Network disconnection games: A game theoretic approach to checkpoint evaluation in networks

Mourad Baïou, Francisco Barahona

NETWORK DISCONNECTION GAMES: A GAME THEORETIC APPROACH TO CHECKPOINT-PLACEMENT IN NETWORKS

MOURAD BAÏOU AND FRANCISCO BARAHONA

ABSTRACT. We study a Network Security question that consists of learning where are the most important locations to place checkpoints to intercept an adversary traveling from an origin to a destination. For that we define a multi-agent cooperative game where every agent controls an arc, and is able to place a checkpoint on it. We assume that the adversary is trying to travel from s to t. If the adversary crosses a checkpoint there is no certainty that he will be detected, therefore the value of a coalition should reflect the number of checkpoints, in the coalition, that the adversary has to cross (the more, the better). This suggests defining the value of a coalition S as the maximum number of disjoint st-cuts included in S; this is a lower bound for the number of checkpoints in S that the adversary has to cross. The adversary can choose any shortest path (minimum cardinality) from s to t, so in any particular coalition one cannot know the exact number of checkpoints that he will cross, and thus we propose to work with a lower bound instead. We give a polynomial combinatorial algorithm for computing the nucleolus of this game. The nucleolus is an allocation of resources to each arc that reflects the contribution of each checkpoint to the common objective of detecting the adversary. Since in reality one might not be able to put a checkpoint on every arc, the larger values in the nucleolus indicate the most important locations to place checkpoints.

1. INTRODUCTION

We study a Network Security issue, namely deciding where are the most important locations to place checkpoints, in a transportation or a communication network, to detect an adversary trying to travel from a node s to a node t. For that we define a multi-agent cooperative game where each agent owns an arc and is able to place a checkpoint on it. If the adversary crosses a checkpoint there is no certainty that he will be detected, and thus the value of a coalition should reflect the number of checkpoints, in the coalition, that the adversary has to cross (the more, the better). This suggests to define the value of a coalition S as the maximum number of disjoint st-cuts included in S; this is a lower bound for the number of checkpoints in the coalition that the adversary has to cross. The adversary can choose any shortest path (minimum cardinality), and thus one cannot know the exact number of checkpoints in a coalition that he will cross. For that reason we have chosen to work with a lower bound instead. In Figure 1 we show an example with two coalitions. The arcs in Coalition 1 are drawn with thick arrows, and the arcs in Coalition 2 are drawn with thick dashed arrows. The circles and ovals represent sets of nodes. Coalition 1 has value 2, notice that arc (c, d) does not belong to it. Coalition 2 has

value 3 because it consists of three bridges. We depict with a dashed line one shortest path from s to t. It contains three arcs in each of these two coalitions. We use a dotted line to represent another shortest path. It contains two arcs from Coalition 1 and three arcs from Coalition 2. The adversary can choose any shortest path, so we can be sure that he will cross *at least* two checkpoints in Coalition 1. Coalition 2 consists of three bridges, here the adversary has to cross *exactly* three checkpoints.

