2015 IEEE 56th Annual Symposium on Foundations of Computer Science

An $O(1)$-Approximation for Minimum Spanning Tree Interdiction

Rico Zenklusen
Department of Mathematics
ETH Zurich
Zurich, Switzerland
Email: ricoz@math.ethz.ch

Abstract

Network interdiction problems are a natural way to study the sensitivity of a network optimization problem with respect to the removal of a limited set of edges or vertices. One of the oldest and best-studied interdiction problems is minimum spanning tree (MST) interdiction. Here, an undirected multigraph with nonnegative edge weights and positive interdiction costs on its edges is given, together with a positive budget B. The goal is to find a subset of edges R, whose total interdiction cost does not exceed B, such that removing R leads to a graph where the weight of an MST is as large as possible. Frederickson and Solis-Oba (SODA 1996) presented an $O(\log m)$-approximation for MST interdiction, where m is the number of edges. Since then, no further progress has been made regarding approximations, and the question whether MST interdiction admits an $O(1)$-approximation remained open.

We answer this question in the affirmative, by presenting a 14-approximation that overcomes two main hurdles that hindered further progress so far. Moreover, based on a well-known 2-approximation for the metric traveling salesman problem (TSP), we show that our $O(1)$-approximation for MST interdiction implies an $O(1)$-approximation for a natural interdiction version of metric TSP.

Keywords

approximation algorithms; combinatorial optimization; interdiction problems; minimum spanning trees; submodular functions

I. Introduction

Network interdiction studies the sensitivity of a network optimization problem with respect to the removal of some limited set of its edges or vertices. For example, in the minimum spanning tree (MST) interdiction problem, we are given an undirected loopless multigraph $G = (V, E)$ with nonnegative edge weights $w : E \to \mathbb{Z}_{\geq 0}$, positive edge interdiction costs $c : E \to \mathbb{Z}_{>0}$, and an interdiction budget $B \in \mathbb{Z}_{>0}$. The goal is to remove a subset of edges whose total interdiction cost is bounded by B, and such that the weight of an MST in the graph on the non-removed edges is as large as possible. To avoid trivial cases, we assume that the budget is not large enough to disconnect the graph. Along the same lines, interdiction problems have been considered for a wide variety of other underlying network optimization problems, including maximum s-t flows, maximum matchings, shortest paths, maximum edge-connectivity, and maximum stable sets (see Section I-B for references and some further details). As highlighted in the example of interdicting MSTs, interdiction problems can naturally be interpreted as two-player problems, where an *interdictor* first removes edges and plays against an *operator*, who solves an optimization problem over the remaining network.

Interdiction problems allow for identifying weak spots in a networked system that may be worth reinforcing, or to obtain strategies to interdict an optimization problem that describes an undesirable process on a network. Therefore, interdiction problems have found applications in a wide variety of areas, including preventing the spread of infections in hospitals [1], inhibiting the production and distribution of illegal drugs [41], prevention of nuclear arms smuggling [28], military planning [14], and infrastructure protection [36], [8]. Even the discovery of the Max-Flow/Min-Cut Theorem was motivated by a Cold War plan to interdict the Soviet rail network in Eastern Europe [37].

0272-5428/15 $31.00 © 2015 IEEE
DOI 10.1109/FOCS.2015.49

Considerable effort has also been spent in getting a better theoretical understanding of interdiction problems. However, large gaps remain. This is especially true regarding their approximability, which is of particular interest since almost all known interdiction problems are easily shown to be NP-hard. One of the oldest and most-studied interdiction problems, for which a large gap in terms of approximability exists, is MST interdiction, which is the focus of this paper. It captures well-known graph optimization problems, like the *maximum components problem* (MCP) [13], which asks to break a graph into as many connected components as possible by removing a given number q of edges. Also the generalization of MCP with interdiction costs on the edges and an interdiction budget B, which was studied in [11] and called the *budgeted graph disconnection* (BGD) problem, remains a special case of MST interdiction. Notice the close relation between MCP and the k-cut problem [15], where the roles of objective and budget are exchanged. In particular, as observed in [13], this connection to the k-cut problem immediately implies strong NP-hardness of MCP, and therefore also of MST interdiction. For completeness, we briefly discuss this connection in Appendix A. Another motivation for studying MST interdiction is that MSTs are often used as building blocks in other optimization problems or approximation algorithms. Results on MST interdiction therefore have the potential to be carried over to further interesting problem settings. In particular, we exploit the well-known property that the weight of an MST is within a factor of 2 of the shortest tour for the metric traveling salesman problem (TSP), to transform approximation results on MST interdiction to metric TSP interdiction.

In 1996, Frederickson and Solis-Oba [13] presented an $O(\log m)$-approximation for MST interdiction, where $m = |E|$ is the number of edges. No improvement on the approximation ratio has been obtained since. We highlight that parallel edges are allowed in the MST interdiction problem, and we thus may have $\log m = \omega(\log n)$, where $n = |V|$. Admitting parallel edges is of particular interest in MST interdiction and also other interdiction problems, since they allow for modeling effects like partial destruction of a connection between two vertices. Hence, so far, no approximation algorithm for MST interdiction is known with an approximation factor that is polylogarithmic in n.

A special case of MST interdiction, which received considerably attention, is the *k most vital edges problem*, which asks to remove k edges to obtain a graph whose MST has a weight as large as possible. Hence, this corresponds to MST interdiction with unit interdiction costs and budget $B = k$. From an approximation point of view, the best known procedure is as well the algorithm of Frederickson and Solis-Oba. However, for the k most vital edges problem this algorithm is known to be an $O(\log k)$-approximation [13]. Interest arose in obtaining fast polynomial algorithms for $k = O(1)$. In particular, the *most vital edge problem*, which corresponds to $k = 1$, is closely related to the *sensitivity analysis problem* for MSTs, as observed in [19]. In the sensitivity analysis problem one is given an edge-weighted graph $G = (V, E)$ and an MST $T \subseteq E$ in G. The goal is to determine for every edge by how much its weight has to be changed so that T is not anymore an MST. Clearly, any algorithm to find an MST combined with an algorithm for the sensitivity analysis problem leads to an algorithm to solve the most vital edge problem. Using this observation leads to the currently fastest algorithms for the most vital edge problem, beating the strongest specialized approaches known previously [18]. In particular, a deterministic $O(m \cdot \alpha(m, n))$ time algorithm for the most vital edge problem is obtained—where $\alpha(m, n)$ is the inverse Ackermann function—by combining Chazelle's [6] $O(m \cdot \alpha(m, n))$ MST algorithm with Tarjan's [40] $O(m \cdot \alpha(m, n))$ algorithm for the sensitivity analysis problem. Moreover, a randomized $O(m)$ time algorithm is obtained for the most vital edge problem by combining an $O(m)$ randomized MST algorithm—like the original algorithm of Klein and Tarjan [24] or a revised version presented by Karger et al. [21]—with a randomized $O(m)$ time algorithm by Dixon et al. [10] for the sensitivity analysis problem.[1] Pettie [32] presented an even faster deterministic $O(m \cdot \log \alpha(m, n))$ time algorithm for the sensitivity analysis problem. However, this does not lead to improvements for currently fastest deterministic algorithms for the most vital edge problem because no deterministic method is known to find an MST faster than in $O(m \cdot \alpha(m, n))$ time. Several exponential-time algorithms have been suggested for the k most vital edges problem for general k, including parallel algorithms [27], [26], [4]. The problem has also been

[1]A simpler randomized $O(m)$ time algorithm for the sensitivity analysis problem was later obtained by King [23].

considered under the aspect of parameterized complexity [17].

Our focus on MST interdiction lies on approximation algorithms. From an approximation point of view, the central open question within MST interdiction is whether it is possible to obtain an $O(1)$-approximation. The main contribution of this paper is to answer this question in the affirmative. As a direct consequence thereof, we obtain an $O(1)$-approximation for a natural interdiction version of metric TSP.

A. Our results and techniques

Our main result is the first $O(1)$-approximation for MST interdiction, improving on Frederickson and Solis-Oba's $O(\log m)$-approximation [13].

Theorem 1. *There is a* 14*-approximation for MST interdiction.*

MSTs are a useful tool in approximation algorithms for other combinatorial optimization problems, like metric TSP. Due to this link, we can use the above result as a black-box to obtain an $O(1)$-approximation for a natural interdiction version of metric TSP. In metric TSP, a complete graph is given with lengths on the edges that satisfy the triangle inequality, and the task is to find a shortest Hamiltonian cycle. Metric TSP often stems from settings where a graph $G = (V, E)$ with edge lengths $\ell : E \to \mathbb{Z}_{>0}$ is given, and the goal is to find a shortest closed walk that visits every vertex *at least* once. Such settings easily translate to metric TSP by considering a complete graph $\overline{G} = (V, \overline{E})$ over V such that to every edge $\{u, v\} \in \overline{E}$ the distance $d(\{u, v\})$ is assigned, where $d(\{u, v\})$ is the length of a shortest u-v path in G. A natural interdiction version is obtained by considering interdiction costs $c : E \to \mathbb{Z}_{>0}$ in G and a budget $B \in \mathbb{Z}_{>0}$; the task is to find a subset of edges $R \subseteq E$ such that the shortest closed walk in $(V, E \setminus R)$ that visits each vertex at least once is as large as possible. For brevity, we call this problem *metric TSP interdiction*. Combining Theorem 1 with a well-known 2-approximation for metric TSP that is based on MSTs, we obtain:

Theorem 2. *Metric TSP interdiction admits a* 28*-approximation.*

To obtain our main result, Theorem 1, we overcome two main hurdles for obtaining $O(1)$-approximations for MST interdiction. First, it is hard to find a good upper bound for MST interdiction. In particular, no strong LP relaxations are known. We note that even for the related k-cut problem and variants of it, it is nontrivial to find LP relaxations with constant integrality gap (see [30], [7], [35] and references therein).

