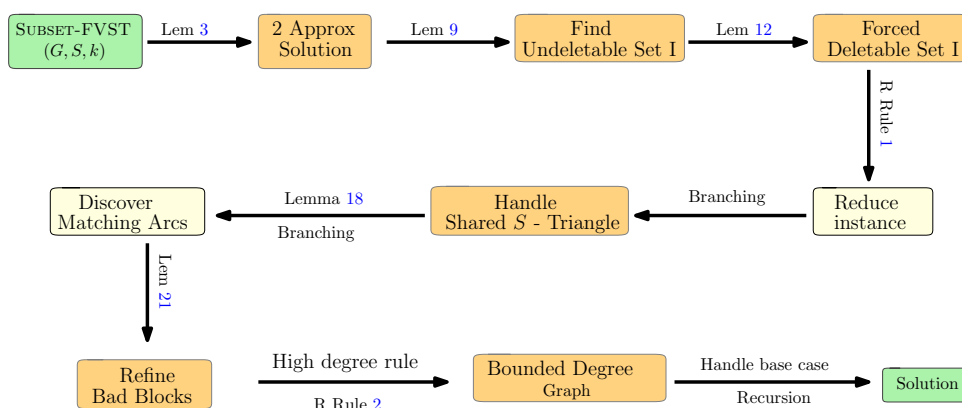
Fixed parameter tractable. A parameterized problem Π is a subset of $\Gamma^* \times \mathbb{N}$ for some finite alphabet Γ . An instance of a parameterized problem consists of (X, k) , where k is called the parameter. A parameterized problem L is considered to have a fixed parameter tractable (FPT) algorithm if there is an algorithm \mathcal{A} that can determine whether $(X, k) \in L$ in time $f(k) \cdot n^{\mathcal{O}(1)}$ for some computable function f , where n is the size of the input. An important tool from the FPT toolkit is *kernelization* [10, 11]. A kernelization replaces, in polynomial time, an instance by a decision equivalent instance (the kernel) whose size can be bounded by a function of the parameter k , that is, it will not depend on the original problem size n anymore.

3 FPT algorithm for SUBSET-FVST

In this section, we prove the following result.

► **Theorem 1.** SUBSET-FVST is solvable in $\mathcal{O}(1.6181^k + n^{\mathcal{O}(1)})$ time.

A schematic diagram showing the main steps of our algorithm is shown in Figure 1.



■ **Figure 1** A summary of the steps of our algorithm.

3.1 Preprocessing Step

It is well-known that a tournament is acyclic if and only if it does not contain any triangle [7], which allows us to formulate FVST as a 3-HITTING SET problem. We first observe a similar statement for the subset variant in the next lemma proved in [2].

► **Lemma 2** ([2, Lemma 2]). *A tournament is S -acyclic if and only if it does not contain an S -triangle.*

Lemma 2 immediately gives rise to a greedy 3-approximation algorithm for SUBSET-FVST. However, Gupta et al. [12] designed a 2-factor approximation algorithm for SUBSET-FVST. Using this result, we get the following lemma.

► **Lemma 3.** *Given a tournament T with n vertices, a subset $S \subseteq V(T)$ and integer k , in $n^{\mathcal{O}(1)}$ time we correctly conclude that T has no S -feedback vertex set of size at most k or outputs a S -feedback vertex set of size at most $2k$.*

The initial phase of our algorithm for SUBSET-FVST involves reducing the problem to its kernel. By reducing the instance of the SUBSET-FVST problem into an instance of the 3-HITTING SET problem, by using a kernel for 3-HITTING SET [1], one can derive a kernel on $\mathcal{O}(k^2)$ vertices for SUBSET-FVST, which is an induced subgraph of the input tournament. Formally, we have the following lemma.

► **Lemma 4** ([1, 2]). *Given a tournament T with n vertices, a subset $S \subseteq V(T)$ and an integer k , in $n^{\mathcal{O}(1)}$ time we can output a tournament T' (an induced subgraph), a vertex set $S' \subseteq V(T')$ and an integer k' such that $|V(T')| \leq \mathcal{O}(k^2)$, $k' \leq k$, and T' has a S' -feedback vertex set of size at most k' if and only if T has a S -feedback vertex set of size at most k .*

In what follows, we assume that we have applied Lemma 4 and obtained an equivalent instance, (T, S, k) , such that $|V(T)| \leq \mathcal{O}(k^2)$. We call such instances *reduced*.

3.2 Discovering Structure I: Universal Undeletable Family \mathcal{M}

In the second step of our algorithm, we find a family of subsets of S such that any solution avoids at least one set in our family. Toward defining the family we first need a notion of block which relies on the notion of between and consecutive.

17:6 Subset FVS in Tournaments as Fast as Without the Subset

- **Definition 5** (Between and Consecutive). Let D be a directed graph.
- For any pair of vertices $u, v \in V(D)$, the set $\text{between}(D; u, v)$ is defined as $N^+(u) \cap N^-(v) \setminus \{u, v\}$.
 - Let $X \subseteq V(D)$. Two vertices $u, v \in X$ are called X -consecutive if $(u, v) \in A(D)$ and $\text{between}(D; u, v) \cap X = \emptyset$.

► **Definition 6** (X -block). Let D be a directed graph and $X \subseteq V(D)$. We define the set of X -blocks in D as follows.

- For each pair of X -consecutive vertices u and v , we define the X -block, denoted by $\text{block}(X; u, v)$, as $\text{between}(D; u, v)$. That is, $\text{block}(X; u, v) = \text{between}(D; u, v)$.
- For each vertex $u \in X$ with no in-neighbors in X we define the X -block, denoted by $\text{block}(X; u)$, as $N^-(u)$. That is, $\text{block}(X; u) = N^-(u)$.
- For each vertex $v \in X$ with no out-neighbors in X we define the X -block, denoted by $\text{block}(X; v)$, as $N^+(v)$. That is, $\text{block}(X; v) = N^+(v)$.

These definitions immediately imply the following observation for an acyclic tournament.

► **Observation 7.** Let T be an acyclic tournament and $X \subseteq V(T)$. Furthermore, let σ be the unique topological order of $V(T)$. Then the following holds.

1. For any pair of vertices $u, v \in V(T)$ the set $\text{between}(T; u, v)$ is exactly the set of vertices in T that appear between u and v in σ .
2. Two vertices $u, v \in X$ are X -consecutive if no vertex of X appears between u and v in σ .
3. X -blocks form a unique partition of $V - X$, where two vertices belong to a different block if and only if there exists a vertex of X that appears between them in σ .

Now we are ready to define the notion of universal undeletable family.

