RC25637 (WAT1611-043) November 10, 2016
Mathematics

IBM Research Report

Faster Algorithms for Security Games on Matroids

Mourad Baïou
CNRS and Université Clermont II
Campus des Cézeaux, BP 125, 63173 Aubière Cedex
France

Francisco Barahona
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598 USA

Research Division
Almaden – Austin – Beijing – Brazil – Cambridge – Dublin – Haifa – India – Kenya – Melbourne – T.J. Watson – Tokyo – Zurich

Faster algorithms for security games on matroids

Mourad Baïou[1] and Francisco Barahona[2]

[1] CNRS and Université Clermont II, Campus des Cézeaux, BP 125, 63173 Aubière Cedex, France
[2] IBM T. J. Watson research Center, PO Box 218, Yorktown Heights, NY 10589, USA

Abstract. Given a matroid M defined by an independence oracle on a ground set E, the *Matroid Base Game* is played by two players: the *defender* chooses a basis B and (simultaneously) the *attacker* chooses an element $e \in E$. The attacker incurs in a cost $c(e)$ for choosing an element e, and if $e \in B$ then there is a probability $p(e)$ that the attacker will detect the defender. The defender has to find a bases-selection strategy that minimizes the average probability of being detected. The attacker has to find a probabilistic selection strategy that maximizes the average detection probability minus its average cost.
An algorithm to compute Nash-equilibrium mixed strategies was given in [21]. Its time complexity is $O(|E|^{10}IO)$, where IO is the time that it takes one call to the independence oracle. Here we present an algorithm that requires $O(|E|^6 IO)$ time. For graphic matroids, i.e., when the defender chooses a spanning tree in a graph $G = (V, E)$, and the attacker chooses an edge, we give an algorithm that takes $O(|V|^5)$ time. This type of game is extended to common bases of two matroids. For this case we give a strongly polynomial algorithm, settling a question that was left open in [21]. We also treat the case when the defender chooses a rooted arborescence in a directed graph $D = (V, \mathscr{A})$, and the attacker chooses an arc, we use this structure to give an algorithm that requires $O(|V||\mathscr{A}|^3 \log(|V|^2/|\mathscr{A}|) \log |\mathscr{A}|)$ time.

1 Introduction

Given a matroid M defined by an independence oracle on a ground set E, the *Matroid Base Game* is played by two players: the *defender* chooses a basis B and (simultaneously) the *attacker* chooses an element $e \in E$. The attacker pays a penalty $c(e)$ for choosing an element e, this might come from a cost or from the degree of difficulty of choosing the different elements. If $e \in B$ then there is a probability $p(e)$ that the attacker will detect the defender. The defender has to find a bases-selection strategy that minimizes the average probability of being detected. The attacker has to find a probabilistic selection strategy that maximizes the average detection probability minus its average penalty. This type of games was studied in [12] for graphic matroids, i.e., when the defender chooses a spanning tree in a graph, and the attacker chooses an edge. The case when M is a uniform matroid was studied in [14], and the general case was studied in [21].

This type of games falls in the category of *Security Games*, here the detection probability gives a measure of the security of the system. Also the structure of the solution gives insight on the behavior of the players. See [1], [11], [12], [14], and [21] for applications to Network Security and Steganography.

In a recent paper [21], an algorithm to compute Nash-equilibrium mixed strategies was given. Its time complexity is $O(|E|^{10}IO)$, where IO is the time that it takes one call to the independence oracle. Here we present an algorithm that requires $O(|E|^6 IO)$ time. For graphic matroids, i.e., when the game is played in a graph $G = (V, E)$, a characterization of the Nash-equilibrium was given in [12], but no algorithm was presented. For this case we give an algorithm that takes $O(|V|^5)$ time, to the best of our knowledge this is the first algorithm that takes advantage of the graphic structure. Also for the graphic case, the special case when $p(e) = 1$ and $c(e) = 0$, for each edge e, was studied in [1] and [11]. Algorithms to compute the attacker's strategy are given in these two papers, but the computation of the defender's strategy was left open.

This game is extended in [21] to the case when the defender chooses a common basis of two matroids. A (weakly) polynomial time algorithm is given, leaving open the question of whether it exists a strongly polynomial time algorithm. Here we give a strongly polynomial algorithm for this case. Then we study the game where the defender chooses a rooted arborescence in a directed graph, and the attacker chooses an arc. We take advantage of this structure to give an algorithm faster than the straightforward application of the common basis algorithm.

This paper is organized as follows. In Section 2 we give some definitions, notation, and review results on matroid polyhedra and Newton's method for ratio optimization. In Section 3 we treat the Matroid Base Game. Section 4 is devoted to graphic matroids. In Section 5 we treat common bases of two matroids. Rooted arborescences are treated in the Appendix.

2 Preliminaries

Here we define some notation, and review some results to be used in the subsequent sections. For a matroid M with ground set E, we use m to denote $m = |E|$. Its *rank function* r gives for each set $S \subseteq E$, the maximum cardinality of an independent set included in S. A function $f : 2^E \to I\!R$ is called *submodular* if $f(S \cup T) + f(S \cap T) \leq f(S) + f(T)$, for all $S, T \subseteq E$. The rank function is submodular. For a vector $y : E \to I\!R$, and $S \subseteq E$, we use $y(S)$ to denote $\sum_{e \in S} y(e)$. For $F \subseteq E$ the *incidence vector* of F, $x^F : E \to I\!R$, is defined by $x^F(e) = 1$ if $e \in F$, and $x^F(e) = 0$ otherwise. For $A \subseteq E$ we use $\bar{A}$ to denote $E \setminus A$.