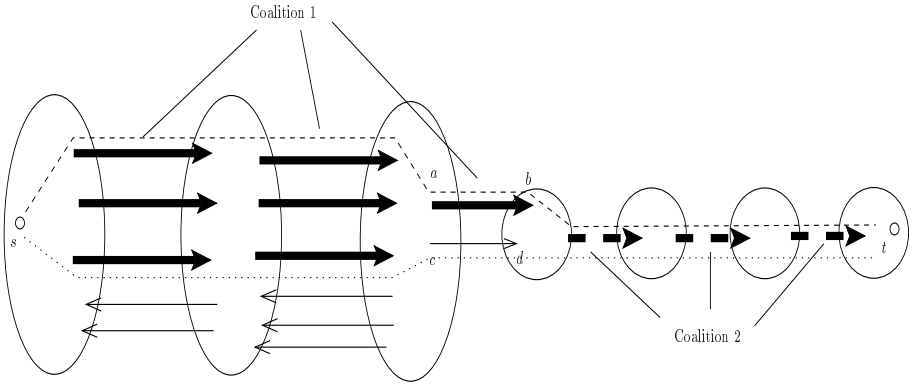

FIGURE 1. An example of two coalitions. Coalition 1 is depicted with thick arrows. Coalition 2 is drawn with thick dashed arrows. The circles and ovals represent sets of nodes. An st-path is depicted with dashed lines. This path uses three arcs in Coalition 1 and three arcs in Coalition 2. A second st-path is shown with a dotted line, it uses two arcs from Coalition 1 and three arcs from Coalition 2.

1.1. **Our contribution.** For this type of cooperative games, computing the nucleolus can be accomplished by solving a sequence of linear programs, each of them involving an exponential number of inequalities. Thus the challenge is to find a way to solve all these linear programs in polynomial time. Our contribution consists of giving a polynomial combinatorial algorithm for computing the nucleolus of this game. The key is to identify a subset of inequalities that has polynomial size, and that is sufficient for the computation of the nucleolus. Then we show how to treat these inequalities with a combinatorial algorithm. The nucleolus is an allocation of resources to the arcs, so that for each arc it reflects its relative contribution to the common objective of detecting the adversary. Since in reality one might not be able to put a checkpoint on every arc, the larger values in the nucleolus show which are the most important locations to place checkpoints. Our development uses techniques of Linear Programming and Network Flows, and thus we expect the reader to be familiar with these two subjects. A reference for Linear Programming is [6], and for Network Flows is [1]. We also expect the reader to be familiar with Cooperative Game Theory, in particular with concepts like the core and the nucleolus.

1.2. **Related work.** Related work on this type of games is the following. This game is dual to a flow game studied in [18], where the value of a coalition $S \subseteq A$ is the maximum number of disjoint st-paths included in S. The core of this flow game has been studied in [18], and the nucleolus has been studied in [9] and [22]. A number of network interdiction models have been studied in the literature where the goal is to identify network elements that, if lost, could result in a significant interruption to supply, services or communication, see for instance [5, 2, 17, 28, 27]. Our case is a security game where the goal is to intercept *every* path from s to t. This type of games has been studied in [4] and [3], and have been called *Path Disruption Games* (PDG). For PDG the value of a coalition is one if it controls a set of arcs whose removal disconnects s and t, and the value is zero otherwise. A polynomial algorithm to compute the ϵ-core was given in [3]. This (zero-one) value function is useful under the assumption that the adversary is detected each time that he crosses a checkpoint, on the other hand, our (different) value function let us deal with the uncertainty of detection. Other references for the study of combinatorial games are [21] and [10]. Other work on the nucleolus of combinatorial games appears in [26], [16], [19], [12]. See also the surveys [7] and [8].

1.3. **Organization.** This paper is organized as follows. In Section 2 we give some notation and show how to compute the value function of the game. In Section 3 we study the core and the nucleolus of the game. In Section 4 we show two small examples. Section 5 contains some final remarks.

2. PRELIMINARIES

In this section we define some notation, and show how to compute the value function of the game.

Let $G = (V, A)$ be a directed graph, for $U \subseteq V$, we denote by $\delta^+(U)$ the set

$$\delta^+(U) = \{(u, v) \in A \,|\, u \in U, v \notin U\},$$

also $\delta^-(U)$=$\delta^+(V \setminus U)$. If there are two distinguished vertices s and t, for a set $U \subset V$, with $s \in U, t \notin U$, the set of arcs $\delta^+(U)$ is called an *st-cut*. We use $\delta^+(u)$ (resp. $\delta^-(u)$) instead of $\delta^+(\{u\})$ (resp. $\delta^-(\{u\})$).

An st-path is a sequence of arcs $(u_1, u_2), (u_2, u_3), \ldots, (u_{k-1}, u_k)$, where $s = u_1$, $u_k = t$, and all nodes $\{u_i\}$ are distinct.

