On short paths interdiction problems: Total and node-wise limited interdiction.

Leonid Khachiyan[a] Endre Boros[b] Konrad Borys[c]
Khaled Elbassioni[c] Vladimir Gurvich[c]
Gabor Rudolf[c] Jihui Zhao[d]

RRR 25-2006, October 2006

[a]Our co-author Leonid Khachiyan passed away with tragic suddenness on April 29th, 2005.

[b]RUTCOR, Rutgers University, 640 Bartholomew Road, Piscataway, New Jersey 08854; ({boros,kborys,gurvich,grudolf}@rutcor.rutgers.edu).

[c]Max Plamck Institute fur Informatik, Saarbruken, Germany; (elbassio@mpi-sb.mpg.de).

[d]Department of Computer Science, Rutgers University, 110 Frelinghuysen Road, Piscataway, New Jersey 08854. (zhaojih@cs.rutgers.edu).

Rutcor Research Report
RRR 25-2006, October 2006

On short paths interdiction problems: Total and node-wise limited interdiction.

Leonid Khachiyan   Endre Boros   Konrad Borys
Khaled Elbassioni   Vladimir Gurvich   Gabor Rudolf   Jihui Zhao

Abstract. Given a directed graph $G = (V, A)$ with a non-negative weight (length) function on its arcs $w : A \to \mathbb{R}_+$ and two terminals $s, t \in V$, our goal is to destroy all short directed paths from s to t in G by eliminating some arcs of A. This is known as the *short paths interdiction problem*. We consider several versions of it, and in each case analyze two subcases: *total limited interdiction*, when a fixed number k of arcs can be removed, and *node-wise limited interdiction*, when for each node $v \in V$ a fixed number $k(v)$ of out-going arcs can be removed. Our results indicate that the latter subcase is always easier than the former one. In particular, we show that the short paths node-wise interdiction problem can be efficiently solved by an extension of Dijkstra's algorithm. In contrast, the short paths total interdiction problem is known to be NP-hard. We strengthen this hardness result by deriving the following inapproximability bounds: Given k, it is NP-hard to approximate within a factor $c < 2$ the maximum $s - t$ distance $d(s, t)$ obtainable by removing (at most) k arcs from G. Furthermore, given d, it is NP-hard to approximate within a factor $c < 10\sqrt{5} - 21 \approx 1.36$ the minimum number of arcs which has to be removed to guarantee $d(s, t) \geq d$. Finally, we also show that the same inapproximability bounds hold for non-directed graphs and/or node elimination.
Keywords: approximation algorithm, Dijkstra's algorithm, most vital arcs problem, cyclic game, maxmin mean cycle, minimal vertex cover, network inhibition, network interdiction.

Acknowledgements: This research was supported in part by NSF grant IIS-0118635 and by DIMACS, the NSF Center for Discrete Mathematics & Theoretical Computer Science. Preprints DTR-2005-04 and DTR-2006-13 are available at http://dimacs.rutgers.edu/TechnicalReports/2005.html and 2006.html .

1 Introduction

1.1 Node-wise limited interdiction

Let $G = (V, A)$ be a directed graph (digraph) with given arc-weights $w(e)$, $e \in A$. For each vertex $v \in V$, we are allowed to delete (remove, block, interdict) a subset $X(v)$ of the arcs $A(v) = \{e \in A \mid e = (v, u)\}$ leaving v. We assume that these arc-sets $X(v) \subseteq A(v)$ are selected for all vertices $v \in V$ independently, and we call the collection $\mathcal{B}(v)$ of all admissible arc-sets $X(v)$ a *blocking system* at v. We also assume that for each v, the family $\mathcal{B}(v)$ forms an *independence system*, i.e., if $X(v) \in \mathcal{B}(v)$ is an admissible arc-set at v, then so is any subset of $X(v)$. Hence, we could replace $\mathcal{B}(v)$ by the collection of all inclusion-wise maximal admissible arc-sets. In general, we will only assume that the blocking system $\mathcal{B}(v)$ is given by a *membership oracle* $\mathcal{O}$:

($\mathcal{B}_0$) Given a list $X(v)$ of out-going arcs for some vertex v, the oracle can determine whether or not the arcs in the list belong to $\mathcal{B}(v)$ and hence can be simultaneously deleted.

A similar formalization of blocking sets via membership oracles is used by Pisaruk [37]. We will also consider two special types of blocking systems:

($\mathcal{B}_1$) The blocking system is given by a function $k(v) : V \to \mathbb{Z}_+$, where $k(v) \leq |A(v)| =$ out-deg(v). For each vertex v, we can delete any collection of (at most) $k(v)$ arcs leaving v. The numbers $k(v)$ define *digraphs with prohibitions* considered by Karzanov and Lebedev [30].

($\mathcal{B}_2$) There are two types of vertices: *control vertices*, where we can select any out-going arc $e \in A(v)$ and block all the remaining arcs in $A(v)$, and *regular vertices*, where we can block no arc. This case, considered in [6, 23], is a special case of $\mathcal{B}_1$: $k(v) = |A(v)| - 1$ for control vertices, and $k(v) = 0$ otherwise.

We call a digraph $G' = (V, A')$ *admissible* for $G = (V, A)$ if $A' \subseteq A$ is obtained by deleting some admissible subsets $X(v) \in \mathcal{B}(v)$ of outgoing arcs for each vertex $v \in V$.

1.2 Interdiction of directed cycles

We proceed with an obvious observation. Let $G = (V, A)$ be a directed graph (digraph) and our goal is to destroy all directed cycles in G. In this case the total limited interdiction problem is stated as follows. Is it possible to destroy all directed cycles of G by eliminating (at most) k arcs of A, or in other words, whether G has a feedback arc set of size (at most) k? In 1972 Karp [28] proved that this decision problem is NP-hard.

Node-wise limited cycle interdiction problem of type $\mathcal{B}_1$ can be stated as follows. Is it possible to destroy all directed cycles in G by deleting for each node $v \in V$ (at most) $k(v)$ arcs going from v? This problem is trivial. Indeed, if $k(v) <$ out-deg(v) for each $v \in V$ then the answer is negative, since for each $v \in V$ at least one out-going arc will remain and they will form a directed cycle, since G is finite. If $k(v) \geq$ out-deg(v) for a vertex $v \in V$

then obviously all out-going arcs from v should be removed, since after this we can delete the node v itself together with all in-going arcs. Repeating this simple procedure recursively we get a linear time algorithm for the node-wise limited directed cycle interdiction problem. The above algorithm can obviously be generalized for the interdiction of type $\mathcal{B}_0$.

1.3 Interdiction of negative directed cycles and mean payoff games

Now let us assume that weights may be negative and consider the negative cycle interdiction problem. Clearly, the total limited version of it is NP-hard, since if all weights are negative then all directed cycles are negative too, and we obtain the previous NP-hard problem as a special case.

In contrast, the node-wise limited negative directed cycles interdiction is equivalent to a famous open problem, solving mean payoff games, [12, 13, 23, 34, 35], that is very unlikely NP-hard, since it is known to be in NP $\cap$ co-NP, [30], and there is a sub-exponential algorithm for it, [6] and [26].

A *mean payoff game* is a zero-sum game played by two players on a finite arc-weighted digraph G all vertices of which have positive out-degrees, in other words, there are no dead-ends in G. The vertices of G (*positions*) are partitioned into two sets controlled by two players, who move a chip along the arcs of the digraph, starting from a given vertex $s \in V$ (the *initial position*). A *positional strategy* of a player is a mapping which assigns an out-going arc to each of his positions. If both players select positional strategies then the sequence of moves (the *play*) starts in s and settles on a simple directed cycle of G whose average arc-weight is called the *effective payoff* corresponding to the selected strategies.

Ehrenfeucht and Mycielski [12, 13] and Moulin [34, 35] introduced mean payoff games on bipartite digraphs and proved the existence of the value for such games in positional strategies. Gurvich, Karzanov, and Khachiyan [23] extended this result to arbitrary digraphs and suggested a potential-reduction algorithm to compute the value and optimal positional strategies of the players. In many respects this algorithm for mean payoff games is similar to the simplex method for linear programming.

Let us assume that the vertices assigned to the maximizing (respectively, to the minimizing) player are controlled (respectively, regular) vertices for $\mathcal{B}_2$. Then the determination of an optimal positional strategy for the maximizing player reduces to computing a $\mathcal{B}_2$-admissible digraph $G = (V, A')$ that maximizes the minimum average arc-cost for the cycles reachable from the initial position s. Beffara and Vorobyov [4] report on computational experiments with the potential-reduction algorithm [23] in which it was used to solve very large instances of mean payoff games. However, for some special instances with exponentially large arc-weights, this algorithm may require exponentially many steps [23, 5]. Interestingly, computational experiments [5] seem to indicate that such hard instances become easily solvable if the game is modified into an equivalent one by a random potential transformation.

Karzanov and Lebedev [30] extended the potential-reduction algorithm [23] to so-called mean payoff games with prohibitions, that is to interdiction of type $\mathcal{B}_1$. Pisaruk [37] further extended these results to interdiction of type $\mathcal{B}_0$ defined by an arbitrary membership oracle, and showed that in this general setting, the potential-reduction algorithm [23] is pseudo-polynomial. Zwick and Paterson [45] gave another pseudo-polynomial algorithm for interdiction of type $\mathcal{B}_2$.

As mentioned above, mean payoff games can be reduced to the node-wise limited negative directed cycles interdiction. Indeed, if we fix a start vertex s, then determining whether the value of a mean payoff game on $G = (V, A)$ exceeds some threshold ξ is equivalent to the following decision problem:

(ξ) : Is there an admissible digraph G' such that the average arc-weight of each cycle reachable from s in G' is at least ξ?

After the substitution $w(e) \to w(e) - \xi$ we may assume without loss of generality that $\xi = 0$. This completes the reduction.

Björklund, Sandberg and Vorobyov [6] recently showed that mean payoff games can be solved in *expected sub-exponential time.* A deterministic sub-exponential algorithm was proposed by Jurdzinski, Paterson, and Zwick [26]. However, the question as to whether this class of games can be solved in polynomial time remains open, even though the decision problem (ξ) is in NP$\cap$ co-NP [30, 45].

Finally, let us consider a special case when digraph G contains only one negative arc, $w(t, s) = -d$, where d is a positive threshold. Let us also assume that, except (t, s), there is no other arc going from t and that $\mathcal{B}(t) = \emptyset$, or in other words, that (t, s) cannot be deleted. It is easy to see that in this case negative directed cycles of G are in one-to-one correspondence with directed paths from s to t that are shorter than d. So we will destroy these paths rather than negative cycles. Since the arc (t, s) becomes irrelevant, we can delete it and get a network with non-negative weights. Thus, we come to the short paths interdiction problem that is studied in the rest of the paper. As usual, we consider two cases: total limited and node-wise limited interdiction.