A second obstacle, which also makes clear why MST interdiction seems substantially more difficult to approximate than MCP, is the fact that MST interdiction can be interpreted as a multilevel BGD problem, with interactions between the levels that are hard to control. To highlight this connection, which goes back to [13], we first observe that one can assume that each edge weight is either zero or a power of two, by losing at most a factor of 2 in the approximation guarantee. This is achieved by rounding down all edge weights to the next power of 2 (without changing zero-edges). Let $E_{\leq i}$ be all edges with weight at most 2^i. Now one can observe, and we will formalize this in Section II, that the weight of an MST is determined by the number of connected components of $G_i = (V, E_{\leq i})$ for each i. Hence, MST interdiction seeks to break the graphs G_i into as many components as possible, where breaking a graph G_i into an additional component has an impact on the weight of MSTs that is the higher, the larger the index i is. The approximation algorithm of Frederickson and Solis-Oba [13] essentially focusses only on one level where a high impact can be achieved, thus reducing the problem to a BGD problem, or an MCP for the case of unit interdiction costs. No algorithm is known so far that exploits the interactions between the different levels, which seems crucial for obtaining $O(1)$-approximations.

The way we address these two obstacles is as follows. First we obtain a good upper bound ν^* for the optimal value OPT by formulating a parametric submodular minimization problem. However, instead of finding a way to directly compare against ν^*, we focus on what we call *efficiencies* of potential edge sets to remove. More precisely, the efficiency of a set $U \subseteq E$—which does not need to fulfill the budget constraint—is defined as follows. Let $\mathrm{val}(U)$ be the weight of an MST in $(V, E \setminus U)$. Then the efficiency of U is given by $\mathrm{val}(U)/c(U)$. Apart from simple special cases, our algorithm computes a set $U \subseteq E$ that is over budget, and whose efficiency is close to ν^*/B, which is at least as good as the efficiency of an optimal interdiction set. The core part of our algorithm is a procedure that, given a set $U \subseteq E$ with $c(U) > B$, computes a set $R \subseteq U$ fulfilling the budget constraint and whose efficiency is close to the efficiency of U. Since we choose U to have a close-to-optimal efficiency, this allows us to compare our solution to ν^*.

To design this core part of the algorithm, we overcome the above-explained difficulty coming from the interpretation of MST interdiction as multilevel BGD problem as follows. We exploit that $U \subseteq E$ is a high-efficiency set, which implies that it has a good overall impact over the different levels i. To obtain a solution R that largely inherits this property from U, we start with $R = \emptyset$ and successively add to R appropriate subsets of U that are guaranteed to have a good impact over several levels, as long as $c(R) \leq B$.

We highlight that, in the interest of clarity, we did not try to heavily optimize constants.

B. Further related work

Many interdiction problems beyond the minimum spanning tree setting have been studied. This includes interdiction versions of the maximum s-t flow problem [41], [33], [43] (a setting often called *network flow interdiction*), the shortest path problem [2], [22], the maximum matching problem [42], [9], interdicting the connectivity of a graph [44], interdiction of packings [9], stable set interdiction [3], and variants of facility location [8]. However, the theoretical understanding of most interdiction problems still seems rather limited. A good example for which a large gap remains between the best known hardness results and approximation algorithms is network flow interdiction. Network flow interdiction is a strongly NP-hard problem [41] for which no approximation results are known, except for a pseudo-approximation [5] which is allowed to violate the budget by a factor of 2.

A related line of research is the study of a *continuous* version of interdiction problems, where the weight of edges can be increased continuously at a given weight per cost ratio which depends on the edge. These models are typically much more tractable then their discrete counterparts, i.e., the classical interdiction problems. The reason for this is that they can often be written as a single linear program. In particular, efficient algorithms for continuous interdiction have been obtained for maximum weight independent set in a matroid [13], maximum weight common independent sets in two matroids and the minimum cost circulation problem [20].

We highlight that [39] claims to present a 2-approximation for the k most vital edges problem for MST. However, the results in [39] are based on an erroneous lemma about spanning trees. In Appendix C we provide details on this erroneous lemma.

Organization of the paper

We formally define the problem and present some basic observations in Section II. Section III outlines our algorithmic approach, and reduces the task of finding an $O(1)$-approximation for MST interdiction to one specific subproblem, for which we present an algorithm in Section IV. The analysis of this algorithm is provided in Section V. Finally, Section VI provides the details of our result for metric TSP interdiction, thus proving Theorem 2.

II. Preliminaries

Throughout this paper, $G = (V, E)$ is an undirected multigraph with edge weights $w : E \to \mathbb{Z}_{\geq 0}$, edge costs $c : E \to \mathbb{Z}_{>0}$, and a global budget $B \in \mathbb{Z}_{>0}$. Furthermore, we assume that each edge weight is either a power of two or zero, i.e., $w : E \to \{0, 1, \dots, 2^p\}$. This can be achieved by rounding down all edge weights to the next power of two (without changing zero-edges). Clearly, this rounding changes the weight of any MST in G or any of its subgraphs by at most a factor of two. Hence, any α-approximation for MST interdiction with weights being powers of two is a 2α-approximation for general MST interdiction.

The MST interdiction problem asks to find a subset of edges $R \subseteq E$ with $c(R) \leq B$ that maximizes the weight of an MST in $(V, E \setminus R)$; we denote the weight of such an MST by $\mathrm{val}(R)$. Hence, $\mathrm{val}(\emptyset)$ is the weight of a minimum spanning tree in G, and the MST interdiction problem can formally be described by

$$\max\{\mathrm{val}(R) \mid R \subseteq E, c(R) \leq B\}. \tag{1}$$

Let OPT be the optimal value of problem (1). We call a set $R \subseteq E$ with $c(R) \leq B$ an *interdiction set*. When talking about edge sets $U \subseteq E$ that may not satisfy the budget constraint, but about which we still think of edges to be removed, we use the notion *removal set*.

To easily distinguish the different weight-levels we define

$$E_{-1} = \{e \in E \mid w(e) = 0\}, \quad E_i = \{e \in E \mid w(e) = 2^i\} \quad \forall i \in \{0, \dots, p\}, \text{ and}$$
$$E_{\leq i} = E_{-1} \cup \cdots \cup E_i \quad \forall i \in \{-1, \dots, p\}.$$

To avoid trivial cases, we assume that no interdiction set disconnects the graph, i.e., $c(\delta(S)) > B$ for all $S \subsetneq V, S \neq \emptyset$, where $\delta(S) \subseteq E$ is the set of all edges with precisely one endpoint in S. Due to this, there is always an optimal interdiction set that does not remove any edge from E_p, i.e., the edges with heaviest weight. Indeed, removing edges of heaviest weight cannot increase the weight of an MST, except if one could break the graph into several components, which we excluded. For simplicity we can therefore assume that (V, E_p) is a connected graph. This can be achieved by adding a non-interdictable spanning tree consisting of edges of weight 2^p to G. By the above discussion, adding such edges does not have any impact on the MST interdiction problem. Since there are optimal interdiction sets not containing any edge of E_p, we will consider throughout the paper only removal sets that are subsets of $E_{\leq p-1}$. Moreover, we assume to have at least 3 levels, i.e., $p \geq 1$, to simplify the presentation.

Furthermore, we assume that there is an interdiction set $R \subseteq E_{\leq p-1}$ such that $(V, E_{\leq p-1} \setminus R)$ has more connected components than $(V, E_{\leq p-1})$. Without this assumption, there is no interdiction set R that increases the number of edges in E_p that must be used in any MST in $(V, E \setminus R)$. In such a case, independent of the interdiction set R, any MST in $(V, E \setminus R)$ would use the same number of edges in E_p, namely a minimal set of edges connecting the connected components of $(V, E_{\leq p-1})$. Hence, one could reduce the problem by contracting any minimum edge set in E_p that connects the connected components of $(V, E_{\leq p-1})$.

For our analysis we focus on a well-known formula to describe the weight of an MST, which highlights the level-structure. For $U \subseteq E$, let $\sigma(U)$ be the number of connected components of the graph (V, U). For any $U \subseteq E_{\leq p-1}$, the weight $\mathrm{val}(U)$ of an MST in $(V, E \setminus U)$ is given by

$$\mathrm{val}(U) = \sigma(E_{-1} \setminus U) - 1 + \sum_{i=0}^{p-1} 2^i \Big(\sigma(E_{\leq i} \setminus U) - 1\Big). \tag{2}$$

This formula readily follows from the optimality of the greedy algorithm to find an MST, or from known results on matroid optimization (see [38, Volume B]).[2] Furthermore, it shows explicitly that for every additional component that is created on level $i \in \{0, \dots, p-1\}$—i.e., in the graph $(V, E_{\leq i})$—when removing U, the weight of MSTs increases by 2^i. Moreover, we highlight the well-known fact that $\sigma(U)$, and therefore also $\sigma(E_{\leq i} \setminus U)$ for $i \in \{-1, \dots, p-1\}$, is a supermodular function in U, i.e., $\sigma(A) + \sigma(B) \leq \sigma(A \cup B) + \sigma(A \cap B)$ for $A, B \subseteq E$. This follows from the fact that $\sigma(U) = n - r(U)$, where r is the rank function of the graphic matroid, which is submodular. This also implies that $\mathrm{val}(U)$ is supermodular in U, a fact we use later to find a removal set of high efficiency via submodular function minimization.

III. Outline of our approach

A core part of our algorithm is described in the following theorem. Before proving the theorem in Section IV, we will show how it can be used to obtain an $O(1)$-approximation for MST interdiction.

[2]In particular, (2) is a consequence of Theorem 40.2 in [38], which describes the weight of a maximum spanning tree in terms of the rank function $r : 2^E \to \mathbb{Z}_{\geq 0}$ of the graphic matroid, which satisfies $r(U) = n - \sigma(U)$. Notice that the MST problem can easily be reduced to the maximum spanning tree problem with nonnegative weights by replacing each edge weight $w(e)$ by $M - w(e)$ for a sufficiently large constant M.

Theorem 3. *There is an efficient algorithm (to be described in Section IV) that, for any set $U \subseteq E_{\leq p-1}$ with $c(U) > B$, returns an interdiction set $R \subseteq E$ with*

$$\mathrm{val}(R) \geq \frac{1}{2} \cdot B \cdot \frac{\mathrm{val}(U)}{c(U)} - 2^{p+1}. \tag{3}$$

We can get rid of the additive term 2^{p+1} in (3) by a best-of-two algorithm that either returns the interdiction set R claimed by Theorem 3 or an interdiction set that increases the number of connected components in $(V, E_{\leq p-1})$, which exists by assumption.