For a set $S \subseteq A$, its incidence vector $x^S \in I\!R^A$ is defined by $x^S(a) = 1$ if the arc $a \in S$, and $x^S(a) = 0$ otherwise.

For a vector $x \in I\!R^A$ and for $S \subseteq A$, we use $x(S)$ to denote $x(S) = \sum_{a \in S} x(a)$. For a graph $G = (V, A)$ we use n to denote $|V|$, and m to denote $|A|$.

2.1. **Computing the value function.** For an arc-set S we need to compute the maximum number of disjoint st-cuts included in S, here we show that a primal-dual version of Dijkstra's algorithm [11], for shortest paths, can be used. We start with a linear programming formulation of the shortest path problem, as follows. Let B be a matrix whose rows are the incidence vectors of all st-cuts. For a weight

function $w : A \to \mathbb{R}_+$ consider the linear program below

$$\min wx, \quad Bx \geq 1, \quad x \geq 0. \tag{1}$$

Here we used 1 to denote a column vector of ones, and 0 to denote a column of zeroes. Its dual is

$$\max y1, \quad yB \leq w, \quad y \geq 0. \tag{2}$$

Here y is a row vector, we used 1 to denote a column of ones, and 0 to denote a row of zeroes. For the primal problem there is a variable $x(u, v)$ for each arc (u, v). The inequalities in (1) are of the form

$$\sum_{(u,v)\in C} x(u, v) \geq 1,$$

for each st-cut C. For the dual problem there is a variable y_C for each st-cut C. The inequalities in (2) are of the form

$$\sum_{C:(u,v)\in C} y_C \leq w(u, v),$$

for each arc (u, v). In other words, this says that for a fixed arc (u, v), the sum of the variables y_C for all st-cuts C containing the arc (u, v), is at most the weight $w(u, v)$. This can be solved with the following primal-dual version of Dijkstra's algorithm [11].

Dijkstra's Algorithm

Step 0. Set $S = \{s\}$; $d(u, v) = w(u, v)$, $x(u, v) = 0$ for all $(u, v) \in A$; $y_C = 0$ for each st-cut C.

Step 1. Let $(k, l) = \operatorname{argmin}\{d(u, v) \mid (u, v) \in \delta^+(S)\}$.

Set

$predecessor(l) = k$,
$y_C = d(k, l)$, with $C = \delta^+(S)$.
$d(u, v) \leftarrow d(u, v) - y_C$, for all $(u, v) \in C = \delta^+(S)$,
$S \leftarrow S \cup \{l\}$.

Step 2. If $l \neq t$ go to Step 1.

If $l = t$, use the predecessors to retrace a path from s to t, set $x(u, v) = 1$, for each arc (u, v) in this path, and stop.

At the end, the set of arcs (u, v) with $x(u, v) = 1$, forms a directed path from s to t, so x satisfies the inequalities in (1). At every iteration the algorithm produces a vector y that satisfies the inequalities in (2).

Now we have to see that the pair (x, y) satisfies the complementary slackness conditions of Linear Programming. For that notice that in the algorithm, each time that a variable $x(u, v)$ is set to the value $x(u, v) = 1$, then we also have

$$\sum_{C:(u,v)\in C} y_C = w(u, v).$$

This sum is over all st-cuts C that contain the arc (u, v). Thus $x(u, v) > 0 \Longrightarrow \sum_{C:(u,v)\in C} y_C = w(u, v)$.

Now we have to see that

$$y_C > 0 \Longrightarrow \sum_{(u,v)\in C} x(u,v) = 1. \tag{3}$$

Let $\{S^0, S^1, \ldots\}$ be the sequence of sets S obtained in Step 1. We have $S^0 = \{s\}$, and $S^{i+1} = S^i \cup \{l_i\}$ for some node l_i, $i = 0, 1, \ldots$ These are the only node sets such that y_{C^i} takes a positive value, with $C^i = \delta^+(S^i)$. Also if $x(u,v) = 1$, then $u \in S^i$ and $v \in S^{i+j}$, with $j \geq 1$. See Figure 2. The arcs (u,v) with $x(u,v) = 1$ form a path from s to t, therefore if $y_{C^i} > 0$ we cannot have more than one arc in C^i whose associated variable takes the value one. Thus condition (3) is satisfied.