► **Definition 8** (Universal Undeletable Family). Let (T, S, k) be an instance of SUBSET-FVST. A *universal undeletable family*, \mathcal{M} , is a family of subsets of S , which satisfies the following properties. For every S -feedback vertex set H of T size at most k there exists a set $M \in \mathcal{M}$ such that

1. $H \cap M = \emptyset$;
2. in each M -block B in $T - H$ we have $|B \cap S| \leq 2 \log^2 k$.

Our main result in this section is the following.

► **Lemma 9.** Let (T, S, k) be a reduced instance. There exists an algorithm that takes (T, S, k) as input and outputs a universal undeletable family \mathcal{M} of size $2^{\mathcal{O}(\frac{k}{\log k})}$.

Proof. Let X be a S -feedback vertex set of size at most $2k$ obtained using Lemma 3. Let $Y := V(T) \setminus X$, $S_X := X \cap S$, $S_Y := Y \cap S$ and $v_1, v_2, \dots, v_{|S_Y|}$ be the topological sort of $T[S_Y]$ such that the edges in $T[S_Y]$ are directed from left to right. We color S_Y using $\lfloor \log^2 k \rfloor$ colors $\{1, 2, \dots, \lfloor \log^2 k \rfloor\}$ such that for each $d \in [|S_Y|]$, v_d gets color $d \bmod \lfloor \log^2 k \rfloor$. For each $c \in [\lfloor \log^2 k \rfloor]$, let S_c be the set of vertices in S_Y which get color c . Each $M \in \mathcal{M}$ is specified by a 4-tuple $\langle c, \hat{H}, \hat{R}, \hat{X} \rangle$ where

- c is a color in $[\lfloor \log^2 k \rfloor]$,
- $\hat{H} \subseteq S_c$ such that $|\hat{H}| \leq \frac{k}{\log^2 k}$,
- $\hat{R} \subseteq S_Y \setminus S_c$, $|\hat{R}| \leq |\hat{H}|$, and
- $\hat{X} \subseteq X$ such that $|\hat{X}| \leq \frac{2k}{\log^2 k}$.

For each 4-tuple $\langle c, \hat{H}, \hat{R}, \hat{X} \rangle$, let $M := (S_c \setminus \hat{H}) \cup \hat{R} \cup \hat{X}$. Hence $|\mathcal{M}|$ is upper bounded by the maximum possible number of such 4-tuples.

$$\begin{aligned}
|\mathcal{M}| &\leq \log^2 k \times \left(\frac{\mathcal{O}(k^2)}{k}\right) \times \left(\frac{\mathcal{O}(k^2)}{\log^2 k}\right) \times \left(\frac{2k}{\log^2 k}\right) \\
&\leq 2^{\log(\log^2 k)} \times \mathcal{O}(k^2)^{\frac{2k}{\log^2 k}} \times (2k)^{\frac{2k}{\log^2 k}} \\
&= 2^{\mathcal{O}(\frac{k}{\log k})}
\end{aligned}$$

This completes the description of the enumeration of \mathcal{M} . Next, we prove the correctness of the above algorithm by showing that for every S -feedback vertex set H of T with size at most k , \mathcal{M} contains a set M that satisfies the properties listed in the statement of the lemma. Consider an arbitrary S -feedback vertex set H of T with size at most k .

For each $j \in [\lceil \log^2 k \rceil]$, let $H_j := S_j \cap H$. By the pigeonhole principle, there is a color c such that $0 < |H_j| \leq \frac{k}{\log^2 k}$. For this color c , let $\hat{H} := H_c$. Note that $H_c \subseteq S_c \subseteq S_Y$. Now, consider a set \hat{R} obtained as follows: for every vertex $v \in H_c$, pick the first vertex after v (if there is any) in $S_Y \setminus (S_c \cup H)$, in the topological ordering of $T[S_Y]$. Note that $T[X \setminus H]$ is S -acyclic. Let $S_X^H := S \cap (X \setminus H)$. Now $|S_X^H| \leq 2k$ and there is a topological ordering of $T[S_X^H]$. We then color S_X^H using $\lceil \log^2 k \rceil$ colors in a similar way as we did for S_Y . Let $\hat{X} \subseteq S_X^H$ be the set of all vertices with color 1 in this coloring. Clearly, $|\hat{X}| \leq \frac{2k}{\log^2 k}$.

The 4-tuple $\langle c, \hat{H}, \hat{R}, \hat{X} \rangle$ described above satisfies all the properties listed in the construction of \mathcal{M} . Let $M := (S_c \setminus \hat{H}) \cup \hat{R} \cup \hat{X}$. Clearly, $M \subseteq S$, $M \cap H = \emptyset$, and $M \in \mathcal{M}$. Since in any $[(S_c \setminus H_c) \cup \hat{R}]$ -block B in Y , it holds $|B \cap S| \leq \log^2 k$, it also holds in each M -block B in $T - H$ that $|B \cap S| \leq 2 \log^2 k$. \blacktriangleleft

3.3 Discovering Structure II: Forced Deletable Set I

Lemma 9 gets us one step closer to our goal. Let H be a hypothetical solution of size at most k of (T, S, k) . We know that there exists a set $M \in \mathcal{M}$ such that in each M -block in $T - H$, we have a small number of S -vertices, i.e., each M -block contains a small number ($\leq 2 \log^2 k$) of non-solution S -vertices. However, it is possible that there is a M -block of T that contains too many vertices of S , because that block contains *many solution S -vertices, that is, vertices of $H \cap S$* . In this section, we deal with this case. We assume that we know M corresponding to the hypothetical solution H . Indeed, we can achieve this by going over each set in \mathcal{M} , and that will only cost us a factor of $2^{\mathcal{O}(\frac{k}{\log k})}$ in the running time of our algorithm. It will be useful to remember that $T[M]$ is a directed acyclic tournament.

We start with a simple definition that will be useful.

► **Definition 10 (Consistency).** Let T be a tournament and $M \subseteq V(T)$. For a vertex $v \in V(T)$, we say that v is *consistent* with M if there is no $\{v\}$ -cycle in $T[M \cup v]$.

Observe that if v is not consistent with M , then there is a v -cycle in $T[M \cup v]$, which is also a S -cycle, since $M \subseteq S$. Thus, since we know that $M \cap H = \emptyset$, v must belong to H . We next integrate such vertices and blocks that may contain a lot of S vertices (in the solution) into the notion of universal undeletable families. We call this notion universal (deletable-undeletable) families (in short, *du-family*).