For a graph $G = (V, E)$, and a family $\{S_1, \ldots, S_p\}$ of disjoint subsets of V, we denote by $\delta(S_1, \ldots, S_p)$ the set of edges with both endpoints in different sets $\{S_i\}$. We use n to denote $|V|$ and m to denote $|E|$.

For a directed graph $D = (V, \mathscr{A})$, and $S \subset V$, we use $\delta^+(S)$ to denote $\delta^+(S) = \{(i, j) \in \mathscr{A} \mid i \in S, j \notin S\}$. Also we use n to denote $|V|$ and m to denote $|\mathscr{A}|$.

2.1 Matroid Polyhedra

Here we review some classic results about polyhedra associated with matroids. Let M be a matroid with ground set E, let $P(M)$ be the convex hull of incidence vectors of independent sets of M, and let r denote the rank function of M. Edmonds [6] proved the following two theorems.

Theorem 1 *The polytope $P(M)$ is*
$$P(M) = \{x \,|\, x \geq 0, x(S) \leq r(S) \text{ for all } S \subseteq E\}.$$

Theorem 2 *For $u \in \mathbb{R}^E_+$,*
$$\max\{x(E) \,|\, x \in P(M), x \leq u\} = \min\{r(A) + u(\bar{A}) \,|\, A \subseteq E\}.$$

For the graphic case, it follows from a theorem of Tutte & Nash-Williams [22,16] that the dominant of the convex hull of incidence vectors of spanning trees of a graph is described as below.

Theorem 3 *The dominant of the spanning tree polytope of a graph $G = (V, E)$ is defined by*
$$x(\delta(S_1, \ldots, S_k)) \geq k - 1 \quad \text{for all partitions } \{S_1, \ldots, S_k\} \text{ of } V, \quad \text{and } x \geq 0.$$

Suppose now that M_1 and M_2 are two matroids on the same ground set E. Let r_1 and r_2 be their respective rank functions. Let P be the convex hull of incidence vectors of common independent sets. For $S \subseteq E$, let $r(S)$ be the maximum cardinality of a common independent set included in S. Edmonds [6] proved the following two theorems.

Theorem 4 *The polytope P is*
$$P = \{x \,|\, x \geq 0, x(S) \leq r_1(S),\ x(S) \leq r_2(S) \text{ for all } S \subseteq E\}.$$

Theorem 5 *For $u \in \mathbb{R}^E_+$,*
$$\max\{x(E) \,|\, x \in P,\ x \leq u\} = \min\{r(A) + u(\bar{A}) \,|\, A \subseteq E\}.$$

For a directed graph, Edmonds [8] proved that the dominant of the convex hull of incidence vectors of rooted arborescences is described as follows.

Theorem 6 *If $D = (V, \mathscr{A})$ is a directed graph, and $r \in V$ is a fixed vertex, the dominant of the convex hull of incidence vectors of arborescences rooted at r is defined by*
$$x(\delta^+(S)) \geq 1, \quad \text{for } S \subset V,\ r \in S, \quad \text{and } x \geq 0.$$

2.2 Newton's method for ratio maximization

Let $\mathcal{F}$ be a family of subsets of E. Let $f : \mathcal{F} \to \mathbb{R}_+$ and $g : \mathcal{F} \to \mathbb{R}_+$ two functions. In the subsequent sections we need a subroutine for finding $\max\{f(S)/g(S) \,|\, S \in \mathcal{F}\}$. This can be done with Newton's method as follows, see [5].

Newton's method

Step 0. Pick any set $\hat{S} \in \mathcal{F}$ with $g(\hat{S}) > 0$. Set $\mu = f(\hat{S})/g(\hat{S})$.
Step 1. Find $\bar{S} \in \mathcal{F}$ such that $f(\bar{S}) - \mu g(\bar{S}) = \max\{f(S) - \mu g(S) \mid S \in \mathcal{F}\}$.
Step 2. If $f(\bar{S}) - \mu g(\bar{S}) > 0$, then $f(\bar{S})/g(\bar{S}) > f(\hat{S})/g(\hat{S})$.
In this case set $\mu = f(\bar{S})/g(\bar{S})$, $\hat{S} \leftarrow \bar{S}$, and go to Step 1.
If $f(\bar{S}) - \mu g(\bar{S}) = 0$, stop, $\hat{S}$ is an optimal solution.

Notice that in Step 1, it is possible to have $\bar{S} = \hat{S}$, thus the maximum is always greater than or equal to zero. The value μ increases at each iteration, thus the number of iterations is finite. We give bounds for the number of iterations in the following sections.