Corollary 4. *There is an efficient algorithm that, for any set $U \subseteq E$ with $c(U) > B$, returns an interdiction set $R \subseteq E$ with*

$$\mathrm{val}(R) \geq \frac{1}{6} \cdot B \cdot \frac{\mathrm{val}(U)}{c(U)}.$$

Proof: Let $U \subseteq E$ with $c(U) > B$, and let $R_1 \subseteq E$ be an interdiction set as claimed by Theorem 3. Furthermore, by assumption, there exists an interdiction set $R_2 \subseteq E$, such that $(V, E_{\leq p-1} \setminus R_2)$ has at least two components. (Actually, the assumption even implies that there is an interdiction set R_2 such that $(V, E_{\leq p-1} \setminus R_2)$ has at least one more component than $(V, E_{\leq p-1})$.) Such a set R_2 can be found efficiently by finding a minimum cost cut in $(V, E_{\leq p-1})$. Hence, $\mathrm{val}(R_2) \geq 2^p$. Let $R \in \mathrm{argmax}_{i \in \{1,2\}} \mathrm{val}(R_i)$. The set R satisfies the conditions of Theorem 4 since

$$\begin{aligned} \frac{1}{2} B \frac{\mathrm{val}(U)}{c(U)} &\leq \mathrm{val}(R_1) + 2^{p+1} && \text{(by (3))} \\ &\leq \mathrm{val}(R_1) + 2\,\mathrm{val}(R_2) \leq 3\,\mathrm{val}(R). \end{aligned}$$

■

In the following we show that either we can get an $O(1)$-approximation to MST interdiction with a quite direct approach, or we can find a removal set $U \subseteq E$ with $c(U) > B$ and high efficiency $\mathrm{val}(U)/c(U)$. For this, we take a somewhat different, bi-objective look on MST interdiction that is independent of the budget value B. Namely, for all sets $U \subseteq E_{\leq p-1}$ we consider the tuple $(c(U), \mathrm{val}(U))$. We are interested in sets $U \subseteq E$ with a large MST value $\mathrm{val}(U)$ and small cost $c(U)$, which can be interpreted as two objectives on U. Using standard notions of multi-objective optimization, we say that a tuple $(c(U), \mathrm{val}(U))$ is *non-dominated* if there is no other set $U' \subseteq E_{\leq p-1}$ with $c(U') \leq c(U)$, $\mathrm{val}(U') \geq \mathrm{val}(U)$ and at least one of these two inequalities being strict. The Pareto front in the cost-value space consists therefore of all non-dominated tuples $(c(U), \mathrm{val}(U))$ for $U \subseteq E_{\leq p-1}$, which can all be interpreted as optimal solutions to problem (1) when varying the budget.

Whereas finding a particular point on the Pareto front through solving problem (1) is NP-hard (since it is precisely the MST interdiction problem), one can efficiently compute so-called *extreme supported solutions* or *extreme supported tuples*, which are all vertices of $\mathrm{conv}(\{(c(U), \mathrm{val}(U)) \mid U \subseteq E_{\leq p-1}\}) + \mathbb{R}_{\geq 0} \times \mathbb{R}_{\leq 0}$, where conv is the convex hull operator. Hence, a tuple $(c(U), \mathrm{val}(U))$ for some $U \subseteq E_{\leq p-1}$ is an extreme supported tuple if there is a $\lambda \geq 0$ such that this tuple is the unique minimizing tuple for

$$\min\{\lambda \cdot c(U) - \mathrm{val}(U) \mid U \subseteq E_{\leq p-1}\}. \text{ [3]} \tag{4}$$

Notice that there may be several edge sets $U \subseteq E_{\leq p-1}$ that correspond to the same (extreme supported) tuple. Figure 1 shows an example of a Pareto front where the filled dots correspond to all extreme supported tuples, which we denote by $\mathcal{X}$. Notice that for any $\lambda \geq 0$, the objective $\lambda \cdot c(U) - \mathrm{val}(U)$ is a submodular function in U because $\mathrm{val}(U)$ is supermodular and $\lambda \cdot c(U)$ is modular in U. Problem (4) is therefore a parametric submodular function minimization problem, which is a well-studied problem (see [12], [29]). In particular, there is a set

[3]Notice that (4) can also be interpreted as a Lagrangean dual of $\min\{-\mathrm{val}(R) \mid R \subseteq E, c(R) \leq B\}$. We focus on the Pareto front interpretation since it is natural for properties we want to highlight later. The Lagrangian dual approach has been employed in similar problems in budgeted optimization (see [34], [16] and references therein).

of at most $\beta \le |E_{\le p-1}| + 1 \le m+1$ different solutions $U_1, \dots, U_\beta$, such that for each $\lambda \ge 0$, one of these solutions is optimal for (4). In other words, the optimal value of (4) is a piecewise linear function in λ with at most $m+1$ segments. The upper bound of $|E_{\le p-1}| + 1$ on β follows by the fact that one can choose sets U_i that are nested. Furthermore, Nagano [29] showed that such a family of sets $U_1, \dots, U_\beta$ can be determined by a variation of Orlin's submodular function minimization algorithm [31] within the same strongly polynomial time complexity. In summary, we can find in strongly polynomial time all $O(m)$ points in $\mathcal{X}$ each with a corresponding set $U \subseteq E_{\le p-1}$.

To find a good interdiction set, we distinguish the following three cases, depending on the budget B.

Case 1: There is a tuple $(\mathrm{val}(U), c(U)) \in \mathcal{X}$ such that $c(U) = B$. In this case U is an optimal solution to (1) that we can find efficiently and return.

Case 2: B is larger than the largest first coordinate among all points in $\mathcal{X}$. This implies that all edges in $E_{\le p-1}$ can be removed simultaneously without exceeding the budget. Hence, we return the interdiction set $R = E_{\le p-1}$ which is clearly optimal.

Case 3: There are two tuples $p_1 = (c(U_1), \mathrm{val}(U_1)), p_2 = (c(U_2), \mathrm{val}(U_2)) \in \mathcal{X}$ such that $c(U_1) < B < c(U_2)$, and p_1 and p_2 are consecutive in the sense that there is no other tuple $(c(U), \mathrm{val}(U)) \in \mathcal{X}$ with $c(U_1) < c(U) < c(U_2)$.

Since we easily get optimal solutions for the first two cases, we assume from now on to be in the third case. Figure 1 highlights a possible set $\mathcal{X}$ that corresponds to the third case. We can now upper bound OPT as follows. Consider the point $p = (x, y)$ on the segment between p_1 and p_2 such that $x = B$ (see Figure 1). Clearly, y is then equal to the following value, which we denote by ν^*:

$$y = \nu^* = \mathrm{val}(U_1) + (B - c(U_1)) \frac{\mathrm{val}(U_2) - \mathrm{val}(U_1)}{c(U_2) - c(U_1)}, \tag{5}$$

and we have $\nu^* \ge \mathrm{OPT}$ since all solutions are below the line that goes through p_1 and p_2, because p_1 and p_2 are consecutive points on the convex hull of the Pareto front. We will show that the following algorithm is an $O(1)$-approximation for the third case.

Algorithm 1: $O(1)$-approximation for third case

if $\mathrm{val}(U_1) \ge \frac{1}{7} \cdot \nu^*$ **then**
    Return U_1.
else
    Return an interdiction set $R \subseteq E$ satisfying

$$\mathrm{val}(R) \ge \frac{1}{6} \cdot B \cdot \frac{\mathrm{val}(U_2)}{c(U_2)} \quad ,$$

    which can be obtained by Corollary 4.
end

Theorem 5. *Algorithm 1 is a 7-approximation for the third case.*

Proof: If $\mathrm{val}(U_1) \ge \frac{1}{7}\nu^*$, then Algorithm 1 is clearly a 7-approximation since ν^* upper bounds OPT. Hence, assume

$$\mathrm{val}(U_1) < \frac{1}{7}\nu^*. \tag{6}$$

Notice that the slope from p_1 to p_2 is not larger than the one from the origin to p_2, i.e.,

$$\frac{\mathrm{val}(U_2) - \mathrm{val}(U_1)}{c(U_2) - c(U_1)} \le \frac{\mathrm{val}(U_2)}{c(U_2)}. \tag{7}$$

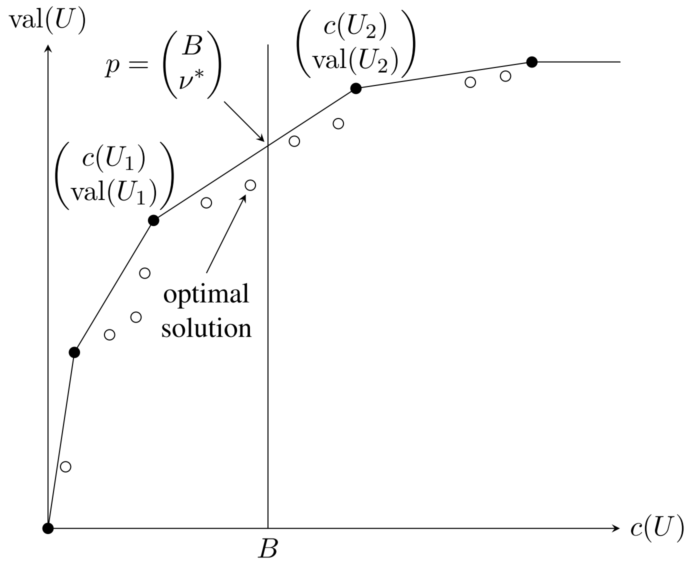

Figure 1: A possible constellation for the third case. The dots correspond to all non-dominated solutions, i.e., to the Pareto front. The filled dots represent the set $\mathcal{X}$ of all extreme supported tuples.

We therefore obtain

$$\begin{aligned}
\operatorname{val}(R) &\geq \frac{1}{6} \cdot B \cdot \frac{\operatorname{val}(U_2)}{c(U_2)} \\
&\geq \frac{1}{6} \cdot B \cdot \frac{\operatorname{val}(U_2) - \operatorname{val}(U_1)}{c(U_2) - c(U_1)} && \text{(using (7))} \\
&\geq \frac{1}{6} \cdot (B - c(U_1)) \frac{\operatorname{val}(U_2) - \operatorname{val}(U_1)}{c(U_2) - c(U_1)} \\
&= \frac{1}{6} \cdot (\nu^* - \operatorname{val}(U_1)) && \text{(using (5))} \\
&> \frac{1}{7} \nu^*, && \text{(using (6))}.
\end{aligned}$$

■

Thus, it remains to show Theorem 3. Finally, our main result, Theorem 1, is a direct consequence of the fact that we have a 7-approximation for all three cases under the assumption that each weight is either zero or a power of two. Hence, this implies a 14-approximation for general weights.