1.4 Node-wise limited short paths interdiction

Given a digraph $G = (V, A)$ with a non-negative weigh function $w : A \to \mathbb{R}_+$, two terminals $s, t \in V$ and a blocking system $\mathcal{B}$, find an admissible digraph G' that maximizes the distance from a given start vertex s to a given terminal vertex t:

$$\ell(s,t) \overset{\text{def}}{=} \max\{ \text{ } s\text{-}t \text{ distance in } G' \mid G' \text{ is an admissible digraph of } G\}.$$

We will see from what follows that, for any fixed terminal vertex $t \in V$, we can select an optimal admissible digraph that simultaneously maximizes the distances from all start vertices s. In other words, there exists an admissible digraph G^o such that for all vertices $v \in V \setminus \{t\}$, we have

$$\ell(v,t) \equiv \text{ } v\text{-}t \text{ distance in } G^o.$$

For this reason, it is convenient to consider the single-destination version of the above problem:

MASPNLAI (Maximizing all shortest paths to a given terminal by node-wise limited arc interdiction): *Given an arc-weighted digraph $G = (V, A)$, a non-negative weight function w, a terminal vertex $t \in V$, and a blocking system $\mathcal{B}$, find an optimal admissible digraph G^o that maximizes the distances from all vertices $v \in V \setminus \{t\}$ to t.*

Let us remark however, that if we fix a start vertex s, instead of t, then distinct terminal vertices may require distinct optimal admissible digraphs. The same happens if we consider in-going rather than out-going arcs.

In section 2 we show that MASPNLAI can be solved in strongly polynomial time by a natural extension of Dijkstra's algorithm.

Theorem 1 *Given a digraph $G = (V, A)$, a non-negative weight function $w : A \to \mathbb{R}_+$, and a terminal vertex $t \in V$,*
(i) *The special case of problem MASPNLAI for blocking systems $\mathcal{B}_1$ can be solved in time*

$$O\Bigg(|A| + |V| \log |V| + \sum_{v \in V \setminus \{t\}} [\textit{out-deg}(v) - k(v)] \log(k(v) + 1)\Bigg).$$

In particular, for blocking systems $\mathcal{B}_2$ the problem can be solved in $O(|A| + |V| \log |V|)$ time;
(ii) *For arbitrary blocking systems defined by membership oracles, MASPNLAI can be solved in $O(|A| \log |V|)$ time and at most $|A|$ monotonically increasing membership tests;*
(iii) *When all of the arcs have unit weight, problem MASPNLAI can be solved in $O(|A|+|V|)$ time and at most $|A|$ monotonically increasing blocking tests. The special cases $\mathcal{B}_1$ and $\mathcal{B}_2$ can be solved in $O(|A| + |V|)$ time.*

We show parts **(ii)** and **(iii)** of the theorem by using an extension of Dijkstra's algorithm and breadth-first search, respectively. As mentioned in the theorem, both of these algorithms employ monotonically increasing membership queries and never de-block a previously blocked arc. This is not the case with the variant of Dijkstra's algorithm used in the proof of part **(i)**. Note also that for blocks of type $\mathcal{B}_1$ and $\mathcal{B}_2$, the above bounds include the blocking tests overhead, and that the bound stated in **(i)** for $\mathcal{B}_2$ is as good as the running time of the fastest currently known strongly-polynomial algorithm by Fredman and Tarjan [14] for the standard shortest path problem, without interdiction.

Let us also mention that by Theorem 1, problem MASPNLAI can be solved in strongly polynomial time for any digraph $G = (V, A)$ that has no directed cycles of negative total arc-weight. Indeed, Gallai [16] proved that if G has no negative cycles then all input arc-weights $w(v, u)$ can be made non-negative by a potential transformation $w(v, u) \to w(v, u) + \varepsilon(v) - \varepsilon(u)$, where $\varepsilon : V \to \mathbb{R}$ are some vertex weights (potentials); see [1, 39]. Clearly, the total weights of all directed cycles remain unchanged and the total weight of a directed path p from s to t is transformed as follows: $w(p(s, t)) \to w(p(s, t)) + \varepsilon(s) - \varepsilon(t)$. Hence, the set of optimal arc blocks for MASPNLAI remain unchanged, too. Karp [29] showed that such a potential transformation can be found in $O(|A||V|)$ time.

We proceed with a negative observation. It is well known that the standard shortest path problem is in NC, that is it can be efficiently solved in parallel. In contrast, problem MASPNLAI is P-complete already for blocking systems of type $\mathcal{B}_2$ and acyclic digraphs $G = (V, A)$ of out-degree 2. This is because determining whether the blocking distance between a pair of vertices s, t is finite: $d(s, t) < +\infty$ includes, as a special case, the well-known monotone circuit value problem [20, 21].

1.5 Total limited short paths interdiction and similar problems

Given a digraph $G = (V, A)$, terminals $s, t \in V$, a non-negative weight function $w : A \to \mathbb{R}_+$, and two thresholds $k \in \mathbb{Z}_+$ and $d \in \mathbb{R}_+$, is it possible to remove k arcs from A so that $d(s, t) \geq d$ in the remaining digraph? For any constant k this clearly can be accomplished in polynomial time (see e.g., Corley and Shaw [8]), however, in general the problem is NP-hard, as it was shown by Bar-Noy, Khuller, and Schieber in [3]. In this paper we strengthen this result by deriving inapproximability bounds. For a given k let us denote by $\ell_A(G, s, t, k)$ the maximum of $d(s, t)$ over all digraphs obtainable from G by deleting (at most) k arcs.

Theorem 2 *It is NP-hard to approximate ℓ_A within a factor smaller than 2, even for bipartite graphs.*

Given a positive integer d, let us denote by $b_A(G, s, t, d)$ the smallest integer k such that $d(s, t) \geq d$ after k appropriate arcs are deleted from G.

Theorem 3 *It is NP-hard to approximate b_A within a factor smaller than $10\sqrt{5}-21 \approx 1.36$, even for bipartite graphs.*

The inapproximability bound $10\sqrt{5}-21 \approx 1.36$ was recently obtained by Dinur and Safra [11] for the Minimum Vertex Cover Problem in graphs improving the previous bound $7/6 \approx 1.17$ given by Håstad [24]. In our proof we reduce the problem to the Minimum Vertex Cover Problem.

We prove also that the same bounds 2 and $10\sqrt{5}-21 \approx 1.36$ hold for non-directed graphs and/or vertex interdiction.

Clearly, the functions ℓ_A and b_A establish an inverse connection between the distance $d(s, t)$ and the number of deleted arcs k: the more arcs we delete, the higher the distance between s and t. In some situations we might be interested to know the tradeoff between the number of deleted arcs and the distance achieved. We can however show that even this tradeoff cannot be approximated arbitrarily well, unless P=NP.

Let us say that it is NP-hard to *distinguish* two disjoint subsets of (di)graphs, $\mathcal{A}$ and $\mathcal{B}$, if no polynomial time algorithm can accept all graphs $G \in \mathcal{A}$ and reject all graphs $G \in \mathcal{B}$, unless P=NP.

Theorem 4 *For every fixed $\epsilon > 0$ it is NP-hard to distinguish graphs having $d(s, t) \geq d$ after the removal of some k arcs from those having $d(s, t) \leq \frac{1}{2-\epsilon} d$ in all subgraphs obtained by removing $(\frac{34}{33} - \epsilon)k$ arcs, where d and k are part of the input.*

1.6 Multiple cuts

Let us next recall that similar sounding problems about multiple cuts can be solved in polynomial time.

Given a directed graph $G(V, A)$, two terminals $s, t \in V$, a subset $A' \subseteq A$ of the arcs is called an ℓ-cut, if any $s - t$ directed path contains at least ℓ arcs from A'. Then, the following problem can be solved in polynomial time:

$\mathcal{M}$: *Given a digraph $G = (V, A)$ with two distinguished vertices $s, t \in V$ and positive integers k and ℓ, determine whether there exists an ℓ-cut $A' \subseteq A$ consisting of (at most) k arcs.*

Suppose without loss of generality that t is reachable from s in G, and let A' be an arbitrary ℓ-cut, that is, $|A' \cap P| \geq \ell$ for any s-t path $P \subseteq A$. Then, denoting by V_i the set of vertices that can be reached from s by using at most i arcs from A', we conclude that A' contains ℓ disjoint s-t cuts $C_i = \text{cut}(V_{i-1}, V_i)$ for $i = 1, \ldots, \ell$. Conversely, the union of any ℓ arc-disjoint s-t cuts is an ℓ-cut separating t from s. Hence problem $\mathcal{M}$ can be equivalently stated as follows:

$\mathcal{M}'$: *Given a digraph $G = (V, A)$, two distinguished vertices $s, t \in V$, and positive integers k and ℓ, determine whether there exist ℓ arc-disjoint s-t-cuts $C_1, \ldots, C_\ell$ such that $|C_1| + \ldots + |C_\ell| \leq k$.*

The latter problem is polynomial. Moreover, Wagner [41] showed that its weighted optimization version can be solved in strongly polynomial time.

$\mathcal{M}'_w$: *Given a digraph $G = (V, A)$ with two distinguished vertices $s, t \in V$, a weight function $w : A \to \mathbb{R}_+$, and a positive integer ℓ, find ℓ arc-disjoint s, t-cuts $C_1, \ldots, C_\ell$ of minimum total weight $w(C_1) + \ldots + w(C_\ell)$.*

1.7 Network interdiction and its applications

Problem MASPNLAI is a special (polynomially solvable) case of the so-called *network interdiction problem.* Interdiction (or inhibition) is an attack on arcs to destroy them, or increase their effective lengths, or decrease their capacities. The goal of the interdiction is to utilize a given budget most efficiently, that is to maximize the shortest path or minimize the maximum flow between two given terminals. These problems were originally motivated by military applications, McMasters and Mustin [32], Ghare, Montgomery, and Turner [18]. Later analogous models of pollution and drug interdiction were developed by Wood and Washburn [44, 42]. The problem of minimizing the maximum flow was considered by Phillips [36], while the maximization of the shortest path was first studied by Fulkerson and Harding [15] and also by Golden [19] (see Israeli and Wood [25] for a short survey). An important special case of the latter problem is the so-called *k-most-vital-arcs problem* [2, 3, 8, 31] in which it is allowed to destroy exactly k arcs. All the above mentioned problems are known to be NP-hard, in general.

Problem MASPNLAI is the short paths interdiction problem under the assumption that the budget is node-wise limited. This problem is polynomially solvable.

To illustrate possible applications of this polynomially solvable case, suppose that for each arc $e = (u, v)$ we are given a probability $p(e)$ that some undesirable transition (for example, contraband smuggling) from u to v can be carried out undetected. Then, assuming independence and letting $w(e) = -\log p(e) \geq 0$, we can interpret problem MASPNLAI as the uniform maximization of interception capabilities for a given target t under limited inspection resources distributed over the nodes of G.

2 Proof of Theorem 1

We first describe an extension of Dijkstra's algorithm for problem MASPNLAI that uses *blocking queues* and may temporarily block and then de-block some arcs. This extension, presented in Section 2.2, is used to show part **(i)** of Theorem 1. Then in Section 2.4 we present another implementation of the extended algorithm to prove part **(ii)** of the theorem. Part **(iii)** is shown in Section 2.5.