3 The Matroid Base Game

We assume that M is a matroid with ground set E, given by an independence oracle. The defender chooses a basis B, and (simultaneously) the attacker chooses an element e. If the attacker chooses e, and $e \in B$, there is a probability $p(e) > 0$ that the attacker will detect the defender. Also there is a cost $c(e)$ for the attacker to choose element e. The defender has to minimize the average probability of being detected. The attacker has to maximize the average detection probability minus its average cost. Let $y(B)$ be the probability that the defender chooses basis B. For the attacker, let $x(e)$ be the probability of choosing element e. Let P be a diagonal matrix that contains the probabilities $\{p(e)\}$, and x a row vector, then xP is a row with components $\{x(e)p(e)\}$. Let D be a matrix whose rows correspond to the elements of E, and whose columns are the incidence vectors of all bases. Let y be a column vector whose components are $\{y(B)\}$. Then Dy is a column whose component associated with an element e is the probability that the element e is used by the defender. Thus $xPDy$ is the probability that the attacker finds the defender. Let c be a column so that $c(e)$ is the cost for the attacker of choosing element e. We have to concentrate on the following two-person game:

$$\max_x \min_y \; xPDy - xc \tag{1}$$

$$\sum_e x(e) = 1, \quad \sum_B y(B) = 1, \quad x \geq 0, \quad y \geq 0. \tag{2}$$

Von Neumann's classic Minimax Theorem [17] shows the existence of a Nash-equilibrium for this type of games. This can be computed by solving a linear program. One difficulty here is that this linear program has an exponential number of variables, that could be treated with the ellipsoid method [10]. Here we give polynomial combinatorial algorithms to compute both strategies.

3.1 Computing the Nash-equilibrium payoff

If we fix y we have

$$\max_x \; xPDy - xc, \quad \sum_e x(e) = 1, \quad x \geq 0.$$

And its dual is

$$\min_\mu \mu, \quad \mu \geq p(e) \sum \{y(B) \,|\, e \in B\} - c(e), \quad \text{for each element } e.$$

Then (1)-(2) is equivalent to

$$\begin{aligned} &\min_{\mu,y} \; \mu \\ &\mu - p(e) \sum \{y(B) \,|\, e \in B\} \geq -c(e), \quad \text{for each element } e, \\ &\sum_B y(B) = 1, \quad y \geq 0. \end{aligned}$$

This can be written as

$$\min_{\mu,y} \; \mu \tag{3}$$

$$\sum \{y(B) \,|\, e \in B\} \leq \frac{\mu + c(e)}{p(e)}, \quad \text{for each element } e, \tag{4}$$

$$\sum_B y(B) = 1, \quad y \geq 0. \tag{5}$$

Here we are looking for the minimum value of μ such that there is a convex combination of incidence vectors of bases that satisfies the element-capacities $\{(\mu + c(e))/p(e)\}$. We consider this in the sequel.

Define $u(e) = (\mu + c(e))/p(e)$ for $e \in E$. For a vector y satisfying (4)-(5) we define $z(e) = \sum \{y(B) \,|\, e \in B\}$. We should have $z(E) = r(E)$. From Theorem 2 we have

$$\max\{z(E) \,:\, z(S) \leq r(S), \text{ for } S \subseteq E,\, 0 \leq z \leq u\} = \min\{r(A) + u(\bar{A}) \,:\, A \subseteq E\} \tag{6}$$

Since we should have $z(E) = r(E)$, this implies $r(E) \leq r(A) + u(\bar{A}) = r(A) + \mu\, p'(\bar{A}) + q(\bar{A})$, for all $A \subseteq E$, where $p'(e) = 1/p(e)$, $q(e) = c(e)/p(e)$.

Thus we have to find the minimum value of μ such that $\mu\, p'(\bar{A}) \geq r(E) - r(A) - q(\bar{A})$, for $A \subset E$. This is

$$\hat{\mu} = \max_{A \subset E} \frac{r(E) - r(A) - q(\bar{A})}{p'(\bar{A})}. \tag{7}$$

As seen in Section 2, we can apply Newton's method. In each iteration we need to maximize $r(E) - r(A) - q(\bar{A}) - \mu p'(\bar{A})$, or minimize $r(A) - q'(A)$, where $q'(A) = q(A) + \mu p'(A)$. This can be solved with Narayanan's algorithm [15]. Its

time complexity is $O(m^3\rho^2 IO)$, where ρ is the rank of the matroid, and IO is the time that it takes one call to the independence oracle. So we assume that its complexity is $O(m^5 IO)$. In sub-section 3.3 we show that Newton's method takes at most m iterations, so it takes $O(m^6 IO)$ time to compute the Nash-equilibrium payoff of the game.

3.2 Computing both mixed strategies

Once the value $\hat{\mu}$ has been computed, we need a solution of (6). This can be found with the *Greedy* algorithm [7], as below. For a vector $\bar{z}$, we say that a set $S \subseteq E$ is *tight*, if $\bar{z}(S) = r(S)$.

Greedy Algorithm

Step 0. Start with any feasible vector $\bar{z}$, set $\bar{E} = E$, $A = \emptyset$.

Step 1. Pick any $e \in \bar{E}$. Let $S_e = \text{argmin}\{r(S) - \bar{z}(S) \mid e \in S\}$, and $\alpha = r(S_e) - \bar{z}(S_e)$. If $\alpha \leq u(e) - \bar{z}(e)$, set $\bar{z}(e) \leftarrow \bar{z}(e) + \alpha$, $A \leftarrow A \cup S_e$. Here A is a tight set, S_e becomes tight after adding α to $\bar{z}(e)$, and it follows from submodularity that the union of tight sets is also tight.
If $\alpha > u(e) - \bar{z}(e)$, set $\bar{z}(e) = u(e)$. In both cases set $\bar{E} \leftarrow \bar{E} \setminus e$.

Step 2. If $\bar{E} = \emptyset$, we have $\bar{z}(A) = r(A)$, and $\bar{z}(\bar{A}) = u(\bar{A})$. It follows from Theorem 2 that we have an optimal solution, and we stop.
If $\bar{E} \neq \emptyset$ go to Step 1.

