DIMACS Technical Report 2005-04
February 2005

Extending Dijkstra's Algorithm to Shortest Paths with Blocks[1]

by

Jihui Zhao[2]
Dept. of Computer Science
Rutgers University
New Brunswick, New Jersey 08854

Vladimir Gurvich[3,4]
RUTCOR
Rutgers University
New Brunswick, New Jersey 08854

Leonid Khachiyan[3]
Dept. of Computer Science
Rutgers University
New Brunswick, New Jersey 08854

[1]This research was supported in part by NSF grant IIS-0118635.
[2]Affiliated Graduate Student Member.
[3]DIMACS Permanent Member.
[4]The author gratefully acknowledges the partial support by DIMACS.

ABSTRACT

We consider the problem of computing shortest paths in a directed arc-weighted graph $G = (V, A)$ in the presence of an adversary that can block, for each vertex $v \in V$, some subsets $X(v)$ of the arcs $A(v) = \{e \in A \mid e = (v, u)\}$ leaving v. We assume that if $X(v) \subseteq A(v)$ is an admissible blocking set of arcs at v, then so is any subset of $X(v)$. In other words, for each $v \in V$, the hypergraph $\mathcal{B}(v)$ of all admissible blocks at v is an independence system in $2^{A(v)}$. We show that when the independence systems $\mathcal{B}(v),\ v \in V$ are given by an arbitrary membership oracle, and the input arc-weights are non-negative, the single-destination version of the problem can be solved by a natural extension of Dijkstra's algorithm in $O(|A| \log |V|)$ time and at most $|A|$ monotonically increasing blocking tests. We obtain better bounds for the special case where each blocking system $\mathcal{B}(v)$ consists of all arc-sets $X(v) \subseteq A(v)$ of given cardinality $p(v) \leq$ out-deg(v), and also show that computing shortest paths with blocks becomes NP-hard when the adversary can block a given number of arcs arbitrarily distributed in the input graph.

1 Introduction

Let $G = (V, A)$ be a digraph with given arc-weights $w(e)$, $e \in A$, and let $s, t \in V$ be two distinguished vertices of G. We consider the problem of computing the shortest s-t path in G when an adversary can block, for each vertex $v \in V$, some subsets $X(v)$ of the arcs $A(v) = \{e \in A \mid e = (vu)\}$ leaving v. We assume that the blocking arc-sets $X(v) \subseteq A(v)$ at v can be selected independently of the blocks $X(u)$ selected for all other vertices of the digraph, and that for each vertex v, the collection $\mathcal{B}(v)$ of all admissible blocks $X(v)$ forms an *independence system*: if $X(v) \in \mathcal{B}(v)$ is an admissible block at v, then so is any subset of $X(v)$. Alternatively, we could replace the independence systems $\mathcal{B}(v)$ by the collections of all inclusion-wise maximal blocking arc-sets. In general, we will only assume that the blocking systems $\mathcal{B}(v)$ are given by a *membership oracle* $\mathcal{O}$:

> 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 blocked.

A similar formalization of blocking sets via membership oracles is used by Pisaruk [15]. We will also consider two special types of blocking systems:

($\mathcal{B}_1$) The blocking system is given by a function $p(v) : V \to \mathcal{Z}_+$, where $p(v) \leq |A(v)| =$ out-deg(v). For each vertex v, the adversary can block any collection of (at most) $p(v)$ arcs leaving v. The numbers $p(v)$ define *digraphs with prohibitions* introduced in [12];

($\mathcal{B}_2$) There are two types of vertices: *control vertices*, where the adversary can select any out-going arc $e \in A(v)$ and block all the remaining arcs in $A(v)$, and *regular vertices*, where the adversary can block no arc. This case, considered in [11, 4], is a special case of $\mathcal{B}_1$: $p(v) = |A(v)| - 1$ for control vertices, and $p(v) = 0$ otherwise.

Let us call a digraph $G' = (V, A')$ *admissible for* $G = (V, A)$ if A' is obtained from A by deleting some admissible sets of outgoing arcs $X(v) \in \mathcal{B}(v)$ for each vertex $v \in V$. Consider the following problem:

Given an arc-weighted digraph $G = (V, A, w)$ and a blocking system $\mathcal{B}$, find an admissible graph G' that maximizes the distance from a given start vertex s to a given terminal vertex t:

$$d(s,t) \stackrel{\text{def}}{=} \max\{\ s\text{-}t \text{ distance in } G' \mid G' \text{ admissible digraph of } G\}.$$

We call $d(s, t)$ the *blocking distance* from s to t. We will see from what follows that, for any fixed terminal vertex $t \in V$, the adversary 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[1]

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

[1]Note, however, that if we fix a start vertex s, then distinct terminal vertices t may require distinct optimal admissible digraphs.