2.1 Blocking Queues

Let $\mathcal{B}$ be a blocking (i.e. independence) system on a finite set A, for example on the set $A(v)$ of arcs leaving a given vertex v of G. Given a mapping $p : A \to \mathbb{R}$, and a set $Y \subseteq A$, let

$$p_{\mathcal{B}}(Y) = \max_{X \in \mathcal{B}} \min_{e \in Y \setminus X} p(e), \tag{1}$$

where, as usual, it is assumed that the minimum over the empty set is $+\infty$. For instance, if $Y = \{e_1, e_2, e_3, e_4\}$ and $(p(e_1), p(e_2), p(e_3), p(e_4)) = (1, 3, 3, 5)$, then

$$p_{\mathcal{B}}(Y) = \begin{cases} 1, & \text{if } \{e_1\} \notin \mathcal{B}; \\ 3, & \text{if } \{e_1\} \in \mathcal{B} \text{ but } \{e_1, e_2, e_3\} \notin \mathcal{B}; \\ 5, & \text{if } \{e_1, e_2, e_3\} \in \mathcal{B} \text{ but } Y \notin \mathcal{B}; \\ +\infty, & \text{if } Y \in \mathcal{B}. \end{cases}$$

Considering the image $\{p(e),\ e \in Y\}$ as a set of keys, we define a *$\mathcal{B}$-queue* as a data structure for maintaining a dynamic set of keys under the following operations:
1. *Make_queue*: Create an empty queue $Y = \emptyset$;
2. *Insert*: Expand Y by adding a new element e with a given key value $p(e)$;
3. *Return* $p_{\mathcal{B}}(Y)$: Compute the right-hand side of (1) for the current key set.
Note that when the independence system is trivial, $\mathcal{B} = \{\emptyset\}$, we obtain the customary definition of a minimum priority queue.

When $\mathcal{B}$ is a blocking system of type $\mathcal{B}_1$, that is, $X \in \mathcal{B}$ whenever $|X| \le k$ for some given integer $k \le |A|$, then

$$p_{\mathcal{B}}(Y) = \begin{cases} +\infty, & \text{if } |Y| \le k; \\ (k+1)^{st} \text{ smallest key of } Y, & \text{if } |Y| \ge k+1. \end{cases}$$

Hence, by maintaining a regular maximum priority queue of at most $k+1$ elements of A,

- *A sequence of $d \ge k$ queue operations for an initially empty $\mathcal{B}_1$-queue can be implemented to run in $O(k + (d-k)\log(k+1))$ time.*

For general blocking systems $\mathcal{B}$, each $\mathcal{B}$-queue operation can be performed in $O(\log|Y|)$ time and $O(\log|Y|)$ oracle queries. This can be done by using a balanced binary search tree on the set of keys in Y. Specifically, inserting a new key into Y takes $O(\log|Y|)$ time and no oracle queries, while computing the value of $p_{\mathcal{B}}(Y)$ can be done by searching for the largest key p in the tree for which the oracle can block the set of all keys smaller than p. Note that each query to the blocking oracle can be specified by a *list* of keys if we additionally maintain a sorted list of all keys in Y along with pointers from the search tree to the list.

We close this subsection by defining, for each set $Y \subseteq A$ of keys, a (unique) inclusion-wise minimal blocking set $\hat{X}(Y) \in \mathcal{B}$ such that

$$p_{\mathcal{B}}(Y) = \min_{e \in Y \setminus \hat{X}(Y)} p(e).$$

We will refer to $\hat{X}(Y) \subseteq Y$ as the *lazy block for Y*. For instance, if, as before, $Y = \{e_1, e_2, e_3, e_4\}$ and $(p(e_1),\ p(e_2),\ p(e_3),\ p(e_4)) = (1, 3, 3, 5)$, then

$$\hat{X}(Y) = \begin{cases} \emptyset, & \text{if } \{e_1\} \notin \mathcal{B}; \\ \{e_1\}, & \text{if } \{e_1\} \in \mathcal{B}, \text{ but } \{e_1, e_2, e_3\} \notin \mathcal{B}; \\ \{e_1, e_2, e_3\}, & \text{if } \{e_1, e_2, e_3\} \in \mathcal{B}, \text{ but } Y \notin \mathcal{B}; \\ Y, & \text{if } Y \in \mathcal{B}. \end{cases}$$

For an unsorted list of keys $\{p(e),\ e \in Y\}$, the lazy block $\hat{X}(Y)$ can be computed in $O(|Y|)$ time and $O(\log|Y|)$ oracle queries by recursively splitting the keys around the median. For blocking systems $\mathcal{B}_1$ this computation takes $O(|Y|)$ time.

2.2 Extended Dijkstra's Algorithm for MASPNLAI

Given a digraph $G = (V, A)$, a non-negative weight function $w(v) : A \to \mathbb{R}^+$, a vertex $t \in V$, and a blocking system $\mathcal{B}$, we wish to find an admissible graph G^o that maximizes the distance from each start vertex $v \in V$ to t. In the statement of extended Dijkstra's algorithm below we assume without loss of generality that the out-degree of the terminal vertex t is 0, and the input arc-weights $w(e)$ are all finite. By definition, we let $\ell(t,t) = 0$.

Similarly to the regular Dijkstra's algorithm, the extended version maintains, for each vertex $v \in V$, an upper bound $\rho(v)$ on the blocking v-t distance:

$$\rho(v) \geq \ell(v,t) \overset{\text{def}}{=} \max_{G' \text{ admissible}} \{\text{distance from } v \text{ to } t \text{ in } G'\}.$$

Initially, we let $\rho(t) = 0$ and $\rho(v) = +\infty$ for all vertices $v \in V \setminus \{t\}$. As the regular Dijkstra's algorithm, the extended version runs in at most $|V| - 1$ iterations and (implicitly) partitions V into two subsets S and $T = V \setminus S$ such that $\rho(v) = \ell(v,t)$ for all $v \in T$. We iteratively grow the initial set $T = \emptyset$ by removing, at each iteration, the vertex u with the smallest value of $\rho(v)$ from S and adding it to T. For this reason, the values of $\rho(v),\ v \in S$ are stored in a minimum priority queue, e.g., in a Fibonacci heap. Once we remove the minimum-key vertex u from S (and thus implicitly declare that $\rho(u) = \ell(u,t)$), we update $\rho(v)$ for all those vertices $v \in S$ that are connected to u by an arc in G. Recall that the regular version of Dijkstra's algorithm uses updates of the form $\rho(v) \leftarrow \min\{\rho(v),\ w(v,u)+\rho(u)\}$. The updates performed by the extended version use blocking queues $Y(v)$ maintained at all vertices $v \in V \setminus \{t\}$. Initially, all these $\mathcal{B}(v)$-queues are empty, and when the value of $\rho(v)$ needs to be updated for some vertex $v \in S$ such that $e = (v,u) \in A$, we first insert arc e with the key value $p(e) = w(v,u) + \rho(u)$ into $Y(v)$, and then let $\rho(v) \leftarrow p_{\mathcal{B}}(Y(v)) \overset{\text{def}}{=} \max_{X \in \mathcal{B}(v)} \min_{e \in Y(v) \setminus X} p(e)$. In particular, for the standard shortest path problem, we obtain the regular updates.

Finally, as the regular Dijkstra's algorithm, the extended version terminates as soon as $\rho(u) = \min\{\rho(v),\ v \in S\} = +\infty$ or $|S| = 1$.

EXTENDED DIJKSTRA'S ALGORITHM

Input: **A digraph $G = (V, A)$ with arc-weights $\{w(e) \in [0, +\infty),\ e \in A\}$, a destination vertex $t \in V$, and a blocking system $\mathcal{B}$.**

```
Initialization:
1. $\rho(t) \leftarrow 0$;
2. For all vertices $v \in V \setminus \{t\}$ do:
3.    $\rho(v) \leftarrow +\infty$; Set up an empty blocking queue $Y(v)$;
4. Build a minimum priority queue (Fibonacci heap) $S$ on
the key values $\rho(v),\ v \in V$.

Iteration loop:
5. While $|S| > 1$ do:
6.   If $\min\{\rho(v),\ v \in S\} = +\infty$, break loop and go to line 12;
7.    Extract the vertex $u$ with the smallest key value $\rho(\cdot)$
from $S$;
8.      For all arcs $e = (v, u) \in A$ such that $v \in S$, do:
9.           $p(e) \leftarrow w(e) + \rho(u)$;
10.          Insert $p(e)$ into $Y(v)$;
11.          Update the value of $\rho(v)$ : $\rho(v) \leftarrow p_{\mathcal{B}}(Y(v))$.

Output:
12. For each vertex $v \in V \setminus \{t\}$, return $\rho(v)$ with the lazy
block $\hat{X}(Y(v))$.
```

Bounds on running time for blocks of type $\mathcal{B}_1$. Line 12 and the initialization steps in lines 1-4 take linear time $O(|V| + |A|)$. Let $n \leq |V| - 1$ be the number of iterations performed by the algorithm. Denote by $Y_i(v)$ (the set of key values in) the blocking queue at a fixed vertex $v \in V \setminus \{t\}$ after the execution of iteration $i = 1, \ldots, n$, and let $Y_0(v) = \emptyset$ be the initial queue at v. As $Y_0(v) \subseteq Y_1(v) \subseteq \ldots \subseteq Y_n$, the values of $\rho_i(v) = p_{\mathcal{B}}(Y_i(v))$ are monotonically non-increasing: $+\infty = \rho_0(v) \geq \rho_1(v) \geq \ldots \geq \rho_n(v)$. Since S is a (minimum) Fibonacci heap, the decrease-key operations in line 11 can be executed in constant amortized time per iteration, provided that the values of $p_{\mathcal{B}}(Y_i(v))$ are known. Lines 6 and 7 take $O(1)$ and $O(\log |V|)$ time per iteration, respectively. In view of the bounds on the $\mathcal{B}_1$- queue operations 10-11 stated in Section 2.1, the overall running time of the algorithm is thus within the bound stated in part **(i)** of Theorem 1.

To complete the proof of part **(i)** it remains to show that the extended algorithm is correct.