Here the computing time is dominated by the time that it takes to find the minimum in Step 1. This can be done with Narayanan's algorithm [15]. Since we have m iterations, the greedy algorithm takes $O(m^6 IO)$ time.

Let $\bar{z}$ be a solution of (6). We have to express $\bar{z}$ as a convex combination of incidence vectors of bases. For that we propose to use Orlin's method [18] to minimize a submodular function. We define $f(S) = r(S) - \bar{z}(S)$ for $S \subseteq E$, then Orlin's method minimizes f and express $\bar{z}$ as a convex combination of incidence vectors of bases. This method has time complexity $O(m^5 EO + m^6)$, where EO is the time that it takes to evaluate f, thus we assume $EO = O(mIO)$, and we have an $O(m^6 IO)$ algorithm.

At this point we have $\bar{z} = \sum\{\bar{y}(B) z^B \mid B \in \mathscr{B}\}$, and $\sum \bar{y}(B) = 1$, $\bar{y} \geq 0$. Here z^B is the incidence vector of the basis B, and $\mathscr{B}$ denotes the set of bases of M. Let A be a solution of (7) then the attacker's strategy is given by

$$\bar{x}(e) = \begin{cases} \dfrac{\hat{\mu} p'(e)}{r(E) - r(A) - q(\bar{A})} & \text{if } e \in \bar{A}, \\ 0 & \text{otherwise.} \end{cases}$$

Thus $\sum_{e \in \bar{A}} \bar{x}(e) = 1$.

We have to see that $\bar{x}$ and $\bar{y}$ are solutions of (1)-(2). The Greedy algorithm produced a vector $\bar{z}$ with $\bar{z}(\bar{A}) = u(\bar{A})$, and for each $e \in \bar{A}$, $\sum\{\bar{y}(B) \mid e \in B\} =$

$\bar{z}(e) = u(e)$. Therefore

$$\bar{x}PD\bar{y} - \bar{x}c = \sum_{e\in\bar{A}} p(e)\frac{\hat{\mu}p'(e)}{r(E)-r(A)-q(\bar{A})}\frac{\hat{\mu}+c(e)}{p(e)} - \sum_{e\in\bar{A}}\frac{\hat{\mu}p'(e)}{r(E)-r(A)-q(\bar{A})}c(e)$$
$$= \sum_{e\in\bar{A}}\frac{\hat{\mu}^2 p'(e)}{r(E)-r(A)-q(\bar{A})} = \hat{\mu}$$

3.3 Analysis of Newton's method

Here we plan to show that Newton's method takes at most m iterations. Recall that each time one has to solve

$$\text{minimize} \quad r(A) - q(A) - \mu p'(A), \tag{8}$$

for $A \subseteq E$. If μ_1 and μ_2 are the values of μ in two consecutive iterations, then $\mu_2 > \mu_1$. The key is in the following lemma.

Lemma 7 *Let A_1 and A_2 be solutions of (8) for μ_1 and μ_2 respectively. Then we can assume that $A_1 \subset A_2$.*

Proof. It follows from Theorem 2 that a solution of (8) for μ, can be obtained by solving

$$\max z(E) \tag{9}$$
$$z(S) \leq r(S), \text{ for } S \subseteq E, \tag{10}$$
$$0 \leq z(e) \leq u(e), \text{ for } e \in E. \tag{11}$$

Here $u(e) = q(e) + \mu p'(e)$. This can be solved with the greedy algorithm above.

Let $u_i(e) = q(e)+\mu_i p'(e)$, for $e \in E$, $i = 1, 2$. Since $\mu_2 > \mu_1$, we have $u_2 \geq u_1$. Let A_1 be the solution obtained for μ_1, and $\bar{z}_1$ be the associated vector. We have $\bar{z}_1(A_1) = r(A_1)$. Then for μ_2 we can start the algorithm with $\bar{z} = \bar{z}_1$, and treat the elements of E starting with the elements in A_1, since this is a tight set, it will be included in the tight set A produced by the algorithm. □

Since we propose to solve (8) with the algorithm of [15], we should set $q(e) = M$, for $e \in A_1$, where M is a big number, then A_1 will be included in the solution for u_2. Notice that in all intermediate iterations of Newton's method we have $A_2 \neq A_1$, and we have equality only at the end.

4 The graphic case

Here we assume that for an undirected graph $G = (V, E)$, the defender picks a spanning tree T with probability $y(T)$, and the attacker picks an edge e with

probability $x(e)$. We take advantage of this structure to give an $O(n^5)$ algorithm. Recall that $n = |V|$, $m = |E|$. Here (3)-(5) becomes

$$\min_{\mu,y} \ \mu$$

$$\sum\{y(T) \,|\, e \in T\} \le \frac{\mu + c(e)}{p(e)}, \quad \text{for each edge } e, \text{ and} \quad \sum y(T) = 1, \quad y \ge 0.$$