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

> $\mathcal{SPB}$ (Shortest paths with blocks) : *Given an arc-weighted digraph* $G = (V, A, w)$, *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 start vertices* $v \in V \setminus \{t\}$ to t.

To illustrate, suppose that for each arc $e = (uv)$ we are given a probability $p(e)$ that some undesirable transition (e.g, contraband smuggling) from u to v can be carried out undetected. Then, under the independence assumption and letting $w(e) = -\log p(e) \geq 0$, problem $\mathcal{SPB}$ can be interpreted as the uniform maximization of interception capabilities for a given target t under limited inspection resources distributed over G.

Another application of problem $\mathcal{SPB}$ is related to a class of games on graphs known as *cyclic* or *mean payoff games* [5, 6, 13, 14, 11]. Björklund, Sandberg and Vorobyov [4] observed that this class of games is polynomially reducible to problem $\mathcal{SPB}$ with blocks of type $\mathcal{B}_2$, provided that the arc-weights in G have arbitrary signs. A mean payoff game is a zero-sum game played by two players on a finite directed arc-weighted graph G all vertices of which have positive out-degrees. The vertex set of the digraph is partitioned into two subsets of vertices controlled by the players, who move a chip along the arcs of the digraph, starting from some given vertex $s \in V$. When the players select some *positional strategies*, i.e. select one outgoing arc for each of the vertices they control, the sequence of moves settles on a simple directed cycle of G, whose average arc-weight is the payoff in the game corresponding to the selected strategies.

Ehrenfeucht and Mycielski [5, 6] and Moulin [13, 14] introduced mean payoff games on bipartite digraphs and proved the existence of the value for such games in positional strategies. Gurvich, Karzanov and Khachiyan [11] extended this result to arbitrary digraphs and suggested a potential reduction algorithm to compute optimal positional strategies for the two players. Assuming that the vertices assigned to the maximization player are controlled vertices for $\mathcal{B}_2$, while the vertices assigned to the minimization player are regular vertices for $\mathcal{B}_2$, the determination of an optimal positional strategy for the maximization 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 start vertex s. Beffrara and Vorobyov [2] report on computational experiments with the potential-reduction algorithm [11] in which it was used to solve very large instances of mean payoff games. However, for some special instances of games with exponentially large arc-weights, the simplex-like pivoting algorithm [11] may require exponentially many steps (see [11, 3]. Interestingly, computational experiments [3] 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 [12] extended the potential reduction algorithm [11] to so-called mean payoff games with prohibitions, i.e, to blocking systems of type $\mathcal{B}_1$. Pisaruk [15] further extended these results to blocking systems defined by arbitrary membership oracles $\mathcal{O}$, and showed that in this general setting, the potential reduction algorithm [11] is pseudo-polynomial. Zwick and Paterson [18] gave another pseudo-polynomial algorithm for blocks of type $\mathcal{B}_2$.

As mentioned above, mean payoff games can be reduced to shortest paths with blocks and arc-weights of arbitrary sign. For instance, 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 all cycles of G' that are reachable from s is at least ξ?

After the substitution $w(e) \to w(e) - \xi$ we may assume without loss of generality that $\xi = 0$, and then (ξ) becomes equivalent to determining whether or not the blocking distance $d(s, v)$ is equal to $-\infty$ for some vertex $v \in V$.

Björklund, Sandberg and Vorobyov [4] recently showed that mean payoff games can be solved in *expected sub-exponential time.* 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 obviously in NP$\cap$coNP [12, 18]. Accordingly, for arc-weights of arbitrary sign and magnitude, the polynomial-time complexity of shortest paths with blocks also remains open – it is only known that this problem can be solved in pseudo-polynomial time, see e.g. [4]. In this paper, we show that for non-negative arc-weights, shortest paths with blocks can be computed 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 $w : A \to \Re_+$, and a terminal vertex $t \in V$,*
(i) *The special case of problem $\mathcal{SPB}$ for blocking systems $\mathcal{B}_1$ can be solved in time*

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

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, $\mathcal{SPB}$ 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 $\mathcal{SPB}$ 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 [7] for the regular shortest path problem without blocks.