2.3 Correctness of Extended Dijkstra's Algorithm

Let us show that *upon the termination of the extended Dijkstra's algorithm,*

- $\rho(v) = \ell(v,t) \overset{\text{def}}{=} \max_{G' admissible}\{$*distance from* v *to* t *in* $G'\}$ *for all vertices* $v \in V$*, and*
- *The digraph* $G^o = \big(V, A \setminus \bigcup_{v\in V\setminus\{t\}} \hat{X}(Y(v))\big)$ *obtained by deleting the lazy blocking sets of arcs* $\hat{X}(Y(v))$ *is an optimal admissible digraph for all vertices:* $\ell(v,t) \equiv$ v*-*t *distance in* G^o*.*

Let S_i and $T_i = V \setminus S_i$ be the vertex partition maintained by the algorithm for $i = 0, 1, \ldots, n \leq |V| - 1$. We have $S_0 = V \supset S_1 = V \setminus \{t\} \supset \ldots \supset S_{n-1} \supseteq S_n$, where $S_{n-1} = S_n$ if and only if the algorithm terminates due to the stopping criterion in line 6. For the given arc weights $w(e),\ e \in A$, consider the following weight functions $w_i : A \to \mathbb{R}_+ \cup \{+\infty\}$:

$$w_i(e) = \begin{cases} +\infty, & \text{if both endpoints of } e \text{ are in } S_i, \\ w(e), & \text{otherwise.} \end{cases} \tag{2}$$

Clearly, we have $w_0(e) = +\infty \geq w_1(e) \geq \ldots \geq w_n(e) \geq w(e)$. Let

$$\ell_i(v,t) \overset{\text{def}}{=} \max_{G' \ admissible} \{w_i\text{-distance from } v \text{ to } t \text{ in } G'\},$$

then $\ell_0(v,t) = +\infty \geq \ell_1(v,t) \geq \ldots \geq \ell_n(v,t) \geq \ell(v,t)$ for all $v \in V \setminus \{t\}$. The correctness of the algorithm will follow from the following two invariants: *for all* $i = 0, 1, \ldots, n$,

$\mathcal{I}_i^S$: $\rho_i(v) = \ell_i(v,t)$ *for all vertices* $v \in S_i$*;*

$\mathcal{I}_i^T$: *If* $v \in T_i = V \setminus S_i$*, then* $\rho_i(v) = \ell(v,t)$ *and the admissible digraph* $G_i^o = \big(V, A \setminus \bigcup_{v\in V\setminus\{t\}} \hat{X}(Y_i(v))\big)$ *is an optimal blocking digraph for* v*. Moreover,* $\min\{\rho_i(v),\ v \in S_i\} \geq \max\{\ell(v,t),\ v \in T_i\}$ *and for each vertex* $v \in T_i$ *there exists a shortest* v*-*t *path in* G_i^o *which lies entirely in* T_i*.*

Note that by $\mathcal{I}_i^T$, the algorithm removes vertices from S and determines their blocking distances in non-decreasing order.

Proof of invariants $\mathcal{I}_i^S$and $\mathcal{I}_i^T$ is similar to that for the regular Dijkstra's algorithm. Since $T_0 = \emptyset$, invariant $\mathcal{I}_0^T$ holds trivially. $\mathcal{I}_0^S$ follows from the initialization steps of the algorithm: for $S_0 = V$ we have $w_0(e) \equiv +\infty$, and hence $\rho_0(t) = \ell_0(t,t) = 0$ and $\rho_0(v) = \ell_0(v,t) = +\infty$ for all vertices $v \in V \setminus \{t\}$.

In order to prove by induction that $\mathcal{I}_{i+1}^S$ and $\mathcal{I}_{i+1}^T$ follow from $\mathcal{I}_i^S$ and $\mathcal{I}_i^T$, let us first suppose that the ith iteration loop breaks due to the stopping criterion in line 6: $\min\{\rho_i(v),\ v \in S_i\} = +\infty$. Then $i = n-1$ and $S_{n-1} = S_n$, which means that $\ell_n(v,t) \equiv \ell_{n-1}(v,t)$ and $\rho_n(v) \equiv \rho_{n-1}(v)$. Consequently, the statements of $\mathcal{I}_n^S$ and $\mathcal{I}_n^T$ become identical to $\mathcal{I}_{n-1}^S$ and $\mathcal{I}_{n-1}^T$, and we have nothing to prove. Moreover, as all vertices of S_n are disconnected from t in $G^o = G_n^o$, invariant $\mathcal{I}_n^T$also shows that the algorithm correctly computes the blocking distances and the optimal blocking digraph G^o for all vertices.

We may assume henceforth that $n = |V| - 1$ and $|S_n| = 1$. Consider the vertex u that moves from S_i to T_{i+1} at iteration i:

$$\rho_i(u) = \min\{\rho_i(v), \quad v \in S_i\} < +\infty. \tag{3}$$

To show that $\rho_i(u) = \ell(u,t)$, observe that by $\mathcal{I}_i^S$, $\rho_i(u) = \ell_i(u,t) \geq \ell(u,t)$. In other words, $\rho_i(u)$ is an upper bound on the w-cost of reaching t from u, regardless of any admissible blocks selected by the adversary. So we will have $\rho_i(u) = \ell(u,t)$ if we can find an admissible digraph G' such that

$$\rho_i(u) = w\text{-distance from } u \text{ to } t \text{ in } G'. \tag{4}$$

Let $G' = G_i^o$ be the admissible digraph defined in $\mathcal{I}_i^T$. Then (4) follows from $\mathcal{I}_i^T$, the non-negativity of the input arc-weights, and the fact that $\rho_i(u) = p(e_*) = w(e_*) + \rho_i(v)$, where $e_* = (u,v) \in A$ is the arc with the smallest key value in the $(S_i\,, T_i)$-cut of G'.

After u gets into T_{i+1}, the value of $\rho(u)$ never changes. Hence $\rho_{i+1}(u) = \ell(u,t)$, as stated in $\mathcal{I}_{i+1}^T$. Note that (3) and invariant $\mathcal{I}_i^T$also tell us that $\min\{\rho_i(v),\ v \in S_i\} = \ell(u,t) \geq \max\{\ell(v,t),\ v \in T_i\}$. Let us now show that after the algorithm updates $\rho(v)$ on S_{i+1}, we still have

$$\min\{\rho_{i+1}(v),\ v \in S_{i+1}\} \ \geq\ \ell(u,t)\ = \max\{\ell(v,t),\ v \in T_{i+1}\}, \tag{5}$$

again as stated in $\mathcal{I}_{i+1}^T$. Suppose to the contrary, that $\rho_{i+1}(v) < \ell(u,t) = \rho_i(u)$ for some vertex $v \in S_{i+1}$. Then from (3) it would follow that $e = (v,u)$ is an arc of $G = (V,A)$ and consequently $Y_{i+1}(v) = Y_i(v) \cup \{e\}$. Moreover, we must have $e \in \hat{X}(Y_{i+1}(v))$, for otherwise the value of $\rho_{i+1}(v) = p_{\mathcal{B}}(Y_{i+1}(v))$ could not have dropped below the minimum of $\rho_i(v)$ and $p(e) = w(v,u) + \ell(u,t)$, which is at least $\ell(u,t)$. But if $e \in \hat{X}(Y_{i+1}(v))$ then again $p_{\mathcal{B}}(Y_{i+1}(v)) \geq p(e)$, contradiction.

To complete the proof of $\mathcal{I}_{i+1}^T$, it remains to show that G_{i+1}^o is an optimal admissible digraph for each vertex $v \in T_{i+1}$, and that some shortest v-t path in G_{i+1}^o lies in T_{i+1}. This readily follows from (5) and the fact that the sub-graphs of G_i^o and G_{i+1}^o induced by T_{i+1} are identical.

Finally, $\mathcal{I}_{i+1}^S$ follows from the updates $\rho_{i+1}(v) \leftarrow p_{\mathcal{B}}(Y_{i+1}(v))$ performed by the algorithm in lines 8-11. $\square$

Since we assume that $n = |V| - 1$ and $|S_n| = 1$, the correctness of the algorithm readily follows from $\mathcal{I}_n^S$ and $\mathcal{I}_n^T$. When S_n is a singleton $s \in V$, then $w_n(e) \equiv w(e)$, see (2). Hence $\ell_n(v,t) \equiv \ell(v,t)$, and $\mathcal{I}_n^S$ yields $\rho_n(s) = \ell_n(s,t) = \ell(s,t)$. By $\mathcal{I}_n^T$, we also have $\rho_n(v) = \ell(v,t)$ for the remaining vertices $v \in T_n = V \setminus \{s\}$. Invariant $\mathcal{I}_n^T$ also guarantees that $G^o = G_n^o$ is an optimal admissible digraph for all vertices $v \in V$.

2.4 Modified Dijkstra's Algorithm

In this section we prove part **(ii)** of Theorem 1 by modifying the algorithm stated in Section 2.2.

The modified algorithm keeps all arcs across the current (S,T)-cut in a minimum priority queue $\mathcal{A}$, implemented as a binary heap. As in the previous algorithm, each arc $e = (v,v')$

across the cut is assigned the key value $p(e) = w(e) + \rho(v')$, where $\rho(v') = \ell(v', t)$ for all vertices $v' \in T$. In addition to the arcs in the current cut, $\mathcal{A}$ may also contain some arcs $e = (v, v')$ for which both endpoints v, v' are in T. In order to compute the vertex u to be moved from S to T, we repeatedly extract the minimum-key arc e from $\mathcal{A}$, and check whether $e = (v, v')$ belongs to the current cut and can be blocked along with the arcs that have already been blocked at v. The first arc $e = (v, v')$ in the cut that cannot be blocked defines the vertex $u = v$. We then move u to T, insert all arcs $e = (vu) \in A$ for which $v \in S$ into $\mathcal{A}$, and iterate.

```
MODIFIED ALGORITHM

Input: A digraph $G = (V, A)$ with arc-weights $\{w(e) \in [0, +\infty),\ e \in A\}$, a terminal vertex $t \in V$, and a blocking system $\mathcal{B} \subseteq 2^A$ defined via a membership testing subroutine.

Initialization:
1. Initialize arrays $T[1:V] \equiv FALSE$ and $\ell[1:V,t] \equiv +\infty$ ;
2. $T[t] \leftarrow TRUE$,   $d[t,t] \leftarrow 0$;
3. For each vertex $v \in V \setminus \{t\}$ initialize an empty list $\hat{X}(v)$;
4. For each arc $e = (v,t) \in A$, insert $e$ with key $p(e) = w(e)$ into an initially empty binary heap $\mathcal{A}$.

Iteration loop:
5. While $\mathcal{A} \neq \emptyset$ do:
6.     Extract the minimum-key arc $e = (u,v)$ from $\mathcal{A}$;
7.     If $T[u] = FALSE$ and $T[v] = TRUE$ do:
8.         If $\hat{X}(u) \cup \{e\}$ can be blocked at $u$, insert $e$ into $\hat{X}(u)$
9.         else { $T[u] \leftarrow TRUE$;  Return $\hat{X}(u)$ and $\ell[u,t] = p(e)$;
10.                For all arcs $e = (v,u) \in A$ such that $T[v] = FALSE$,
                   Insert $e$ with key value $p(e) = w(e) + \ell[u,t]$ into $\mathcal{A}$}.
```

The outputs of the modified algorithm and the extended Dijkstra's algorithm presented in Section 2.2 are identical. It is also easy to see that the running time and the number of membership tests required by the modified algorithm satisfy the bounds stated in part **(ii)** of Theorem 1.

2.5 Unit arc-weights

When $w(e) = 1$ for all $e \in A$, and the blocking systems $\mathcal{B}(v)$ are all empty, the single-destination shortest path problem can be solved in linear time by breadth-first search. The extended Dijkstra's algorithm for problem MASPNLAI can be similarly simplified to prove part **(iii)** of Theorem 1.

```
BREADTH-FIRST SEARCH FOR MASPNLAI

Input: A digraph $G = (V, A)$ with a destination vertex $t \in V$,
and a blocking system $\mathcal{B}$ defined by a membership subroutine.

Initialization:
1. Initialize $\ell(1 : V, t) \equiv +\infty$ and an empty first-in first-out
queue $T$;
2. $\ell(t, t) \leftarrow 0$; Enqueue $t$ into $T$;
3. For each vertex $v \in V \setminus \{t\}$ initialize an empty list $\hat{X}(v)$;

Iteration loop:
4. While $T \neq \emptyset$ do:
5.     Extract the first vertex $u$ from $T$;
6.     For all arcs $e = (v, u) \in A$, do:
7.         If $\ell(v, t) = +\infty$ and $\hat{X}(v) \cup \{e\}$ can be blocked, insert $e$
into $\hat{X}(v)$;
8.         else $\ell(v, t) \leftarrow \ell(u, t) + 1$, enqueue $v$ into $T$, and return
$\ell(v, t), \hat{X}(v)$.
```

The above algorithm runs in at most $|A|$ iterations. It follows by induction on $\ell(v, t)$ that it correctly computes the blocking distances and that the admissible digraph $G^o = \big(V, A \setminus \bigcup_{v \in V \setminus \{t\}} \hat{X}(v)\big)$ is optimal.