Here we are looking for the minimum value of μ such that there is a convex combination of incidence vectors of spanning trees that satisfies the edge capacities $\{(\mu + c(e))/p(e)\}$. In other words, we have to minimize μ so that the vector with components $\{(\mu + c(e))/p(e)\}$ belongs to the dominant of the spanning tree polytope. It follows from Theorem 3 that we have to solve

$$\min \mu$$
$$\mu\, p'(\delta(S_1, \ldots, S_k)) + q(\delta(S_1, \ldots, S_k)) \ge k - 1, \text{ for all partitions } \{S_1, \ldots, S_k\} \text{ of } V,$$

where $p'(e) = 1/p(e)$, $q(e) = c(e)/p(e)$. Thus the payoff is

$$\hat{\mu} = \max \frac{k - 1 - q(\delta(S_1, \ldots, S_k))}{p'(\delta(S_1, \ldots, S_k))}, \tag{12}$$

where the maximum is taken among all partitions $\{S_1, \ldots, S_k\}$ of V. We use Newton's method, thus qiven $\bar{\mu}$ at each iteration we solve

$$\min \quad \bar{\mu} p'(\delta(S_1, \ldots, S_k)) + q(\delta(S_1, \ldots, S_k)) - (k - 1). \tag{13}$$

This reduces to n minimum cut problems, as shown in [2]. Theorem 16 in the Appendix shows that Newton's method takes at most n iterations. Therefore the computing time of this part of the algorithm is dominated by the time that it takes to solve n^2 minimum cut problems. Thus this part takes $O(n^5)$ time.

Let $\delta(S_1, \ldots, S_k)$ be a solution of (12). To obtain the attacker's strategy we set

$$\bar{x}(e) = \begin{cases} \dfrac{\hat{\mu} p'(e)}{k - 1 - q(\delta(S_1, \ldots, S_k))} & \text{if } e \in \delta(S_1, \ldots, S_k), \\ 0 & \text{otherwise.} \end{cases}$$

The defender's strategy can be obtained from a maximum packing of spanning trees with capacities $\{(\hat{\mu} + c_e)/p_e\}$. This can be done with the algorithm of [3], or with the algorithm of [9]. This last algorithm requires $O(n^3 m \log(n^2/m))$ time. Let $\bar{y}$ be the vector obtained, then as in Section 3, it is easy to see that

$$\bar{x} P D \bar{y} - \bar{x} c = \hat{\mu}.$$

5 Matroid Intersection

Here we assume that we have two matroids M_1 and M_2, and that the defender picks a common basis. This case was studied in [21], and a (weakly) polynomial

algorithm was given, leaving open the question of whether it exists a strongly polynomial algorithm. Here we show that an approach similar to the one in Section 3 is strongly polynomial. Again we have to solve

$$\min_{\mu,y} \ \mu \tag{14}$$

$$\sum\{y(B) \,|\, e \in B\} \leq \frac{\mu + c(e)}{p(e)}, \quad \text{for each element } e, \tag{15}$$

$$\sum_B y(B) = 1, \quad y \geq 0. \tag{16}$$

This time B represents a basis of both matroids, and we are looking for the minimum value of μ such that there is a convex combination of incidence vectors of common bases that satisfies the element-capacities $\{(\mu + c(e))/p(e)\}$.

Define $u(e) = (\mu + c(e))/p(e)$ for $e \in E$. For a vector y satisfying (15)-(16) we define $z(e) = \sum\{y(B) \,|\, e \in B\}$. We should have $z(E) = r(E)$. This time $r(S)$ is the maximum cardinality of a common independent set included in S, $S \subseteq E$. From Theorem 5 we have

$$\begin{aligned} &\max\{z(E) \,:\, z(S) \leq r_i(S), \text{ for all } S \subseteq E,\, i = 1, 2, \quad 0 \leq z \leq u\} = \\ &= \min\{r(A) + u(\bar{A}) \,:\, A \subseteq E\}. \end{aligned} \tag{17}$$

Since we should have $z(E) = r(E)$, we obtain $r(E) \leq r(A) + u(\bar{A}) = r(A) + \mu\, p'(\bar{A}) + q(\bar{A})$, for $A \subseteq E$, where $p'(e) = 1/p(e)$, $q(e) = c(e)/p(e)$. Thus we have to find the minimum value of μ such that $\mu\, p'(\bar{A}) \geq r(E) - r(A) - q(\bar{A})$, for $A \subset E$. Thus the payoff is

$$\hat{\mu} = \max_{A \subset E} \frac{r(E) - r(A) - q(\bar{A})}{p'(\bar{A})}. \tag{18}$$

In order to apply Newton's method, we need to maximize $r(E) - r(A) - q(\bar{A}) - \mu p'(\bar{A})$, or

$$\text{minimize} \quad r(A) - q'(A), \tag{19}$$

where $q'(S) = q(S) + \mu p'(S)$. Cunningham [4] gave a strongly polynomial algorithm that solves (19), so we need a bound for the number of iterations of Newton's method, this is given below.

Theorem 8 *Newton's method takes $O(m^2 \log m)$ iterations to solve* (18)*. The proof of this is given in the next sub-section.*

Thus we have a strongly polynomial algorithm to compute the Nash-equilibrium payoff of the game. Once the value $\hat{\mu}$ has been computed, we need a solution of (17). Cunningham [4] gave a strongly polynomial algorithm for this. Let $\bar{z}$ be the vector obtained, we have to express $\bar{z}$ as a convex combination of incidence vectors of common bases. That can be done with the algorithm in Theorem 41.13 of [20], that is strongly polynomial. This gives the defender's strategy. The attacker's strategy is obtained with a formula similar to the one in Section 3. Thus this game is solvable in strongly polynomial time. For rooted arborescences we give a faster algorithm in the Appendix.