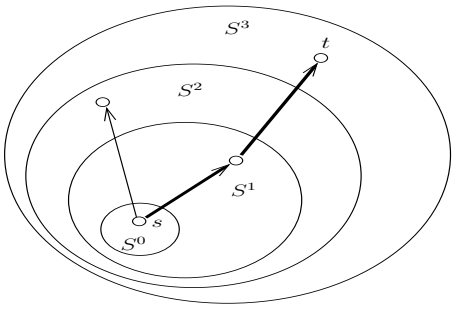

FIGURE 2. Here we show the sets $\{S^i\}$ produced by the algorithm. We depict with thick lines the arcs (u,v) with $x(u,v) = 1$.

Therefore, since x and y satisfy the complementary slackness conditions, we have that x and y are optimal solutions of (1) and (2).

The dual problem (2) gives a *maximum packing of st-cuts* with arc-capacities w. Notice that if the weights w are integer, then the dual vector y produced by this algorithm is also integral. So in particular if w is the vector of all ones, the primal solution gives an st-path of minimum cardinality, and the dual solution gives the maximum number of disjoint st-cuts contained in A. This will be used in Section 3. To compute the value of a coalition $S \subseteq A$, we set $w(u,v) = 1$ for $(u,v) \in S$, and $w(u,v) = 0$ otherwise. Then the algorithm above gives the maximum number of disjoint st-cuts included in S. This algorithm also shows that the extreme points of the polyhedron defined in (1) are the incidence vectors of all st-paths.

2.2. **Maximum Flow.** For a directed graph $G = (V, A)$, with two distinguished vertices s and t, we associate a variable $x(u,v)$ with every arc $(u,v) \in A$. Then

for a capacity function $w : A \to \mathbb{R}$, a *maximum flow* can be found by solving

$$\begin{aligned}
&\max f \\
&\text{subject to} \\
&x(\delta^-(v)) - x(\delta^+(v)) = \begin{cases} f, \text{ if } v = t, \\ -f, \text{ if } v = s, \\ 0, \text{ if } v \neq s, t, \end{cases} \\
&0 \leq x(u,v) \leq w(u,v), \text{ for all } (u,v) \in A.
\end{aligned}$$

If C is an st-cut, its capacity is $w(C)$. A *minimum st-cut* is an st-cut of minimum capacity. The *Max-flow Min-cut Theorem* [13, 1], states that the value of a maximum flow is equal to the capacity of a minimum st-cut.

3. The Network Disconnection Game

Given a directed graph $G = (V, A)$ with two distinguished nodes s and t, we assume that each arc is owned by a different player that is able to place a checkpoint on it, and thus we identify the player set with A. We also assume that there is an adversary trying to travel from s to t, and he will use a shortest (minimum cardinality) st-path. If the adversary crosses a checkpoint there is no certainty of detecting him. To deal with this situation we define a cooperative game $(A, \mathbf{v})$, where the characteristic function $\mathbf{v} : 2^A \to \mathbb{R}_+$, gives for each coalition $S \subseteq A$, the maximum number of disjoint st-cuts included in S. This is a lower bound for the number of checkpoints in the coalition that the adversary will cross. The idea here is to relate the value of a coalition with the number of checkpoints, in the coalition, that the adversary will cross, (the more, the better). Since the adversary could take any shortest path, it is not possible to know in advance how many checkpoints he will cross in the coalition. For this reason we are using a lower bound instead. The value of the grand coalition is the length of a shortest st-path.