Let us also mention that by Theorem 1, problem $\mathcal{SPB}$ can be solved in strongly polynomial time for any digraph $G = (V, A)$ that has no cycles of negative total arc weight. When G has no negative cycles, the input arc-weights $w(vu)$ can be transformed into non-negative arc-weights by a potential transformation $w(vu) \to w(vu) + \varepsilon(u) - \varepsilon(v)$, where $\varepsilon : V \to \Re$ are some vertex weights [1, 16]. Such a transformation can be found in $O(|A||V|)$ time and it does not change the set of optimal arc blocks for $\mathcal{SPB}$.

We close with two negative observations.

1) It is well known that the regular shortest path problem is in NC, i.e., it can be efficiently solved in parallel. In contrast, problem $\mathcal{SPB}$ 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 [9, 10].

2) The independence systems $\mathcal{B} \subseteq 2^A$ considered in this paper are Cartesian products of independence systems $\mathcal{B}(v) \subseteq 2^{A(v)}$ defined on the sets $A(v)$ of out-going arcs for each vertex v of $G = (V, A)$:

$$\mathcal{B} = \bigotimes_{v \in V \setminus \{t\}} \mathcal{B}(v).$$

When $\mathcal{B} \subseteq 2^A$ is not decomposable as above, computing the blocking distances in G may be NP-hard for very simple blocking systems and unit arc-weights. In particular, in Section 3 we show the NP-hardness of this problem for directed (or undirected) graphs when an adversary can block a given number p of arcs (or edges) arbitrarily distributed in the input graph.

Theorem 2 *The following two decision problems are NP-complete:*

$\mathcal{A}$: *Given a digraph $G = (V, A)$ with two distinguished vertices $s, t \in V$ and two integers p and q, determine whether there exists a set $A' \subseteq A$ of at least $|A| - p$ arcs such that any s, t-path in $G' = (V, A')$ contains at least q arcs.*

$\mathcal{E}$: *Given a graph $G = (V, E)$ with two distinguished vertices $s, t \in V$ and two integers p and q, determine whether there exists a set $E' \subseteq E$ of at least $|E| - p$ edges such any s, t-path in $G' = (V, E')$ contains at least q edges.*

The remainder of the paper is the proof of Theorems 1 and 2 in Sections 2 and 3, respectively.

2 Dijkstra's Algorithm for Shortest Paths with Blocks

We first describe an extension of Dijkstra's algorithm for problem $\mathcal{SPB}$ 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 E (e.g., on the set $A(v)$ of arcs leaving a given vertex v of G). Given a mapping $k : E \to \Re$, and a set $Y \subseteq E$, let

$$k_{\mathcal{B}}(Y) = \max_{X \in \mathcal{B}} \; \min_{e \in Y \setminus X} k(e), \tag{1}$$

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 $(k(e_1),\, k(e_2),\, k(e_3),\, k(e_4)) = (1, 3, 3, 5)$, then

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

Considering the image $\{k(e),\ e \in Y\}$ as a sets 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 $k(e)$;
3. *Return* $k_{\mathcal{B}}(Y)$: Compute the right-hand side of (1) for the current key set.

Note that when the independence system is empty, $|\mathcal{B}| = 0$, we obtain the customary definition of a minimum priority queue.

When $\mathcal{B}$ is a blocking system of type $\mathcal{B}_1$, i.e., $X \in \mathcal{B}$ whenever $|X| \leq p$ for some given integer $p \leq |E|$, then

$$k_{\mathcal{B}}(Y) = \begin{cases} +\infty, & \text{if } |Y| \leq p; \\ (p+1)^{st} \text{ smallest key of } Y, & \text{if } |Y| \geq p+1. \end{cases}$$

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

- *A sequence of $d \geq p$ queue operations for an initially empty $\mathcal{B}_1$-queue can be implemented to run in $O(p + (d-p)\log(p+1))$ 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 $k_{\mathcal{B}}(Y)$ can be done by searching for the largest key k in the tree for which the oracle can block the set of all keys smaller than k. 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 $Y \subseteq E$ of keys, the (unique) inclusion-wise minimal blocking set $\hat{X}(Y) \in \mathcal{B}$ such that

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

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 $(k(e_1),\, k(e_2),\, k(e_3),\, k(e_4)) = (1, 3, 3, 5)$, 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 $\{k(e),\ e \in Y\}$, 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 $\mathcal{SPB}$

Given a digraph $G = (V, A)$, a non-negative weight function $w(v) : A \to \Re^+$, 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 $d(t,t) = 0$.