5.1 Proof of Theorem 8

Here we plan to give a strongly polynomial bound for the number of iterations of Newton's method. For a ground set E, let $\mathcal{F}$ be a family of subsets of E. Let $f : \mathcal{F} \to \mathbb{R}_+$ and $g : \mathcal{F} \to \mathbb{R}_+$ be two functions. Newton's method is used to find $\max\{f(S)/g(S) \mid S \in \mathcal{F}\}$. Consider the special case when

$$f(S) = \sum_{e \in S} a(e), \quad \text{and} \quad g(S) = \sum_{e \in S} b(e), \tag{20}$$

for $S \in \mathcal{F}$, where $a : E \to \mathbb{R}_+$ and $b : E \to \mathbb{R}_+$ are two weight functions for the elements of E. For this case it was shown in [19] that it takes takes $O(m^2(\log m)^2)$ iterations, where $m = |E|$. This bound was improved to $O(m^2 \log m)$ in [23]. In our case we have to solve

$$\max_{A \subset E} \frac{r(E) - r(A) - q(\bar{A})}{p'(\bar{A})},$$

so neither the numerator nor the denominator are like in (20), and we cannot just apply the above results. However the proof in [23] can be adapted, we give the details below.

For $A \subset E$, we define $f(A) = r(E) - r(A) - q(E) + q(A)$ (the numerator), and $g(A) = p(E) - p(A)$ (the denominator). Let A_i be the set obtained at iteration i, and $f_i = f(A_i)$, $g_i = g(A_i)$, $\delta_{i+1} = f_i/g_i$, $h_i = f_i - \delta_i g_i$. The following three lemmas were proved in [19].

Lemma 9 *Let t be the index of the last iteration, then $h_1 > h_2 > \ldots > h_t$ and $g_1 > g_2 > \ldots > g_{t-1} \geq g_t$*

Lemma 10 *For all i, $\quad h_{i+1}/h_i + g_{i+1}/g_i \leq 1$.*

Lemma 11 *Let $c \in \mathbb{R}^p$ be a nonnegative vector. Let $\{y_1, \ldots, y_q\}$ be a set of vectors in $\{-1, 0, 1\}^p$. If for all $i = 1, \ldots, q-1$,*

$$0 < y_{i+1} \cdot c \leq (1/2) y_i \cdot c,$$

then $q = O(p \log p)$.

We have used $a \cdot b$ to denote the inner product between the vectors a and b. Now we need the following two lemmas.

Lemma 12 *There are at most $O(m \log m)$ iterations k such that $h_{k+1} \geq (1/2)h_k$.*

Proof. Let $k_1 < k_2 < \ldots < k_q$ be all indices k such that $h_{k+1} \geq (1/2)h_k$. Since $h_{i+1}/h_i + g_{i+1}/g_i \leq 1$, we have $0 < g_{k_i+1} \leq (1/2)g_{k_i}$. Since $\{g_i\}$ is decreasing, $g_{k_{i+1}} \leq g_{k_i+1}$, we have

$$0 < g_{k_{i+1}} \leq (1/2) g_{k_i}.$$

In order to apply Lemma 11 we have to write $g_{k_i} = y_i \cdot c$, where c is a nonnegative vector, and y_i is a vector with components in $\{-1, 0, 1\}$. So we define c as a vector

with components $\{p(e)\}$ and with a last component equal to $p(E)$. Then y_i is a vector that has a component -1 for each element $e \in A_i$, a component 0 for each element $e \notin A_i$, and a component equal to 1 in the position $m+1$. Then Lemma 11 implies that there are at most $O(m \log m)$ vectors y_i. □

Lemma 13 *There are at most $O(m^2 \log m)$ iterations k such that $h_{k+1} \leq (1/2)h_k$.*

Proof. Let $k_1 < k_2 < \ldots < k_q$ be all indices k such that $h_{k+1} \leq (1/2)h_k$. Here

$$h_i = f_i - \delta_i g_i = f_i - \frac{f_{i-1}}{g_{i-1}} g_i.$$

Since $\{h_i\}$ is decreasing, $h_{k_{i+1}} \leq h_{k_i+1} \leq (1/2)h_{k_i}$. Thus

$$f_{k_{i+1}} - \frac{f_{k_{i+1}-1}}{g_{k_{i+1}-1}} g_{k_{i+1}} \leq \frac{1}{2}\Big(f_{k_i} - \frac{f_{k_i-1}}{g_{k_i-1}} g_{k_i}\Big).$$

The sequence $\{g_i\}$ is decreasing, so $g_{k_{i+1}-1} < g_{k_i-1}$. Therefore

$$f_{k_{i+1}} g_{k_{i+1}-1} - f_{k_{i+1}-1} g_{k_{i+1}} \leq \frac{1}{2}\Big(f_{k_i} g_{k_i-1} - f_{k_i-1} g_{k_i}\Big).$$

Let $s_i = f_{k_i} g_{k_i-1} - f_{k_i-1} g_{k_i}$, then

$$0 < s_{i+1} \leq (1/2)s_i,$$

for $i = 1, \ldots, q$.