3.1. Non-convexity.

We start by showing that this game is not convex. Convex games were introduced in [25] where many nice properties were established. A game is convex if its value function is supermodular, i.e., for any two coalitions S and T,

$$\mathbf{v}(S \cup T) + \mathbf{v}(S \cap T) \geq \mathbf{v}(S) + \mathbf{v}(T). \tag{4}$$

In our case consider a graph $G = (V, A)$, where $V = \{s, a, b, t\}$, and $A = \{(s, a), (s, b), (s, t), (a, t), (b, t)\}$. Let

$$S = \{(s, a), (s, b), (s, t)\} \text{ and } T = \{(a, t), (b, t), (s, t)\}.$$

Both of them are st-cuts, so $\mathbf{v}(S) = \mathbf{v}(T) = 1$. We have $S \cap T = \{(s, t)\}$, so this does not contain an st-cut and $\mathbf{v}(S \cap T) = 0$. The coalition $S \cup T$ does not contain two disjoint st-cuts, then $\mathbf{v}(S \cup T) = 1$. Thus inequality (4) is violated.

3.2. **The core.** The core of a game is a notion introduced in [14] seeking stability, we study some algorithmic questions here. A vector $x : A \to I\!R$ is called an *allocation* if $x(A) = \mathbf{v}(A)$. In our case $x(a)$ represents the amount of resources allocated to player a to satisfy the common objective of detecting the adversary. The definition of the core is based on the following stability condition: No subgroup of players does better if they break away from the joint decision of all players to form their own coalition. Thus the core of the game is the set of allocations satisfying the inequalities below.

$$x(A) = \mathbf{v}(A)$$
$$x(S) \geq \mathbf{v}(S),\ \forall S \subseteq A.$$

First we need a simpler description of the core as follows.

Lemma 1. *Let k be the length of an st-path of minimum cardinality. Then the core is also defined by*

$$x(A) = k, \tag{5}$$
$$x(C) \geq 1, \text{ for each } st\text{-cut } C, \tag{6}$$
$$x \geq 0. \tag{7}$$

Proof. It follows from Dijkstra's algorithm in Subsection 2.1, that the value of a minimum cardinality st-path is equal to the maximum number of disjoint st-cuts included in A, therefore $\mathbf{v}(A) = k$.

Consider now an inequality

$$x(S) \geq \mathbf{v}(S), \tag{8}$$

for $S \subset A$. If $\mathbf{v}(S) = q > 0$, then S contains q disjoint st-cuts $C_1, \ldots, C_q$. Thus (8) can be obtained by adding $x(C_i) \geq 1$, for $i = 1, \ldots, q$, and $x(a) \geq 0$ for $a \in S \setminus (\cup_i C_i)$.

If $\mathbf{v}(S) = 0$, then $x(S) \geq 0$ can be obtained by adding $x(a) \geq 0$, for $a \in S$.

Thus inequalities (8) are implied by (6) and (7). □

Inequalities (6)-(7) correspond to the ones in (1), and thus the extreme points of the core are the incidence vectors of all minimum cardinality st-paths. In other words, any vector in the core is a convex combination of incidence vectors of shortest st-paths.

Now we use Network Flows to characterize the core. Given a directed graph $G = (V, A)$, we associate a variable $x(u, v)$ with every arc $(u, v) \in A$. Then for a function $b : V \to I\!R$, and cost function $c : A \to I\!R$, a *minimum cost flow* can be found by solving

$$\begin{aligned} &\min \sum_{(u,v)\in A} c(u,v)x(u,v) \\ &\text{subject to} \\ &x(\delta^-(v)) - x(\delta^+(v)) = b(v), \text{ for all } v \in V, \\ &x(u,v) \geq 0, \text{ for all } (u,v) \in A. \end{aligned} \tag{9}$$

Equations (9) are called *flow conservation equations*.