IV. Algorithm proving Theorem 3

In this section, we present an algorithm that proves Theorem 3. For brevity, we define $[k] = \{1, \ldots, k\}$ for $k \in \mathbb{Z}_{\geq 0}$; in particular, $[0] = \emptyset$. Throughout this section let $U \subseteq E_{\leq p-1}$ with $c(U) > B$. Furthermore, for $i \in \{-1, \ldots, p\}$, we define

$$U_{\leq i} = U \cap E_{\leq i}.$$

For each $i \in \{-1, \ldots, p\}$, let $\mathcal{A}_i \subseteq 2^V$ be the partition of V that corresponds to the connected components of $(V, E_{\leq i} \setminus U)$. Notice that the partitions $\mathcal{A}_i$ become coarser with increasing index i. Furthermore, $\mathcal{A}_p = \{V\}$, since we assume that (V, E_p) is connected and U does not contain any edges of E_p. See Figure 2 for an example. For $i \in \{-1, \ldots, p\}$ and $A \in \mathcal{A}_i$, we denote by $\mathcal{C}_i(A) \subseteq \mathcal{A}_{i-1}$ the sets in $\mathcal{A}_{i-1}$ that are included in A, which

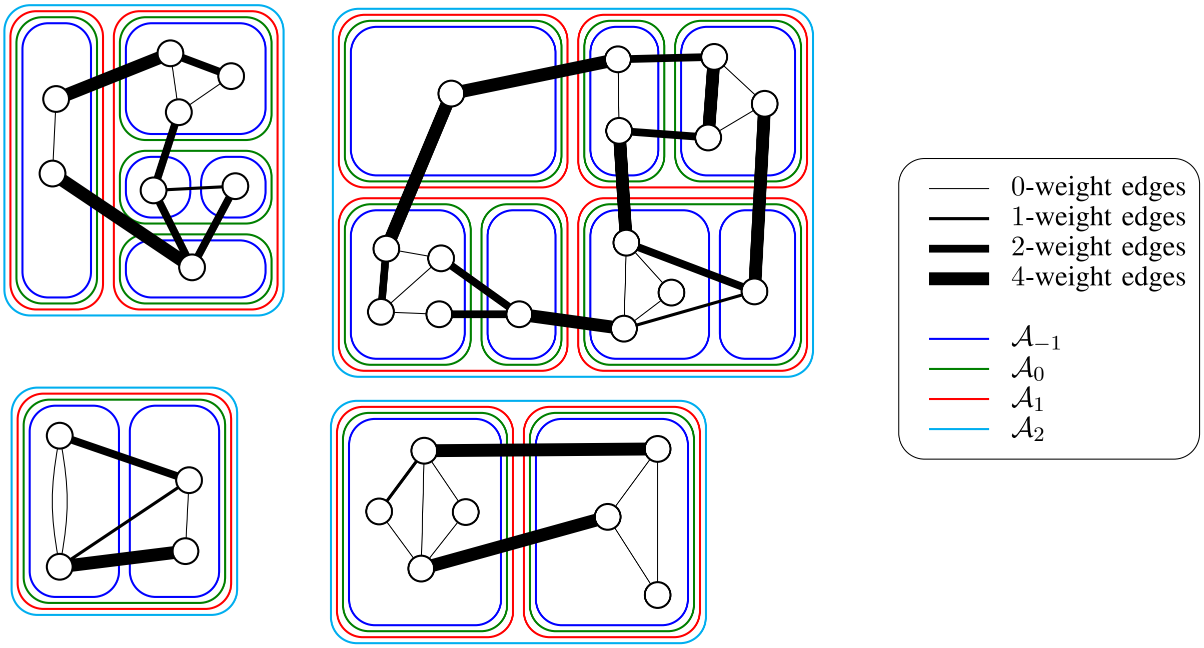

Figure 2: Example of a graph $(V, E_{\leq p-1} \setminus U)$ for $p = 3$ together with its corresponding partitions $\mathcal{A}_{-1}$, $\mathcal{A}_0$, $\mathcal{A}_1$, and $\mathcal{A}_2$. The edges in E_p, which connect all vertices by assumption, and the coarsest partition $\mathcal{A}_3 = \{V\}$ are not shown.

we call the *children* of A (on level i). More formally:

$$\mathcal{C}_i(A) = \begin{cases} \emptyset & \text{if } i = -1, \\ \{C \in \mathcal{A}_{i-1} \mid C \subseteq A\} & \text{if } i \geq 0. \end{cases}$$

Notice that when talking about children, we must indicate on which level i we consider the set A, since A may be a set that exists in several consecutive partitions. In this case, one has $\mathcal{C}_i(A) = \{A\}$ for all levels i such that $A \in \mathcal{A}_i$, except for the most fine-grained one (smallest i such that $A \in \mathcal{A}_i$).

Our algorithm greedily constructs what we call a removal pattern.

Definition 6 (Removal pattern)**.** *A removal pattern* $\mathcal{W} = \{(W_1, i_1), \ldots, (W_\beta, i_\beta)\}$ *is a family of tuples, where* $i_1, \ldots, i_\beta \in \{-1, \ldots, p-1\}$*,* $W_q \in \mathcal{A}_{i_q}$ *for* $q \in [\beta]$*, and* $W_1, \ldots, W_\beta$ *are all disjoint sets.*

To each removal pattern we assign a set of corresponding edges $R(\mathcal{W})$ to be removed:

$$R(\mathcal{W}) = \bigcup_{(W,i) \in \mathcal{W}} \{e \in U_{\leq i} \mid |e \cap W| = 1\},$$

where $|e \cap W|$ counts the number of endpoint that e has in W. In general, we treat an edge $e = \{u, v\}$ as a set containing its two endpoints u and v.

The motivation for the use of a removal pattern $\mathcal{W}$ to define an interdiction set, is that when removing all edges $U_{\leq i}$ that touch W_q, we have locally the same impact on the levels $-1, \ldots, i$ as U has when removing it from the graph. This allows us to exploit synergies between different levels that exist when removing U. For notational convenience, we denote the cost of the edges $R(\mathcal{W})$ that correspond to $\mathcal{W}$ by

$$c(\mathcal{W}) = c(R(\mathcal{W})).$$

To decide which sets to add to $\mathcal{W}$, we define for $i \in \{-1, \dots, p\}$ an auxiliary cost function $\kappa_i : \mathcal{A}_i \to \mathbb{Z}_{\geq 0}$ and an auxiliary impact function $g_i : \mathcal{A}_i \to \mathbb{Z}_{\geq 0}$ as follows: Let $A \in \mathcal{A}_i$, then

$$\kappa_i(A) = c(\{e \in U_{\leq i} \mid |e \cap A| = 1\}) + 2c(\{e \in U_{\leq i} \mid |e \cap A| = 2\}),$$
$$g_i(A) = |\{D \in \mathcal{A}_{-1} \mid D \subseteq A\}| + \sum_{\ell=0}^{i} 2^\ell \cdot |\{D \in \mathcal{A}_\ell \mid D \subseteq A\}|.$$

Notice that for $i \in \{0, \dots, p\}$,

$$\kappa_i(A) \geq \sum_{C \in \mathcal{C}_i(A)} \kappa_{i-1}(C), \text{ and} \tag{8}$$

$$g_i(A) = 2^i + |\{D \in \mathcal{A}_{-1} \mid D \subseteq A\}| + \sum_{\ell=0}^{i-1} 2^\ell |\{D \in \mathcal{A}_\ell \mid D \subseteq A\}| = 2^i + \sum_{C \in \mathcal{C}_i(A)} g_{i-1}(C). \tag{9}$$

These recursive relations are a main reason why we use g_i and κ_i as proxys for measuring locally the impact and cost of the removal set U. Moreover we have the following basic properties.

Lemma 7.

$$g_p(V) - 2^{p+1} = \text{val}(U), \tag{10}$$
$$\kappa_p(V) = 2c(U). \tag{11}$$

Proof: Equation (10) holds since

$$\begin{aligned} g_p(V) &= |\{D \in \mathcal{A}_{-1} \mid D \subseteq V\}| + \sum_{\ell=0}^{p} 2^\ell \cdot |\{D \in \mathcal{A}_\ell \mid D \subseteq V\}| = |\mathcal{A}_{-1}| + \sum_{\ell=0}^{p} 2^\ell \cdot |\mathcal{A}_\ell| \\ &= \sigma(E_{-1} \setminus U) + \sum_{\ell=0}^{p} 2^\ell \cdot \sigma(E_{\leq \ell} \setminus U) = (\sigma(E_{-1} \setminus U) - 1) + \sum_{\ell=0}^{p} 2^\ell \cdot (\sigma(E_{\leq \ell} \setminus U) - 1) + 2^{p+1} \\ &= \text{val}(U) + 2^{p+1}. \end{aligned}$$

Furthermore, (11) follows immediately from the definition of κ_i and the observation that $U_{\leq p} = U$:

$$\kappa_p(V) = c(\underbrace{\{e \in U_{\leq p} \mid e \cap V = 1\}}_{=\emptyset}) + 2c(\underbrace{\{e \in U_{\leq p} \mid |e \cap V| = 2\}}_{=U_{\leq p}}) = 2c(U).$$

■

For $i \in \{-1, \dots, p\}$ and $A \in \mathcal{A}_i$, we define the *auxiliary efficiency* of A by

$$\rho_i(A) = \frac{g_i(A)}{\kappa_i(A)},$$

with the convention that $\rho_i(A) = \infty$ if $\kappa_i(A) = 0$. Our algorithm, as described in Algorithm 2, adds sets to $\mathcal{W}$ iteratively starting at level $p-1$ and descending to level -1. Among the sets considered in each level, preference is given to sets with higher auxiliary efficiency. In the following we will show that the interdiction set $R(\mathcal{W})$ returned by Algorithm 2 satisfies the conditions of Theorem 3.

V. Analysis of the algorithm

We first formalize a particular structure of the removal pattern returned by Algorithm 2 which follows immediately from the fact that Algorithm 2 considers elements to add to $\mathcal{W}$ with respect to decreasing order of their auxiliary efficiencies.