In order to use Lemma 11 we have to write $s_i = y_i \cdot c$, where c is a nonnegative vector, and y_i is a vector with components in $\{-1, 0, 1\}$. To simply notation let $R = r(E) - q(E)$, $P = p(E)$, also we write A'_i instead of A_{k_i}. Then

$$\begin{aligned} s_i = {} & (R - r(A'_i))P - (R - r(A'_i))p(A'_{i-1}) + Pq(A'_i) - q(A'_i)p(A'_{i-1}) \\ & -(R - r(A'_{i-1}))P + (R - r(A'_{i-1}))p(A'_i) - Pq(A'_{i-1}) + q(A'_{i-1})p(A'_i). \end{aligned} \tag{21}$$

For the first term in (21) we create a vector c^1 with components $|R-1|P, |R-2|P, \ldots, |R - r(E)|P$, and a vector y^1 with all components equal to zero, except for the component in position $r(A'_i)$, where we put $sg(R - r(A'_i))$. Here $sg(x) = 1$ if $x \geq 0$, and $sg(x) = -1$ otherwise.

Now we treat the last term of (21), the other six terms are treated in a similar way. We create an $m \times m$ matrix M whose coefficient in row r and column s is $M_{r,s} = |q(e_r)p(e_s)|$. We also create an $m \times m$ matrix Y whose coefficient in row r and column s is $Y_{r,s} = sg\big(q(e_r)p(e_s)\big)$ if $e_r \in A'_{i-1}$ and $e_s \in A'_i$, and $Y_{r,s} = 0$ otherwise. Then $M \cdot Y = q(A'_{i-1})p(A'_i)$, where $M \cdot Y = \sum_{i,j} M_{i,j} Y_{i,j}$. Finally we create a vector c^8 consisting of all rows of M, and a vector y^8 containing all rows of Y.

Assuming that the vectors $c^2, \ldots, c^6$ and $y^2, \ldots, y^6$ have been created in a similar way, we define the vectors $c = [c^1, \ldots, c^8]$, $y_i = [y^1, \ldots, y^8]$. Then $s_i = y_i \cdot c$. The number of components of c is $O(m^2)$, then Lemma 11 implies $q = O(m^2 \log m)$. □

Then Theorem 8 follows from lemmas 12 and 13.

Acknowledgments. We are grateful to H. Narayanan and to S.T. McCormick for some helpful discussions.

References

1. H. Aziz, O. Lachish, M. Paterson, and R. Savani, *Wiretapping a hidden network*, in Internet and Network Economics, Springer, 2009, pp. 438–446.
2. F. Barahona, *Separating from the dominant of the spanning tree polytope*, Operations research letters, 12 (1992), pp. 201–203.
3. ———, *Packing spanning trees*, Mathematics of Operations Research, 20 (1995), pp. 104–115.
4. W. H. Cunningham, *Optimal attack and reinforcement of a network*, Journal of the ACM (JACM), 32 (1985), pp. 549–561.
5. W. Dinkelbach, *On nonlinear fractional programming*, Management Science, 13 (1967), pp. 492–498.
6. J. Edmonds, *Submodular functions, matroids, and certain polyhedra*, Combinatorial structures and their applications, (1970), pp. 69–87.
7. ———, *Matroids and the greedy algorithm*, Mathematical programming, 1 (1971), pp. 127–136.
8. ———, *Edge-disjoint branchings*, Combinatorial algorithms, 9 (1973), pp. 91–96.
9. H. N. Gabow and K. Manu, *Packing algorithms for arborescences (and spanning trees) in capacitated graphs*, Mathematical Programming, 82 (1998), pp. 83–109.
10. M. Grötschel, L. Lovász, and A. Schrijver, *The ellipsoid method and its consequences in combinatorial optimization*, Combinatorica, 1 (1981), pp. 169–197.
11. A. Gueye, J. C. Walrand, and V. Anantharam, *Design of network topology in an adversarial environment*, in Decision and Game Theory for Security, Springer, 2010, pp. 1–20.
12. ———, *A network topology design game: How to choose communication links in an adversarial environment*, in Proc. of the 2nd International ICST Conference on Game Theory for Networks, GameNets, vol. 11, 2011.
13. J. Hao and J. B. Orlin, *A faster algorithm for finding the minimum cut in a directed graph*, Journal of Algorithms, 17 (1994), pp. 424–446.
14. A. Laszka and D. Szeszlér, *Hide and seek in digital communication: the steganography game*, in Proceedings of the 9th Hungarian-Japanese Symposium on Discrete Mathematics and its Applications, Fukuoka, Japan, 2015, pp. 126–136.
15. H. Narayanan, *A rounding technique for the polymatroid membership problem*, Linear algebra and its applications, 221 (1995), pp. 41–57.
16. C. S. J. Nash-Williams, *Edge-disjoint spanning trees of finite graphs*, Journal of the London Mathematical Society, 1 (1961), pp. 445–450.
17. J. v. Neumann, *Zur theorie der gesellschaftsspiele*, Mathematische Annalen, 100 (1928), pp. 295–320.
18. J. B. Orlin, *A faster strongly polynomial time algorithm for submodular function minimization*, Mathematical Programming, 118 (2009), pp. 237–251.
19. T. Radzik, *Fractional combinatorial optimization*, in Handbook of combinatorial optimization, Springer, 2013, pp. 1311–1355.
20. A. Schrijver, *Combinatorial optimization: polyhedra and efficiency*, vol. 24, Springer Science & Business Media, 2002.
21. D. Szeszlér, *Security games on matroids*, Mathematical Programming, (2016), pp. 1–18.
22. W. T. Tutte, *On the problem of decomposing a graph into n connected factors*, Journal of the London Mathematical Society, 1 (1961), pp. 221–230.
23. Q. Wang, X. Yang, and J. Zhang, *A class of inverse dominant problems under weighted l_∞ norm and an improved complexity bound for Radzik's algorithm*, Journal of Global Optimization, 34 (2006), pp. 551–567.