When $c(u,v) \geq 0$ for all $(u,v) \in A$, $b(s) = -1, b(t) = 1, b(v) = 0$ for $v \neq s,t$; then a minimum cost flow gives a shortest st-path. Here we are sending one unit of flow from s to t. Thus any convex combination of incidence vectors of shortest st-paths also corresponds to a minimum cost flow from s to t, with arc-costs all equal to one; also the amount of flow is one. We summarize this below.

Theorem 2. *The core is also defined by the system*

$$x(\delta^-(v)) - x(\delta^+(v)) = \begin{cases} -1 & \text{if } v = s, \\ 0 & \text{if } v \neq s,t, \\ 1 & \text{if } v = t, \end{cases} \tag{10}$$

$$x(u,v) \geq 0 \quad \text{for all } (u,v) \in A, \tag{11}$$

$$x(u,v) = 0 \quad \text{for all } (u,v) \in A_0\ . \tag{12}$$

Here A_0 is the set of arcs that do not belong to any shortest st-path.

Theorem 3. *For the Network Disconnection Game, the core is non-empty if and only if there is a path from s to t.*

Theorem 4. *Given a vector $\bar{x}$, we can test whether $\bar{x}$ belongs to the core in polynomial time.*

Proof. To test if $\bar{x}$ satisfies (5)-(7), we have to solve a shortest path problem to test equation (5). Then we solve a minimum cut problem with capacities $\bar{x}$, to check if inequalities (6) are satisfied.

Alternatively we can test if $\bar{x}$ satisfies (10)-(12). For this we need to identify the set A_0, this reduces to a sequence of shortest path problems as follows. Let k be the cardinality of a shortest st-path. Then for each arc (u,v) we compute a shortest path from s to u, and a shortest path from v to t. If the sum of the cardinalities of these two paths is greater than $k-1$, then $(u,v) \in A_0$. □

3.3. **The nucleolus.** For a coalition S and a vector $x \in \mathbb{R}^A$, their *excess* is $e(x,S) = x(S) - \mathbf{v}(S)$. The nucleolus is the allocation that lexicographically maximizes the vector of non-decreasingly ordered excesses, cf. [24]. Roughly speaking, we would like to maximize the resources allocated to each coalition, since this involves multiple (opposing) objectives, the nucleolus is used as a compromise. The basic idea is to maximize the minimum satisfaction.

The nucleolus can be computed with a sequence of linear programs as described below, cf. [20]. Also this procedure gives an alternative (and perhaps easier to understand) definition of the nucleolus.

First we have to solve

$$\begin{aligned} &\max \epsilon \\ &x(S) \geq \mathbf{v}(S) + \epsilon, \quad \forall S \subset A \\ &x(A) = \mathbf{v}(A). \end{aligned}$$

Here the variables are ϵ and x. Let ϵ_1 be the optimal value of this, and $P_1(\epsilon_1)$ be the polytope defined above, with $\epsilon = \epsilon_1$, i.e., $P_1(\epsilon_1)$ is the set of optimal solutions

of the linear program above. For a polytope $P \subset I\!R^A$ let

$$\mathcal{F}(P) = \{S \subseteq A \,|\, x(S) = y(S), \forall x, y \in P\}$$

denote the set of coalitions *fixed* for P. For instance, for $P(\epsilon_1)$, these are the coalitions whose associated inequalities are satisfied with equation for each vector in $P(\epsilon_1)$. In general given ϵ_{r-1} we solve

$$\max \epsilon \tag{13}$$

$$x(S) \geq \mathbf{v}(S) + \epsilon, \forall S \notin \mathcal{F}(P_{r-1}(\epsilon_{r-1})) \tag{14}$$

$$x \in P_{r-1}(\epsilon_{r-1}). \tag{15}$$

We denote by ϵ_r the optimal value of this, and $P_r(\epsilon_r)$ the polytope above with $\epsilon = \epsilon_r$. We continue for $r = 2, ..., |A|$, or until $P_r(\epsilon_r)$ is a singleton. In other words $P_r(\epsilon_r)$ is the set of optimal solutions of a linear program r, then inside this set we solve the next linear program. Notice that each time the dimension of $P_r(\epsilon_r)$ decreases by at least one. To see this notice that after each new linear program, there is at least one inequality $x(S) \geq \mathbf{v}(s) + \epsilon$ that becomes equation. Thus it takes at most $|E|$ steps for $P_r(\epsilon_r)$ to be a singleton.

