Exploiting Spanning Trees for Directed Acyclicity

Sergei Khargeliia
ITMO University, St. Petersburg, Russian Federation
Saint Petersburg State University, Russian Federation

Danil Sagunov
Markov Lab, Saint Petersburg State University, Russian Federation

Abstract

We study the weighted case of the Maximum Acyclic Subgraph (MAS) problem, where each edge of a given directed graph has a positive weight assigned, and the task is to find a maximum-weight acyclic edge set. The famous and well-studied random ordering lower bound guarantees the existence of an acyclic set that gives at least the half of the total edge weight.

The maximum spanning tree (MaxST) guarantee, which is the weight of a maximum-weight acyclic subgraph of the underlying undirected graph of G, is another natural lower bound for the weight of an acyclic subgraph. A solution of this weight dominates the random ordering solution on instances where MaxST spans the most of the total edge weight.

Our main contribution are two parameterized algorithms that find acyclic subgraphs of total weight larger than the weight of the MaxST of G. Both our algorithms find a solution of total weight at least $\mathrm{MaxST}(G) + k$, for a given integer $k \geq 0$, or report that it does not exist, and

- First of our algorithms runs in time $2^{k^{\mathcal{O}(1)}} \cdot |\mathcal{I}|^{\mathcal{O}(1)}$ and works when all weights are integers;
- Our second algorithm handles rational weights not less than 1, and its running time is upper-bounded by $n^{k^{\mathcal{O}(1)}} \cdot |\mathcal{I}|^{\mathcal{O}(1)}$. This positive result is rather surprising since solving MAS above the random ordering lower bound is NP-hard in the same rational weights scenario, when $k = 1$.

Our findings unravel intricate connections between structure of MaxSTs and directed cycles, use perfect graph theorem to tackle rational weights, and raise graph-theoretic questions that are interesting on their own. Of another importance, this is one of the few examples of positive "above guarantee" results for a weighted problem on directed graphs, especially for rational weights.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases parameterized algorithms, spanning tree, above guarantee parameterizations, perfect graph theorem, feedback arc set, maximum acyclic subgraph

Digital Object Identifier 10.4230/LIPIcs.ESA.2026.118

Related Version *Full Version*: `https://arxiv.org/abs/2607.07705` [17]

Funding *Sergei Khargeliia*: Supported by PJSC «Gazprom Neft», aggr. # ГПН-26/03000/00502/P. *Danil Sagunov*: Supported by the Ministry of Science and Higher Education of the Russian Federation (agreement 075-15-2025-344 dated 29/04/2025 for Saint Petersburg Leonhard Euler International Mathematical Institute at PDMI RAS).

1 Introduction

In the Maximum Acyclic Subgraph problem, MAS for short, we are given a directed graph (digraph) G with n vertices and m edges, and an integer $k \geq 0$, and the task is to find an acyclic subgraph of G that contains at least k edges. In the dual version of the problem, alternatively, one can ask to delete at most $k' = m - k$ edges from G. This formulation is well-known as Feedback Arc Set and comes from the seminal work of Karp [16]. In this paper, we study the weighted version of MAS, here we discuss the unweighted case first.

One can easily see that any digraph contains an acyclic subgraph with at least $m/2$ edges. To construct such a subgraph in polynomial time, take an arbitrary ordering of $V(G)$, and take all edges that go from the left to the right into solution. If there are less than $m/2$ such edges, take all edges that go from the right to the left in the solution instead.

This is known as the *random ordering* lower bound and is quite significant to MAS from the perspective of approximation algorithms. Obviously, it provides a polynomial-time 2-approximation algorithm for MAS. In [12], Guruswami, Manokaran and Raghavendra proved that this is tight and no better approximation ratios for MAS are possible, a *"first tight inapproximability result for an ordering problem"*, under the Unique Games Conjecture [18]. One year before that, Charikar, Makarychev and Makarychev [2] designed a polynomial-time algorithm that finds an acyclic subgraph with at least $(\frac{1}{2}+\Omega(\delta/\log n))\cdot m$ edges in any given n-vertex graph that admits an acyclic subgraph with at least $(\frac{1}{2}+\delta)\cdot m$ edges, for any $\delta > 0$.

The same lower bound was the original starting point of a quite successful concept in parameterized complexity. It was initiated from the search for an efficient algorithm that finds an acyclic subgraph with at least $m/2+k$ edges. The question of an existence of such an algorithm was posed as an open problem by Raman and Saurabh in [24], who obtained partial results. Later, Mahajan, Raman and Sikdar [21] addressed this question again among other similar ones. Importantly, [21] is the first paper that addresses parameterizations above (or below) tight lower (or upper) bounds as a general concept, now known as *above-guarantee* or *below-guarantee* parameterizations (see, e.g. a survey by Gutin and Mnich [14] on this topic). Finally, in [13], Gutin, Kim, Szeider and Yeo gave a positive answer to the question. They showed that the problem of finding an acyclic set of at least $m/2+k$ edges admits a $2^{\mathcal{O}(k\log k)}\cdot n^{\mathcal{O}(1)}$-running-time fixed-parameter tractable (FPT) algorithm.

Another tight lower bound for MAS was considered independently by Mnich, Philip, Saurabh and Suchý [22] and by Crowston, Gutin and Jones [5]. It comes from the work of Poljak and Turzík from 1986 [23], and guarantees that if G is an oriented[1] weakly-connected[2] graph, then G has an acyclic subgraph with at least $m/2+(n-1)/4$ edges. In [22, 5] it was shown that the problem of finding an acyclic subgraph with at least $m/2+(n-1)/4+k$ edges in a given oriented graph is fixed-parameter tractable.

Later, Etscheid and Mnich [10] showed that a variety of problems (with MAS and Max Cut among them) admit linear kernels, when parameterized above the Poljak–Turzík bound. In particular, they showed a kernel with $\mathcal{O}(k)$ vertices for MAS above Poljak–Turzík, and improved the running time to $2^{\mathcal{O}(k)}\cdot n^{\mathcal{O}(1)}$. Whereas the Poljak–Turzík bound is applicable only to oriented graphs, this parameterization is superior to the parameterization above $m/2$. Finding an acyclic subgraph with at least $m/2+k$ edges in an arbitrary digraph can be simply reduced to the same problem on an oriented graph.

There is one particular lower bound for MAS that has been overlooked in this line of research. This lower bound is the *spanning tree* guarantee, and it speaks for itself: any weakly-connected digraph admits an acyclic graph with at least $n-1$ edges. The spanning tree bound is also (algorithmically) inferior to the Poljak–Turzík bound.

The situation with these three guarantees changes slightly when we introduce edge weights in G. The algorithm of [13] for MAS above random ordering works in the multigraph setting when we are allowed to have several copies of the same edge in G. This can alternatively

[1] G is directed, and each pair of vertices is connected by at most one directed edge, going in either of the two directions, but not both simultaneously.

[2] G is connected if we remove edge orientations.

be seen as a weighted scenario. Instead of having several copies of the same edge, we keep G a simple digraph, and we assign a positive integer weight $\mathbf{w}(e)$ to each edge $e \in E(G)$. Then the random ordering guarantee is at least the half of the total edge weight, $\mathbf{w}(E(G))/2$. Within this context, the mentioned algorithm of [13] finds an acyclic subgraph of total edge weight at least $\mathbf{w}(E(G))/2 + k$ in $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$ running time.

The Poljak–Turzík lower bound was originally formulated in [23] for the weighted case, and it essentially guarantees that an edge-weighted *oriented* graph G contains an acyclic subgraph of weight at least $\mathbf{w}(E(G))/2 + \mathbf{w}(\mathrm{MinST}(G, \mathbf{w}))/4$, where $\mathrm{MinST}(G, \mathbf{w})$ is the *minimum* spanning tree of the underlying undirected graph of G (or its minimum spanning forest, if G is not weakly-connected). In [10], the authors posed an open question on whether the efficient FPT-algorithms above Poljak–Turzík can be generalized to the weighted case.

But are these two lower bounds that good when the edge weights are not evenly distributed? If, for example, the graph has just one heavy edge, whose weight is greater than the sum of all other weights, then this edge alone is better than any of the two guarantees can provide.

The spanning tree guarantee becomes the *maximum* spanning tree (MaxST) guarantee in the weighted setting. Clearly, G admits an acyclic subgraph of weight at least $\mathrm{MaxST}(G, \mathbf{w})$, a weight of the maximum spanning tree of the underlying undirected graph of G. This lower bound captures the case in the paragraph above and is better on instances where edge weights are concentrated around a sparse structure. For instance, it dominates both the random ordering and the Poljak–Turzík guarantees on instances where $\mathrm{MaxST}(G, \mathbf{w})$ is at least $\frac{3}{4} \cdot \mathbf{w}(E(G))$. The MaxST lower bound is the central guarantee for MAS in our work. We give a formal definition to the corresponding parameterized problem below.

Maximum Weight Acyclic Subgraph Above MaxST (MAS/MaxST)

Input:	Weakly-connected n-vertex digraph G with integer weight function $\mathbf{w} : E(G) \to \mathbb{Z}_{\geq 1}$ assigned to its edges, and integer $k \geq 0$.
Task:	Find an acyclic subgraph G' of G such that $\mathbf{w}(E(G'))$ is at least $\mathbf{w}(\mathrm{MaxST}(G, \mathbf{w})) + k$, or report that G' does not exist.

This parameterization differs significantly from the Poljak–Turzík one. First, we do not have to enforce that G is oriented for our guarantee to work. The restriction on the weak connectivity of G can also be lifted, because G can always be made weakly-connected by adding bridges between connected components of G. Second, these bounds are generally incomparable since they behave differently on sparse and dense graphs.

Our contribution. Our first main result is an FPT-algorithm for MAS/MaxST. It is obtained by a novel approach that combines several constructive ideas and precise analysis of the edges outside the maximum spanning tree. Integrality of the weight function is crucial, since the edges of MaxST are classified according to their *profits* – the amount of extra weight we could possibly get from edges outside of MaxST if we remove this edge of MaxST from the solution. Since all profits are integral, and large profits guarantee the solution, we are able to achieve only $k^{\mathcal{O}(1)}$ equivalence classes, and we are interested only in the "most suitable" edges in each class. We then perform branching on which edges should be deleted from the MaxST.

▶ **Theorem 1.** *MAS/MaxST($\mathbb{Z}_{\geq 1}$) admits an algorithm with $2^{k^{\mathcal{O}(1)}} \cdot |\mathcal{I}|^{\mathcal{O}(1)}$ running time.*

As an example, we can use Theorem 1 to find an (unweighted) acyclic subgraph that contains two prescribed edges e_1, e_2 (that are not opposites) and at least $(n-3)+k$ other edges. This can be extended to more than two edges, but only if they form an undirected forest. Note that the random ordering or Poljak–Turzík guarantees cannot provide us such use since they could only ensure that the half of the prescribed edges belong to the solution.

Theorem 1 works only in a restricted setting of integer weights. What can we do for the general case when weights are allowed to be rational numbers? In this setting, the choice of the parameter k is not very obvious. If we allow arbitrary rational edge weights in the definition of MAS/MaxST, then we can reduce an arbitrary weakly-connected instance of (unweighted) MAS to an instance of MAS/MaxST with $k = 1$. Hence it is NP-hard for $k = 1$.