► **Definition 11 (du-Family).** Let (T, S, k) be an instance of SUBSET-FVST. A *du family*, $\mathcal{F} = \{(M_1, P_1), \dots, (M_\ell, P_\ell)\}$, is a family of disjoint pairs of vertex sets, which satisfies the following properties. For every S -feedback vertex set H of T size at most k there exists a set pair $(M, P) \in \mathcal{F}$ such that the following holds.

1. $M \subseteq S$, $M \cap H = \emptyset$.
2. $P \subseteq H$ (recall, $M \cap P = \emptyset$).

3. Every vertex of $T - P$ is consistent with M .
 4. In each M -block B in $T - P$, we have $|B \cap S| \leq 2 \log^4 k$.
- Furthermore, the set H is said to be *compatible* with the pair (M, P) .

► **Lemma 12.** *Let (T, S, k) be a reduced instance. There exists an algorithm that takes (T, S, k) as input and outputs a du-family \mathcal{F} of size $2^{\mathcal{O}(\frac{k}{\log k})}$.*

Proof. We define a function f that for a given tournament T and a subset $M \subseteq V(T)$, outputs a set of vertices that are *not consistent* with M . We denote this output set as $f(T, M)$. More specifically

$$f(T, M) := \{v \mid v \in V(T) \setminus M \text{ and } T[M \cup \{v\}] \text{ has a } \{v\}\text{-cycle}\}$$

Observe that each v -cycle here is also a S -cycle since $M \subseteq S$. Thus, since we know that $M \cap H = \emptyset$, v must belong to H . Furthermore, we define another function g that, given a tournament T , two subsets $M \subseteq S \subseteq V(T)$, and an integer k , outputs the S -vertices contained within some M -block B in $T - f(T, M)$ with $|B \cap S| > 2 \log^4 k$. We denote this output set as $g(T, S, M, k)$. We can compute f and g in time $k^{\mathcal{O}(1)}$.

We use the algorithm of Lemma 9 to compute \mathcal{M} . Then for each $M \in \mathcal{M}$, we compute the sets $f(T, M)$ and $g(T, S, M, k)$. For each $Z \subseteq g(T, S, M, k)$ such that $|Z| \leq \frac{2k}{\log^2 k}$, we output a pair of sets $(M, P) = (M, f(T, M) \cup (g(T, S, M, k) \setminus Z))$. The family \mathcal{F} is the collection of all such pairs of sets.

We prove that the algorithm satisfies the stated properties. Consider a S -feedback vertex set H of size at most k . By Lemma 9 there exists $M \in \mathcal{M}$ such that $M \subseteq S$ and $M \cap H = \emptyset$. Because of the definition of $f(T, M)$ and $M \cap H = \emptyset$, the vertex set $f(T, M)$ must be contained in H . Now for every vertex $u \in V(T - M) \setminus f(T, M)$ the induced subtournament $T[M \cup \{u\}]$ of T is acyclic. So u can be placed uniquely in the topological order of $T[M \cup \{u\}]$. Hence, for each $u \in V(T - M) \setminus f(T, M)$, there is a unique M -block containing it. Since in each M -block B in $T - H$ we have $|B \cap S| \leq 2 \log^2 k$, we also have in each M -block B in $T - f(T, M)$ that $|B \cap S| \leq k + 2 \log^2 k$.

We say that a M -block B in $T - C$ is *large* if $|B \cap S| \geq 2 \log^4 k$. From each large M -block at least $(2 \log^4 k - 2 \log^2 k)$ many S -vertices belong to H . So the number of large blocks is bounded by $\frac{k}{2 \log^4 k - 2 \log^2 k}$. Hence in total at most $\frac{k}{2 \log^4 k - 2 \log^2 k} \times 2 \log^2 k \leq \frac{2k}{\log^2 k}$ many S -vertices from the union of large M -blocks do not belong to H . Since the algorithm loops over all choices of subsets $Z \subseteq g(T, S, M, k)$ with $|Z| \leq \frac{2k}{\log^2 k}$, the family \mathcal{F} contains a pair (M, P) satisfying the properties listed in the lemma.

Moreover, $|\mathcal{F}|$ is bounded by the product of $|\mathcal{M}|$ and the number of subsets Z . Now $Z \subseteq S$ and $|Z| \leq \frac{2k}{\log^2 k}$ together imply that the number of subsets Z is at most $\mathcal{O}(k^2)^{\frac{2k}{\log^2 k}} = 2^{\mathcal{O}(\frac{k}{\log k})}$. Hence $|\mathcal{F}| = |\mathcal{M}| \times 2^{\mathcal{O}(\frac{k}{\log k})} \leq 2^{\mathcal{O}(\frac{k}{\log k})} \times 2^{\mathcal{O}(\frac{k}{\log k})} = 2^{\mathcal{O}(\frac{k}{\log k})}$. ◀

Our algorithm for SUBSET-FVST applies Lemma 12 and obtains a du-family \mathcal{F} . The algorithm then iterates over each pair $(M, P) \in \mathcal{F}$, and looks for a S -feedback vertex set H of size at most k that is compatible with (M, P) . Henceforth, our problem reduces to the following.

Given an instance (T, S, k) , and a pair (M, P) of vertex sets in T . The objective is to find a S -feedback vertex set H for T of size at most k that is compatible with (M, P) .

We first apply the following reduction rule to make the instance smaller. The correctness of Reduction Rule 1 follows from the fact that we are looking for a S -feedback vertex set H for T of size at most k that is compatible with (M, P) (Lemma 12).

► **Reduction Rule 1.** Delete P from the graph. The resultant instance is $(T - P, S \setminus P, k - |P|)$.

The pair (M, P) naturally partitions the vertices of $T - (P \cup M)$ into local subtournaments corresponding to the induced graphs on the M -blocks in $T - P$. For clarity of notation, we denote the reduced instance $(T - P, S \setminus P, k - |P|)$ by (T, S, k) itself. The properties of $(T - P, S \setminus P, k - |P|) = (T, S, k)$ that we need in the future are encapsulated below.

1. For all $v \in V(T) \setminus M$, we have that $T[M \cup \{v\}]$ does not contain a $\{v\}$ -cycle. In particular, $T[M \cup \{v\}]$ is acyclic.
2. In each M -block B in T , we have $|B \cap S| \leq 2 \log^4 k$.

Following the implementation of Reduction Rule 1, we categorize S triangles in T into two distinct groups, as described below.

- (i) **Local S -triangle:** All three vertices are within one M -block in T .
- (ii) **Shared S -triangle:** Those that are not local.