3 Inapproximability Bounds

In the rest of the paper we prove the inapproximability results for total limited short paths interdiction stated in the introduction, and a few analogous claims for vertex interdiction and/or undirected graphs. Let us first give precise formulations for these results.

3.1 Problems and results

We consider a graph (or digraph) $G = (V, E)$ with a nonnegative length associated with every edge (or arc), two distinct vertices s and t, and a threshold $d \in \mathbb{Z}_+$, and denote this input by (G, s, t, d).

A *vertex blocker* of (G, s, t, d) is a set of vertices different from s and t whose removal increases the s-t distance to at least d. We define the Minimum Vertex Blocker to Short Paths Problem (MVBP) as follows:

Minimum Vertex Blocker to Short Paths Problem (MVBP)

Input: A graph (digraph) G with a nonnegative length associated with every edge (arc), two vertices s, t and a threshold d

Output: The size $b_V(G, s, t, d)$ of the smallest vertex blocker:

$$b_V(G, s, t, d) = min\{ \ |U| \mid d_{G[V \setminus U]}(s, t) \geq d, \ U \subseteq V \setminus \{s, t\}\}.$$

Theorem 5 *It is NP-hard to approximate the size of the smallest vertex blocker within a factor smaller than* $10\sqrt{5} - 21 \approx 1.36$*, even for bipartite graphs.*

An *edge blocker* of (G, s, t, d) is a set of edges (arcs) whose removal increases the s-t distance to at least d. We define the Minimum Edge Blocker to Short Paths Problem (MEBP) as follows:

Minimum Edge Blocker to Short Paths Problem (MEBP)

Input: A graph (digraph) G with a nonnegative length associated with every edge (arc), two vertices s, t and a threshold d

Output: The size $b_E(G, s, t, d)$ of the smallest edge blocker:

$$b_E(G, s, t, d) = min\{ \ |F| \mid d_{(V, E \setminus F)}(s, t) \geq d, \ E \subseteq F\}.$$

Then, the analogous statement to Theorem 5 above is equivalent with Theorem 3 as we claimed in the introduction:

Theorem 6 *It is NP-hard to approximate the size of the smallest edge blocker within a factor smaller than* $10\sqrt{5} - 21 \approx 1.36$*, even for bipartite graphs.*

The Most Vital Vertices Problem (MVVP) is defined as follows:

Most Vital Vertices Problem (MVVP)

Input: A graph (digraph) $G = (V, E)$ with a nonnegative length associated with every edge (arc), two special vertices s, t and a threshold k

Output: The maximum $\ell_V(G, s, t, k)$ of s-t distances in all graphs obtained from G by removing k vertices. More precisely:

$$\ell_V(G, s, t, k) = max\{d_{G[V \setminus U]}(s, t) \mid U \subseteq V \setminus \{s, t\}, |U| = k\}.$$

Theorem 7 *It is NP-hard to approximate ℓ_V within a factor smaller than 2, even for bipartite graphs.*

The Most Vital Edges Problem (MVEP) is defined as follows:

The Most Vital Edges Problem (MVEP)

Input: A graph (digraph) $G = (V, E)$ with a nonnegative length associated with every edge (arc), two vertices s, t and a threshold k

Output: The maximum $\ell_E(G, s, t, k)$ of s-t distances in all graphs obtained from G by removing k edges. More precisely:

$$\ell_E(G, s, t, k) = max\{d_{(V, E \setminus F)}(s, t) \mid F \subseteq E, |F| = k\}.$$

Then, Theorem 2 can be reformulated as follows:

Theorem 8 *It is NP-hard to approximate ℓ_E within a factor smaller than 2, even for bipartite graphs.*

3.2 Restricted problems

In this section we define restricted versions of the above problems by introducing the assumption that some vertices (edges) cannot be removed. We called these vertices (edges) *fixed*. The remaining vertices (edges) are called *removable*.

We obtain *restricted*-MVBP and *restricted*-MVVP from MVBP and MVVP, respectively, by fixing some vertices (in addition to s and t). Similarly we obtain *restricted*-MEBP and *restricted*-MVEP from MEBP and MVEP, respectively, by fixing some edges.

For a graph G, two vertices s, t, a set of fixed vertices V' (or a set of fixed edges E') and thresholds d and k, let $b'_V(G, s, t, V', d)$, $b'_E(G, s, t, E', d)$, $\ell'_V(G, s, t, V', k)$ and $\ell'_E(G, s, t, E', k)$ denote the solutions to restricted-MVBP, restricted-MEBP, restricted-MVVP and restricted-MVEP, respectively.

Given an instance (G, s, t, V', d) of restricted-MVBP we assume that all removable vertices form a vertex blocker. Similarly given an instance (G, s, t, E', k) of restricted-MEBP we assume that all removable edges form an edge blocker.

3.3 Inapproximability of minimum vertex cover

In this section we present previously known results on which the proofs of our results are based. A *vertex cover* of an undirected graph G is a subset of vertices incident to every edge. Let $\tau(G)$ denote the size of the smallest vertex cover of G.

Deciding if G has a vertex cover of size at most k is NP-hard [17], even for tripartite graphs [38]. However, $\tau(G)$ can be easily approximated within a factor 2, since the vertex cover consisting of both vertices of edges belonging to the maximum matching can be computed in polynomial time and its size is at most $2\tau(G)$. Improving this simple 2-approximation algorithm has been quite a nontrivial task. The best known approximation algorithm has a factor of $2 - \Theta(\frac{1}{\sqrt{\log n}})$, where n is the number of vertices [27].

On the other hand, in 1997 Håstad [24] proved that it is NP-hard to approximate $\tau(G)$ within a factor smaller than $\frac{7}{6} \approx 1.17$. Recently Dinur and Safra [11] obtained the better inapproximability factor of $10\sqrt{5} - 21 \approx 1.36$. For tripartite graphs it is NP-hard to approximate $\tau(G)$ within a factor smaller than $\frac{34}{33} \approx 1.03$ [7].

4 Proof of Theorem 5

In this section we prove Theorem 5 by reducing the minimum vertex cover problem to restricted-MVBP. As shown in Section 7.1 and Section 7, for each instance of restricted-MVBP we can construct an instance of MVBP with the same optimal value and a bipartite input graph. Therefore Theorem 9 below implies Theorem 5.

Theorem 9 *It is NP-hard to approximate b'_V within a factor smaller than* $10\sqrt{5} - 21 \approx 1.36$.

Proof. Let G be an undirected graph with vertices $v_1, \ldots, v_n$ (see Figure 1). We construct an instance of restricted-MVBP. We obtain an undirected graph H from G by adding to it a path $su_1u_2 \ldots u_nt$ and connecting v_i to u_i for $i = 1, \ldots, n$ (see Figure 2). Let W denote the vertex set of H. We assign length 1 to edges u_1u_2, $u_2u_3, \ldots, u_{n-1}u_n$ and 0 to all other

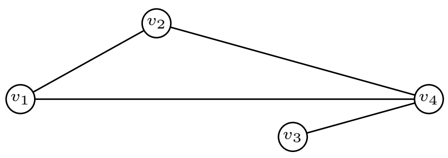

Figure 1: Graph G.

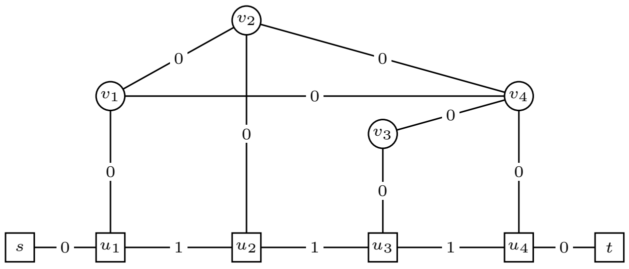

Figure 2: Graph H. Squares are fixed vertices.

edges. Let $V' = \{u_1, \dots, u_n\}$ be the set of fixed vertices. The threshold is $n-1$. Note that the set of all removable vertices forms a vertex blocker.
Recall that $\tau(G)$ denotes the size of the smallest vertex cover of G.

Claim 1 $\tau(G) = b'_V(H, s, t, V', n-1)$.

Proof. Let $U \subseteq \{v_1, \dots, v_n\}$ be a set of removable vertices. We show that U is a vertex cover of G if and only if U is a vertex blocker of $(H, s, t, n-1)$.$H[W \setminus U]$ is at least $n-1$.

Suppose U is a vertex cover of G. Since $V \setminus U$ is an independent set of G, there is only one s-t path, $su_1u_2 \dots u_nt$, in $H[W \setminus U]$ and the length of this path is $n-1$ (see Figure 3).

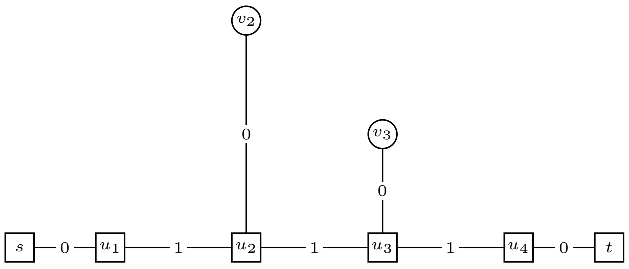

Figure 3: Graph $H[W \setminus U]$ obtained from H by removal of the vertex cover $U = \{v_1, v_4\}$ of G.

Conversely, suppose U is a vertex blocker of $(H, s, t, n-1)$. Note that for every $i < j$ there is no edge between vertices v_i and v_j in $H[W \setminus U]$, since otherwise there would exist a path $su_1 \dots u_iv_iv_ju_j \dots u_nt$ in $H[W \setminus U]$ shorter than $n-1$. Thus U is a vertex cover of G. $\square$

Since it is NP-hard to approximate the minimum vertex cover within a factor smaller than $10\sqrt{5} - 21 \approx 1.36$ [11], Theorem 9 follows.

We can similarly reduce the minimum vertex cover problem to restricted-MVBP for directed graphs. Let H be a digraph obtained from G by replacing every edge v_iv_j, $i < j$, of G by an arc v_iv_j, adding to it a dipath $su_1u_2 \dots u_nt$ and connecting v_i to u_i with two arcs v_iu_i and u_iv_i for $i = 1, \dots, n$ (see Figure 4).

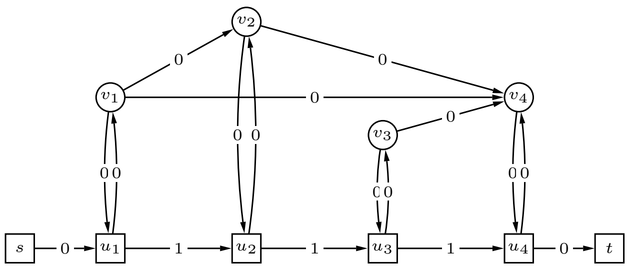

Figure 4: Digraph H. Squares are fixed vertices.