To overcome this trivial obstacle and capture the actual parameterized complexity, we don't allow k to be too large compared to edge weights. We keep the original definition of MAS/MaxST and restrict weights to be rationals *not less than one*, and denote this version as MAS/MaxST($\mathbb{Q}_{\geq 1}$). Our second main contribution is an XP-algorithm for this problem.

▶ **Theorem 2.** *MAS/MaxST($\mathbb{Q}_{\geq 1}$) admits an algorithm with $n^{k^{\mathcal{O}(1)}} \cdot |\mathcal{I}|^{\mathcal{O}(1)}$ running time.*

To prove Theorem 2, we use some of the structural profit-related insights that are used to prove Theorem 1. Main obstacle when transitioning from integer to rational weights is that edge weights differences are not integers anymore, and can be arbitrarily close to 0. This raises two new challenges that we have to deal with, even to obtain an XP-algorithm. First, the profits of the edges of MaxST could be very small and even to obtain just "+1" excess weight, we could have to remove a lot of them. Second, the profits are not integers and hence cannot be simply split into $f(k)$ equivalence classes by their profit. We resolve these challenges by providing two new structural insights (alternative to "profit classification" used for the integral case) that focus on the structure of directed paths of the MaxST instead.

Related work. MAS and Feedback Arc Set are known for notorious hardness from the exact exponential algorithms perspective. For both weighted and unweighted versions of these problems, the best known exact algorithms are still the ones that run in $2^n \cdot n^{\mathcal{O}(1)}$ time. On the positive side, Kim, Kratsch, Pilipczuk and Wahlström in their recent work [19] introduced the *flow augmentation* technique. They showed that, if k is the number of edges that we are allowed to remove, then Weighted Feedback Arc Set is solvable in $2^{k^{\mathcal{O}(1)}} \cdot |\mathcal{I}|^{\mathcal{O}(1)}$ time. We use this algorithm in the last step of our proof of Theorem 1.

While the above-guarantee approach is now well developed in parameterized complexity, relatively few works study weighted guarantees specifically. One of the few is the work by Gutin and Patel [15], which investigates FPT-algorithms for finding weighted Hamiltonian cycles below the average cycle weight in complete undirected graphs with integer edge weights. The authors leave the complexity for complete digraphs as an open question. There is also a related line of works [6, 4, 3] on weighted CSPs for linear systems over $\mathbb{F}_2$, parameterized above or below guarantees, where the parameter k is also the excess weight.

Organization of the paper. In the next section, we introduce the notation used throughout the paper and provide some preliminary results. In Section 3, we introduce definitions and properties of edges around MaxST$(G, \mathbf{w})$ that are fundamental to our approach. In Section 4, we show the main milestones of our proof of Theorem 1, supporting them with necessary intuition and discussion. In Section 5, we give an overview of the proof of Theorem 2 in the same way. Formal proofs of most of the results are omitted and can be found in the full version of this paper [17]. We mark such results with " $\star$ ".

2 Preliminaries

Notation. We use standard graph terminology and notation (we refer the reader to the book of Diestel [8]). We also use several additional notions. We use $G - S$ to denote a graph obtained by deleting all edges of S from G. We write $G + S$ to denote a graph obtained by adding all edges of S to G. We refer to both undirected and directed edges as just *edges*, the actual variant is always clear from the context.

For a graph $(G, \mathbf{w})$ with edge weights, we always assume that $\mathbf{w}$ is additively extended to all subsets of edges. That is, for each $S \subseteq E(G)$, we have $\mathbf{w}(S) = \sum_{e \in S} \mathbf{w}(e)$. For a subgraph $X \subseteq G$, we also shorten the notation $\mathbf{w}(E(X))$ to just $\mathbf{w}(X)$.

For the comprehensive introduction to parameterized algorithms and terminology of parameterized complexity, we refer the reader to the Parameterized Algorithms book [7].

Partially ordered sets. A *partially ordered set* (poset for short) is an ordered pair $\mathcal{P} = (X, \preceq)$ where X is a set and $\preceq$ is a partial order on X. A relation $\preceq^*$ is called a *linear extension* of $\mathcal{P}$ if it is a total order on X that is compatible with $\preceq$, that is, for all $x, y \in X$, $x \preceq y$ implies $x \preceq^* y$.

The *order dimension* of a poset $\mathcal{P} = (X, \preceq)$ is the least integer t for which there exists a family $(\preceq_1, \preceq_2, \ldots, \preceq_t)$ of linear extensions of $\mathcal{P}$ with the following property: for each $x, y \in X$, it holds that $x \preceq y$ if and only if $x \preceq_i y$ for all i. This concept was introduced by Dushnik and Miller in [9]. We will use the following result on the order dimension.

▶ **Proposition 3** ([25], [1]). *Let T be an oriented tree, and let $\mathcal{P} = (V(T), \preceq)$ be a poset such that for each $u, v \in V(T)$, $u \preceq v$ if and only if there exists a directed path from u to v in T. Then the order dimension of $\mathcal{P}$ is at most three, and the corresponding linear extensions $(\preceq_1, \preceq_2, \preceq_3)$ of $\mathcal{P}$ can be constructed in polynomial time.*

Trotter and Moore [25] proved that the order dimension of such posets is at most three, but they did not explicitly state the existence of a polynomial-time algorithm that constructs the corresponding linear extensions. Although such an algorithm follows from their proof, we refer to a recent work of Abram and Segovia for a more direct construction (see Corollary 6.3 in [1]).

3 Classification of directed edges and their basic properties

In this section, we introduce the basic toolbox that we use extensively throughout our proof. Our tools are mostly related to classification of edges of G relative to its MaxST and several useful properties surrounding this classification for *arbitrary positive edge weights*.

3.1 Maximum spanning tree, allowed and blocked edges

We start with properly defining the "guaranteed" subgraph of G. We informally refer to it as a maximum spanning tree, or MaxST for short.

▶ **Definition 4** (Maximum spanning tree). *Let G be a weakly-connected n-vertex digraph with an edge set $\{e_1, \ldots, e_m\}$ and edge weights $\mathbf{w}$. By $\mathrm{MaxST}(G, \mathbf{w})$ we denote a subgraph T of G, such that:*

- *$|V(T)| = n$ and $|E(T)| = n - 1$;*
- *T is weakly-connected;*
- *$\mathbf{w}(E(T))$ is maximum possible;*

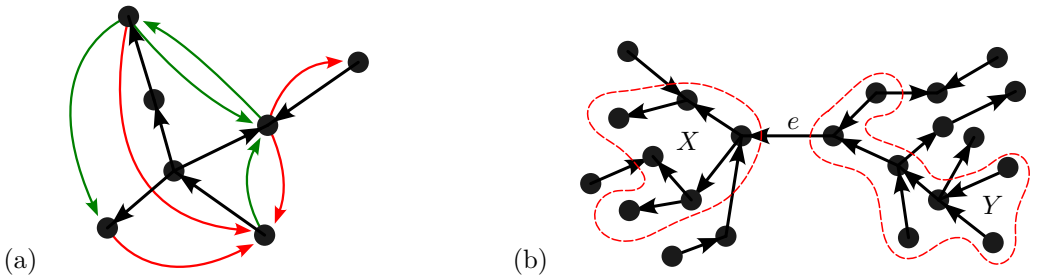

Figure 1 (a) An instance with six tree edges (thick black), four allowed edges (green) and four blocked edges (red). (b) $\mathrm{Inv}(e)$ is the set of all edges that start in X (vertices reachable from e in T) and end in Y (vertices e is reachable from in T).

- *a sorted sequence of indices of edges from the set $E(T)$ is lexicographically smallest possible among all subgraphs satisfying the previous constraints.*

Throughout the paper, we use T as a shortcut for $\mathrm{MaxST}(G, \mathbf{w})$.

Note that in the definition above, we put the last property in order that T is defined unambiguously. Moreover, T can still be found in polynomial time via the standard argument: we first find the maximum weight w of a spanning tree of G. Then we construct T iteratively, starting with the empty subgraph. In each iteration, we find an edge e_i of minimum index such that $T + e_i$ can be extended to a spanning tree of G of weight w, and add this e_i to T.

Naturally, some edges outside $E(T)$ do not form a cycle, if added to T. For example, if $T + e$ is acyclic for $e \in E(G) \setminus E(T)$, then $(G, \mathbf{w}, k)$ is a yes-instance of MAS/MaxST for any $k \leq \mathbf{w}(e)$. We proceed with a proper formal definition that includes the concept of such edges.

▶ **Definition 5** (Classification of graph edges relative to MaxST). *For a weakly-connected digraph G with edge weights $\mathbf{w}$, we say that*

- *Each edge $e \in E(T)$ is a tree edge.*
- *Each edge $e \in E(G) \setminus E(T)$ such that $T + e$ has no directed cycles is an **allowed** edge.*
- *Each edge $e \in E(G) \setminus E(T)$ such that $T + e$ has a directed cycle is a **blocked** edge.*

By $A(G, \mathbf{w})$ we denote the set of all allowed edges of G and by $B(G, \mathbf{w})$ we denote the set of all blocked edges of G. We shorten this notation to just A and B, unless the arguments are different from $(G, \mathbf{w})$.

The main challenge in our two algorithms is to handle both edge types simultaneously. In the rest of this section, we focus on blocked edges.

3.2 Inverse edges and their properties

Note that removing an edge from T can "unblock" some edges in B. That is, if b is blocked, and $e \in E(T)$ belongs to a directed cycle in $T + b$, then $(T - e) + b$ is acyclic. In this situation, we say that b is an *inverse edge* of e. We give a formal definition below.

▶ **Definition 6** (Inverse edges). *For a weakly-connected digraph with positive edge weights $(G, \mathbf{w})$ and a tree edge $e \in E(T)$, by $\mathrm{Inv}(G, \mathbf{w}, e) = \{b \in B : T - e + b \text{ is acyclic}\}$ we denote the set of all* inverse *edges of e in G.*

For a tree edge set $S \subseteq E(T)$, by $\mathrm{Inv}(G, \mathbf{w}, S) = \bigcup_{e \in S} \mathrm{Inv}(G, \mathbf{w}, e)$ we denote the union of all inverse edges among all edges in S. We shorten the notation to just $\mathrm{Inv}(e)$ and $\mathrm{Inv}(S)$.

We start by observing basic properties of inverse edges that follow from their definition.

▶ **Observation 7** (⋆). *Let $b \in B$ be a blocked edge that goes from u to v. Then T contains a directed path P from v to u. Moreover, for a tree edge e, an edge $b \in \mathrm{Inv}(e)$ iff e lies on P.*

We also note that if $X \subseteq G$ is acyclic and S is the set of all tree edges of G outside $E(X)$, then X can contain only blocked edges from the set $\mathrm{Inv}(S)$.

▶ **Observation 8** (⋆). *If $X \subseteq G$ is acyclic and $S = E(T) \setminus E(X)$, then $X \subseteq T + A - S + \mathrm{Inv}(S)$.*

Finally, we observe that blocked and inverse edges have to follow specific directions derived from a cut of T (see Figure 1 (b)).