Notice that in each shared S -triangle (not containing a vertex of M) there exists a pair of vertices in the triangle that belong to different M -blocks in $T - P$. Next, we understand how M participates in a S -triangle. Observe that every S -triangle contains at most one vertex from M . Furthermore, we show that every S -triangle containing a M vertex (which we say M -triangle) must be a shared S -triangle.

► **Observation 13.** Every M -triangle is a shared S -triangle. That is, there is no pair of vertices u, v such that both vertices u, v belong to the same M -block in T and there is a vertex $w \in M$ that forms a triangle with u, v .

Proof. Let u, v be a pair of vertices in T . Since we have applied Reduction Rule 1, it follows that the vertex $u \in V(T)$ is consistent with M , meaning that there is no $\{u\}$ -cycle in $T[M \cup \{u\}]$. That implies $T[M \cup \{u\}]$ is a DAG. Thus, there exists a unique partition $M_1 \cup M_2$ of M such that for all $x \in M_1, y \in M_2$ we have $(x, u), (u, y) \in V(T)$. Similarly, since $T[M \cup \{v\}]$ is also a DAG, there exists a unique partition $M'_1 \cup M'_2$ of M such that for all $x' \in M'_1, y' \in M'_2$ we have $(x', v), (v, y') \in V(T)$. Furthermore, since u and v belong to the same M -block, it must be that $M_1 = M'_1$ and $M_2 = M'_2$. Therefore, both $T[M_1 \cup \{u, v\}]$ and $T[M_2 \cup \{u, v\}]$ are DAGs. Now, the vertex w is in either M_1 or M_2 . In either case, w cannot form a cycle with $\{u, v\}$. This concludes the proof. ◀

Next, we show that having shared S -triangle is equivalent to having a M -triangle

► **Lemma 14.** If there is a shared S -triangle uvw , then there exists a vertex $m \in M$ such that there is a shared S -triangle umw .

Proof. If there is a shared S -triangle, say uvw , then it must contain vertices from two different M blocks, say B_1 and B_2 , where the vertices in B_1 appear before B_2 and there is an arc (v, u) with $u \in B_1$ and $v \in B_2$. Let m be a vertex of M that appears before any vertex of B_2 and after every vertex of B_1 . This implies we have arcs (u, m) and (m, v) . This implies that we have a triangle umv containing a M vertex. ◀

Observation 13 and Lemma 14 imply the following.

► **Lemma 15.** There is a shared S -triangle if and only if there is a M -triangle.

17:10 Subset FVS in Tournaments as Fast as Without the Subset

Before we proceed further, we design an algorithm for SUBSET-FVST that runs in time $2^{k+o(k)} + n^{\mathcal{O}(1)}$. Branch on each of the triangles containing M -vertices. This will lead to 2^k branching factor. Lemma 15 implies that the only S -triangle that remains after branching are local S -triangles, and hence we can solve the problem independently on each block. However, observe that in each M -block B in T , we have $|B \cap S| \leq 2 \log^4 k$. This implies that the size of an optimal solution in B is upper-bounded by $2 \log^4 k$. Thus, we can solve the problem for each block independently in time $k^{\mathcal{O}(\log^4 k)}$. This results in the following.

► **Theorem 16.** SUBSET-FVST is solvable in time $2^{k+o(k)} + n^{\mathcal{O}(1)}$.

This algorithm is comparable to the best known algorithm of Bai and Xiao [2]. We move on to designing the faster algorithm. This algorithm also follows the route of Theorem 1. We move on to Section 3.4 when there is at least one shared S -triangle, otherwise we move to Section 3.7.

3.4 Branching Structure I: Shared S -triangle

Our objective is to reduce to “vertex cover” (either a vertex or all of its neighbors must go in any solution) like branching in this case. Towards this, we first define the set of edges for which we need vertex cover.

Dealing with shared S -triangles. Consider the ordered partition of the vertices of T based on M vertices in the following way: each M vertex is a singleton and they are ordered according to the unique topological ordering σ of $T[M]$. The other parts are defined by M -block with corresponding order. For example, let $|M| = \ell$ and m_1, \dots, m_ℓ be the vertices of M such that $m_1 < \dots < m_\ell$ in σ . Then our ordered partition of T would be

$$B_1 = \text{block}(X, m_1) \cup \{m_1\} \cup B_2 = \text{block}(M; m_1, m_2) \cup \dots \cup \{m_\ell\} \cup B_{\ell+1} = \text{block}(X, m_\ell).$$

Observe that every vertex v not in M belongs to a unique block as $T[M \cup \{v\}]$ is acyclic.

Let R be the set of arcs (u, v) of $A(T)$ such that $u \in B_j$ and $v \in B_i$ and $i, j \in [\ell + 1]$, and $i < j$. We call R a set of *red arcs*. Next, we show that dealing with shared triangles is equivalent to hitting arcs in R .

► **Lemma 17.** *Every shared S -triangle has a red arc. Furthermore, for every red arc $e = (u, v) \in R$ we have a M -triangle containing e . This implies that we have a shared S -triangle if and only if there is a red arc.*

Proof. If a S -triangle is not local, then it contains vertices of two distinct blocks or two vertices of the same block and a M vertex. By Observation 13 we know that there is no S -triangle containing two vertices of the same block and a M vertex. It is clear that any S -triangle containing vertices of two distinct blocks contains a red arc.

Let $e = (u, v) \in R$. Then there exists $i, j \in [\ell + 1]$ such that $u \in B_j$ and $v \in B_i$ where $i < j$. Let m be a vertex of M that appears before any vertex of B_j and after every vertex of B_i . We know from the properties of the M blocks that we have arcs (v, m) and (m, u) . This implies that we have a triangle vmu containing a M vertex. This concludes the proof. ◀

► **Lemma 18.** *Let H be a solution to (T, S, k) . Then H is a vertex cover of R (that is, it intersects each edge in R).*

Proof. For every red arc $e = (u, v) \in R$ we have a M -triangle containing e (Lemma 17). Since, by construction no red arc has an end-point in M and $M \cap H = \emptyset$ by assumption, we have that H must contain either u or v . This concludes the proof. ◀

We first consider the case where there are some vertices that are incident to more than one red arc. Once we deal with that case, the only situation left is that all the red arcs that remain to hit form a matching. Now we describe a recursive algorithm which searches for a potential solution H of size at most k by branching. Let $G = (V(T), R)$ be an undirected graph with vertex set $V(T)$ and edge set R (without orientation). For any vertex v , let $\text{Red}(v)$ (the *red-degree*) denote the degree of v in G . Consider a vertex v of highest red-degree. We know $\text{Red}(v) > 0$. If $k = 0$, then return No. From now on, assume that $k \geq 1$. If $\text{Red}(v) \geq 2$, the algorithm branches into two cases: $v \in H$ or $v \notin H$. In the branch where v is added to H , k drops by 1. Hence, we return the instance $(T - v, S \setminus \{v\}, k - 1)$. In the other branch where v is not added to H , we know that all the neighbors in G must be in the solution (by Lemma 18). Hence, we return the instance $(T - N_G(v), S \setminus N_G(v), k - \text{Red}(v))$. Here, $N_G(v)$ denotes the neighbors of v in G . This (1, 2) branching step dominates the running time of the algorithm and corresponds to $\mathcal{O}(1.6181^k \times k^{\mathcal{O}(1)})$ time.