As before we assign length 1 to arcs u_1u_2, $u_2u_3, \dots, u_{n-1}u_n$ and 0 to all other arcs, vertices $u_1, \dots, u_n$ are fixed and the threshold is $n - 1$. The proof that $\tau(G) = b'_V(H, s, t, V', n - 1)$ is analogous. $\square$

5 Proof of Theorem 6

In this section we prove Theorem 6 similarly to the proof of Theorem 5. We reduce the minimum vertex cover problem to restricted-MEBP. As shown in Section 7.2 and Section 7, for each instance of restricted-MEBP we can construct an instance of MEBP with the same optimal value and a bipartite input graph. Therefore Theorem 10 below implies Theorem 6.

In the proof of Theorem 10 we use a gadget first described in [3], where it was used to prove NP-hardness of the Most Vital Edges Problem.

Theorem 10 *It is NP-hard to approximate b'_E within a factor smaller than* $10\sqrt{5} - 21 \approx 1.36$.

Proof. Let G be a undirected graph with vertices $v_1, \dots, v_n$ (see Figure 1). We construct an instance of restricted-MEBP. We obtain an undirected graph H from G by

- replacing every vertex v_i of G by two vertices v'_i and v''_i connected by an edge $v'_iv''_i$ of length 1 for $i = 1, \dots, n$,
- replacing every edge v_iv_j, $i < j$, of G by $v''_iv'_j$ of length $5(j - i) - 2$,

- adding to it a path $P = su'_1u''_1u'_2u''_2 \dots u'_nu''_nt$, where $u'_iu''_i$ has length 5 for $i = 1, \dots, n$ and other edges have length 0,
- adding two edges $v'_iu'_i$ and $v''_iu''_i$ of length 2 for $i = 1, \dots, n$ (see Figure 5).

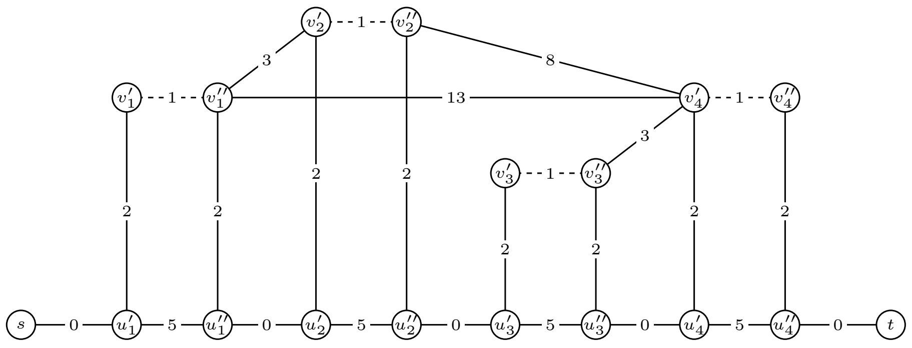

Figure 5: Graph H. Solid lines are fixed edges.

All edges except for $v'_1v''_1, \dots, v'_nv''_n$ are fixed. We denote the set of fixed edges by E'. The threshold is $5(n-1)$. Note that the set of all removable edges forms a vertex blocker. Let W and E denote the vertex set and the edge set of H, respectively.

Let $x \in \{u'_i, u''_i\}$, $y \in \{u''_j, u''_j\}$, where $i \neq j$. We call the subpath of P from x to y an *x-y line*. An *x-y detour* is an x-y path D in H, where no vertices of D, apart from the first and the last, belong to P. An *i-j shortcut* is the path $v'_iv''_iv'_jv''_j$ (see Figure 6).

Let length(Q) denote the length of a path Q.

Claim 2 *If x-y detour D contains no shortcuts then* length$(D) \geq$ length(*x-y line*).

Proof. There are four possible kinds of x-y detours containing no shortcuts:

Case 1: $u'_iv'_iv''_iu''_i$, for $i = 1, \dots, n$. Then length(*x-y line*) $= 5$ and length(*detour*(x, y)) $= 5$

Case 2: $u'_iv'_iv''_iv'_ju'_j$, for $i = 1, \dots, n$. Then length(*x-y line*) $= 5(j-i)$ and length(*detour*(x, y)) $= 5(j-i)+3$,

Case 3: $u''_iv''_iv'_ju'_j$, for $i = 1, \dots, n$. Then length(*x-y line*) $= 5(j-i-1)$ and length(*detour*(x, y)) $= 5(j-i)+2$.

Case 4: $u''_iv''_iv'_jv''_ju''_j$, for $i = 1, \dots, n$. Then length(*x-y line*) $= 5(j-i)$ and length(*detour*(x, y)) $= 5(j-i)+3$.

Thus all four kinds of x-y detours are at least as long as x-y line. □

Claim 3 *Let Q be an s-t path in H. If for every detour D contained in P* length$(D) \geq$ length(*x-y line*)*, where x and y are ends of D, then the length of Q is at least* $5(n-1)$.

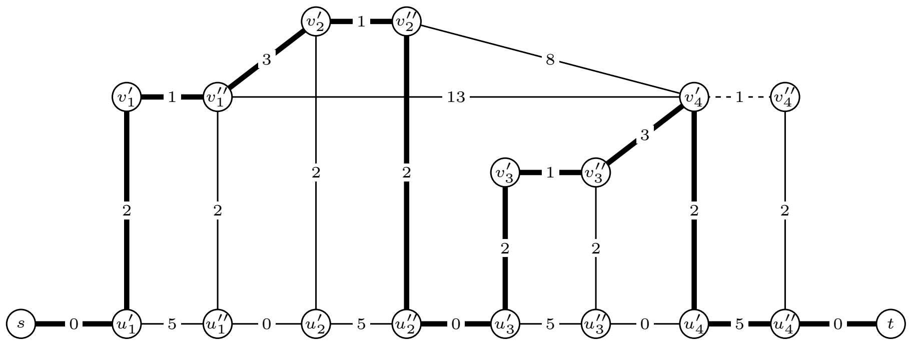

Figure 6: Thick lines are edges of the s-t path consisting of the s-u_1' line, a u_1'-u_2'' detour D_1, the u_2''-u_3' line, a u_3'-u_4' detour D_2 and the u_4'-t line. Note that the detour D_1 contains the 1-2 shortcut.

Proof. Note that Q starts with the edge su_1' and ends with the edge $u_n''t$. Thus we can decompose Q into an alternating sequence of lines and detours the s-ℓ_1 line, an ℓ_1-r_1 detour D_1, the r_1-ℓ_2 line, an ℓ_2-r_2 detour D_2, ..., an ℓ_m-r_m detour D_m, the r_m-t line (see Figure 6). Since no x-y detour is shorter than the x-y line, we have

$$\begin{aligned}\text{length}(Q) &= \text{length}(s\text{-}\ell_1 \textit{ line}) + \text{length}(D_1) + \text{length}(r_1\text{-}\ell_2 \textit{ line}) \\ &+ \text{length}(D_2) + \ldots + \text{length}(r_m\text{-}t \textit{ line}) \\ &\geq \text{length}(s\text{-}l_1 \textit{ line}) + \text{length}(\ell_1\text{-}r_1 \textit{ line}) + \text{length}(r_1\text{-}\ell_2 \textit{ line}) \\ &+ \text{length}(\ell_2\text{-}r_2 \textit{ line}) + \ldots + \text{length}(r_m\text{-}t \textit{ line}) \\ &\geq \text{length}(P) = 5(n-1).\end{aligned}$$

□

Recall that $\tau(G)$ denotes the size of the smallest vertex cover of G.

Claim 4 $\tau(G) = b_E'(H, s, t, E', 5(n-1))$.

Proof. Let F be a set of removable edges. We show that $\{v_i \mid v_i'v_i'' \in F\}$ is a vertex cover of G if and only if F is an edge blocker of $(H, s, t, 5(n-1), E')$.

Suppose $\{v_i \mid v_iv_i' \in F\}$ is a vertex cover of G. Thus there is no shortcut in the graph $(W, E \setminus F)$. By Claim 2 all x-y detours are longer than x-y lines, which by Claim 3 implies that every s-t path has length at least $5(n-1)$.

Conversely, suppose F is an edge blocker of $(H, s, t, E', 5(n-1))$ and suppose that $v_i'v_i'', v_j'v_j'' \notin F$, for some edge v_iv_j of G. Then F does not block the path consisting of the s-u_i' line, the u_i'-u_j'' detour $u_i'v_i'v_i''v_j'v_j''u_j''$ and the u_j''-t line which has a total length $5(n-1)-1$, a contradiction. □

Since it is NP-hard to approximate the minimum vertex cover within a factor smaller than $10\sqrt{5} - 21 \approx 1.36$ [11], Theorem 10 follows.

Note that we can similarly reduce the Minimum Vertex Cover Problem to restricted-MEBP for directed graphs. Let H be a digraph obtained from G by

- replacing every vertex v_i of G by two vertices v_i' and v_i'' connected by an arc $v_i'v_i''$ of length 0 for $i = 1, \dots, n$,
- replacing every edge v_iv_j, $i < j$, of G by $v_i''v_j'$ of length 0,
- adding to it a dipath $su_1u_2 \dots u_nt$, where arcs u_1u_2, $u_2u_3, \dots, u_{n-1}u_n$ have length 1 and all other arcs have length 0,
- adding two arcs u_iv_i' and $v_i''u_i$ of length 0 for $i = 1, \dots, n$ (see Figure 7).

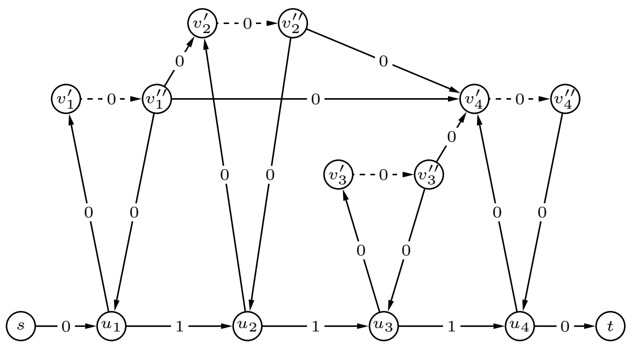

Figure 7: Digraph H. Solid lines are fixed arcs.

As before, all edges except for $v_1'v_1'', \dots, v_n'v_n''$ are fixed, we denote the set of fixed edges by E' and the threshold is $5(n-1)$.

Analogously to proof of Claim 1 we show that $\tau(G) = b_E'(H, s, t, E', n-1)$, implying the theorem. □

6 Proof of Theorem 7

In this section we prove Theorem 7 by reducing the problem of deciding whether a tripartite graph has a vertex cover of size at most k, which is known to be NP-hard [38], to restricted-MVVP. As shown in Section 7.3 and Section 7, for each instance of restricted-MVVP we can construct an instance of MVVP with the same optimal value and a bipartite input graph. Therefore Theorem 11 below implies Theorem 7.

Theorem 11 *It is NP-hard to approximate l_V' within a factor smaller than* 2.