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:

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

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 $\rho(v) = d(v,t)$ 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 $\rho(u) = d(u,t)$), 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 $\rho(v) \leftarrow \min\{\rho(v),\, w(vu)+\rho(u)\}$. 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 $e = (vu) \in A$, we first insert arc e with the key value $k(e) = w(vu) + \rho(u)$ into $Y(v)$, and then let $\rho(v) \leftarrow k_{\mathcal{B}}(Y(v)) \stackrel{\text{def}}{=} \max_{X \in \mathcal{B}(v)} \min_{e \in Y(v) \setminus X} k(e)$. In particular, for the regular 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. $\quad \rho(v) \leftarrow +\infty$; **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. $\quad$ **If** $\min\{\rho(v),\ v \in S\} = +\infty$, **break loop and go to line 12;**
7. $\quad$ **Extract the vertex** u **with the smallest key value** $\rho(\cdot)$ **from** S;
8. $\quad\quad$ **For all arcs** $e = (vu) \in A$ **such that** $v \in S$, **do:**
9. $\quad\quad\quad$ $k(e) \leftarrow w(e) + \rho(u)$;
10. $\quad\quad\quad$ **Insert** $k(e)$ **into** $Y(v)$;
11. $\quad\quad\quad$ **Update the value of** $\rho(v)$: $\rho(v) \leftarrow k_{\mathcal{B}}(Y(v))$.

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 iteration 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 $\rho_i(v) = k_{\mathcal{B}}(Y_i(v))$ 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 $k_{\mathcal{B}}(Y_i(v))$ 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 algorithm,*

- $\rho(v) = d(v,t) \stackrel{\text{def}}{=} \max_{G' admissible}\{$*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:* $d(v,t) \equiv$ 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 $w_i : A \to \Re_+ \cup \{+\infty\}$:

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

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

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

then $d_0(v,t) = +\infty \geq d_1(v,t) \geq \ldots \geq d_n(v,t) \geq d(v,t)$ for all $v \in V \setminus \{t\}$. The correctness of the algorithm will follow from the following two invariants: *for all* $i = 0, 1, \ldots, n$,

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

$\mathcal{I}_i^T$: *If* $v \in T_i = V \setminus S_i$, *then* $\rho_i(v) = d(v,t)$ *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,* $\min\{\rho_i(v),\ v \in S_i\} \geq \max\{d(v,t),\ v \in T_i\}$ *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 $\rho_0(t) = d_0(t,t) = 0$ and $\rho_0(v) = d_0(v,t) = +\infty$ 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 $d_n(v,t) \equiv d_{n-1}(v,t)$ 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 $\rho_i(u) = d(u,t)$, observe that by $\mathcal{I}_i^S$, $\rho_i(u) = d_i(u,t) \geq d(u,t)$. 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 $\rho_i(u) = d(u,t)$ 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 $\rho_i(u) = k(e_*) = w(e_*) + \rho_i(v)$, where $e_* = (uv) \in A$ 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 $\rho_{i+1}(u) = d(u,t)$, as stated in $\mathcal{I}_{i+1}^T$. Note that (3) and invariant $\mathcal{I}_i^T$also tell us that $\min\{\rho_i(v),\ v \in S_i\} = d(u,t) \geq \max\{d(v,t),\ v \in T_i\}$. Let us now show that after the algorithm updates $\rho(v)$ on S_{i+1}, we still have

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

again as stated in $\mathcal{I}_{i+1}^T$. Suppose to the contrary, that $\rho_{i+1}(v) < d(u,t) = \rho_i(u)$ for some vertex $v \in S_{i+1}$. Then from (3) it would follow that $e = (vu)$ 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 $\rho_{i+1}(v) = k_{\mathcal{B}}(Y_{i+1}(v))$ could not have dropped below the minimum of $\rho_i(v)$ and $k(e) = w(vu) + d(u,t)$, which is at least $d(u,t)$. But if $e \in \hat{X}(Y_{i+1}(v))$ then again $k_{\mathcal{B}}(Y_{i+1}(v)) \geq k(e)$, 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 $\rho_{i+1}(v) \leftarrow k_{\mathcal{B}}(Y_{i+1}(v))$ 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 $d_n(v,t) \equiv d(v,t)$, and $\mathcal{I}_n^S$yields $\rho_n(s) = d_n(s,t) = d(s,t)$. By $\mathcal{I}_n^T$, we also have $\rho_n(v) = d(v,t)$ 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 for $\mathcal{SPB}$

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

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

Iteration loop:

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

The modified algorithm keep 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 $e = (vv')$ across the cut is assigned the key value $k(e) = w(e) + \rho(v')$, where $\rho(v') = d(v',t)$ for all vertices $v' \in T$. In addition to the arcs in the current cut, $\mathcal{A}$ may also contain some arcs $e = (vv')$ 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 $e = (vv')$ belongs to the current cut and can be blocked along with the arcs that have already been blocked at v. The first arc $e = (vv')$ 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.