▶ **Observation 9** (⋆). *Let $e \in E(T)$ and let X, Y be the vertex sets of two weakly-connected components of $T - e$ labeled so that e goes from X to Y in G. Then*

1. *For each $b \in B$, b does not go from X to Y in G;*
2. *For each $b \in \mathrm{Inv}(e)$, b goes from Y to X in G.*

The most crucial property of inverse edges is that, once unblocked by deleting some edges in T, we can add them to the remaining graph together without forming any directed cycle.

▶ **Lemma 10** (⋆). *Let $(G, \mathbf{w})$ be a weakly-connected digraph with positive edge weights. Let $S \subseteq E(T)$ be its tree edge set. Then, the graph $T - S + \mathrm{Inv}(S)$ is acyclic.*

3.3 Tree edge profits

The strategy that comes naturally from Lemma 10 can informally be described as trading tree edges for the union of all their inverse edges. The profit of this trade is a difference between total weights of tree edges and their inverse edge union. We formalize this intuition.

▶ **Definition 11** (Tree edge profits). *Let $(G, \mathbf{w})$ be a weakly-connected digraph with positive edge weights. For a tree edge $e \in E(T)$, by*

$$\mathbf{p}(G, \mathbf{w}, e) = \mathbf{w}(\mathrm{Inv}(e)) - \mathbf{w}(e)$$

we denote the profit *of e in $(G, \mathbf{w})$. For a tree edge set $S \subseteq E(T)$, by*

$$\mathbf{p}(G, \mathbf{w}, S) = \mathbf{w}(\mathrm{Inv}(S)) - \mathbf{w}(S)$$

we denote the profit *of S in $(G, \mathbf{w})$. As usual, we use shortened notation $\mathbf{p}(e)$ and $\mathbf{p}(S)$.*

We observe some straightforward properties of edge profits below.

▶ **Observation 12.** *For a weakly-connected digraph with positive edge weights $(G, \mathbf{w})$, and tree edge set $S \subseteq E(T)$, and an integer $k \geq 0$, we have that*

- *If $\mathbf{p}(S) \geq k$, then $(G, \mathbf{w}, k)$ is a yes-instance of MAS/MaxST.*
- $\mathbf{p}(S) \leq \sum_{s \in S} \mathbf{p}(s)$.
- *If for each distinct $e_1, e_2 \in S$ we have $\mathrm{Inv}(e_1) \cap \mathrm{Inv}(e_2) = \emptyset$, then $\mathbf{p}(S) = \sum_{e \in S} \mathbf{p}(e)$.*

Throughout our algorithms for MAS/MaxST, we assume that the profit of each single tree edge is at most k. Otherwise, we have a yes-instance that is recognizable in polynomial time.

3.4 Remaining profits

While the basic concept of profits clearly suggests to look for a profitable enough tree edge set, it does not really allow us to construct a such a tree set iteratively.

Assume, for example, that $A = \emptyset$ and that we guess correctly that an edge $e_1 \in E(T)$ is not a part of the solution. We know that if we remove e_1 from T and add all its inverse edges, we gain $\mathbf{p}(e_1)$ profit from this. Afterwards, we move on to guessing the second edge e_2 that should be removed from T. But the profits are not "up-to-date" anymore, since it might be the case that $\mathrm{Inv}(e_1) \cap \mathrm{Inv}(e_2) \neq \emptyset$, and we gain less profit than $\mathbf{p}(e_2)$ from removing e_2.

We introduce the concept of *remaining profit*, which is aimed to support the additive edge removal and represent the actual profit when some edges are already removed from T.

▶ **Definition 13** (Remaining profits). *Let $(G, \mathbf{w})$ be a weakly-connected digraph with positive edge weights. For a tree edge $e \in E(T)$, and for a tree edge set $S \subseteq E(T)$, we denote by*

$$\mathbf{rp}(G, \mathbf{w}, S, e) = \mathbf{w}(\mathrm{Inv}(e) \setminus \mathrm{Inv}(S)) - \mathbf{w}(e)$$

the remaining profit *of e in $(G, \mathbf{w})$ with respect to S. As usual, we shorten this to $\mathbf{rp}(S, e)$.*

The following properties of remaining profits are straightforward from the definition.

▶ **Observation 14.** *Let $(G, \mathbf{w})$ be a weakly-connected digraph with positive edge weights, let $e \in E(T)$, let $S \subseteq E(T)$ and let $e_1, e_2, \ldots, e_q \in E(T)$ such that $e_i \neq e_j$ for $i \neq j$. Then*

- $\mathbf{rp}(S, e) \leq \mathbf{p}(e)$.
- $\mathbf{p}(\{e_1, \ldots, e_q\}) = \sum_{i=1}^{q} \mathbf{rp}(\bigcup_{j=1}^{i-1} \{e_j\}, e_i)$.

In the following observation, we highlight that picking k edges from $E(T)$ consecutively, each time with positive remaining profit, is a winning strategy for MAS/MaxST.

▶ **Observation 15** (⋆). *Let $\mathcal{I} = (G, \mathbf{w}, k)$ be an instance of MAS/MaxST. If there exist $e_1, e_2, \ldots, e_k \in E(T)$ with $\mathbf{rp}(\bigcup_{j=1}^{i-1} \{e_j\}, e_i) \geq 1$ for each $i \in [k]$, then $\mathcal{I}$ is a yes-instance.*

For $A = \emptyset$, solving an instance is equivalent to finding a set $S \subseteq E(T)$ with $\mathbf{p}(S) \geq k$.

4 FPT-algorithm for integral edge weights

In this section, we highlight and formulate the four major blocks of the proof of Theorem 1. Each block corresponds to a lemma, which we formulate and explain (by giving some intuition, formal definitions and informal discussion) in this section.

4.1 Limiting allowed edges

In Section 3 we gave a lot of insight on blocked edges. Allowed edges, if given in instance without inverse edges, are somewhat simpler to deal with, even for rational weights. We are able to show an algorithm that constructs a solution if their total weight is at least $3k$.

▶ **Lemma 16.** *Let $\mathcal{I} = (G, \mathbf{w}, k)$ be an instance of MAS/MaxST($\mathbb{Q}_{\geq 1}$). If $\mathbf{w}(A) \geq 3k$, then $\mathcal{I}$ is a yes-instance. The solution to $\mathcal{I}$ can be constructed in polynomial time.*

Proof. We first apply Proposition 3 to the maximum spanning tree T of $(G, \mathbf{w})$. Let $\mathcal{P} = (V(T), \preceq)$ be the poset defined as in the proposition statement, that is, $u \preceq v$ if and only if there is a directed path in T from u to v. Then, in polynomial time, we can construct linear extensions $\preceq_1$, $\preceq_2$ and $\preceq_3$ of $\mathcal{P}$ such that for each $x, y \in V(T)$, $x \preceq y$ if and only if $x \preceq_i y$ for each $i \in \{1, 2, 3\}$.

We now provide a partition $A_1 \sqcup A_2 \sqcup A_3$ of A such that $T + A_i$ is acyclic for each $i \in \{1, 2, 3\}$. For each edge $uv \in A$, we determine its part as follows. By the definition of allowed edges, $T + uv$ does not contain directed cycles. In particular, this means that there is no directed path from v to u in T. Hence, $v \preceq u$ does not hold. Then, by the definition of $(\preceq_1, \preceq_2, \preceq_3)$, we know that u must precede v in one of these total orders. We choose an arbitrary $\preceq_i$ with this property and add uv to the set A_i. Note that the construction of A_1, A_2 and A_3 also takes polynomial time.

We claim that for each $i \in \{1, 2, 3\}$, all edges from $E(T) \cup A_i$ go from smaller to larger elements with respect to $\preceq_i$. For the edges from A_i, this property is ensured by the definition of A_i. Now consider an edge $uv \in E(T)$. This edge gives a trivial path from u to v in T, and thus we have $u \preceq v$, implying $u \preceq_i v$. Because uv is a tree edge, it cannot be a loop, and hence u strictly precedes v in $\preceq_i$.

It follows that for each $i \in \{1, 2, 3\}$, the graph $T + A_i$ is acyclic. Let A_j be a set of maximum weight among A_1, A_2 and A_3. If $\mathbf{w}(A) \geq 3k$, then $\mathbf{w}(A_j) \geq k$, and hence $\mathbf{w}(T + A_j) \geq \mathbf{w}(T) + k$. Therefore, $T + A_j$ is a solution to $\mathcal{I}$, and it can be constructed in polynomial time. ◀

It is an interesting open problem to determine whether the bound in Lemma 16 is tight or can be improved.

4.2 Decomposing MST into paths

Our next block is a polynomial-time algorithm that either constructs a solution, or obtains an equivalent instance of MAS/MaxST($\mathbb{Z}_{\geq 1}$) via edge contractions. In the resulting instance, all but at most $\mathcal{O}(k + |A|)$ vertices have exactly one ingoing and exactly one outgoing edge.

▶ **Lemma 17** (⋆). *There is a polynomial-time algorithm, that, given an instance $\mathcal{I} = (G, \mathbf{w}, k)$ of MAS/MaxST($\mathbb{Z}_{\geq 1}$), either*

- *outputs a solution to $\mathcal{I}$, or*
- *outputs an equivalent instance $\mathcal{I}' = (G', \mathbf{w}', k)$ of MAS/MaxST($\mathbb{Z}_{\geq 1}$) such that in* MaxST$(G', \mathbf{w}')$ *all but at most $\mathcal{O}(k + |A'|)$ vertices have both in- and out-degree one, where $A' = A(G', \mathbf{w}')$.*

Additionally, a solution to $\mathcal{I}'$ can be transformed into a solution to $\mathcal{I}$ in polynomial time.

The proof of Lemma 17 involves several sophisticated ideas. One of the difficulties arises when handling long chains of vertices of degree two. While they form a single *undirected* path, reducing the number of directed paths they could form becomes challenging.

Lemma 17 does not provide any upper bound on the size of G' in $\mathcal{I}'$. Nevertheless, the MaxST of $(G', \mathbf{w}')$ exhibits a simple structure, since it consists of $\mathcal{O}(k + |A|)$ pivotal vertices connected by pairwise-disjoint directed paths.

4.3 Greedy-like approach to remaining profits

Our third building block relies fundamentally on remaining profits and directed paths in MaxST. Somewhat surprisingly, there is a strategy that provides a limited number of choices of removing a single edge from $E(T)$ iteratively, depending on its current remaining profit, given that T is partitioned into a limited number of directed paths (guaranteed by Lemma 17).

This strategy works even if A is not empty, and it highly relies on that $E(T)$ is ordered in a topological manner. To describe it, we have to introduce the notion of *proper path covers*.

▶ **Definition 18** (Proper path cover). *A* proper path cover $\mathcal{P}$ *of* $T = \mathrm{MaxST}(G, \mathbf{w})$ *is a sequence* $P_1, P_2, \ldots, P_t$ *of directed paths of* T*, such that*

- *each edge* $e \in E(T)$ *belongs to exactly one path in* T*, and*
- *each internal vertex of each* P_i *has in-degree and out-degree one in* T*, and*
- *if* $v \in V(G)$ *is incident to an allowed edge* $a \in A$*, then* v *is not an internal vertex in any* P_i*, and*
- *if there is a directed path in* T*, that starts with* $s \in E(T)$ *and ends with* $e \in E(T)$*, then* $i \leq j$*, where* $i, j \in [t]$ *are such that* $e \in P_i$ *and* $s \in P_j$*.*

Note that the last point in the definition above is a topological-like ordering of paths required for our strategy. We also have to introduce the notion of edge orderings that agree with proper path covers. Order the edges inside the paths, so the last edge goes first, and the first edge goes last. We call the resulting total edge ordering the $\mathcal{P}$*-respecting ordering*. The formal definition is given below.