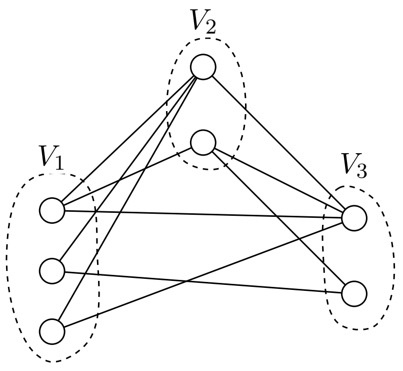

Figure 8: Tripartite graph G.

Proof. We will show that a $(2-\epsilon)$-approximation algorithm, where $\epsilon > 0$, can decide whether a tripartite graph has a vertex cover of size k in polynomial time.

Let G be a tripartite graph with vertex set $V = V_1 \cup V_2 \cup V_3$, where V_1, V_2 and V_3 are independent sets (see Figure 8). We construct an instance of restricted-MVVP. We obtain an undirected graph H from G by adding to it a path $su_1u_2u_3t$ and connecting every $v \in V_i$ to u_i, for $i = 1, 2, 3$ (see Figure 9). Let W denote the vertex set of H. We assign length 1 to edges u_1u_2, u_2u_3 and 0 to all other edges. Vertices u_1, u_2, u_3 are fixed.

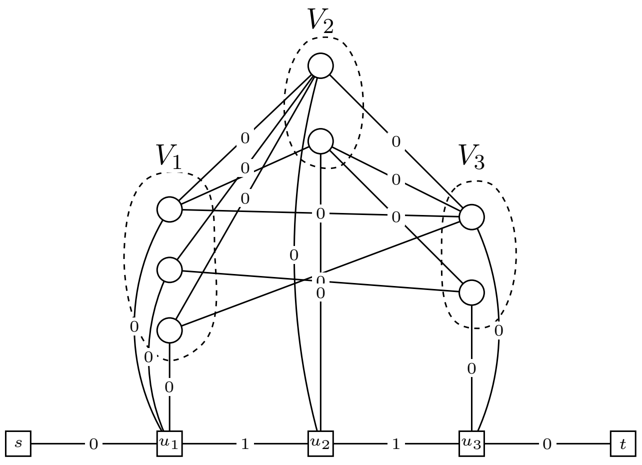

Figure 9: Graph H. Squares are fixed vertices.

Claim 5

(i) If G has a vertex cover of size at most k then $\ell'_V(H, s, t, V', k) = 2$.

(ii) If G does not have a vertex cover of size at most k then $\ell'_V(H, s, t, V', k) \leq 1$.

Proof. *(i)* Let U be a vertex cover of G such that $|U| \leq k$. Since $V \setminus U$ is an independent set in G, there is only one s-t path, $su_1u_2u_3t$, in $H[W \setminus U]$ and the length of this path is 2.

(ii) Since G has no vertex cover of size k, for every k-element subset U of removable vertices, $V \setminus U$ is not independent in G. Thus there is an edge xy in $H[W \setminus U]$ with x and y belonging to different parts of G. There are three cases:

Case 1: $x \in V_1$, $y \in V_2$. Then $su_1xyu_2u_3t$ is an s-t path of length 1.

Case 2: $x \in V_1$, $y \in V_3$. Then su_1xyu_3t is an s-t path of length 0.

Case 3: $x \in V_2$, $y \in V_3$. Then $su_1u_2xyu_3t$ is an s-t path of length 1.

Thus the s-t distance in $H[W \setminus U]$ is 0 or 1 for every k-element set U of removable vertices. $\square$

Since a $(2 - \epsilon)$-approximation algorithm, when run on H, must produce a solution smaller than 2 when $\ell'_V(H, s, t, V', k) \in \{0, 1\}$ and a solution greater than or equal to 2 when $l'_V(H, s, t, V', k) = 2$, such an algorithm could distinguish graphs that have a vertex cover of size k from graphs that do not.

We can similarly reduce the Most Vital Vertices Problem to restricted-MVVP for directed graphs. We obtain a directed graph H from G by replacing every edge vw, where $v \in V_i$, $w \in V_j$, $i < j$, of G by an arc vw, adding to it a dipath $su_1u_2u_3t$ and two arcs vu_i and u_iv for every $v \in V_i$, for $i = 1, 2, 3$ (see Figure 10). We assign length 1 to arcs u_1u_2, u_2u_3 and 0 to all other arcs. Vertices u_1, u_2, u_3 are fixed. The proof of Claim 5 is essentially the same as in the undirected case. $\square$

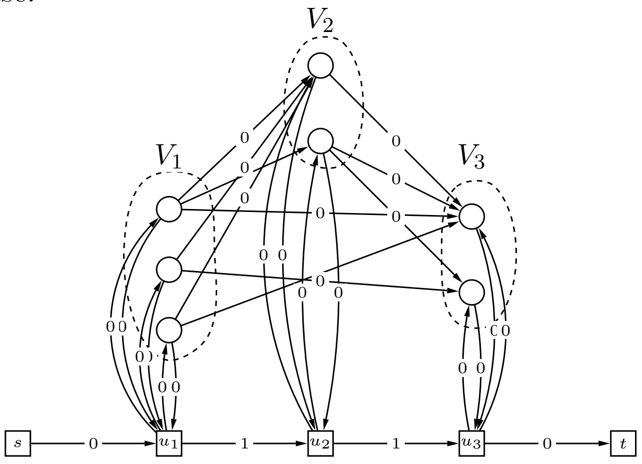

Figure 10: Digraph H. Squares are fixed vertices.

7 Reduction from Restricted to Original Problems

In this section for each instance of a restricted problem we construct in polynomial time an instance of the original problem with the same optimal value.

For an undirected graph we define the operation of *splitting a vertex x into n copies* as follows: we replace x by vertices $x^1, \dots, x^n$ and each edge xy of length l by edges $x^1y, \dots, x^ny$ of length l (see Figure 11). We call vertices $x^1, \dots, x^n$ *split vertices of x*.

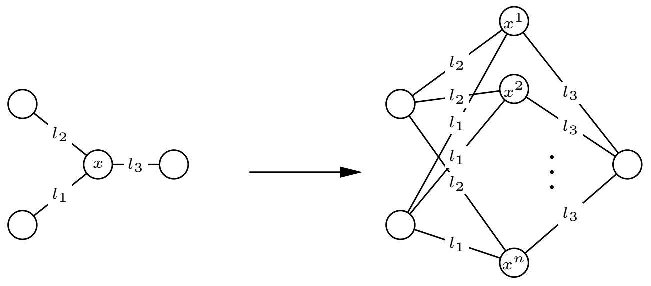

Figure 11: Operation of splitting x into n copies.

Analogously, for a directed graph we define the operation of *splitting a vertex x into n copies* as follows: we replace x by vertices $x^1, \dots, x^n$, each arc xy of length l by edges $x^1y, \dots, x^ny$ of length l and each arc yx of length l by edges $yx^1, \dots, yx^n$ of length l (see Figure 12).

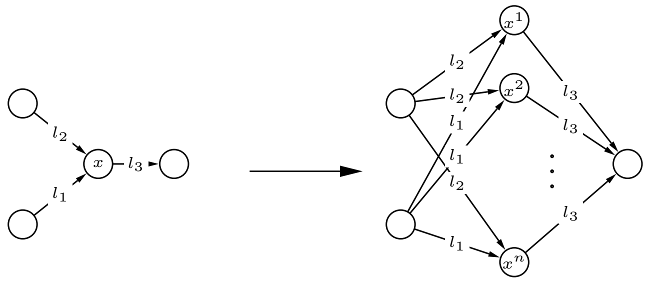

Figure 12: Operation of splitting x into n copies in directed graphs.

For a graph (digraph) we define the operation of *splitting an edge (arc) xy into n copies* as follows: we add vertices $z^1, \dots, z^n$, then replace the edge (arc) xy of length l by edges (arcs) $xz^1, \dots, xz^n$ of length l and edges (arcs) $z^1y, \dots, z^ny$ of length 0 (see Figure 13). We call vertices $z^1, \dots, z^n$ *division vertices of xy.*

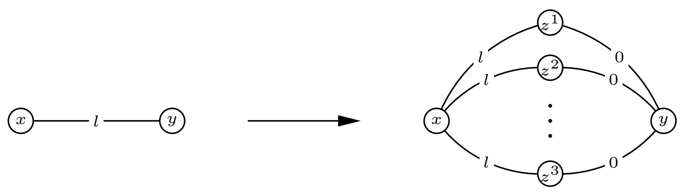

Figure 13: Operation of splitting xy into n copies.

7.1 Reduction from Restricted-MVBP to MVBP

For each instance of restricted-MVBP we construct in polynomial time an instance of MVBP with the same size of the minimum vertex blocker.

Let (G, s, t, V', d) be an instance of restricted-MVBP. Recall that we assume that all removable vertices form a vertex blocker. Let n be the number of vertices of G. We obtain a graph H from G by consecutively splitting every fixed vertex $x \in V'$ into n copies. Let W denote the vertex set of H.

Observation 1 *Let U be a subset of removable vertices. U is a vertex blocker of (H, s, t, d) if and only if U is a vertex blocker of (G, s, t, V', d).*

Claim 6 *Let U be a minimum vertex blocker of (H, s, t, k). If U contains a split vertex y of some fixed vertex x, then U contains all split vertices of x.*

Proof. Since $y \in U$, there is an s-t path in in $H[(W \setminus U) \cup y]$ through y which is shorter than k. Suppose there is a split vertex z of x such that $z \notin U$. Since the neighborhoods of y and z are the same we can replace y by z in this path and obtain a path of the same length in $H[(W \setminus U)]$, a contradiction with U being a vertex blocker. Thus all split vertices of x belong to U. □

Proposition 1 $b'_V(G, s, t, V', d) = b_V(H, s, t, d)$.

Proof. By Observation 1 every vertex blocker of (G, s, t, V', d) is a vertex blocker of (H, s, t, d). Thus $b'_V(G, s, t, V', d) \geq b_V(H, s, t, d)$.

Suppose $b'_V(G, s, t, V', d) > b_V(H, s, t, d)$. Let U be a minimum vertex blocker of (H, s, t, d). Since by our assumption all removable vertices form a vertex blocker of (G, s, t, V', d), we obtain $|U| < n$. Thus by Claim 6 U cannot contain split vertices. By Observation 1 U is a vertex blocker of (G, s, t, V', d), a contradiction. □

7.2 Reduction from restricted-MEBP to MEBP

For each instance of restricted-MEBP we construct in polynomial time an instance of MEBP with the same size of the minimum edge blocker.

Let (G, s, t, E', k) be an instance of restricted-MEBP. Recall that we assume that all removable edges (arcs) form a edge blocker. Let m be the number of edges (arcs) of G. We obtain a graph H from G by consecutively splitting every fixed edge (arcs) $xy \in E'$ into m copies.

Similarly to Proposition 1 we can show that the minimum edge blockers of (G, s, t, E', d) and (H, s, t, d) have the same size.

Proposition 2 $b'_E(G, s, t, E', d) = b_V(H, s, t, d)$.

7.3 Reduction from restricted-MVVP to MVVP

For each instance of restricted-MVVP we construct in polynomial time an instance of MVVP with the same optimal value.