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 Computing blocking distances for 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 bread-first search. The extended Dijkstra's algorithm for problem $\mathcal{SPB}$ can be similarly simplified to prove part **(iii)** of Theorem 1.

BREADTH-FIRST SEARCH FOR $\mathcal{SPB}$

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 $d(1 : V, t) \equiv +\infty$ and an empty first-in first-out queue T;
2. $d(t, t) \leftarrow 0$; 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 $e = (vu) \in A$, do:
7. If $d(v, t) = +\infty$ and $\hat{X}(v) \cup \{e\}$ can be blocked, insert e into $\hat{X}(v)$;
8. else $d(v, t) \leftarrow d(u, t) + 1$, enqueue v into T, and return $d(v, t), \hat{X}(v)$.

The above algorithm runs in at most $|A|$ iterations. It follows by induction on $d(v, t)$ 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 Proof of Theorem 2

Problem $\mathcal{E}$ can be polynomially reduced to problem $\mathcal{A}$ by replacing each undirected edge (u, v) in the input graph $G = (V, E)$ by the four-vertex digraph shown in Fig. 1. (A similar reduction for another pair of decision problems on graphs and digraphs is described in [16], p. 1224.) Note that for the digraph shown in Fig. 1, removing arc (a, b) simultaneously blocks all directed paths between u and v, and so we can assume without loss of generality that if the arc-blocking adversary is allowed to block a total of p arcs, then all of the blocked arcs will be of type (a, b). Accordingly, in the directed version of the problem we also replace p and q by p and $3q$, respectively.

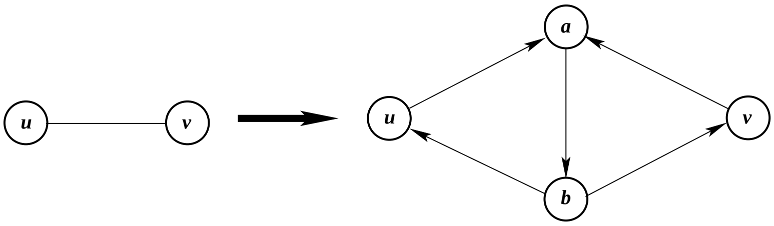

Figure 1: Reducing $\mathcal{E}$ to $\mathcal{A}$

To show the NP-completeness of problem $\mathcal{E}$, we reduce $\mathcal{E}$ from the following well-known NP-complete problem [8]:

> *Exact Cover by 3-Sets*: Given a set $U = \{u_1, \ldots, u_{3n}\}$ of $3n$ elements and a collection $\mathcal{T} = \{T_1, \ldots, T_k\}$ of k three-element subsets $T_i = \{u_{i_1}, u_{i_2}, u_{i_3}\}$ of U, determine if there exists a sub-collection $\mathcal{T}' = \{T_{i_1}, \ldots, T_{i_n}\}$ of n disjoint sets from T.

We let $p = k - n$ and consider the three-layered graph G shown in Fig. 2.

The bottom and middle layers of G contain one vertex for each element of U and T, respectively, and each vertex t_i of the middle layer is connected to those three vertices $u_{i_1}, u_{i_2}, u_{i_3}$ of the bottom layer which represent the elements contained in the triangle $T_i = \{u_{i_1}, u_{i_2}, u_{i_3}\}$. We make the $3k$ thick edges between the bottom and middle layers of G "indestructible" by replacing each of these edges by $p + 1$ parallel paths containing an intermediate vertex on each path[2], as shown in Fig. 2. Since the edge-blocking adversary cannot delete more than p edges, this effectively guarantees that each of the thick edges has length 2 and survives any blocking attack.

The top layer of G is a path P of k vertices $r_1, \ldots, r_k$. The edges in P are also indestructible but they have length 7 each. In other words, each of the dotted edges (r_i, r_{i+1}) actually

[2]We introduce the $p + 1$ intermediate vertices to avoid parallel edges and multi-graphs.