▶ **Definition 19** ($\mathcal{P}$-respecting orderings). *Let* $(G, \mathbf{w})$ *be a weakly-connected digraph with positive edge weights, and let* $\mathcal{P}$ *be a proper path cover of* G*. We say that an ordering* $e_1, e_2, \ldots, e_{n-1}$ *of* $E(T)$ *is* $\mathcal{P}$-respecting *if*

- *if there is a directed path in* T *that starts with* e_j *and ends with* e_i*, then* $i \leq j$ *holds, and*
- *for each* $1 \leq i \leq j < n$*, it holds that* $\ell \leq \ell'$*, where* $e_i \in P_\ell$ *and* $e_j \in P_{\ell'}$*.*

The definition of $\mathcal{P}$-respecting orderings implies that edges of a single path of $\mathcal{P}$ form consecutive segments in the ordering. Such a segment starts with the last edge of a path and ends with a first edge of a path. We are now ready to formulate the lemma itself.

▶ **Lemma 20** ($\star$). *Let* $\mathcal{I} = (G, \mathbf{w}, k)$ *be an instance of MAS/MaxST*$(\mathbb{Z}_{\geq 1})$*, and let* $\mathcal{P}$ *be a proper path cover of* T*. Let* $e_1, e_2, \ldots, e_{n-1}$ *be a* $\mathcal{P}$*-respecting ordering of* $E(T)$*. If* $\mathcal{I}$ *is a yes-instance, then there exists a solution* G' *to* $\mathcal{I}$ *with* $S = E(T) \setminus E(G')$ *such that*

- *If* $e_i \in S$*, then* $\mathbf{rp}(S_{i-1}, e_i) \geq -\mathbf{w}(A)$*, and*
- *If* $e_i, e_j \in S$ *for* $i < j$*, and* e_i, e_j *belong to the same path in* $\mathcal{P}$*, then* $\mathbf{rp}(S_{j-1}, e_j) > 0$*, and*
- *If* $e_i \notin S$ *and* $e_j \in S$ *for* $i < j$*,* $S_{j-1} = S_{i-1}$ *and* e_i, e_j *belong to the same path in* $\mathcal{P}$*, then* $\mathbf{rp}(S_{i-1}, e_i) < \mathbf{rp}(S_{i-1}, e_j)$*,*

where $S_0 = \emptyset$ *and* $S_i = S_{i-1} \cup (S \cap \{e_i\})$ *for each* $i \in [n-1]$*.*

Let us demonstrate the power of Lemma 20. First, it follows that there is a solution to $\mathcal{I}$, where only at most $|\mathcal{P}|$ tree edges with non-positive remaining profits are removed iteratively. Second, Observation 15 guarantees that k positive remaining profits are enough to obtain a solution. Therefore, it is enough to consider edge subsets $S \subseteq E(T)$ of size at most $|\mathcal{P}| + k$.

On the other hand, Lemma 20 gives us a limited number of choices of what edge to remove next. Indeed, when we have to remove an edge, we know that its remaining profit is in $[-\mathbf{w}(A), k]$. The third point of Lemma 20 guarantees that we can always choose the earliest (with respect to the ordering) edge that belongs to P_ℓ with the value of remaining profit equal to p, for small integer values of p and ℓ.

4.4 Solution when the removed tree edges are fixed

It comes naturally from Lemma 20 that we have to solve the following special version of the problem. We have a fixed edge set $S \subseteq E(T)$ of bounded size, and we have to find a solution G' to $(G, \mathbf{w}, k)$ that contains all edges in $E(T - S)$ and contains no edge of S. G' cannot contain any blocked edge outside $\mathrm{Inv}(S)$, so it is a subgraph of $H = (T - S) + (\mathrm{Inv}(S) \cup A)$.

Since the case $\mathbf{p}(S) \geq k$ is a solution to $(G, \mathbf{w}, k)$, the total edge weight of H is bounded with $\mathbf{w}(E(T)) + k + \mathbf{w}(A)$. Therefore, we have to delete at most $\mathbf{w}(A)$ edges from H to obtain the solution. Armed up with this idea, we formulate our last major algorithmic lemma.

▶ **Lemma 21** (⋆). *There is an algorithm with the running time $2^{(\mathbf{w}(A))^{\mathcal{O}(1)}} \cdot n^{\mathcal{O}(1)}$ that, given an instance $\mathcal{I} = (G, \mathbf{w}, k)$ of MAS/MaxST($\mathbb{Z}_{\geq 1}$), and given a tree edge set $S \subseteq E(T)$, either*

- *reports that $\mathcal{I}$ is a no-instance, if $\mathcal{I}$ is a no-instance, or*
- *outputs a solution to $\mathcal{I}$, if there is a solution G' to $\mathcal{I}$ such that $E(T) \setminus E(G') = S$, or*
- *outputs any of the two outcomes above, otherwise.*

Note that the algorithm from Lemma 21 can report false-negatives, if the choice of S is incorrect, but always outputs a solution if the choice of S is correct. Therefore, we can run the algorithm of Lemma 21 with many choices of S, and we will get a solution if at least one of these choices was correct.

In the proof of Lemma 21, we use the FPT-algorithm for Weighted Feedback Arc Set based on flow augmentation [19].

4.5 Putting all together

To prove our first main result, Theorem 1, we pipeline the lemmas into an algorithm that solves MAS/MaxST($\mathbb{Z}_{\geq 1}$) in $2^{k^{\mathcal{O}(1)}} \cdot n^{\mathcal{O}(1)}$ running time. The core internal subroutine of this algorithm works with a proper path cover and outputs a solution in FPT-time (see Alg. 1).

Algorithm 1 The recursive subroutine `extend` of the algorithm of Theorem 1. It takes (as an input) an instance $\mathcal{I}$, a proper path cover $\mathcal{P}$ of T, and a $\mathcal{P}$-respecting ordering $e_1, \ldots, e_n$ of $E(T)$. `extend`($\mathcal{I}$, $\mathcal{P}$, $e_1, \ldots, e_n$, $\emptyset$, 0) returns Yes iff $\mathcal{I}$ is a yes-instance.

```
1  function extend(I, P = (P_1, P_2, ..., P_t), e_1, e_2, ..., e_{n-1}, S, i)
2      if |S| ≥ k + t then
3          return Yes
4      if Lemma 21 applied to I and S returns Yes then
5          return Yes
6      if i ≥ n − 1 then
7          return No
8      if i = 0 then
9          ℓ ← 0
10     else
11         ℓ ← integer in [t] such that e_i ∈ P_ℓ
12     if i > 0 then                                   // next edge is in the same path
13         foreach p ∈ [1, k] do                // it is the earliest edge with rem. profit p
14             j ← smallest integer in [i + 1, n − 1] with e_j ∈ P_ℓ and rp(S, e_j) = p, or n;
15             if extend (I, P, e_1, e_2, ..., e_{n-1}, S ∪ {e_j}, j) is Yes then
16                 return Yes
17     foreach ℓ' ∈ [ℓ + 1, t] do                  // next edge is in another path P_ℓ'
18         foreach p ∈ [−w(A), k] do                       // it has remaining profit p
19             j ← smallest integer in [i + 1, n − 1] with e_j ∈ P_ℓ' and rp(S, e_j) = p, or n;
20             if extend (I, P, e_1, e_2, ..., e_{n-1}, S ∪ {e_j}, j) is Yes then
21                 return Yes
22     return No;
```

5 XP-algorithm for rational edge weights

This section is dedicated to the proof of Theorem 2: given a graph $(G, \mathbf{w})$ with rational edge weights not less than one, an acyclic subgraph of G of weight at least $\mathbf{w}(T) + k$ can be found in time $n^{k^{\mathcal{O}(1)}} \cdot |\mathcal{I}|^{\mathcal{O}(1)}$.

We highlight that with rational edge weights, the problem becomes non-trivial even when $k = 1$ and $A = \emptyset$. In this case, we need to check whether a set $S \subseteq E(T)$ with $\mathbf{p}(S) \geq 1$ exists. Every such S must contain an element with positive profit. Hence, for integer weights this implies that $\mathbf{p}(s) \geq 1$ for some $s \in S$, and we can consider only single-element sets. In contrast, when edge weights are rational, each $\mathbf{p}(s)$ may be arbitrarily close to zero. Thus, to reach $\mathbf{p}(S) \geq 1$, we may now need to combine many small profits. Consequently, there are no longer any natural bounds on the size of S, which significantly increases the difficulty of the problem. Nevertheless, an XP algorithm must handle the case $k = 1$ in polynomial time.

We overcome this obstacle in Section 5.1. Then, in Section 5.2, we address the additional challenges arising from the presence of allowed edges in G.

5.1 Maximizing profit under restrictions

A natural way to decide whether some set reaches a target profit is to find a set of maximum profit. It turns out that if the elements are restricted to not share inverse edges, this can be done efficiently. This subproblem is a key building block of our XP algorithm.

For a digraph $(G, \mathbf{w})$, let $H_{G,\mathbf{w}}$ be the undirected graph on $E(T)$ where each vertex e has weight $\mathbf{p}(e)$ and an edge connects e_1 and e_2 iff $\mathrm{Inv}(e_1) \cap \mathrm{Inv}(e_2) \neq \emptyset$. Recall that if the sets $\mathrm{Inv}(s)$ for $s \in S$ are pairwise disjoint, then the profit is additive. Thus, maximizing the profit of S under this restriction is equivalent to finding the maximum-weight independent set in $H_{G,\mathbf{w}}$, which is NP-hard on general graphs. Fortunately, $H_{G,\mathbf{w}}$ has a special structure.

▶ **Lemma 22** (⋆). *For each edge-weighted digraph $(G, \mathbf{w})$, $H_{G,\mathbf{w}}$ is a perfect graph.*

As shown by Grötschel, Lovász, and Schrijver in [11], one can find a maximum-weight independent set in a perfect graph with positive integer vertex weights in polynomial time. We remark that this result can be extended to rational weights of an arbitrary sign.

▶ **Proposition 23** (⋆) ([11], Section 6). *For a perfect graph G with rational vertex weights, a maximum-weight independent set can be found in time polynomial in $|V(G)| + \log C$, where C is the maximum among the numerators and denominators of the weights.*

Even when certain tree edges are forbidden from S (set F), others are required to be in S (set R), and a set of blocked edges $B' \subseteq \mathrm{Inv}(R)$ may be the inverse of multiple edges in S, the maximum-profit S can still be found in polynomial time.