Algorithm 2: Construction of interdiction set R fulfilling conditions of Theorem 3.

```
W = ∅
ℓ = p − 1                                                   // current level
A = V            // current vertex set to break into components on levels ≤ ℓ
while ℓ ≠ −2 do
    Let C_{ℓ+1}(A) = {Q_1, ..., Q_h} , where the numbering is chosen such that
                      ρ_ℓ(Q_1) ≥ ρ_ℓ(Q_2) ≥ ··· ≥ ρ_ℓ(Q_h).
    Let
          s = max { j ∈ {0, ..., h} | c(W ∪ {(Q_k, ℓ) | k ∈ [j]}) ≤ B }.
    Set
                      W = W ∪ {(Q_k, ℓ) | k ∈ [s]}.
    if s < h then
        ℓ = ℓ − 1
        A = Q_{s+1}
    else
        ℓ = −2    (i.e., leave the while-loop)
    end
end
return R(W)
```

Definition 8 (efficient removal pattern)**.** *Let* $\mathcal{W}$ *be a removal pattern.* $\mathcal{W}$ *is called* efficient *if for every* $i \in \{0,\dots,p\}$ *and* $A \in \mathcal{A}_i$*, one of the following holds:*

(i) No descendant of A *is contained in* $\mathcal{W}$*, i.e., for every* $\ell \in \{-1,\dots,i-1\}$ *and* $D \in \mathcal{A}_\ell$ *with* $D \subseteq A$*, we have* $(D,\ell) \notin \mathcal{W}$*, or*

(ii) all sets $(W,i') \in \mathcal{W}$ *for* $i' \in \{-1,\dots,i-1\}$ *are descendants of* (A,i)*. Moreover, there is a numbering of the elements in* $\mathcal{C}_i(A)$*, say* $\mathcal{C}_i(A) = \{Q_1,\dots,Q_h\}$*, and* $s \in \{0,\dots,h\}$ *such that* $\rho_i(Q_1) \geq \dots \geq \rho_i(Q_h)$ *and the following holds:*
- $(Q_k, i-1) \in \mathcal{W}$ *for* $k \in \{1,\dots,s\}$*,*
- $(Q_k, i-1) \notin \mathcal{W}$ *for* $k \in \{s+1,\dots,h\}$*,*
- *all tuples in* $\mathcal{W}$ *on levels* $\{-1,\dots,i-2\}$ *are descendants of* $(Q_{s+1}, i-1)$*. In particular, if* $s = h$*, then* $\mathcal{W}$ *contains no tuples on levels* $\{-1,\dots,i-2\}$*.*

Clearly, Algorithm 2 returns an efficient removal pattern. The key motivation for concentrating on efficient removal patterns is that we can relate, for any efficient removal pattern $\mathcal{W}$, its corresponding value $\mathrm{val}(R)$, where $R = R(\mathcal{W})$, to its cost $c(R)$. To do so, we first introduce variants $\kappa_i^{\mathcal{W}}$ and $g_i^{\mathcal{W}}$ of the auxiliary cost and impact functions κ_i and g_i, that measure cost and impact of the efficient removal pattern $\mathcal{W}$. In what follows, let $\mathcal{W}$ be an efficient removal pattern with corresponding removal set $R = R(\mathcal{W})$.

As usual we use the notation $R_{\leq i} = R \cap E_{\leq i}$ for $i \in \{-1,\dots,p\}$. For $\ell \in \{-1,\dots,p\}$ we define $\mathcal{S}_\ell \subseteq \mathcal{A}_\ell$ to be all sets of $\mathcal{A}_\ell$ that are descendants of sets added to $\mathcal{W}$, i.e.,

$$\mathcal{S}_\ell = \{A \in \mathcal{A}_\ell \mid \exists (W,i) \in \mathcal{W} \text{ with } i \geq \ell \text{ and } A \subseteq W\}.$$

Notice that contrary to $\mathcal{A}_\ell$, the family $\mathcal{S}_\ell$ is generally not a partition.

Similarly to the definitions of the auxiliary impact function g_i and auxiliary cost function κ_i, which are defined in terms of the set U, we define corresponding functions $g_i^{\mathcal{W}}$ and $\kappa_i^{\mathcal{W}}$ for the efficient removal pattern $\mathcal{W}$. For

$i \in \{-1, \dots, p\}$ and $A \in \mathcal{A}_i$, let

$$\kappa_i^{\mathcal{W}}(A) = \sum_{\substack{(W,j)\in\mathcal{W} \text{ with} \\ W\subseteq A, j\leq i}} \kappa_j(W), \text{ and}$$

$$g_i^{\mathcal{W}}(A) = \sum_{\substack{(W,j)\in\mathcal{W} \text{ with} \\ W\subseteq A, j\leq i}} g_j(W) = |\{S \in \mathcal{S}_{-1} \mid S \subseteq A\}| + \sum_{\ell=0}^{i} 2^l |\{S \in \mathcal{S}_\ell \mid S \subseteq A\}|.$$

The functions $\kappa_i^{\mathcal{W}}$ and $g_i^{\mathcal{W}}$ are thus analogous to κ_i and g_i with the difference that they only consider sets of the partitions $\mathcal{A}_i$ that are subsets of a set in the removal pattern $\mathcal{W}$. Since each edge in R crosses at least one of the sets in the efficient removal pattern $\mathcal{W}$, we obtain

$$\kappa_p^{\mathcal{W}}(V) \geq c(R). \tag{12}$$

Notice that if $(A, i) \in \mathcal{W}$ then $\kappa_i^{\mathcal{W}}(A) = \kappa_i(A)$ and $g_i^{\mathcal{W}}(A) = g_i(A)$. Furthermore, for $i \in \{0, \dots, p-1\}$ and $(A, i) \notin \mathcal{W}$ we have

$$\kappa_i^{\mathcal{W}}(A) = \sum_{C\in\mathcal{C}_i(A)} \kappa_{i-1}^{\mathcal{W}}(C), \text{ and} \tag{13}$$

$$g_i^{\mathcal{W}}(A) = \sum_{C\in\mathcal{C}_i(A)} g_{i-1}^{\mathcal{W}}(C). \tag{14}$$

The following shows a basic lower bound on $\text{val}(R)$ in terms of $g_i^{\mathcal{W}}$.

Proposition 9. *Let $\mathcal{W}$ be an efficient removal pattern and $R = R(\mathcal{W})$ the corresponding removal set. Then*

$$\text{val}(R) \geq g_p^{\mathcal{W}}(V) - 2^{p-1}.$$

Proof:

For each $i \in \{-1, \dots, p-1\}$, the number $\sigma(E_{\leq i} \setminus R)$ of connected components of $(V, E_{\leq i} \setminus R)$ is at least $|\mathcal{S}_i|$, since each $S \in \mathcal{S}_i$ is a connected component of $(V, E_{\leq i} \setminus R)$. Furthermore, only if $\mathcal{S}_i$ is a partition of V we have $\sigma(E_{\leq i} \setminus U) = |\mathcal{S}_i|$, otherwise there is at least one more connected component in $(V, E_{\leq i} \setminus R)$, and thus $\sigma(E_{\leq i} \setminus U) > |\mathcal{S}_i|$. Notice that $\mathcal{S}_{p-1}$ does not form a partition of V, since this would imply $R = U$ which contradicts $c(R) \leq B < c(U)$. Hence, $\sigma(E_{\leq p-1} \setminus U) > |\mathcal{S}_{p-1}|$ and we obtain

$$\begin{aligned}\text{val}(R) &= \sigma(E_{-1} \setminus U) - 1 + \sum_{i=0}^{p-1} 2^i \cdot (\sigma(E_{\leq i} \setminus U) - 1) \geq 2^{p-1} \cdot |\mathcal{S}_{p-1}| + |\mathcal{S}_{-1}| - 1 + \sum_{i=0}^{p-2} 2^i \cdot (|\mathcal{S}_i| - 1) \\ &= |\mathcal{S}_{-1}| + \left(\sum_{i=0}^{p-1} 2^i \cdot |\mathcal{S}_i|\right) - 2^{p-1} = g_p^{\mathcal{W}}(V) - 2^{p-1}.\end{aligned}$$

■

The following lemma relates cost and impact function for the sets U and R.

Lemma 10. *Let $\mathcal{W}$ be an efficient removal set, let $i \in \{-1, \dots, p\}$, and let $A \in \mathcal{A}_i$ such that $\kappa_i(A) > 0$. Then*

$$\frac{\kappa_i^{\mathcal{W}}(A)}{\kappa_i(A)} \cdot \left(g_i(A) - 2^i\right) \leq g_i^{\mathcal{W}}(A) + 2^i.$$

To prove Lemma 10, we need the following basic result, which is proven in Appendix B.

Lemma 11. *Let $k \in \mathbb{Z}_{>0}$, and let $a_j, b_j \geq 0$ for $j \in [k]$ be reals satisfying $\frac{a_1}{b_1} \geq \cdots \geq \frac{a_k}{b_k}$, where we interpret $\frac{a}{b} = \infty$ if $b = 0$, independent of whether $a = 0$. Let $\lambda \in [0, 1]$. Then for any $q \in [k]$ with $\left(\sum_{j=1}^{q-1} b_j\right) + \lambda b_q > 0$ we have*

$$\frac{\sum_{j=1}^{k} a_j}{\sum_{j=1}^{k} b_j} \leq \frac{\left(\sum_{j=1}^{q-1} a_j\right) + \lambda a_q}{\left(\sum_{j=1}^{q-1} b_j\right) + \lambda b_q}.$$

Proof of Lemma 10: Let $i \in \{-1, \ldots, p\}$ and $A \in \mathcal{A}_i$ such that $\kappa_i(A) > 0$. The result trivially holds if $\kappa_i^{\mathcal{W}}(A) = 0$; we thus assume $\kappa_i^{\mathcal{W}}(A) > 0$. We prove the lemma by induction on i, starting at $i = -1$. First observe that if $(A, i) \in \mathcal{W}$, then $g_i^{\mathcal{W}}(A) = g_i(A)$ and $\kappa_i^{\mathcal{W}}(A) = \kappa_i(A)$, and the result follows trivially. This observation also covers the base case $i = -1$ of the induction as $\kappa_{-1}^{\mathcal{W}}(A) > 0$ implies $(A, -1) \in \mathcal{W}$.