6 Appendix

6.1 Analysis of Newton's method in the graphic case

Here we show that for the graphic case, Newton's method requires at most n iterations to solve (12). We need the following two lemmas.

Lemma 14 *Let $u(e) = \bar{\mu}p'(e) + q(e)$, for $e \in E$. If $\delta(S_1, \ldots, S_k)$ is a solution of (13), and if $\{T_1, \ldots, T_r\}$ is a partition of some set S_j, $1 \leq j \leq k$, then*

$$u(\delta(T_1, \ldots, T_r)) \geq r - 1.$$

Proof. If $u(\delta(T_1, \ldots, T_r)) < r - 1$, then we could replace S_j with $\{T_1, \ldots, T_r\}$ and we would obtain a better solution of (13). □

Lemma 15 *Let μ_1 and μ_2 be two consecutive values of μ produced by Newton's method. If $\delta(S_1, \ldots, S_k)$ is a solution of (13) for μ_1, and if $\delta(T_1, \ldots, T_r)$ is a solution of (13) for μ_2, then for each $i = 1, \ldots, k$, there is an index $j(i)$ such that $S_i \subseteq T_{j(i)}$.*

Proof. Let $u_s(e) = \mu_s p'(e) + q(e)$, for $e \in E$, $s = 1, 2$. Suppose that for some index i, we have $S_i \cap T_{j_l} \neq \emptyset$, for $l = 1, \ldots, p$, $p \geq 2$. Lemma 14 implies

$$u_1(\delta(S_i \cap T_{j_1}, \ldots, S_i \cap T_{j_p})) \geq p - 1.$$

Since $\mu_2 > \mu_1$ we have

$$u_2(\delta(T_{j_1}, \ldots, T_{j_p})) > u_1(\delta(T_{j_1}, \ldots, T_{j_p})) \geq u_1(\delta(S_i \cap T_{j_1}, \ldots, S_i \cap T_{j_p})) \geq p - 1.$$

Then we could combine $\{T_{j_1}, \ldots, T_{j_p}\}$ into one set and obtain a better solution of (13). A contradiction. □

This lemma shows that the cardinality of the partition produced by Newton's method decreases at each iteration. Thus we have the following.

Theorem 16 *In the graphic case Netwon's method takes at most n iterations.*

6.2 Rooted Arborescences

Here we assume that for a directed graph $D = (V, \mathscr{A})$ with a *root* node r, the defender chooses an arborescence rooted at r, and the attacker picks an arc. Arborescences can be seen as common bases of two matroids, here we take advantage of their structure to derive a faster algorithm.

As before, we have to solve

$$\begin{aligned}
&\min_{\mu, y} \ \mu \\
&\sum \{y(A) \,|\, a \in A\} \leq \frac{\mu + c(a)}{p(a)}, \quad \text{for each arc } a, \\
&\sum_{A \in \mathcal{A}} y(A) = 1, \quad y \geq 0.
\end{aligned}$$

Here $y(A)$ is the probability of choosing arborescence A that is rooted at r. Also $\mathcal{A}$ is the set of all arborescences rooted at r. We are looking for the minimum value of μ such that there is a convex combination of incidence vectors of arborescences that satisfies the arc capacities $\{(\mu + c(a))/p(a)\}$. In other words, the vector with components $\{(\mu + c(a))/p(a)\}$ should belong to the dominant of the Arborescence polytope. It follows from Theorem 6 that we have to solve

$$\begin{aligned} &\min \mu \\ &\mu\, p'(\delta^+(S)) + q(\delta^+(S)) \geq 1,\ S \subset V,\, r \in S, \end{aligned}$$

where $p'(a) = 1/p(a)$, $q(a) = c(a)/p(a)$. Thus the payoff is

$$\hat{\mu} = \max\ \frac{1 - q(\delta^+(S))}{p'(\delta^+(S))}. \tag{22}$$

Here the maximum is taken among all sets $S \subset V$, with $r \in S$. We use Newton's method, then at each iteration, qiven $\bar{\mu}$ we solve

$$\max\ 1 - q(\delta^+(S)) - \bar{\mu} p'(\delta^+(S))$$

or

$$\min\ \bar{\mu} p'(\delta^+(S)) + q(\delta^+(S)) - 1.$$

This reduces to a minimum cut problem that can be solved with the algorithm of Hao & Orlin [13] that takes $O(nm \log(n^2/m))$ time. Newton's method takes $O(m^2 \log m)$ iterations, the proof of this is similar to the proof of Theorem 8. Thus this part takes $O(nm^3 \log(n^2/m) \log m)$ time.

Let $\hat{S}$ be a solution of (22). The attacker's strategy is given by

$$\bar{x}(a) = \begin{cases} \dfrac{\hat{\mu} p'(a)}{1 - q(\delta^+(\hat{S}))} & \text{if } a \in \delta^+(\hat{S}), \\ 0 & \text{otherwise.} \end{cases}$$

The defender's strategy can be obtained from a maximum packing of arborescences with capacities $\{(\hat{\mu} + c(a))/p(a)\}$. This can be done with the algorithm of [9] that requires $O(n^3 m \log(n^2/m))$ time. Thus the computing time is dominated by the time required by Netwton's method that is $O(nm^3 \log(n^2/m) \log m)$.