▶ **Lemma 24** (⋆). *There is a polynomial-time algorithm that given a digraph $(G, \mathbf{w})$ with rational edge weights, sets $F, R \subseteq E(T)$ and $B' \subseteq \mathrm{Inv}(R)$, finds a set with the maximum $\mathbf{p}(S)$ over all $S \subseteq E(T)$ such that:*

- *$F \cap S = \emptyset$ and $R \subseteq S$;*
- *for every pair of distinct elements $s_1, s_2 \in S$, it holds that $\mathrm{Inv}(s_1) \cap \mathrm{Inv}(s_2) \subseteq B'$.*

Observe that no single triple (F, R, B') allows S to have an arbitrary structure. Intuitively, if B' is small, we forbid the elements of S from having strongly overlapping sets of inverse edges. Conversely, if B' is large, then each of its elements has a corresponding edge in $R \subseteq S$, and hence any feasible S must be fairly large as well. Therefore, this lemma alone is not sufficient to solve even instances with $A = \emptyset$. The following result completes the picture.

▶ **Lemma 25** (⋆). *If $\mathcal{I} = (G, \mathbf{w}, k)$ is a yes-instance of MAS/MaxST($\mathbb{Q}_{\geq 1}$), then there exists a solution X to $\mathcal{I}$ with $S = E(T) \setminus E(X)$ such that the weight of*

$$B' = \bigcup_{\substack{s_1, s_2 \in S \\ s_1 \neq s_2}} (\mathrm{Inv}(s_1) \cap \mathrm{Inv}(s_2))$$

is at most $(\mathbf{w}(A) + k)^2$. Additionally, if $\mathcal{I}$ admits a solution of the form $T - S + \mathrm{Inv}(S) + A'$ for some $S \subseteq E(T)$ and $A' \subseteq A$, then X can be required to have the same form.

In combination, Lemma 24 and Lemma 25 supply us with a powerful tool. For example, an instance $(G, \mathbf{w}, k)$ with $A = \emptyset$ can now be solved in time $n^{\mathcal{O}(k^2)}$ as follows. Enumerate all $B' \subseteq B$ of weight at most $(\mathbf{w}(A) + k)^2 = k^2$. For each B', consider all sets R formed by picking, for every $b \in B'$, one tree edge that breaks the cycle in $T + b$. For every such pair (B', R), apply the algorithm from Lemma 24 with the given B', R, and $F = \emptyset$, and obtain a maximum-profit set S under these restrictions. If a solution exists, some such S will yield $\mathbf{p}(S) \geq k$.

5.2 Dealing with allowed edges

When allowed edges come into play, we encounter a further complication: the structure of acyclic subgraphs no longer admits the same clean characterization as before. Given a set of removed tree edges S, the blocked edges of any acyclic subgraph are still confined to $\mathrm{Inv}(S)$. However, once the subgraph includes allowed edges, we lose the guarantee that the whole $\mathrm{Inv}(S)$ can be taken without producing cycles. In this section, we unravel this difficulty.

Our approach, however, will differ significantly from the one used in the integral case, so let us first compare two versions of the problem (integral- and rational-weighted) in the context of allowed edges and discuss why the methods we developed for our FPT-algorithm for integral weights cannot be directly applied to rational weights.

Recall that, at a high level, our FPT-algorithm consisted of two parts: (i) finding a small family of tree edge sets that are candidates for removal, and (ii) for a fixed such set, reducing the problem to Weighted Directed Feedback Arc Set. In this scheme, allowed edges posed no major issue: we simply retained all of them in the reduced instance and let the black-box algorithm for WDFAS deal with them.

Although the second part of the scheme above can be easily modified to work with rational weights, the first part's building blocks (Lemma 17 and Lemma 20) rely heavily on the integrality of profits. Since we could not come up with any analogues of these lemmas for rational weights, we had to abandon the entire scheme and deal with allowed edges by other means.

On the positive side, we note that by Lemma 16, the total weight of allowed edges is still bounded by $3k$.

5.2.1 Inv-respecting subgraphs

As a starting point, we aim to mimic the setting without allowed edges. Although subgraphs of the form $T - S + \mathrm{Inv}(S) + A'$ with $A' \subseteq A$ are not necessarily acyclic, it remains tempting to search for solutions of this kind, because their weight equals $\mathbf{w}(T) + \mathbf{p}(S) + \mathbf{w}(A')$. This would enable us to reuse the ideas tied to edge profits. We call such subgraphs Inv-*respecting*, and extend this terminology to instances that admit a solution of this structure.

Fortunately, we can always seek a solution that is almost Inv-respecting, in the sense that the total weight of the inverse edges it discards is at most $\mathbf{w}(A)$.

▷ Claim 26 (⋆). If $\mathcal{I} = (G, \mathbf{w}, k)$ is a yes-instance of MAS/MaxST($\mathbb{Q}_{\geq 1}$), there exists a solution $X \subseteq G$ to $\mathcal{I}$ with $S = E(T) \setminus E(X)$ such that $\mathbf{w}(\mathrm{Inv}(S) \setminus E(X)) \leq \mathbf{w}(A)$.

This allows us to reduce an arbitrary instance to an Inv-respecting one in time $m^{\mathbf{w}(A)}$: we simply guess which inverse edges are discarded by a solution and remove them from the graph. If the guess is correct, the resulting instance admits an Inv-respecting solution.

5.2.2 Compressed representation of acyclic subgraphs

Note that an Inv-respecting solution is defined by two edge sets: S and A'. Moreover, due to Lemma 16, the number of allowed edges is at most $3k$. Therefore, we can enumerate all $A' \subseteq A$, and for each choice, search for a suitable set S. To ensure that $T - S + \mathrm{Inv}(S) + A'$ is acyclic for a fixed A', we rely on the following alternative characterization.

Let V' be the set of vertices incident to A'. Since $X = T - S + \mathrm{Inv}(S)$ is acyclic, any cycle in $X + A'$ must intersect V'. Furthermore, such a cycle can be decomposed into directed paths in X and edges from A', where the endpoints of each fragment lie in V'. Consequently, instead of working with $X + A'$ directly, we may analyze the compressed graph on V' whose edges represent directed paths in X and the edges of A'. We formalize this idea as follows.

▷ Claim 27 (⋆). Let $X \subseteq T + B$ be an acyclic subgraph. For $A' \subseteq A$, denote by V' the set of vertices incident to A'. Then $X + A'$ is acyclic if and only if there exists a set $\overline{P} \subseteq V' \times V'$ such that:

- for each $(u, v) \in \overline{P}$, there is no directed path from u to v in X;
- let G' be the directed graph on V' where for two distinct $u, v \in V'$, we have $(u, v) \in E(G')$ if $(u, v) \in A'$ or $(u, v) \notin \overline{P}$. Then G' is acyclic.

Therefore, we can ensure the acyclicity of $T - S + \mathrm{Inv}(S) + A'$ by choosing $\overline{P}$ that makes G' acyclic and then prohibiting directed paths in $T - S + \mathrm{Inv}(S)$ for every $(u, v) \in \overline{P}$.

5.2.3 Forbidding paths via constraint sets

Surprisingly, subgraphs of the form $T - S + \mathrm{Inv}(S)$ that do not contain a given directed path can be described by a small collection of the same restrictions as used in Lemma 24. We call a triple (F, R, B') from the statement of that lemma a *restriction triple*, and any set of such triples a *constraint set*. A subgraph X, with $S = E(T) \setminus E(X)$, satisfies (F, R, B') if the lemma conditions hold for S, and a constraint set if it satisfies at least one of its triples.

Recall that each blocked edge corresponds to a path in T (see Observation 7). In our approach, for every edge from B', we will need to identify the first and the last edge of the corresponding path that are absent from the subgraph. We call a restriction triple *strict* if its sets F and R uniquely determine these edges. Formally, (F, R, B') is called strict if for each $(u, v) \in B'$ there exist two edges $e_1, e_2 \in R$ (not necessarily distinct) such that:

- $(u, v) \in \mathrm{Inv}(e_1) \cap \mathrm{Inv}(e_2)$;
- every edge of the path in T from v to the start of e_1 belongs to F;
- every edge of the path in T from the end of e_2 to u belongs to F.

If a subgraph $X = T - S + \mathrm{Inv}(S)$ satisfies such a strict triple, then $e_1, e_2 \in R \subseteq S$, so these two edges are absent from X. Conversely, edges of the path before e_1 and after e_2 must belong to X because they are present in F and $S \cap F = \emptyset$.

Given a strict restriction triple, we can filter the family of subgraphs $T - S + \mathrm{Inv}(S)$ satisfying it: a constraint set, constructed in the next lemma, keeps exactly those subgraphs that avoid the forbidden path. This is by far the most difficult part of our XP-algorithm.

▶ **Lemma 28** (⋆). *Let $(G, \mathbf{w})$ be a digraph with edge weights. For every $u, v \in V(G)$ and every strict restriction triple (F_0, R_0, B'_0), there exists a constraint set $\mathcal{C}$ such that:*

- *For every $S \subseteq E(T)$, let $X = T - S + \mathrm{Inv}(S)$. If X satisfies $\mathcal{C}$, then X contains no directed path from u to v. Conversely, if X satisfies (F_0, R_0, B'_0) and contains no directed path from u to v, then X satisfies $\mathcal{C}$.*
- *For each $(F, R, B') \in \mathcal{C}$, it holds that $B' = B'_0$, $F_0 \subseteq F$, $R_0 \subseteq R$, and each of the sets $F \setminus F_0$ and $R \setminus R_0$ can be represented as a union of at most $|B'_0| + 1$ directed paths in T.*

Finally, we combine this section's results into the main tool for handling allowed edges.

▶ **Lemma 29** (⋆). *If $\mathcal{I} = (G, \mathbf{w}, k)$ is an* Inv*-respecting instance of MAS/MaxST$(\mathbb{Q}_{\geq 1})$, then there exists a family of constraint sets $\{\mathcal{C}_{A'} \mid A' \subseteq A\}$ with the following properties:*

- *for each $S \subseteq E(T)$ and $A' \subseteq A$, if $T - S + \mathrm{Inv}(S) + A'$ satisfies $\mathcal{C}_{A'}$, then it is acyclic;*
- *there exist $S \subseteq E(T)$ and $A' \subseteq A$ such that $T - S + \mathrm{Inv}(S) + A'$ is a solution to $\mathcal{I}$ that satisfies $\mathcal{C}_{A'}$;*
- *for each triple $(F, R, B') \in \mathcal{C}_{A'}$, it holds that $\mathbf{w}(B') \leq (\mathbf{w}(A) + k)^2$, and each of the sets F and R can be represented as a union of at most $4(\mathbf{w}(A) + k + 1)^4$ directed paths in T.*

Although the proofs of the two preceding lemmas are constructive, the final algorithm does not need to build the corresponding constraint sets. Instead, because the sets B', F and R enjoy the special structure, we can enumerate all triples with the required properties. In particular, this enumeration covers every element of the constraint set.

5.3 Summing up

We present a sketch of the XP algorithm below. The formal analysis of correctness and running time is deferred to the appendix.

Proof sketch of Theorem 2. Let $\mathcal{I} = (G, \mathbf{w}, k)$ be an instance of MAS/MaxST$(\mathbb{Q}_{\geq 1})$. If $\mathbf{w}(A) \geq 3k$, Lemma 16 resolves $\mathcal{I}$ in polynomial time. From now on, we assume $\mathbf{w}(A) < 3k$.