Thus, we assume from now on $i > -1$ and $(A, i) \notin \mathcal{W}$. Since $\kappa_i^{\mathcal{W}}(A) > 0$, the efficient removal pattern $\mathcal{W}$ contains at least one descendant of (A, i). Hence, point (ii) of the definition of an efficient removal pattern, i.e., Definition 8, holds for $A \in \mathcal{A}_i$. Let $\mathcal{C}_i(A) = \{Q_1, \ldots, Q_h\}$, where the numbering is chosen according to Definition 8, and let $s \in \{0, \ldots, h\}$ be the index as claimed by Definition 8.

Using (9), we deduce

$$\begin{aligned}\frac{\kappa_i^{\mathcal{W}}(A)}{\kappa_i(A)}\left(g_i(A) - 2^i\right) &= \frac{\kappa_i^{\mathcal{W}}(A)}{\kappa_i(A)} \sum_{j=1}^{h} g_{i-1}(Q_j) && \text{(by (9))} \\ &\leq \frac{\kappa_i^{\mathcal{W}}(A)}{\sum_{j=1}^{h} \kappa_{i-1}(Q_j)} \sum_{j=1}^{h} g_{i-1}(Q_j). && \text{(by (8))} \qquad (15)\end{aligned}$$

In a next step we will apply Lemma 11 with parameters $q = \min\{s+1, h\}$ and $\lambda = \kappa_{i-1}^{\mathcal{W}}(Q_q)/\kappa_{i-1}(Q_q)$ to the ratio $\sum_{j=1}^{h} g_{i-1}(Q_j) / \sum_{j=1}^{h} \kappa_{i-1}(Q_j)$ in (15), i.e., the terms in the terminology of Lemma 11 are $a_j = g_{i-1}(Q_j)$ and $b_j = \kappa_{i-1}(Q_j)$ for $j \in [h]$. To do so, we first check that the conditions of Lemma 11 are fulfilled. More precisely, we have to show that:

(i) λ is well defined, i.e., $\kappa_{i-1}(Q_q) > 0$,
(ii) $\lambda \in [0, 1]$, and
(iii) $(\sum_{j=1}^{q-1} \kappa_{i-1}(Q_j)) + \lambda \kappa_{i-1}(Q_q) > 0$.

First observe that since $(A, i) \notin \mathcal{W}$ we have

$$\sum_{j=1}^{q} g_{i-1}^{\mathcal{W}}(Q_j) = \sum_{j=1}^{h} g_{i-1}^{\mathcal{W}}(Q_j) = g_i^{\mathcal{W}}(A), \qquad \text{(second equality follows by (14))} \qquad (16)$$

$$\sum_{j=1}^{q} \kappa_{i-1}^{\mathcal{W}}(Q_j) = \sum_{j=1}^{h} \kappa_{i-1}^{\mathcal{W}}(Q_j) = \kappa_i^{\mathcal{W}}(A), \qquad \text{(second equality follows by (13))} \qquad (17)$$

where the first equality in the above statements follows from $\kappa_{i-1}^{\mathcal{W}}(Q_j) = 0 = g_{i-1}^{\mathcal{W}}(Q_j)$ for $j \in \{q+1, \ldots, h\}$, since none of the sets $Q_{q+1}, \ldots, Q_h$ or any of its descendants are contained in $\mathcal{W}$, by definition of an efficient removal pattern.

Notice that $\kappa_i^{\mathcal{W}}(A) > 0$ implies by (17) that there is a $\bar{j} \in [q]$ such that $0 < \kappa_{i-1}^{\mathcal{W}}(Q_{\bar{j}}) \leq \kappa_{i-1}(Q_{\bar{j}})$, and hence $\rho_{i-1}(Q_{\bar{j}}) < \infty$. Because the auxiliary efficiencies $\rho_{i-1}(Q_j)$ are nonincreasing in j, we have $\rho_{i-1}(Q_q) < \infty$ which is equivalent to $\kappa_{i-1}(Q_q) > 0$. Hence, λ is well defined and since $\kappa_{i-1}(Q_q) \geq \kappa_{i-1}^{\mathcal{W}}(Q_q)$ we have $\lambda \in [0, 1]$. Furthermore,

$$0 < \kappa_i^{\mathcal{W}}(A) = \sum_{j=1}^{q} \kappa_{i-1}^{\mathcal{W}}(Q_j) = \left(\sum_{j=1}^{q-1} \kappa_{i-1}(Q_j)\right) + \lambda \kappa_{i-1}(Q_q),$$

where the first equality follows from (17). We can thus apply Lemma 11 to the ratio in (15) to obtain

$$\frac{\sum_{j=1}^{h} g_{i-1}(Q_j)}{\sum_{j=1}^{h} \kappa_{i-1}(Q_j)} \leq \frac{\left(\sum_{j=1}^{q-1} g_{i-1}(Q_j)\right) + \lambda g_{i-1}(Q_q)}{\left(\sum_{j=1}^{q-1} \kappa_{i-1}(Q_j)\right) + \lambda \kappa_{i-1}(Q_q)} = \frac{\left(\sum_{j=1}^{q-1} g_{i-1}^{\mathcal{W}}(Q_j)\right) + \lambda g_{i-1}(Q_q)}{\sum_{j=1}^{q} \kappa_{i-1}^{\mathcal{W}}(Q_j)}, \tag{18}$$

where the equality follows by the definition of λ in the denominator, and by using the observation that $(Q_j, i-1) \in \mathcal{W}$ for $j \in \{1, \ldots, q-1\}$, which implies $g_{i-1}^{\mathcal{W}}(Q_j) = g_{i-1}(Q_j)$ and $\kappa_{i-1}^{\mathcal{W}}(Q_j) = \kappa_{i-1}(Q_j)$. We thus obtain

$$\begin{aligned}
\frac{\kappa_i^{\mathcal{W}}(A)}{\kappa_i(A)} \left(g_i(A) - 2^i\right) &\leq \frac{\kappa_i^{\mathcal{W}}(A)}{\sum_{j=1}^{h} \kappa_{i-1}(Q_j)} \sum_{j=1}^{h} g_{i-1}(Q_j) && \text{(by (15))} \\
&\leq \frac{\sum_{j=1}^{q} \kappa_{i-1}^{\mathcal{W}}(Q_j)}{\sum_{j=1}^{h} \kappa_{i-1}(Q_j)} \sum_{j=1}^{h} g_{i-1}(Q_j) && \text{(by (17))} \\
&\leq \left(\sum_{j=1}^{q-1} g_{i-1}^{\mathcal{W}}(Q_j)\right) + \lambda g_{i-1}(Q_q). && \text{(by (18))}
\end{aligned}$$

Applying the induction hypothesis to $\lambda(g_{i-1}(Q_q) - 2^{i-1}) = \frac{\kappa_{i-1}^{\mathcal{W}}(Q_q)}{\kappa_{i-1}(Q_q)}(g_{i-1}(Q_q) - 2^{i-1})$ we get

$$\begin{aligned}
\lambda g_{i-1}(Q_q) &\leq g_{i-1}^{\mathcal{W}}(Q_q) + 2^{i-1}(1+\lambda) && \text{(induction hypothesis)} \\
&\leq g_{i-1}^{\mathcal{W}}(Q_q) + 2^{i}, && (\lambda \leq 1)
\end{aligned}$$

and hence

$$\begin{aligned}
\frac{\kappa_i^{\mathcal{W}}(A)}{\kappa_i(A)}(g_i(A) - 2^i) &\leq \left(\sum_{j=1}^{q} g_{i-1}^{\mathcal{W}}(Q_j)\right) + 2^i \\
&= g_i^{\mathcal{W}}(A) + 2^i, && \text{(by (16))}
\end{aligned}$$

thus proving the lemma. ■

Lemma 12. *Let $\mathcal{W}$ be an efficient removal pattern with corresponding removal set $R = R(\mathcal{W})$. Then*

$$g_p^{\mathcal{W}}(V) \geq \frac{1}{2}\frac{c(R)}{c(U)}\,\mathrm{val}(U) - 2^p.$$

Proof: The statement follows from

$$\begin{aligned}
g_p^{\mathcal{W}}(V) &\geq \frac{\kappa_p^{\mathcal{W}}(V)}{\kappa_p(V)} \cdot (g_p(V) - 2^p) - 2^p && \text{(by Lemma 10)} \\
&\geq \frac{\kappa_p^{\mathcal{W}}(V)}{\kappa_p(V)}\,\mathrm{val}(U) - 2^p && (g_p(V) - 2^p \geq g_p(V) - 2^{p+1} = \mathrm{val}(U) \text{ by (10)}) \\
&= \frac{1}{2}\frac{\kappa_p^{\mathcal{W}}(V)}{c(U)}\,\mathrm{val}(U) - 2^p && \text{(by (11))} \\
&\geq \frac{1}{2}\frac{c(R)}{c(U)}\,\mathrm{val}(U) - 2^p && \text{(by (12)).}
\end{aligned}$$

■

Combining Proposition 9 and Lemma 12 we obtain the following.

Corollary 13. *Let $\mathcal{W}$ be an efficient removal pattern with corresponding removal set $R = R(\mathcal{W})$. Then*

$$\mathrm{val}(R) \geq \frac{1}{2}\frac{c(R)}{c(U)}\,\mathrm{val}(U) - 3 \cdot 2^{p-1}.$$

Now consider the interdiction set R returned by Algorithm 2. If $c(R) = B$, Corollary 13 implies Theorem 3. However, it may be that $c(R)$ only uses a very small fraction of the available budget. To prove Theorem 3 we will show how one can get around this problem by finding another efficient removal pattern $\mathcal{W}'$ that is over budget and whose value can be related to $\mathrm{val}(R)$.

Proof of Theorem 3: We will construct an efficient removal pattern $\mathcal{W}'$ with corresponding removal set $R' = R(\mathcal{W}')$ satisfying the following two conditions:

(i) $c(R') \geq B$, and
(ii) $g_p^{\mathcal{W}}(V) \geq g_p^{\mathcal{W}'}(V) - 2^{p-1}$.