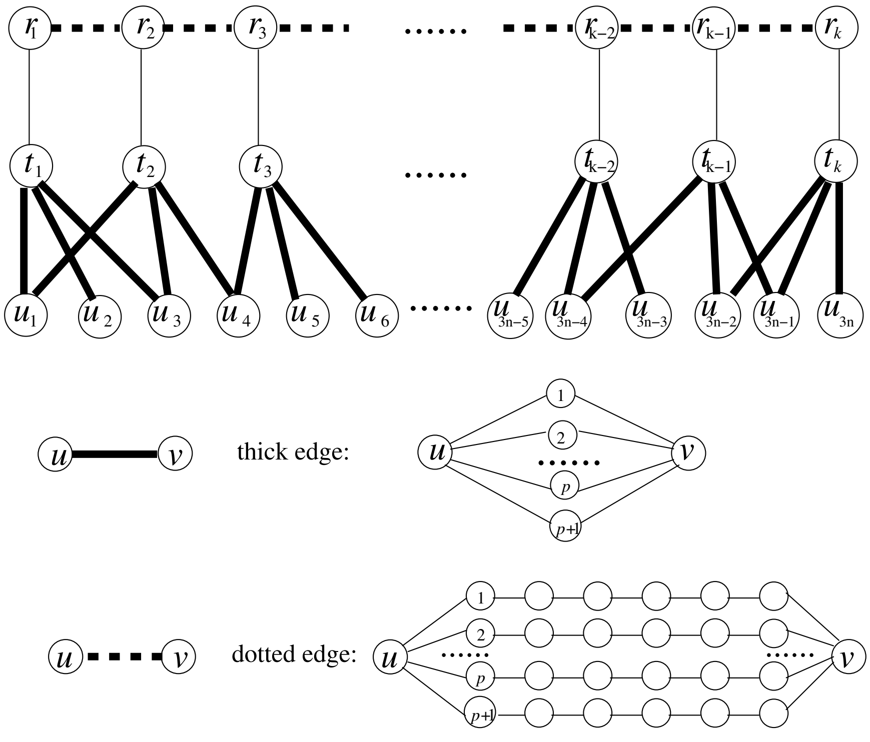

Figure 2: Reducing *Exact Cover by 3-Sets* to $\mathcal{E}$

represents $p+1$ parallel paths with six auxiliary intermediate vertices on each paths., see Fig. 2.

Finally, there are k additional edges (r_i, t_i) between the top and middle layers. These edges form a matching, and the blocking adversary is allowed to delete any $p = k - n$ of them.

It is easy to see that if there are n (disjoint) triangles $\mathcal{T}' = \{T_{i_1}, \ldots, T_{i_n}\}$ that exactly cover U, then by blocking the $p = k-n$ edges (r_i, t_i) for $T_i \notin \mathcal{T}'$ the edge-blocking adversary can guarantee that in the resulting graph, the distance between vertices r_1 and r_k is exactly $q = 7(k-1)$.. Conversely, if the blocking r_1, r_k-distance is at least q, then $\mathcal{T}$ contains n disjoint sets $T_{i_1}, \ldots, T_{i_n}$. $\square$

In conclusion, we mention the following special case of problem $\mathcal{A}$ which can be solved in polynomial time:
$\mathcal{B}$: *Given a digraph $G = (V, A)$ with two distinguished vertices $s, t \in V$ and two positive integers p and q, determine whether there exists a subsets A' of at most p arcs such that any s-t path in G contains at least q arcs from A'.*

Suppose without loss of generality that t is reachable from s in G, and let A' be an arbitrary q-cut, i.e. $|A' \cap P| \geq q$ 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 q disjoint s-t cuts $C_i = \mathrm{cut}(V_{i-1}, V_i)$ for $i = 1, \ldots, q$. Conversely, the union of any q arc-disjoint s-t cuts is a q-cut separating t from s. Hence problem $\mathcal{B}$ can be equivalently stated as follows:

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

The latter problem is known to be solvable in polynomial time. In fact, one can solve in strongly polynomial time the following weighted optimization version of the above problem [17]:

$\mathcal{B}'_w$: *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 q, find q arc-disjoint s, t-cuts $C_1, \ldots, C_q$ of minimum total weight $w(C_1) + \ldots + w(C_q)$.*

References

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

[2] 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**).

[3] 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**).

[4] 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**).

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

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

[7] 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.

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

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

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

[11] 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.

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

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

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

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

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

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

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