We iterate over all choices of a set $D \subseteq B$ of weight at most $\mathbf{w}(A)$, a set $B' \subseteq B \setminus D$ of weight at most $(\mathbf{w}(A) + k)^2$, and sets $P_F, P_R \subseteq V \times V$, each of size at most $4(\mathbf{w}(A) + k + 1)^4$. Additionally, we require that for every (u, v) from $P_F \cup P_R$, T contains a directed path from u to v (we denote the edge set of such a path by $E_{u,v}$). For a fixed choice of D, B', P_F and P_R, we begin by constructing the sets $F = \bigcup_{(u,v) \in P_F} E_{u,v}$ and $R = \bigcup_{(u,v) \in P_R} E_{u,v}$.

Let $G' = G - D$, and let $\mathbf{w}' = \mathbf{w}|_{E(G')}$. Since D consists only of blocked edges, we have $\mathrm{MaxST}(G', \mathbf{w}') = T$. We check whether $B' \subseteq \mathrm{Inv}(G', \mathbf{w}', R)$, and if so, apply the algorithm from Lemma 24 to $(G', \mathbf{w}')$ and the triple (F, R, B'), which returns a set $S \subseteq E(T)$. For each $A' \subseteq A$, we check whether the subgraph $T - S + \mathrm{Inv}(S) + A'$ is a solution to $\mathcal{I}$. If no check succeeds over all choices of D, B', P_F, P_R, and A', we report that $\mathcal{I}$ is a no-instance. ◀

6 Conclusion

We conclude with several open questions and directions for further research. First of all, the parameterized complexity of MAS/MaxST remains unsettled, when we deal with rational weights.

▶ **Open Question 1.** *Is MAS/MaxST$(\mathbb{Q}_{\geq 1})$* W[1]*-hard or fixed-parameter tractable?*

Another standalone question concerns Lemma 16, and it might be of independent interest. Lemma 16 in fact shows that a solution of weight at least $\mathbf{w}(T) + \lceil \mathbf{w}(A)/3 \rceil$ is guaranteed for integer weights and can be found in polynomial time, using known constructions of linear

orderings for oriented trees. Can we improve the denominator 3 in this guarantee using more specific constructions? If yes, can we find the required subset of allowed edges in polynomial time? If no, is there a no-instance with integer weights and $\mathbf{w}(A) = 3k - 3$?

▶ **Open Question 2.** *Can the constant* 3 *in Lemma 16 be improved?*

Our positive results for the MaxST guarantee suggest that Maximum Acyclic Subgraph above the Poljak-Turzík guarantee could also admit efficient algorithms. We re-formulate the corresponding open question, which was already posed in [10] for integral weights and for a broader class of problems.

▶ **Open Question 3.** *Can we find an acyclic subgraph of weight at least*

$$\mathbf{w}(G)/2 + \mathbf{w}(\mathrm{MinST}(G, \mathbf{w}))/4 + k$$

in a directed graph G *with positive integer edge weights* $\mathbf{w} : E(G) \to \mathbb{Z}_{\geq 1}$ *in* FPT*-time?*

We note that for the Maximum Cut problem, the Poljak–Turzík guarantee also applies, and in [20] Lill, Petrova and Weber have shown that a cut of weight at least $\mathbf{w}(E(G))/2 + \mathbf{w}(\mathrm{MinST}(G, \mathbf{w}))/4 + k$ can be found in $2^{\mathcal{O}(k)} \cdot |G|$ time, if G has integer edge weights. Similarly, the weight of the maximum spanning tree is also a viable lower bound for the maximum cut weight in G. This gives rise to the following natural open problem.

▶ **Open Question 4.** *Can we find a cut of weight at least* $\mathbf{w}(\mathrm{MaxST}(G, \mathbf{w})) + k$ *in an undirected graph* G *with positive integer edge weights* $\mathbf{w} : E(G) \to \mathbb{Z}_{\geq 1}$ *in* FPT*-time?*

For the last two open questions, the weights could as well be replaced with rationals not less than one.

One can also study the existence of efficient kernelization algorithms. The main result of [10] demonstrates that many problems parameterized above the Poljak–Turzík bound admit linear kernels, in the unweighted (oriented graph) case. In [20], the authors show that Maximum Cut parameterized above Poljak–Turzík is in FPT in the case of integral weights. Can we improve this result and prove that it admits polynomial kernels? Can we do the same for Theorem 1 and show that MAS/MaxST($\mathbb{Z}_{\geq 1}$) admits polynomial kernels? We summarize this direction in the following question.

▶ **Open Question 5.** *Do polynomial kernels exist for any relevant weighted problem parameterized above the Poljak–Turzík or the maximum spanning tree guarantees?*

Obtaining a negative answer to this question for some weighted parameterized problem, that, on the other hand, admits an FPT-algorithm, would also be very interesting.

References

1 Antoine Abram and Adrien Segovia. Dimension of unicycle posets, 2026. arXiv:2505.16837.

2 Moses Charikar, Konstantin Makarychev, and Yury Makarychev. On the Advantage over Random for Maximum Acyclic Subgraph. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 625–633, 2007. doi:10.1109/FOCS.2007.65.

3 R. Crowston, G. Gutin, M. Jones, and A. Yeo. Parameterized complexity of satisfying almost all linear equations over $\mathbb{F}_2$. *Theor. Comp. Sys.*, 52(4):719–728, May 2013. doi:10.1007/s00224-012-9415-2.

4 Robert Crowston, Michael Fellows, Gregory Gutin, Mark Jones, Frances Rosamond, Stéphan Thomassé, and Anders Yeo. Simultaneously Satisfying Linear Equations Over F2: MaxLin2 and Max-r-Lin2 Parameterized Above Average. In Supratik Chakraborty and Amit Kumar, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2011)*, volume 13 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 229–240, Dagstuhl, Germany, 2011. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.FSTTCS.2011.229.

5 Robert Crowston, Gregory Gutin, and Mark Jones. Directed Acyclic Subgraph Problem Parameterized above the Poljak-Turzik Bound. In Deepak D'Souza, Jaikumar Radhakrishnan, and Kavitha Telikepalli, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2012)*, volume 18 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 400–411, Dagstuhl, Germany, 2012. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.FSTTCS.2012.400.

6 Robert Crowston, Gregory Gutin, Mark Jones, Eun Jung Kim, and Imre Z. Ruzsa. *Systems of Linear Equations over $\mathbb{F}_2$ and Problems Parameterized above Average*, pages 164–175. Springer Berlin Heidelberg, 2010. doi:10.1007/978-3-642-13731-0_17.

7 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

8 Reinhard Diestel. *Graph theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, 5th edition, 2017.

9 Ben Dushnik and E. W. Miller. Partially ordered sets. *American Journal of Mathematics*, 63(3):600–610, 1941. doi:10.2307/2371374.

10 Michael Etscheid and Matthias Mnich. Linear kernels and linear-time algorithms for finding large cuts. *Algorithmica*, 80(9):2574–2615, 2018. doi:10.1007/S00453-017-0388-Z.

11 M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, June 1981. doi:10.1007/BF02579273.

12 Venkatesan Guruswami, Rajsekar Manokaran, and Prasad Raghavendra. Beating the Random Ordering is Hard: Inapproximability of Maximum Acyclic Subgraph. In *Proceedings of the 2008 49th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '08, pages 573–582, USA, 2008. IEEE Computer Society. doi:10.1109/FOCS.2008.51.

13 Gregory Gutin, Eun Jung Kim, Stefan Szeider, and Anders Yeo. A probabilistic approach to problems parameterized above or below tight bounds. *Journal of Computer and System Sciences*, 77(2):422–429, 2011. doi:10.1016/J.JCSS.2010.06.001.

14 Gregory Gutin and Matthias Mnich. A survey on graph problems parameterized above and below guaranteed values. *Computer Science Review*, 58:100795, 2025. doi:10.1016/j.cosrev.2025.100795.

15 Gregory Z. Gutin and Viresh Patel. Parameterized Traveling Salesman Problem: Beating the Average. *SIAM J. Discret. Math.*, 30(1):220–238, 2016. doi:10.1137/140980946.

16 Richard M Karp. On the computational complexity of combinatorial problems. *Networks*, 5(1):45–68, 1975. doi:10.1002/NET.1975.5.1.45.

17 Sergei Khargeliia and Danil Sagunov. Exploiting spanning trees for directed acyclicity, 2026. arXiv:2607.07705.

18 Subhash Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing*, STOC '02, pages 767–775, New York, NY, USA, 2002. Association for Computing Machinery. doi:10.1145/509907.510017.

19 Eun Jung Kim, Stefan Kratsch, Marcin Pilipczuk, and Magnus Wahlström. Flow-augmentation i: Directed graphs. *J. ACM*, 72(1), January 2025. doi:10.1145/3706103.

20 Jonas Lill, Kalina Petrova, and Simon Weber. Linear-time maxcut in multigraphs parameterized above the poljak-turzík bound. *Algorithmica*, 87(7):983–1007, April 2025. doi:10.1007/s00453-025-01306-y.

21 Meena Mahajan, Venkatesh Raman, and Somnath Sikdar. Parameterizing above or below guaranteed values. *Journal of Computer and System Sciences*, 75(2):137–153, 2009. doi:10.1016/J.JCSS.2008.08.004.

22 Matthias Mnich, Geevarghese Philip, Saket Saurabh, and Ondřej Suchý. Beyond max-cut: λ-extendible properties parameterized above the poljak–turzík bound. *Journal of Computer and System Sciences*, 80(7):1384–1403, 2014. doi:10.1016/J.JCSS.2014.04.011.

23 Svatopluk Poljak and Daniel Turzík. A polynomial time heuristic for certain subgraph optimization problems with guaranteed worst case bound. *Discrete Mathematics*, 58(1):99–104, 1986. doi:10.1016/0012-365X(86)90192-5.

24 Venkatesh Raman and Saket Saurabh. Parameterized algorithms for feedback set problems and their duals in tournaments. *Theoretical Computer Science*, 351(3):446–458, 2006. doi:10.1016/J.TCS.2005.10.010.

25 William T Trotter and John I Moore. The dimension of planar posets. *Journal of Combinatorial Theory, Series B*, 22(1):54–67, 1977. doi:10.1016/0095-8956(77)90048-X.

A Other guarantees for Maximum Acyclic Subgraph

In this section, we explain some claims from the introductory part of our paper.

A.1 Superiority of the Poljak–Turzík bound in unweighted case

The Poljak–Turzík bound guarantees that a weakly-connected oriented digraph G has an acyclic subgraph with at least $m/2 + (n-1)/4$ edges, where n and m are the number of vertices and edges of G, respectively. Here we prove that MAS above random ordering ($m/2$) and MAS above spanning tree ($n-1$) are both reducible to MAS above Poljak–Turzík.

We remark that the first reduction was already presented in [22] (see Section 4.2). For the sake of completeness, we provide our own, slightly different, proof below.