First observe that the existence of $\mathcal{W}'$ indeed implies Theorem 3 since

$$\begin{aligned}
\mathrm{val}(R) &\geq g_p^{\mathcal{W}}(V) - 2^{p-1} && \text{(by Proposition 9)}\\
&\geq g_p^{\mathcal{W}'}(V) - 2^{p} && \text{(using (ii))}\\
&\geq \frac{1}{2} \cdot \frac{c(R')}{c(U)}\,\mathrm{val}(U) - 2^{p+1} && \text{(by Lemma 12 applied to } \mathcal{W}')\\
&\geq \frac{1}{2} \cdot \frac{B}{c(U)}\,\mathrm{val}(U) - 2^{p+1}. && \text{(using (i))}
\end{aligned}$$

It remains to show that an efficient removal pattern $\mathcal{W}'$ with the desired properties (i) and (ii) exists. We define $\mathcal{W}'$ in terms of $\mathcal{W}$. Consider the construction of $\mathcal{W}$ through Algorithm 2. Let $\ell \in \{-1, \ldots, p-1\}$ be the last iteration of the while loop of Algorithm 2 where the index s was not equal to h, i.e., the maximum possible value in that iteration. Hence, this corresponds to the lowest value of ℓ in which iteration we have $s \neq h$. Note that there must have been an iteration where $s \neq h$ since for otherwise $R = U$ which violates the fact that R is an interdiction set because $c(U) > B$.

Let $A \in \mathcal{A}_{\ell+1}$ be the set considered by Algorithm 2 at the beginning of iteration ℓ, and let $\mathcal{C}_\ell(A) = \{Q_1, \ldots, Q_h\}$ be the numbering of the children of A considered in that iteration. Moreover, we denote by $\overline{\mathcal{W}}$ the set $\mathcal{W}$ at the beginning of iteration ℓ. We recall that s is defined by

$$s = \max\left\{ j \in \{0, \ldots, h\} \;\middle|\; c\left(\overline{\mathcal{W}} \cup \{(Q_k, \ell) \mid k \in [j]\}\right) \leq B \right\}.$$

Let

$$\mathcal{W}' = \overline{\mathcal{W}} \cup \{(Q_k, \ell) \mid k \in [s+1]\}.$$

Clearly, $\mathcal{W}'$ is an efficient removal pattern. Furthermore, the removal set $R' = R(\mathcal{W})$ satisfies condition (i), i.e., $c(R') > B$, by definition of s. It remains to show that (ii) holds.

Notice that either $\ell = -1$, or all children of (Q_{s+1}, ℓ) are added to $\mathcal{W}$ as sets on level $\ell - 1$, which follows from the fact that ℓ was the last iteration of Algorithm 2 in which not all children were added to $\mathcal{W}$. Moreover, $\mathcal{W}$ contains no sets on levels $-1, \ldots, \ell - 2$: This clearly holds if $\ell = -1$; otherwise, Algorithm 2 left the while loop after having added all children of Q_{s+1}. Hence, $\mathcal{W}$ and $\mathcal{W}'$ are almost identical with the only difference that $\mathcal{W}'$ contains (Q_{s+1}, ℓ), which is not contained in $\mathcal{W}$ and, if $\ell \neq -1$, then $\mathcal{W}$ contains all children of (Q_{s+1}, ℓ), which are not contained in $\mathcal{W}'$. This implies

$$g_p^{\mathcal{W}'}(V) = g_p^{\mathcal{W}}(V) + \max\{1, 2^\ell\}.$$

Point (ii) now follows by observing that $\ell \leq p - 1$ (and $p \geq 1$).

■

VI. An $O(1)$-approximation for metric TSP interdiction

We consider the metric TSP problem as highlighted in Section I. We recall that we are given an undirected connected graph $G = (V, E)$ with edge lengths $\ell : E \to \mathbb{Z}_{>0}$ and the goal is to find a shortest closed walk that visits each vertex at least once. In its interdiction version, every edge is also given an interdiction cost $c : E \to \mathbb{Z}_{>0}$, and there is a global budget $B \in \mathbb{Z}_{>0}$. The goal of metric TSP interdiction is to find a set $R \subseteq E$ of edges to interdict with $c(R) \leq B$, such that the length of a shortest closed walk in $(V, E \setminus R)$ that visits each vertex at least once is as large as possible.

For any set $U \subseteq E$, we denote by $\mathrm{TSP}(U)$ the length of a shortest closed walk in $(V, E \setminus U)$ visiting each vertex at least once. To avoid trivial cases we assume that the graph cannot be disconnected by removing an interdiction set, i.e., for any $R \subseteq E$ with $c(R) \leq B$, the graph $(V, E \setminus R)$ is connected. Formally, metric TSP interdiction can be described as follows:

$$\max\{\mathrm{TSP}(R) \mid R \subseteq E, c(R) \leq B\}. \tag{19}$$

The following result now easily follows by the fact that $\mathrm{MST}(U)$ and $\mathrm{TSP}(U)$ are at most a factor of 2 apart.

Theorem 14. *Let $R \subseteq E$ be an interdiction set obtained by applying an α-approximation to the MST interdiction problem defined on the graph G with weights given by ℓ, interdiction costs given by c, and budget B. Then R is a 2α-approximation for metric TSP interdiction.*

Proof: First observe that for any interdiction set $U \subseteq E$, we have

$$\mathrm{TSP}(U) \geq \mathrm{MST}(U), \tag{20}$$

because any solution to $\mathrm{TSP}(U)$ must connect all vertices and therefore contains a spanning tree. Furthermore, we also have for any interdiction set $U \subseteq E$,

$$\mathrm{TSP}(U) \leq 2\,\mathrm{MST}(U), \tag{21}$$

because doubling a spanning tree leads to a closed walk that visits all vertices. This corresponds to the well-known Double-Tree Algorithm which 2-approximates metric TSP (see [25]). Let R^*_{MST} and R^*_{TSP} be optimal solutions to the MST interdiction problem and the metric TSP interdiction problem on G, respectively. We thus obtain that our α-approximation R for the MST interdiction problem satisfies

$$\begin{aligned}
\mathrm{TSP}(R) &\geq \mathrm{MST}(R) && \text{(by (20))}\\
&\geq \frac{1}{\alpha}\,\mathrm{MST}(R^*_{\mathrm{MST}}) && \text{(R is an α-approximation for MST interdiction)}\\
&\geq \frac{1}{\alpha}\,\mathrm{MST}(R^*_{\mathrm{TSP}}) && \text{(R^*_{MST} is an optimal solution for MST interdiction)}\\
&\geq \frac{1}{2\alpha}\,\mathrm{TSP}(R^*_{\mathrm{TSP}}). && \text{(by (21))}
\end{aligned}$$

■

Finally, Theorem 2 follows from Theorem 14 and Theorem 1, our 14-approximation for MST interdiction.

Acknowledgements

We are grateful to Chandra Chekuri, R. Ravi, and the anonymous reviewers for many helpful comments.

References

[1] N. Assimakopoulos. A network interdiction model for hospital infection control. *Computers in Biology and Medicine*, 17(6):413–422, 1987.

[2] M.O. Ball, B. Golden, and R. V. Vohra. Finding the most vital arcs in a network. *Operations Research Letters*, 8:73–76, 1989.

[3] C. Bazgan, S. Toubaline, and Z. Tuza. The most vital nodes with respect to independent set and vertex cover. *Discrete Applied Mathematics*, 159(17):1933 – 1946, 2011.

[4] C. Bazgan, S. Toubaline, and D. Vanderpooten. Efficient determination of the k most vital edges for the minimum spanning tree problem. *Computers and Operations Research*, 39(11):2888–2898, 2012.

[5] C. Burch, R. Carr, S. Krumke, M. Marathe, C. Phillips, and E. Sundberg. A decomposition-based pseudoapproximation algorithm for network flow inhibition. In *Network Interdiction and Stochastic Integer Programming*, chapter 3, pages 51–68. Springer, 2003.

[6] B. Chazelle. A minimum spanning tree algorithm with inverse-ackermann type complexity. *Journal of the ACM*, 47(6):1028–1047, 2000.

[7] C. Chekuri, S. Guha, and J. Naor. The steiner k-cut problem. *SIAM Journal on Discrete Mathematics*, 20(1):261–271, 2006.

[8] R. L. Church, M. P. Scaparra, and R. S. Middleton. Identifying critical infrastructure: the median and covering facility interdiction problems. *Annals of the Association of American Geographers*, 94(3):491–502, 2004.

[9] M. Dinitz and A. Gupta. Packing interdiction and partial covering problems. In *Proceedings of 16th Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 157–168, 2013.

[10] B. Dixon, M. Rauch, and R. E. Tarjan. Verification and sensitivity analysis of minimum spanning trees in linear time. *SIAM Journal on Computing*, 21(6):1184–1192, 1992.

[11] A. Engelberg, J. Könemann, S. Leonardi, and J. Naor. Cut problems in graphs with a budget constraint. *Journal of Discrete Algorithms*, 5:262–279, 2007.

[12] L. Fleischer and S. Iwata. A push-relabel framework for submodular function minimization and applications to parametric optimization. *Discrete Applied Mathematics*, 131(2):311 – 322, 2003.

[13] G. N. Frederickson and R. Solis-Oba. Increasing the weight of minimum spanning trees. *Journal of Algorithms*, 33:244–266, 1999. Extended abstract appeared in *Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1996.

[14] P. M. Ghare, D. C. Montgomery, and W. C. Turner. Optimal interdiction policy for a flow network. *Naval Research Logistics Quarterly*, 18:37–45, 1971.

[15] O. Goldschmidt and D. S. Hochbaum. A polynomial algorithm for the k-cut problem for fixed k. *Mathematics of Operations Research*, 19(1):24–37, 1994.

[16] F. Grandoni, R. Ravi, M. Singh, and R. Zenklusen. New approaches to multi-objective optimization. *Mathematical Programming, Series A*, 146(1):525–554, 2014.

[17] J. Guo and Y. Shrestha. Parameterized complexity of edge interdiction problems. In *Proceedings of the 20th Computing and Combinatorics Conference (COCOON)*, pages 166–178, 2014.

[18] L.-H. Hsu, R.-H. Jan, Y.-C. Lee, C.-N. Hung, and M.-S. Chern. Finding the most vital edge with respect to minimum spanning tree in weighted graphs. *Information Processing Letters*, 39(5):277–281, 1991.

[19] K. Iwano and N. Katoh. Efficient algorithms for finding the most vital edge of a minimum spanning tree. *Information Processing Letters*, 48(5):211 – 213, 1993.

[20] A. Jüttner. On budgeted optimization problems. *SIAM Journal on Discrete Mathematics*, 20(4):880–892, 2006.