Now we assume that maximum red-degree of any vertex is at most 1. In this case, all the red arcs form a matching in G . We refer to these kind of red arcs as *matching arcs*. Let (T, S, k, M) obtained after branching be called the *matching instance*. In Section 3.5, we look for more structure in the input and take advantage of it.

3.5 Discovering Structure III: Matching Arcs

We now introduce the notion of a *bad* block to represent blocks that are incident to a *large* number of matching arcs.

► **Definition 19** (Bad block). An M -block B is said to be *bad* if the number of matching arcs incident to the vertices in B is at least $\log^4 k$. Otherwise we say that the block B is *good*.

Now, in the following observation, we show that the number of bad blocks will be bounded.

► **Observation 20.** *The number of bad M -blocks in T is at most $\frac{2k}{\log^4 k}$.*

Proof. We know that from each matching-arc, at least one end-point of the arc must belong to H (by Lemma 18). If the number of bad M -blocks in T is more than $\frac{2k}{\log^4 k}$, then the number of vertices incident to the matching arcs is more than $2k$ which in turn implies that the number of matching arcs are more than k . So we cannot hit all the matching arcs using at most k vertices. ◀

► **Lemma 21.** *Let (T, S, k, M) be a matching instance of SUBSET-FVST. Then there exists an algorithm that in $2^{\mathcal{O}(\frac{k}{\log k})}$ time outputs a family $\mathcal{F}' = \{(M'_1, P'_1), \dots, (M'_\ell, P'_\ell)\}$ of sets with $|\mathcal{F}'| = 2^{\mathcal{O}(\frac{k}{\log k})}$ such that if there exists a S -feedback vertex set H of T of size at most k such that $H \cap M = \emptyset$ and there exists $(M', P') \in \mathcal{F}'$ satisfying:*

1. $M \cap M' = \emptyset$
2. $M' \cup P' \subseteq S$,
3. $M' \cap H = \emptyset, P' \subseteq H$,
4. In each $(M \cup M')$ -block B in $T - P'$ we have $|B \cap S| \leq 2 \log^4 k$.
5. In each $(M \cup M')$ -block B in $T - P'$ either B has no S -vertex or B is good.

Proof. Let \mathcal{B} be the set of all bad M -blocks. Let \mathcal{Q} denote the set of all S -vertices in \mathcal{B} . For each $Q \subseteq \mathcal{Q}$ such that $|Q| \leq \frac{4k}{\log^2 k}$, we output a pair of sets $(M', P') = (Q, \mathcal{Q} \setminus Q)$. The family \mathcal{F}' is the collection of all such pairs of sets.

We prove that the algorithm satisfies the stated properties. Clearly, $M' \cup P' \subseteq S$. Consider a S -feedback vertex set H of size at most k such that $H \cap M = \emptyset$. Due to the Lemma 9, in any M -block B in $T - H$, we have $|B \cap S| \leq 2 \log^2 k$. So from each bad M -block in T there are at

17:12 Subset FVS in Tournaments as Fast as Without the Subset

most $2 \log^2 k$ many non solution S -vertices. As the number of bad M -blocks in T is bounded by $\frac{2k}{\log^4 k}$ (by Observation 20), in total there are at most $\frac{2k}{\log^4 k} \times 2 \log^2 k$ many S -vertices from the union of bad M -blocks that do not belong to H . Since the algorithm loops over all choices of subsets $Q \subseteq \mathcal{Q}$, $|Q| \leq \frac{4k}{\log^2 k}$, the family \mathcal{F}' contains a pair (M', P') such that $M' \cap H = \emptyset$, $P' := \mathcal{Q} \setminus M' \subseteq H$. As in each of the M -block in T we have $|B \cap S| \leq 2 \log^4 k$ (by Lemma 12), we also have in each $M \cup M'$ -block B in $T - P'$ that $|B \cap S| \leq 2 \log^4 k$. It remains to show that in each $M \cup M'$ -block B in $T - P'$ either B has no S -triangle or B is good. Note that we only refine the partition for bad blocks. So if any $M \cup M'$ -block B is not good in $T - P'$, then it must be obtained by partitioning a bad block but in that case, we brute force over all non-solution S vertices (part of M') and delete all the solution S vertices (part of P'). So, there is no S -triangle inside (in fact, there are no S -vertices at all) if it is not good.

Now $Q \subseteq S$ and $|Q| \leq \frac{4k}{\log^2 k}$ together imply that the number of pair of sets $(Q, \mathcal{Q} \setminus Q)$ (i.e., (M', P')) is at most $(k^2)^{\frac{4k}{\log^2 k}} = 2^{2 \log k \times \frac{4k}{\log^2 k}} = 2^{\mathcal{O}(\frac{k}{\log k})}$. Hence $|\mathcal{F}'| = 2^{\mathcal{O}(\frac{k}{\log k})}$ and this can be computed in $2^{\mathcal{O}(\frac{k}{\log k})}$ time. \blacktriangleleft

Our algorithm for SUBSET-FVST applies Lemma 21 and obtains a family \mathcal{F}' . The algorithm then iterates over each pair $(M', P') \in \mathcal{F}'$ and looks for a S -feedback vertex set H of size at most k that is compatible with $(M \cup M', P')$. In particular, we delete all the vertices of P' and obtain an instance $(T - P', S \setminus P', k - |P'|, M \cup M') = (T, S, k, M)$ such that the following holds.

1. For all $v \in V(T) \setminus M$, we have that $T[M \cup \{v\}]$ does not contain a $\{v\}$ -cycle. In particular, $T[M \cup \{v\}]$ is acyclic.
2. In each M -block B in T , we have $|B \cap S| \leq 2 \log^4 k$.
3. Either each block B is good or it does not have a S -vertex.

We also need the following lemma, which will allow us to show that Lemma 21 is not applied more than once.

► Observation 22. *Let T be a tournament and $M \subseteq M^*$, then every M^* block is a refinement of M block. That is, for each M^* block B^* , there exists a block M block B such that $B^* \subseteq B$.*

This implies that if we add vertices to M to get M^* , then a good M block can be partitioned into several good blocks or if it does not have a S vertex, then none of the resulting blocks after partitioning will have a S vertex.