Let (G, s, t, V', k) be an instance of restricted-MVVP. We construct an instance (H, s, t, k) as in Section 7.1. Similarly to Proposition 1 we can show that the the maximum of s-t distances in all graphs obtained from G by removing k vertices and the maximum of s-t distances in all graphs obtained from H by removing k vertices are equal.

Proposition 3 $\ell'_V(G, s, t, V', k) = \ell_V(H, s, t, k)$.

7.4 Reduction from restricted-MVEP to MVEP

For each instance of restricted-MVEP we construct in polynomial time an instance of MVEP with the same optimal value.

Let (G, s, t, V', k) be an instance of restricted-MVEP. We construct an instance (H, s, t, k) as in Section 7.2. Similarly to Proposition 1 we can show that the the maximum of s-t distances in all graphs obtained from G by removing k edges and the maximum of s-t distances in all graphs obtained from H by removing k edges are equal.

Proposition 4 $\ell'_E(G, s, t, E', k) = \ell_E(H, s, t, k)$.

8 Reduction to Bipartite Graphs

In this section for each instance of an original problem we construct in polynomial time an instance with a bipartite input graph and the same optimal value.

Let $G = (V, E)$ be a graph (digraph). We construct a graph (digraph) H by splitting every edge of G into 1 copy, where the operation of edge splitting was defined in Section 6. Let W be the set of vertices newly added division vertices. Note that the graph H is bipartite, since every edge of H has one endpoint in V and the other in W. Analogously to Proposition 1, we can prove that $b_E(G, s, t, d) = b_E(H, s, t, d)$ and $\ell_E(G, s, t, k) = \ell_E(H, s, t, k)$.

We next obtain a graph (digraph) H' from H by splitting every vertex of W into $|V|$ copies, where the operation of vertex splitting was defined in Section 6. Note that H' is still bipartite, and we can prove that $b_V(G, s, t, d) = b_V(H', s, t, d)$ and $\ell_V(G, s, t, k) = \ell_V(H', s, t, k)$.

9 Decision Problems

Using the well known connection between optimization and decision problems (see Chapter 29 in [40]) we can restate Theorems 5, 6, 7 and 8 as follows:

Proposition 5 (Reformulation of Theorems 5 and 6) *It is NP-hard to distinguish instances of MVBP having a vertex (edge) blocker of size d to paths of length at most k from those having all vertex (edge) blockers of size greater than $1.36\,d$ to paths of length at most k, where d is also a part of the input.* □

Proposition 6 (Reformulation of Theorems 7 and 8) *For every fixed $\epsilon > 0$ it is NP-hard to distinguish instances of MVVP having s-t distance d after removing some k vertices (edges) from those having s-t distance less than $\frac{1}{2-\epsilon} d$ in all induced subgraphs obtained by removing k vertices (edges), where k is also a part of the input.* □

Note that Theorem 4 is the strengthening of Proposition 6. Similarly it can be viewed as a two-sided generalization of Proposition 5, although the corresponding factor is worse.

10 Proof of Theorem 4

As shown in [7], it is NP-hard to approximate the size of the smallest vertex cover in tripartite graphs within a factor smaller than $\frac{34}{33}$. This can be restated as follows: for every fixed $\epsilon > 0$ it is NP-hard to distinguish tripartite graphs having a vertex cover of size k from those having all vertex covers of size greater than $(\frac{34}{33} - \epsilon)k$, where k is a part of the input.

The claim below immediately follows from Claims 5 and 6.

Claim 7 *Let G be a tripartite graph, let H_V and H_E be the graphs constructed from G in Sections 4 and 5, respectively, and let $\epsilon > 0$.*

(i) If G has a vertex cover of size k then
$l'_V(H_V, s, t, V', k) \geq 2$ *and* $l'_E(H_E, s, t, E', k) \geq 2$.

(ii) If all vertex covers of G have size larger than $(\frac{34}{33} - \epsilon)k$ then
$\ell'_V(H_V, s, t, V', (\frac{34}{33} - \epsilon)k) \leq 1$ *and* $\ell'_E(H_E, s, t, E', (\frac{34}{33} - \epsilon)k) \leq 1$.

Theorem 4 follows from Claim 7 and the inapproximability result stated in the beginning of this subsection.

References

[1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin, Network Flows: Theory, Algorithms, and Applications, Prentice Hall, New Jersey, 1993.

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

[3] A. Bar-Noy, S. Khuller, and B. Schieber, The complexity of finding most vital arcs and nodes, University of Maryland, Institute of Anvanced Computer Studies, College Park, MD, Technical Report CS-TR-3539, 1995.

[4] E. Beffara and S. Vorobyov, Adapting Gurvich-Karzanov-Khachiyan's algorithm for parity games: Implementation and experimentation, Technical Report 020, Department of Information Technology, Uppsala University, 2001 (available at https://www.it.uu.se/research /reports/#2001).

[5] E. Beffara and S. Vorobyov, Is randomized Gurvich-Karzanov-Khachiyan's algorithm for parity games polynomial? Technical Report 025, Department of Information Technology, Uppsala University, 2001 (available at https://www.it.uu.se/research/reports/#2001).

[6] H. Björklund, S. Sandberg, and S. Vorobyov, A Combinatorial strongly subexponential strategy improvement algorithm for mean payoff games, DIMACS Technical Report 2004-05 (2004) (available at http://dimacs.rutgers.edu/TechnicalReports/2004.html).

[7] Andrea E. F. Clementi, Pierluigi Crescenzi, and Gianluca Rossi. On the complexity of approximating colored-graph problems. In *COCOON*, pages 281–290, 1999.

[8] H.W. Corely and D.Y. Shaw, Most vital links and nodes in weighted networks, Operations Research Letters **1** (1982), pp. 157-160.

[9] K. Cormican, D.P. Morton, and R.K. Wood. Stochastic network interdiction. *Operations Research*, 46:184–197, 1998.

[10] W. H. Cunningham. Optimal attack and reinforcement of a network. *J. ACM*, 32(3):549–561, 1985.

[11] I. Dinur and S.Safra. On the hardness of approximating minimum vertex cover. *Annals of Mathematics*, 162:439–485, 2005.

[12] A. Eherenfeucht and J. Mycielski, Positional games over a graph, Notices of the American Mathematical Society **20** (1973), A-334.

[13] A. Ehrenfeucht and J. Mycielski, Positional strategies for mean payoff games, International Journal of Game Theory **8** (1979), pp. 109-113.

[14] M.L. Fredman and R.E. Tarjan, Fibonacci heaps and their uses in improved network optimization algorithms, Journal of the ACM **34(3)** (1987), pp. 596-615.

[15] D.R. Fulkerson and G.C. Harding, Maximizing the minimum source-sink path subject to a budget constraint, Mathematical Programming **13** (1977), pp. 116-118.

[16] T. Gallai, Maximum-minimum Sätze über Graphen. *Acta Mathematica Academiae Scientiarum Hungaricae* **9** (1958) pp. 395-434.

[17] M.R. Garey and D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, 1979.

[18] P.M. Ghare, D.C. Montgomery, and T.M. Turner, Optimal interdiction policy for a flow network, Naval Research Logistics Quarterly **18** (1971), pp. 37-45.

[19] B.L. Golden, A problem in network interdiction, Naval Research Logistics Quarterly **25** (1978), pp. 711-713.

[20] L.M. Goldschlager, The monotone and planar circuit value problem are log space complete for P, SIGACT News **9(2)** (1977), pp. 25-29.

[21] R. Greenlaw, H.J. Hoover, and W.L. Ruzzo, Limits to Parallel Computation: P-Completeness Theory, Oxford University Press, 1995.

[22] M. Grötschel, C.L. Monma, and M. Stoer, Computational results with a cutting plane algorithm for designing communication networks with low-connectivity constraints, Operations Research **40** (1992), pp. 309-330.

[23] V. Gurvich, A. Karzanov, and L. Khachiyan, Cyclic games and an algorithm to find minimax cycle means in directed graphs, USSR Computational Mathematics and Mathematical Physics **28** (1988), pp. 85-91.

[24] J. Håstad. Some optimal inapproximability results. In *STOC '97: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 1–10, New York, NY, USA, 1997. ACM Press

[25] E. Israely and K. Wood, Shortest-path network interdiction, Networks **40(2)** (2002), pp. 97-111.

[26] M. Jurdzinski, M. Paterson, and U. Zwick. A deterministic subexponential algorithm for solving parity games. *SODA 2006*: pp. 117-123.

[27] George Karakostas. A better approximation ratio for the vertex cover problem. In *ICALP*, pages 1043–1050, 2005.

[28] R. Karp, Reducibility among combinatorial problems, in R.E. Miller and J.W.Thatcher, eds. Complexity of computer computations, Plenum Press, New York (1972), pp. 85-103

[29] R. Karp, A Characterization of the Minimum Cycle Mean in a Digraph, Discrete Math. **23** (1978), pp. 309-311.

[30] A.V. Karzanov and V.N. Lebedev, Cyclical games with prohibition, Mathematical Programming **60** (1993), pp. 277-293.

[31] K. Malik, A.K. Mittal, and S.K. Gupta, The k most vital arcs in the shortest path problem, Operations Research Letters **8** (1989), pp. 223-227.

[32] A.W. McMasters and T.M. Mustin, Optimal interdiction of a supply networks, Naval Research Logistics Quarterly 17 (1970), pp. 261-268

[33] D. Medhi, A unified approach to network survivability for teletraffic networks: Models, algorithms and analysis, IEEE Transactions on Communications **42** (1994), pp. 534-548.

[34] H. Moulin, Prolongement des jeux à deux joueurs de somme nulle, Bull. Soc. Math. France, Memoire **45**, (1976).

[35] H. Moulin, Extension of two person zero sum games, Journal of Mathematical Analysis and Apllication **55 (2)** (1976), pp. 490-507.

[36] C.A. Phillips, The network inhibition problem, Proceedings of the 25th Annual ACM Symposium on the Theory of Computing, 1993, pp. 776-785.

[37] N.N. Pisaruk, Mean cost cyclical games, Mathematics of Operations Research **24(4)** (1999), pp. 817-828.

[38] S. Poljak. A note on the stable sets and coloring of graphs. *Commentiones Mathematicae Universitatis Carolinae*, 15:307–309, 1974.

[39] A. Schrijver, Combinatorial Optimization: Polyhedra and Efficiency, Algorithms and Combinatorics **24**, Springer, 2003.

[40] V. Vazirani. *Approximation Algorithms.* Springer Verlag, Berlin, Heidelberg, New York, 2001.

[41] D.K. Wagner, Disjoint (s,t)-cuts in a network, Networks **20** (1990), pp. 361-371.

[42] A. Washburn and K. Wood, Two-person zero-sum games for network interdiction, Operations Research **43(2)** (1995), pp. 243-251.

[43] P.S. Whiteman. Improving single strike effectiveness for network interdiction. *Military Operations Research*, 4:15–30, 1999.

[44] R.K. Wood, Deterministic network interdiction, Mathematical and Computer Modelling **17** (1993), pp. 1-18.

[45] U. Zwick , M. Paterson, The complexity of mean payoff games on graphs, Theoretical Computer Science **158(1-2)** (1996), pp. 343-359.