Random ordering. Let G be a given digraph (without parallel edges), and we are to find an acyclic subgraph with at least $m/2 + k$ edges for $k \geq 0$. Assume as well that we have an algorithm $\mathcal{A}$ that solves MAS above Poljak–Turzík in time $f(k) \cdot n^{\mathcal{O}(1)}$. The only obstacle is that $\mathcal{A}$ works only for oriented weakly-connected graphs. To make G oriented, note that we have to get rid of pairs of opposite edges of form $u \to v$, $v \to u$. Note that if we remove both these edges, $(G - uv - vu, k)$ is equivalent to the initial instance of MAS above random ordering, because *exactly one* of these edges belongs to any inclusion-wise maximal solution. This argument gives an equivalent oriented graph instance in polynomial time. It only remains to apply $\mathcal{A}$ to each separate weakly-connected component of G. For each such component G_i, we find the largest $k_i \in [0, k]$ such that $\mathcal{A}$ reports that G_i has an acyclic subgraph with at least $m_i/2 + (n_i - 1)/4 + k_i$ edges. If the sum of $(n_i - 1)/4 + k_i$ is at least k, then G admits an acyclic subgraph with $m/2 + k$ edges, and otherwise it does not.

Spanning tree. Now let G be a weakly-connected directed graph (without parallel edges), and let us seek an acyclic subgraph of G with at least $(n-1)+k$ edges. We exhaustively apply the following reduction rules, each of which preserves weak connectivity and the absence of parallel edges.

Assume we have a pair of opposite edges uv and vu in G. If $G' = G - uv - vu$ remains weakly-connected, we reduce to the instance $(G', k-1)$. This is safe since exactly one edge of uv and vu can be added to any acyclic subgraph of G' without forming a cycle. Otherwise, to keep the graph weakly-connected, we remove only the edge uv and keep k unchanged. In this case, safety follows because the edge uv in any acyclic subgraph of G can be replaced by vu.

Next, suppose we have a vertex v of degree at most two, and all its incident edges go in the same direction (either all in or all out). Then all these edges belong to every inclusion-wise maximal acyclic subgraph, implying that $\mathrm{mas}(G - v) = \mathrm{mas}(G) - \deg v$, where mas denotes the maximum number of edges in an acyclic subgraph.

If $G - v$ is weakly-connected, we reduce to $(G - v, k - \deg v + 1)$. Here, n decreases by one and mas decreases by $\deg v$. If $G - v$ is disconnected, then v must have two neighbors (since G itself is weakly-connected). Denote them by u and w. In this case, we reduce to $(G - v + uw, k)$. The added edge uw is a bridge, so $\mathrm{mas}(G - v + uw) = \mathrm{mas}(G - v) + 1 = \mathrm{mas}(G) - \deg v + 1 = \mathrm{mas}(G) - 1$. Since the number of vertices also drops by one, the reduction is safe.

We proceed to the last reduction rule. Assume that v is a vertex of degree two with neighbors u and w, where $uv \in E(G)$ and $vw \in E(G)$, but $uw \notin E(G)$. We reduce to $(G - v + uw, k)$. To see that $\mathrm{mas}(G - v + uw) = \mathrm{mas}(G) - 1$, observe the following. If an acyclic subgraph $X \subseteq G - v + uw$ contains uw, then replacing uw with the two edges uv and vw yields an acyclic subgraph of G. If X does not contain uw, we simply add the edge uv to X. The reverse direction is similar. As the number of vertices decreases by one, the reduction is safe.

Since each reduction rule decreases the number of edges and does not increase the parameter, in polynomial time we can obtain an equivalent instance (G', k') with $k' \leq k$ such that neither reduction rule is applicable to G'. By construction, G' is a weakly-connected oriented graph.

Let $S \subseteq V(G')$ be the set of vertices of degree at most two in G'. Since the second reduction rule is not applicable, every $s \in S$ has exactly one incoming edge us and one outgoing edge sv. Moreover, because the third reduction rule is not applicable, we have $uv \in E(G')$. We claim that $G' - S$ is weakly-connected. Take any two vertices a and b in $G' - S$, and let P be a shortest path between them in the underlying graph of G'. Since G' is weakly-connected, such a path exists. If P contained some $s \in S$, then s would have neighbors u and v on P. But uv is an edge in the underlying graph of G', so shortcutting $u - s - v$ by $u - v$ yields a shorter path, a contradiction. Hence, P does not intersect S, and thus $G' - S$ is weakly-connected.

We now show that if $|S| \geq 2k'$, then (G', k') is a yes-instance of MAS above spanning tree. Recall that for each $s \in S$, we have an edge in G' between its neighbors. Choose a vertex ordering of G' such that at least $|S|/2 \geq k'$ of these edges between neighbors are oriented forward. Such an ordering can be constructed by taking either an arbitrary order or its reverse. Let E' be the set of such edges going forward. Note that the edges of E' form an acyclic subgraph of $G' - S$. We arbitrarily extend it to a weakly-connected acyclic subgraph X of $G' - S$.

Now, extend X to the acyclic subgraph of G' as follows. For each $s \in S$ with incident edges us and sv, add both us and sv to X if $uv \in E'$; otherwise, add only us. This preserves acyclicity because if both edges are added, then the edge uv is already in X.

The resulting subgraph is acyclic and has $|E(X)| + |S| + |E'|$ edges. Since X is a weakly-connected subgraph on $|V(G' - S)|$ vertices, we have $|E(X)| \geq |V(G')| - |S| - 1$. Therefore, the number of edges is at least $|V(G')| - 1 + k'$, proving that (G', k') is a yes-instance.

Finally, if $|S| < 2k'$, we aim to apply the FPT algorithm $\mathcal{A}$ for MAS above Poljak–Turzík. Let $n' = |V(G')|$, let $m' = |E(G')|$, and let $d = \frac{m'}{2} + \frac{n'-1}{4} - (n' - 1)$, measuring the difference between Poljak–Turzík and spanning tree guarantees. It suffices to show that d is bounded below by a linear function of k'.

By the definition of S, every vertex outside S has degree at least three, while every vertex in S has degree exactly two. Hence,

$$m' = \frac{\sum_{v \in V(G')} \deg v}{2} \geq \frac{3(n' - |S|) + 2|S|}{2} = \frac{3n' - |S|}{2} \geq \frac{3}{2}n' - k',$$

where the last inequality uses $|S| < 2k'$. Therefore, $\frac{m'}{2} + \frac{n'-1}{4} \geq \frac{3n'}{4} - \frac{k'}{2} + \frac{n'-1}{4} = n' - \frac{1}{4} - \frac{k'}{2} = (n' - 1) + 3/4 - \frac{k'}{2}$, implying $d \geq -k'/2$.

We now run $\mathcal{A}$ on G' with parameter $k'' = k' - d \leq \frac{3k'}{2}$. Thus $\mathcal{A}$ runs in FPT time with respect to k', and since $k' \leq k$, this completes the reduction.

A.2 Weighted MAS above random ordering for rational weights

Here we outline the proof that finding an acyclic subgraph of weight at least $\mathbf{w}(E(G))/2 + k$, for integer k and weights $\mathbf{w} : E(G) \to \mathbb{Q}_{\geq 1}$ is NP-complete when $k = 1$. First of all, note that unweighted MAS is NP-complete for oriented graphs. That is, finding an acyclic subgraph with at least p edges in a given oriented graph G is NP-complete (because digraphs that are not oriented can be made oriented by removing pairs of opposite edges).

Now reduce an unweighted oriented G to a new weighted digraph $(G', \mathbf{w})$ as follows. For each edge uv of G, add an opposite edge vu. Assign weights $\mathbf{w}(vu) = 1$ and $\mathbf{w}(uv) = 1 + \epsilon$. Note that $\mathbf{w}(E(G')) = m \cdot (2 + \epsilon)$, where $m = |E(G)|$. From each pair (uv, vu) in G', we take exactly one into the solution. If we take the "real" copy from G, we obtain $\epsilon/2$ excess weight above average, otherwise we lose $\epsilon/2$ weight. Formally, if an acyclic subgraph of G' contains t "real" copies (so it uniquely corresponds to an acyclic subgraph with t edges in G), its weight is $(m - t) \cdot 1 + t \cdot (1 + \epsilon) = m + t \cdot \epsilon = m + m \cdot \epsilon/2 + (t - m/2) \cdot \epsilon$, or just $\mathbf{w}(E(G'))/2 + (t - m/2) \cdot \epsilon$. Therefore, if we choose ϵ as $1/(p - m/2)$, solving $(G', \mathbf{w}', 1)$ above random ordering is equivalent to finding an acyclic subgraph in G with at least p edges.

This is a polynomial-time reduction, so the NP-completeness follows.

B Proof of Theorem 1

▶ **Theorem 1.** *MAS/MaxST($\mathbb{Z}_{\geq 1}$) admits an algorithm with $2^{k^{\mathcal{O}(1)}} \cdot |\mathcal{I}|^{\mathcal{O}(1)}$ running time.*

Proof of Theorem 1. Given an instance $\mathcal{I} = (G, \mathbf{w}, k)$, the algorithm first applies Lemma 17 to $\mathcal{I}$. If it reports a solution to $\mathcal{I}$, the algorithm outputs it and stops. Otherwise, our algorithm obtains an equivalent instance $\mathcal{I}'$. Lemma 17 guarantees that solving $\mathcal{I}'$ is equivalent to solving $\mathcal{I}$, as the resulting solution to $\mathcal{I}'$ can be transformed into a solution to $\mathcal{I}$. Without loss of generality, we put $\mathcal{I} := \mathcal{I}'$, assuming that the algorithm always transforms the solution as required.

Then, our algorithm identifies T, A and B in polynomial time. Then, for each $e \in E(T)$ the algorithm evaluates $\mathbf{p}(e)$. If $\mathbf{p}(e) \geq k$, the algorithm reports $T - e + \mathrm{Inv}(e)$ as a correct solution (see Observation 12) and stops. Then, the algorithm evaluates $\mathbf{w}(A)$. If $\mathbf{w}(A) \geq 3k$, the algorithm constructs a solution to $\mathcal{I}$ using Lemma 16, outputs it and stops.

The algorithm now targets towards an application of Lemma 20. Let D be the set of all vertices $v \in V(T)$ such that in-degree and out-degree of v in T are both exactly one. From Lemma 17 we know that $|V(T) \setminus D|$ is at most $\mathcal{O}(k + |A|) = \mathcal{O}(k)$, by $|A| \leq \mathbf{w}(A) < 3k$. Let V_A be the set of vertices incident to A in G. Denote $C := (V(T) \setminus D) \cup V_A$, $|C| = \mathcal{O}(k)$. The algorithm evaluates C in polynomial time.

Then, in polynomial time, the algorithm computes a set of directed paths of T with both endpoints in C. This set is defined uniquely since each vertex in $V(T) \setminus C$ has in-degree and out-degree one in T, and has exactly $t = |C| - 1$ paths. Clearly, this set of paths

satisfies the first three points of the Definition 18 of proper path covers. The algorithm then arranges (in polynomial time) the paths in an order $P_1, P_2, \ldots, P_t$ according to the last point in Definition 18. The formed sequence is a proper path cover $\mathcal{P}$ of T with $t = \mathcal{O}(k)$ paths. After that, the algorithm construct a $\mathcal{P}$-respecting ordering $e_1, \ldots, e_{n-1}$ of $E(T)$.