3.6 Branching Structure II: High Red Degree Again

In the beginning of Section 3.5, we were in the situation where red arcs only formed matching arcs in T . That is, we had (T, S, k, M) which was a matching instance. However, after application of Lemma 21, M -blocks can get partitioned and can have a set of red arcs which share a common vertex. This is possible because some local S -triangle for M -blocks in T can become a shared S -triangle for $(M \cup M')$ -blocks in $T - P'$. Thus, in this scenario, we return to the case of Section 3.4. Therefore, we can use the same branching algorithm as before in Section 3.4. However, because of Observation 22 Lemma 21 is not applied more than once.

If the previous scenario does not happen, this means that the collection of red arcs indeed form matching arcs. In that case, we proceed to Section 3.7.

3.7 Discovering Structure IV: Matching Reduction Rule

Let (T, S, k, M^*) be an instance such that the following holds.

1. For all $v \in V(T) \setminus M^*$, we have that $T[M^* \cup \{v\}]$ does not contain a $\{v\}$ -cycle. In particular, $T[M^* \cup \{v\}]$ is acyclic.
2. In each M^* -block B in T , we have $|B \cap S| \leq 2 \log^4 k$.
3. Either each block B is good or it does not have a S -vertex.
4. Red arcs form a matching in $G = (V(T), R)$.

Now observe that, any M -block of T which contains no local S -triangle still may contain end-points of too many (more than $\log^4 k$) matching arcs. In other words, T can have some bad M -blocks that have no local S -triangles. We address this issue in this section.

► **Definition 23** (inner and outer). Given a directed graph D with an ordered partition $V_1 \cup V_2 \cup \dots \cup V_t$ of $V(D)$, and an integer $i \in [t]$, we define two functions $\mathbf{inner}: \{V_1, V_2, \dots, V_t\} \rightarrow V(D)$ and $\mathbf{outer}: \{V_1, V_2, \dots, V_t\} \rightarrow V(D)$ as follows.

- $\mathbf{inner}(V_i) = \{v : (v, u) \in A(D), v \in V_i, u \in V_j, j \in [i-1]\} \cup \{v : (w, v) \in A(D), v \in V_i, w \in V_{j'}, j' \in [i+1, t]\}$
- $\mathbf{outer}(V_i) = \{u : (v, u) \in A(D), v \in V_i, u \in V_j, j \in [i-1]\} \cup \{w : (w, v) \in A(D), v \in V_i, w \in V_{j'}, j' \in [i+1, t]\}$

Notice that we are in the situation where all the red arcs are matching arcs in the ordered partition of M -blocks in T . Consider the directed graph $D = (V(T), R)$ with partitions accorded by M^* . This leads us to the following observation.

► **Observation 24.** For each M -block B in T we have $|\mathbf{inner}(B)| = |\mathbf{outer}(B)|$.

Now consider a bad block B that does not contain any local S -triangle. We will now prove that for an Yes-instance, there is always a solution of size at most k which is disjoint from $\mathbf{inner}(B)$.

► **Lemma 25.** Let H be a solution of size at most k for an instance (T, S, k, M^*) of SUBSET-FVST and let B be a bad M^* -block in T . Then $H^* := (H \setminus \mathbf{inner}(B)) \cup \mathbf{outer}(B)$ is also a solution for (T, S, k, M^*) of SUBSET-FVST with $|H^*| \leq |H|$.

Proof. By Observation 24, $|H^*| \leq |H|$. It remains to show that H^* is also a solution. In contrary, assume that H^* is not a solution. So, there exists a S -triangle denoted by Δ in $T - H^*$ and the triangle Δ must contain a vertex v from $\mathbf{inner}(B)$ where $V(\Delta)$ is disjoint from $\mathbf{outer}(B)$. Now we have following two cases.

Case 1: Δ is a shared S -triangle. Observe that this case can not happen because this triangle Δ must contain a matching-arc (red arc) whose one end point is $v \in \mathbf{inner}(B)$. Hence the other end point, say u must be in $\mathbf{outer}(B)$ which is also a vertex of the triangle Δ , which is a contradiction to the fact that $\mathbf{outer}(B) \subseteq H$.

Case 2: Δ is a local S -triangle. In this case $V(\Delta) \subseteq B$. That means the block B contains a local S -triangle which implies that B is good (by condition 5 of Lemma 21), which is a contradiction to the assumption that B is a bad M^* -block.

This concludes the proof. ◀

Now we obtain the following reduction rule whose correctness follows from Lemma 25.

► **Reduction Rule 2.** *If B is a bad M^* -block in T then we delete $\text{inner}(B) \cup \text{outer}(B)$ from T . The resultant instance is $(T - \{\text{inner}(B) \cup \text{outer}(B)\}, S \setminus \{\text{inner}(B) \cup \text{outer}(B)\}, k - |\text{outer}(B)|, M^*)$.*

After the exhaustive application of Reduction Rule 2, we have the following observation.

► **Observation 26.** *Every M^* -block in T is good.*

Currently, we are in the situation where every M^* -block in T is good, which means that the number of matching arcs incident to the vertices of every M^* -block is at most $\log^4 k$. In the following section, we will construct an undirected multigraph of bounded degree. This graph will guide us in developing a divide-and-conquer algorithm that achieves our final goal.

3.8 Balanced Edge Partitioning

In this section we deal with the case where every block is good.

Construction of a undirected multigraph. Given (T, S, k, M^*) we construct an undirected multigraph \mathcal{T} as follows:

- Each M^* -block B in T corresponds to a vertex v_B in $V(\mathcal{T})$.
- For every matching-arc (u, v) , $u \in B_i$ and $v \in B_j$ where B_i, B_j are two different M^* -blocks in T , we introduce an edge between v_{B_i} and v_{B_j} in $E(\mathcal{T})$.

At this stage, according to Observation 26, every M -block in T is good. Therefore by definition, the number of matching arcs that are incident to the vertices of every M^* -block is at most $\log^4 k$. This leads us to the next observation.

► **Observation 27.** *The maximum degree of \mathcal{T} is bounded by $\log^4 k$.*

The following fact is known regarding graph partitioning which will be useful for us.