[21] D. R. Karger, P. N. Klein, and R. E. Tarjan. A randomized linear-time algorithm to find minimum spanning trees. *Journal of the ACM*, 42(2):321–328, 1995.

[22] L. Khachiyan, E. Boros, K. Borys, K. Elbassioni, V. Gurvich, G. Rudolf, and J. Zhao. On short paths interdiction problems: Total and node-wise limited interdiction. *Theoretical Computer Science*, 43(2):204–233, 2008.

[23] V. King. A simpler minimum spanning tree verification algorithm. *Algorithmica*, 18(2):263–270, 1997.

[24] P. N. Klein and R. E. Tarjan. A randomized linear-time algorithm for finding minimum spanning trees. In *In Proceedings of the 26th Annual ACM Symposium on Theory of Computing (STOC)*, pages 9–15, 1994.

[25] B. Korte and J. Vygen. *Combinatorial Optimization, Theory and Algorithms*. Springer, 5th edition, 2012.

[26] W. Liang. Finding the k most vital edges with respect to minimum spanning trees for fixed k. *Discrete Applied Mathematics*, 113(2-3):319–327, 2001.

[27] W. Liang and X. Shen. Finding the k most vital edges in the minimum spanning tree problem. *Parallel Computing*, 23(13):1889–1907, 1997.

[28] D. P. Morton, F. Pan, and K. J. Saeger. Models for nuclear smuggling interdiction. *IEEE Transactions*, 39:3–14, 2007.

[29] K. Nagano. A faster parametric submodular function minimization algorithm and applications. Technical report, University of Tokyo, 2007. METR 2007-43.

[30] J. Naor and Y. Rabani. Tree packing and approximating k-cuts. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 26–27, 2001.

[31] J. B. Orlin. A faster strongly polynomial time algorithm for submodular function minimization. *Mathematical Programming*, 118:237–251, 2009.

[32] S. Pettie. Sensitivity analysis of minimum spanning trees in sub-inverse-Ackermann time. In *Proceedings of the 16th International Symposium on Algorithms and Computation (ISAAC)*, pages 964–973. 2005.

[33] C. A. Phillips. The network inhibition problem. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing (STOC)*, pages 776–785, 1993.

[34] R. Ravi and M. X. Goemans. The constrained minimum spanning tree problem. In *Proceedings of 5th Scandinavian Workshop on Algorithm Theory (SWAT)*, pages 66–75, 1996.

[35] R. Ravi and A. Sinha. Approximating k-cut using network strength as a Lagrangean relaxation. *European Journal of Operational Research*, 186:77–90, 2008. 2008.

[36] J. Salmeron, K. Wood, and R. Baldick. Worst-case interdiction analysis of large-scale electric power grids. *IEEE Transactions on Power Systems*, 24(1):96–104, 2009.

[37] A. Schrijver. On the history of the transportation and maximum flow problems. *Mathematical Programming*, 91(3):437–445, 2002.

[38] A. Schrijver. *Combinatorial Optimization, Polyhedra and Efficiency*. Springer, 2003.

[39] H. Shen. Finding the k most vital edges with respect to minimum spanning tree. *Acta Informatica*, 36(5):405–424, 1999.

[40] R. E. Tarjan. Applications of path compression on balanced trees. *Journal of the ACM*, 26(4):690–715, 1979.

[41] R. K. Wood. Deterministic network interdiction. *Mathematical and Computer Modeling*, 17(2):1–18, 1993.

[42] R. Zenklusen. Matching interdiction. *Discrete Applied Mathematics*, 158(15):1676–1690, 2010.

[43] R. Zenklusen. Network flow interdiction on planar graphs. *Discrete Applied Mathematics*, 158(13):1441–1455, 2010.

[44] R. Zenklusen. Connectivity interdiction. *Operations Research Letters*, 42(67):450 – 454, 2014.

APPENDIX A. RELATION TO GRAPH DISCONNECTION PROBLEMS

The k-cut problem is closely related to MST interdiction through its budgeted version, the maximum components problem (MCP). We recall that MCP asks to break a graph $G = (V, E)$ into a maximum number of connected components by removing a given number q of edges. The following is a simple way to reduce MCP to an MST interdiction problem: Set $c(e) = 1, w(e) = 0 \ \forall e \in E$, set the budget $B = q$, and add to G a set of $|V| - 1$ edges T forming a spanning tree; for $e \in T$ we set $w(e) = 1$ and make sure that these edges cannot be interdicted by setting $c(e) = B + 1$. One can easily check that this reduction preserves objective values. Another reduction that does not preserve the objective values has been presented in [13]. A generalization of MCP, where edges have interdiction costs, was considered in [11] and called the *budgeted graph disconnection* (BGD) problem. These budgeted versions of the k-cut problem admit $O(1)$-approximations by extending ideas for $O(1)$-approximations for k-cut [13], [11].

APPENDIX B. PROOF OF LEMMA 11

We start by observing that we can assume $b_j > 0$ for $j \in [k]$. Otherwise one can remove all pairs a_j, b_j with $b_j = 0$ from the sequence. Doing so leads to a sharper statement since the left-hand side of the inequality claimed by the lemma decreases at most as much as its right-hand side. Hence, assume $b_j > 0$ for $j \in [k]$.

For brevity we define $r_j = \frac{a_j}{b_j}$ for $j \in [k]$. If $q = k$ and $\lambda = 1$, the statement trivially holds. Hence, assume that either $q < k$ or $\lambda < 1$. We define the following expressions β and γ, where the denominator of γ must be strictly positive since either $q < k$ or $\lambda < 1$:

$$\beta = \frac{\left(\sum_{j=1}^{q-1} b_j r_j\right) + \lambda b_q r_q}{\left(\sum_{j=1}^{q-1} b_j\right) + \lambda b_q} = \frac{\left(\sum_{j=1}^{q-1} a_j\right) + \lambda a_q}{\left(\sum_{j=1}^{q-1} b_j\right) + \lambda b_q}, \qquad \gamma = \frac{(1-\lambda) b_q r_q + \sum_{j=q+1}^{k} b_j r_j}{(1-\lambda) b_q + \sum_{j=q+1}^{k} b_j}.$$

Notice that β can be interpreted as a convex combination of $r_1, \dots r_q$, and since $r_1 \geq \cdots \geq r_q$, we have $\beta \geq r_q$. Similarly, γ is a convex combination of $r_q, \dots, r_k$, and hence $\gamma \leq r_q$. Thus, $\beta \geq \gamma$. The result now follows by

$$\frac{\sum_{j=1}^{k} a_j}{\sum_{j=1}^{k} b_j} = \frac{\sum_{j=1}^{k} b_j r_j}{\sum_{j=1}^{k} b_j} = \frac{1}{\sum_{j=1}^{k} b_j} \left[\left(\left(\sum_{j=1}^{q-1} b_j \right) + \lambda b_q \right) \beta + \left((1-\lambda) b_q + \sum_{j=q+1}^{k} b_j \right) \gamma \right] \leq \beta \ ,$$

where the inequality follows by upper bounding γ by β.

APPENDIX C.
DETAILS ON ERRONEOUS CLAIM IN [39]

The article [39] presents several algorithms for the k most vital edges problem for MST. In particular, they claim to present a 2-approximation. However, their results are based on an erroneous claim about spanning trees, which is stated as Lemma 2 in [39]. In this section, after introducing some basic notions used in [39], we state Lemma 2 of [39] and provide a counterexample for it. Furthermore, we give a brief explanation of why the proof of Lemma 2 that is presented in [39] is erroneous.

Let $G = (V, E)$ be an undirected graph with edge weights $w : E \to \mathbb{Z}_{\geq 0}$, and let $k \in \mathbb{Z}_{>0}$. All edge weights are assumed to be distinct, and hence, the MST is unique, also in any connected subgraph of G. Furthermore, we assume that G is $(k+1)$-edge-connected to avoid the trivial case that the graph can be disconnected. Let $T \subseteq E$ be the unique MST in G. For each $e \in T$, let

$$R(e) = \{f \in E \mid (T \cup \{f\}) \setminus \{e\} \text{ is a spanning tree}\}.$$

In [39], the edges in $R(e)$ are called *replacement edges for* e since they can replace e in T to obtain again a spanning tree. Furthermore $R_e \subseteq R(e)$ is the set containing the k lightest edges in $R(e)$, i.e., these are the k lightest replacement edges for e. Moreover, let $R = \cup_{e \in T} R_e$. We are now ready to state the erroneous lemma in [39].

Lemma 2 in [39]. *Let K be an optimal solution for the k most vital edges problem for MST. Then*

$$K \subseteq T \cup R.$$

The weighted graph depicted in Figure 3 is a counterexample to the above Lemma.

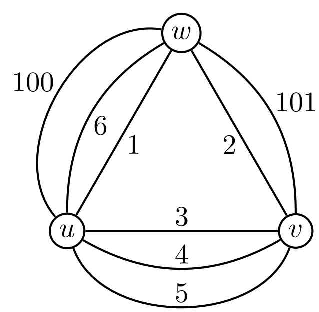

Figure 3: A counterexample to Lemma 2 in [39] for $k = 3$.

Its minimum spanning tree consists of the edges of weight 1 and 2. For each of these edges, the three best replacement edges are the edges of weight 3, 4, and 5. No matter which three edges are removed among the edges of weight 1, 2, 3, 4, and 5, there is always a spanning tree left that uses neither of the two edges of weight 100 and 101, respectively. However, removing the edges of weight 1,2, and 6, leads to a graph whose minimum spanning tree contains the edge of weight 100.

Notice that the example in Figure 3 can easily be converted to a simple graph (i.e., without parallel edges). For example, this can be done by replacing each of the three vertices by a clique of size 5, where all edges in the clique have very low weight and thus are not worth being removed; because no matter which 3 edges get removed, the vertices of any clique can still be connected by low weight edges within the clique. Each remaining edge connects the two cliques that correspond to its endpoints, where it does not matter to which particular vertex of a clique an edge is connected to, as long as no parallel edges are created. Clearly, the edges can be placed in a way to obtain a simple graph.

The main mistake in the proof of Lemma 2 presented in [39] is the assumption that for any subset $U \subseteq T$, one can simultaneously replace in T each edge $e \in U$ by an edge in $R(e)$, still obtaining a spanning tree.