We finally move on to the "heart" of our algorithm. The algorithm performs a recursive branching subroutine `extend` (see Algorithm 1) to guess a correct choice of S given by Lemma 20. For the sake of clarity, `extend` does not return the solution (only reports YES) in the pseudo-code, but it is straightforward to change it so it returns the solution. The algorithm runs $\texttt{extend}(\mathcal{I}, \mathcal{P}, e_1, \ldots, e_{n-1}, \emptyset, 0)$ as an entry-point to this recursive procedure. This finishes the description of the algorithm.

Correctness. Note that `extend` cannot give false-positives, since it returns YES only if $|S| \geq k + t$ or Lemma 21 returns a solution (which cannot be false-positive). In the first case, $|S|$ should have at least k edges with positive remaining profit (which gives a solution by Observation 15), because `extend` takes at most one edge with non-positive **rp** per each path in $\mathcal{P}$.

Now assume that `extend` does not return YES on $\mathcal{I}$. By Lemma 20, there is a set S that satisfies all three properties from the statement, and applying Lemma 21 to $\mathcal{I}$ and S gives a solution to $\mathcal{I}$. `extend` never returned YES, that is, `extend` was never called with this value of S. But the properties of S imply that, if $e_i, e_j \in S$ are consecutive (with respect to the order $e_1, \ldots, e_{n-1}$) edges in S, then either

- e_j belongs to the same path as e_i, its remaining profit is positive, and there is no k with $i < k < j$ such that $\mathbf{rp}(S_i, e_k) = \mathbf{rp}(S_i, e_j)$, or
- e_j belongs to path $P_{\ell'}$ that goes after the path that e_i belongs to, and there is no k such that $e_k \in P_{\ell'}$ and $\mathbf{rp}(S_i, e_k) = \mathbf{rp}(S_i, e_j)$.

In the first case, `extend` will consider the correct choice of $p \in [k]$ that leads to this choice of j, since there is no other edge inbetween e_i and e_j with this value of remaining profit. In the second case, `extend` will consider the correct choice of both $\ell' \in [\ell + 1, t]$ and $p \in [-\mathbf{w}(A), k]$. This will lead to the correct choice of j similarly to the first case. Therefore, we can prove by induction that `extend` will construct S correctly in one of its recursion paths if never returned YES in process. This contradiction finishes the correctness discussion.

Running time. All parts of our algorithm are polynomial, except for the recursive branching subroutine. It invokes Lemma 21, which gives $2^{k^{\mathcal{O}(1)}} \cdot n^{\mathcal{O}(1)}$ multiplier in the running time. To bound the number of recursive calls, note that the depth of recursion is at most $k + t + \mathcal{O}(1)$, that is, at most $\mathcal{O}(k)$, while the number of possible recursive calls produced at one call is bounded by $k + (k + \mathbf{w}(A) + 1) \cdot |\mathcal{P}| = \mathcal{O}(k^2)$. Therefore, the total number of recursive calls is $(k^2)^{\mathcal{O}(k)} = 2^{k^{\mathcal{O}(1)}}$. The total running time is upper bounded with $2^{k^{\mathcal{O}(1)}} \cdot n^{\mathcal{O}(1)}$.

The proof is complete. ◀

C Proof of Theorem 2

▶ **Theorem 2.** *MAS/MaxST($\mathbb{Q}_{\geq 1}$) admits an algorithm with $n^{k^{\mathcal{O}(1)}} \cdot |\mathcal{I}|^{\mathcal{O}(1)}$ running time.*

Proof of Theorem 2. Let $\mathcal{I} = (G, \mathbf{w}, k)$ be an instance of $\text{MAS}/\text{MaxST}_{\mathbb{Q}_{\geq 1}}$. If $\mathbf{w}(A) \geq 3k$, then the algorithm from Lemma 16 constructs a solution to $\mathcal{I}$ in polynomial time. From now on, we assume that $\mathbf{w}(A) < 3k$.

We iterate over all possible choices of a set $D \subseteq B$ of weight at most $\mathbf{w}(A)$, a set $B' \subseteq B \setminus D$ of weight at most $(\mathbf{w}(A)+k)^2$, and sets $P_F, P_R \subseteq V \times V$, each of size at most $4(\mathbf{w}(A)+k+1)^4$. Additionally, we require that for every (u,v) from $P_F \cup P_R$, T contains a directed path from u to v. For a fixed choice of D, B', P_F and P_R, we begin by constructing the sets

$$F = \bigcup_{(u,v)\in P_F} E_{u,v},$$
$$R = \bigcup_{(u,v)\in P_R} E_{u,v},$$

where $E_{u,v}$ denotes the set of edges on the path from u to v in T.

Let $G' = G - D$, and let $\mathbf{w}'$ be the restriction of $\mathbf{w}$ to $E(G')$. Since D consists only of blocked edges, we know that $T \subseteq G'$, and hence $\mathrm{MaxST}(G', \mathbf{w}') = T$. Therefore, $F, R \subseteq E(\mathrm{MaxST}(G', \mathbf{w}'))$ by the definition of these sets.

We check whether $B' \subseteq \mathrm{Inv}(G', \mathbf{w}', R)$, and if so, apply the algorithm from Lemma 24 to $(G', \mathbf{w}')$ and the triple (F, R, B'), which returns a set $S \subseteq E(T)$. For each $A' \subseteq A$, we check whether the subgraph $T - S + \mathrm{Inv}(S) + A'$ is a solution to $\mathcal{I}$.

If no check succeeds over all choices of D, B', P_F, P_R, and A', we report that $\mathcal{I}$ is a no-instance.

Running time. We note that for a fixed choice of the sets D, B', P_F, P_R, and A', our algorithm works in polynomial time. Indeed, we first construct F, R, and G', then we check that $B' \subseteq \mathrm{Inv}(G', \mathbf{w}', R)$, apply the algorithm from Lemma 24, and finally verify that $T - S + \mathrm{Inv}(S) + A'$ is a feasible solution to $\mathcal{I}$. All these parts are polynomial. Therefore, it remains to bound the number of choices of the sets D, B', P_F, P_R and A'.

Recall that all edge weights are at least one. Hence, if the weight of a set is bounded by some value, then the same bound applies for its size. Since $|B| \leq m \leq n^2$, there are at most $n^{2\,\mathbf{w}(A)}$ ways to choose D and at most $n^{2(\mathbf{w}(A)+k)^2}$ possible sets B'. Similarly, because $|E(T)| = n-1$, we have at most $n^{4(\mathbf{w}(A)+k+1)^4}$ ways to choose each of the sets P_F and P_R. Finally, $|A| \leq \mathbf{w}(A)$, and hence there are at most $2^{\mathbf{w}(A)}$ possible sets A'. Combining these bounds with $\mathbf{w}(A) < 3k$, we obtain that there are $n^{\mathcal{O}(k^4)}$ ways to choose these sets.

Correctness. By construction, our algorithm outputs only feasible solutions. Therefore, it remains to check that it does not report false negatives. Suppose that $\mathcal{I}$ is a yes-instance. In this case, by Claim 26, it has a solution X with $S = E(T) \setminus E(X)$ and $D = \mathrm{Inv}(G, \mathbf{w}, S) \setminus E(X)$ such that $\mathbf{w}(D) \leq \mathbf{w}(A)$. Let $G' = G - D$, and let $\mathbf{w}'$ be the restriction of $\mathbf{w}$ to $E(G')$. By the definition of D, it holds that $E(X) \cap D = \emptyset$, and hence X is a subgraph of G'.

Since $D \subseteq B$, we have $\mathrm{MaxST}(G', \mathbf{w}') = T$. Because allowed, blocked, and inverse edges are defined based on the maximum spanning tree, it follows that $A(G', \mathbf{w}') = A(G, \mathbf{w})$, $B(G', \mathbf{w}') = B(G, \mathbf{w}) \setminus D$, and $\mathrm{Inv}(G', \mathbf{w}', e) = \mathrm{Inv}(G, \mathbf{w}, e) \setminus D$ for each $e \in E(T)$. We claim that X is an Inv-respecting subgraph in G'. Since X is a solution to $\mathcal{I}$, it is acyclic. By applying Observation 8 to the graph G', we know that X can contain blocked edges only from the set $\mathrm{Inv}(G', \mathbf{w}', S) = \mathrm{Inv}(G, \mathbf{w}, S) \setminus D$. Additionally, by the definition of D, we have $\mathrm{Inv}(G, \mathbf{w}, S) \setminus E(X) = D$, which implies $\mathrm{Inv}(G', \mathbf{w}', S) \subseteq E(X)$. Therefore, $X = T - S + \mathrm{Inv}(G', \mathbf{w}', S) + A'$ for some $A' \subseteq A$. Note that X is also a solution to $\mathcal{I}' = (G', \mathbf{w}', k)$. Consequently, $\mathcal{I}'$ is an Inv-respecting instance.

Lemma 29 provides a family of constraint sets $\{\mathcal{C}_{A'} \mid A' \subseteq A\}$ for the instance $\mathcal{I}'$. Moreover, there exist $S' \subseteq E(T)$ and $A' \subseteq A$ such that $X' = T - S' + \mathrm{Inv}(G', \mathbf{w}', S') + A'$ is a solution to $\mathcal{I}'$ that satisfies the constraint set $\mathcal{C}_{A'}$. Let $(F, R, B') \in \mathcal{C}_{A'}$ be the satisfied

constraint triple. By definition, we have $B' \subseteq B(G', \mathbf{w}') = B(G, \mathbf{w}) \setminus D$. Additionally, the lemma guarantees that $\mathbf{w}(B') \leq (\mathbf{w}(A) + k)^2$, and each of the sets F and R equals the union of at most $4(\mathbf{w}(A) + k + 1)^4$ directed paths in T. Let P_F and P_R be the sets of endpoints of such paths for F and R, respectively.

From the definition of D, B', P_F, and P_R, it follows that our algorithm must have considered these sets. Moreover, given P_F and P_R, it constructed exactly the sets F and R. Since (F, R, B') is a restriction triple, we have $B' \subseteq \mathrm{Inv}(G', \mathbf{w}', R)$. Hence, our algorithm applied Lemma 24 to $(G', \mathbf{w}')$ and constraints (F, R, B') and obtained a set S''. Since S' is also feasible under these constraints and S'' has a maximum profit, we have $\mathbf{p}(G', \mathbf{w}', S'') \geq \mathbf{p}(G', \mathbf{w}', S')$.

Consider the subgraph $X'' = T - S'' + \mathrm{Inv}(G', \mathbf{w}', S'') + A'$. Since S'' is feasible under the constraints given by (F, R, B'), X'' satisfies this restriction triple, and therefore it also satisfies $\mathcal{C}_{A'}$. Hence, Lemma 29 guarantees that X'' is acyclic. Note that

$$\mathbf{w}(X'') = \mathbf{w}(T) + \mathbf{p}(G', \mathbf{w}', S'') + \mathbf{w}(A') \geq \mathbf{w}(T) + \mathbf{p}(G', \mathbf{w}', S') + \mathbf{w}(A') = \mathbf{w}(X')$$

Since X' is a solution to $\mathcal{I}'$, we have $\mathbf{w}(X') \geq \mathbf{w}(T) + k$, and therefore $\mathbf{w}(X'') \geq \mathbf{w}(T) + k$. Consequently, X'' is a solution to $\mathcal{I}$, which is found by our algorithm. ◀