► **Proposition 28** (Theorem 15 [19]). *Given an undirected multigraph without self-loops and isolated vertices G of maximum degree at most d and $|E(G)| = m$, there exists a partition (A, B) of $V(G)$ such that*

- $\frac{m}{4} - \frac{d}{2} \leq |E(G[A])| \leq \frac{m}{4} + \frac{d}{2}$,
- $\frac{m}{4} - \frac{d}{2} \leq |E(G[B])| \leq \frac{m}{4} + \frac{d}{2}$, and
- $\frac{m}{2} - d \leq |E(G[A, B])| \leq \frac{m}{2} + d$.

where $E(G[A, B])$ is the set of edges with one endpoint in A and other in B . Furthermore, there is a polynomial time algorithm to obtain this partition.

Recursive algorithm. We are now ready to describe our recursive algorithm to deal with good M^* -blocks. First, we construct an undirected multigraph \mathcal{T} from (T, S, k, M^*) . We then use Theorem 28 to find a balanced partition of the M^* -blocks with respect to the number of matching-arcs. We then guess which endpoints of these matching-arcs to add to H and solve the two sides independently. Our algorithm starts with an empty set $W \subseteq V(T)$, initialized as \emptyset . If \mathcal{T} is disconnected, then the algorithm solves each connected component independently and outputs W as the union of solutions returned for each component. Now we have following two cases.

Case 1. $|E(\mathcal{T})| = 0$: This case implies that there is no matching-arc in T . In that case, we solve the problem independently in each M^* -block (by brute force) and return W as the union of solutions returned for each M -block.

Case 2. $|E(\mathcal{T})| > 0$: In this case, first we run the algorithm of Proposition 28 to get a partition (A, B) of the multigraph \mathcal{T} . For a vertex set $X \subseteq V(\mathcal{T})$ and edge set $Y \subseteq E(\mathcal{T})$, the set $V_X(Y)$ denotes the set of all the vertices in T that are incident on the matching-arcs corresponding to Y and belong to M^* -blocks in X . Our algorithm loops over all subsets $C \subseteq E(A, B)$, calling itself recursively on $\mathcal{T}(V(T) \setminus (V_A(C) \cup V_B(E(A, B) \setminus C)))$ (the multigraph corresponding to the subgraph of T without $V_A(C)$ and $V_B(E(A, B) \setminus C)$) and computes $W_C := V_A(C) \cup V_B(E(A, B) \setminus C) \cup S'$ where S' is the set returned at the recursive call. Finally, the algorithm outputs the smallest set W_C over all choices of $C \subseteq E(A, B)$.

3.9 Correctness and Running Time

The correctness of the algorithm follows from each individual pieces we have proved. For the running time observe the following steps:

1. Apply kernelization algorithm in polynomial time. (Lemma 4)
2. Obtain a universal undeletable family \mathcal{M} of size $2^{\mathcal{O}(\frac{k}{\log k})}$. (Lemma 9).
3. Obtain a du-family \mathcal{F} of size $2^{\mathcal{O}(\frac{k}{\log k})}$. (Lemma 12).
4. Let $G = (V(T), R)$ be an undirected graph with the vertex set $V(T)$ and the edge set R (without orientation). Branch on vertices of red-degree at least 2 in G . This leads to (1, 2) branching and corresponds to $\mathcal{O}(1.6181^k \times k^{\mathcal{O}(1)})$.
5. Making each block B good or it does not have a S -vertex by branching in $2^{\mathcal{O}(\frac{k}{\log k})}$ ways (followed by (1, 2) branching). (Lemma 21).
6. Getting rid of blocks that do not contain any S -vertex in $k^{\mathcal{O}(1)}$ time. (Lemma 25)
7. All blocks are good. In this situation, we do divide and conquer based on Proposition 28. Now we proceed to the runtime analysis of this step of the algorithm. When $|E(\mathcal{T})| = 0$, we do brute force on the instance and that takes $k^{\mathcal{O}(\log^4 k)} = 2^{\mathcal{O}(\log^5 k)}$ time. Because when $|E(\mathcal{T})| = 0$, in each connected component the number of S -vertices gets bounded by $2 \log^4 k$ (Lemma 12), so the size of an optimal solution in each component is bounded by $2 \log^4 k$. Let $h(k, d)$ be the maximum number of leaves in the recursion tree of the algorithm when run on an input with parameter k and maximum degree d . Since in each recursive call, k decreases by at least 1, the depth of the recursion tree is at most k . In each internal node of the recursion tree, the algorithm spends $k^{\mathcal{O}(1)}$ time plus constant time for each child (which are at most $h(k, d)$) and in each leaf, it spends at most $2^{\mathcal{O}(\log^5 k)}$ time (as in leaf $|E(\mathcal{T})| = 0$). Thus, the running time of the algorithm on any input with parameters k and d is upper bounded by $h(k, d) \times 2^{\mathcal{O}(\log^5 k)} \times k^{\mathcal{O}(1)}$. To upper bound $h(k, d)$, first note that $h(a, d) + h(b, d) \leq h(a + b, d)$ because $h(a, d)$ and $h(b, d)$ represent the number of leaves of two independent sub-trees. Now for each $C \subseteq E(X, Y)$ there are no edges between A and B in $\mathcal{T}(V(T) \setminus (V_A(C) \cup V_B(E(A, B) \setminus C)))$. Hence, the algorithm effectively solves $\mathcal{T}(V(T) \setminus (V_A(C)))$ and $\mathcal{T}(V(T) \setminus (V_B(E(A, B) \setminus C)))$ independently. By Proposition 28 and since we know that we need to hit all edges in \mathcal{T} , the number of edges in $E(A, B)$ is at most $\frac{k}{4} + \frac{d}{2}$. As we have seen for each C , the algorithm calls itself twice for each C and so in total, the algorithm makes $2^{\frac{k}{2} + d + 1}$ recursive calls with parameter $\frac{k}{4} + \frac{d}{2}$ (and d does not increase). Thus $h(k, d)$ is upper-bounded by the recurrence relation $h(k, d) \leq 2^{\frac{k}{2} + d + 1} h(\frac{k}{4} + \frac{d}{2}, d)$ and $h(k, 0) = 2^{\mathcal{O}(\log^5 k)} \leq 2^{\mathcal{O}(\log^5 k)}$. This solves to $h(k, d) = 1.5874^k \cdot 2^{\mathcal{O}(d \log^5 k)}$. Hence, the running time of the algorithm is bounded by $h(k, d) \times 2^{\mathcal{O}(\log^5 k)} \times k^{\mathcal{O}(1)} = 1.5874^k \cdot 2^{\mathcal{O}(d \log^5 k)} \cdot 2^{\mathcal{O}(\log^5 k)} \cdot k^{\mathcal{O}(1)}$. As $d \leq \log^4 k$, the running time is bounded by $1.5874^k \cdot 2^{\mathcal{O}(\log^9 k)} \cdot k^{\mathcal{O}(1)}$.
8. Taking into account running time in each step we get that the algorithm runs in time $\mathcal{O}(1.6181^k + n^{\mathcal{O}(1)})$. This concludes the proof of Theorem 1.

References

- 1 Faisal N. Abu-Khzam. A kernelization algorithm for d -hitting set. *J. Comput. Syst. Sci.*, 76(7):524–531, 2010. doi:10.1016/J.JCSS.2009.09.002.
- 2 Tian Bai and Mingyu Xiao. A parameterized algorithm for subset feedback vertex set in tournaments. *Theor. Comput. Sci.*, 975:114139, 2023. doi:10.1016/J.TCS.2023.114139.
- 3 Stéphane Bessy, Marin Bougeret, Dimitrios M Thilikos, and Sebastian Wiederrecht. Kernelization for graph packing problems via rainbow matching. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3654–3663. SIAM, 2023. doi:10.1137/1.9781611977554.CH139.
- 4 Jianer Chen, Yang Liu, Songjian Lu, Barry O’Sullivan, and Igor Razgon. A fixed-parameter algorithm for the directed feedback vertex set problem. *J. ACM*, 55(5):21:1–21:19, 2008. doi:10.1145/1411509.1411511.
- 5 Rajesh Hemant Chitnis, Marek Cygan, Mohammad Taghi Hajiaghayi, and Dániel Marx. Directed subset feedback vertex set is fixed-parameter tractable. *ACM Trans. Algorithms*, 11(4):28:1–28:28, 2015. doi:10.1145/2700209.
- 6 Marek Cygan, Marcin Pilipczuk, Michal Pilipczuk, and Jakub Onufry Wojtaszczyk. Subset feedback vertex set is fixed-parameter tractable. *SIAM J. Discret. Math.*, 27(1):290–309, 2013. doi:10.1137/110843071.
- 7 Michael Dom, Jiong Guo, Falk Hüffner, Rolf Niedermeier, and Anke Truß. Fixed-parameter tractability results for feedback set problems in tournaments. *J. Discrete Algorithms*, 8(1):76–86, 2010. doi:10.1016/J.JDA.2009.08.001.
- 8 Guy Even, Joseph Naor, and Leonid Zosin. An 8-approximation algorithm for the subset feedback vertex set problem. *SIAM J. Comput.*, 30(4):1231–1252, 2000. doi:10.1137/S0097539798340047.
- 9 Fedor V. Fomin, Tien-Nam Le, Daniel Lokshtanov, Saket Saurabh, Stéphan Thomassé, and Meirav Zehavi. Subquadratic kernels for implicit 3-hitting set and 3-set packing problems. *ACM Trans. Algorithms*, 15(1):13:1–13:44, 2019. doi:10.1145/3293466.
- 10 Fedor V Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: theory of parameterized preprocessing*. Cambridge University Press, 2019.
- 11 Jiong Guo and Rolf Niedermeier. Invitation to data reduction and problem kernelization. *SIGACT News*, 38(1):31–45, 2007. doi:10.1145/1233481.1233493.
- 12 Sushmita Gupta, Sounak Modak, Saket Saurabh, and Sanjay Seetharaman. Quick-sort style approximation algorithms for generalizations of feedback vertex set in tournaments. In José A. Soto and Andreas Wiese, editors, *LATIN 2024: Theoretical Informatics - 16th Latin American Symposium, Puerto Varas, Chile, March 18-22, 2024, Proceedings, Part I*, volume 14578 of *Lecture Notes in Computer Science*, pages 225–240. Springer, 2024. doi:10.1007/978-3-031-55598-5_15.
- 13 Eva-Maria C. Hols and Stefan Kratsch. A randomized polynomial kernel for subset feedback vertex set. *Theory Comput. Syst.*, 62(1):63–92, 2018. doi:10.1007/S00224-017-9805-6.
- 14 Yoichi Iwata and Yusuke Kobayashi. Improved analysis of highest-degree branching for feedback vertex set. *Algorithmica*, 83(8):2503–2520, 2021. doi:10.1007/S00453-021-00815-W.
- 15 Yoichi Iwata, Magnus Wahlström, and Yuichi Yoshida. Half-integrality, lp-branching, and FPT algorithms. *SIAM J. Comput.*, 45(4):1377–1411, 2016. doi:10.1137/140962838.
- 16 Yoichi Iwata, Yutaro Yamaguchi, and Yuichi Yoshida. 0/1/all csps, half-integral a -path packing, and linear-time FPT algorithms. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 462–473. IEEE Computer Society, 2018. doi:10.1109/FOCS.2018.00051.
- 17 Satyabrata Jana, Daniel Lokshtanov, Soumen Mandal, Ashutosh Rai, and Saket Saurabh. Parameterized approximation scheme for feedback vertex set. In Jérôme Leroux, Sylvain Lombardy, and David Peleg, editors, *48th International Symposium on Mathematical Foundations of Computer Science, MFCS 2023, August 28 to September 1, 2023, Bordeaux, France*, volume 272 of *LIPICs*, pages 56:1–56:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICs.MFCS.2023.56.

- 18 Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972. doi:10.1007/978-1-4684-2001-2_9.
- 19 Mithilesh Kumar and Daniel Lokshтанov. Faster exact and parameterized algorithm for feedback vertex set in tournaments. In Nicolas Ollinger and Heribert Vollmer, editors, *33rd Symposium on Theoretical Aspects of Computer Science, STACS 2016, February 17-20, 2016, Orléans, France*, volume 47 of *LIPICs*, pages 49:1–49:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPICs.STACS.2016.49.
- 20 Jason Li and Jesper Nederlof. Detecting feedback vertex sets of size k in $O^*(2.7k)$ time. *ACM Trans. Algorithms*, 18(4):34:1–34:26, 2022. doi:10.1145/3504027.
- 21 Rolf Niedermeier and Peter Rossmanith. An efficient fixed-parameter algorithm for 3-hitting set. *Journal of Discrete Algorithms*, 1(1):89–102, 2003. doi:10.1016/S1570-8667(03)00009-1.
- 22 Geevarghese Philip, Varun Rajan, Saket Saurabh, and Prafullkumar Tale. Subset feedback vertex set in chordal and split graphs. *Algorithmica*, 81(9):3586–3629, 2019. doi:10.1007/S00453-019-00590-9.
- 23 Magnus Wahlström. *Algorithms, measures and upper bounds for satisfiability and related problems*. PhD thesis, Linköping University, Sweden, 2007. URL: <https://nbn-resolving.org/urn:nbn:se:liu:diva-8714>.