The following lemma gives a simpler formulation of (13)-(15).

Lemma 5. *Instead of solving* (13)-(15)*, we can solve*

$$\max \epsilon \tag{16}$$

$$x(C) \geq 1 + \epsilon, \textit{ for each st-cut}$$

$$C \notin \mathcal{F}(P_{r-1}(\epsilon_{r-1})), \tag{17}$$

$$x(a) \geq \epsilon \textit{ for each arc } a \notin \mathcal{F}(P_{r-1}(\epsilon_{r-1})), \tag{18}$$

$$x \in P_{r-1}(\epsilon_{r-1}). \tag{19}$$

Proof. Consider $S \subset A$, with $S \notin \mathcal{F}(P_{r-1}(\epsilon_{r-1}))$. First assume that $\mathbf{v}(S) = q > 0$, and let $C_1, \ldots, C_q$ be a set of disjoint st-cuts included in S.

If there is at least one of them, C_1 say, with $C_1 \notin \mathcal{F}(P_{r-1}(\epsilon_{r-1}))$, then $x(S) \geq \mathbf{v}(S) + \epsilon$ can be obtained as the sum of $x(C_1) \geq 1 + \epsilon$, $x(C_i) \geq 1$ for $i = 2, \ldots, q$, and $x(a) \geq 0$ for $a \in S \setminus (\cup_j C_j)$.

If $C_i \in \mathcal{F}(P_{r-1}(\epsilon_{r-1}))$, for all i, then there is an arc $\bar{a} \in S \setminus (\cup_j C_j)$ with $\bar{a} \notin \mathcal{F}(P_{r-1}(\epsilon_{r-1}))$. Then $x(S) \geq \mathbf{v}(S) + \epsilon$ can be obtained as the sum of $x(C_i) \geq 1$ for $i = 1, \ldots, q$, $x(a) \geq 0$ for $a \in S \setminus (\cup_j C_j)$, $a \neq \bar{a}$, and $x(\bar{a}) \geq \epsilon$.

If $\mathbf{v}(S) = 0$, then there is an arc $\bar{a} \in S$ with $\bar{a} \notin \mathcal{F}(P_{r-1}(\epsilon_{r-1}))$, then $x(S) \geq \epsilon$ can be obtained adding $x(\bar{a}) \geq \epsilon$, and $x(a) \geq 0$ for $a \in S$, $a \neq \bar{a}$. □

Now we show that to find ϵ_r it is enough to work with constraints $x(a) \geq \epsilon$, for $a \notin \mathcal{F}(P_{r-1}(\epsilon_{r-1}))$, then the constraints $x(\delta^+(U)) \geq 1 + \epsilon$, for $\delta^+(U) \notin \mathcal{F}(P_{r-1}(\epsilon_{r-1}))$, will be automatically satisfied. We treat this in the lemma below. Notice that for $a \in \mathcal{F}(P_{r-1}(\epsilon_{r-1}))$ we denote by $l(a)$ the fixed value that $x(a)$ should take.

Lemma 6. *Assume that x is in the core, $x(a) \geq \epsilon$ for each $a \notin \mathcal{F}(P_{r-1}(\epsilon_{r-1}))$, $x(a) = l(a)$ for $a \in \mathcal{F}(P_{r-1}(\epsilon_{r-1}))$. Then $x(\delta^+(U)) \geq 1 + \epsilon$ for $\delta^+(U) \notin \mathcal{F}(P_{r-1}(\epsilon_{r-1}))$.*

Proof. Consider $U \subset V$, $s \in U$, $t \notin U$. The system (10)-(12) implies $x(\delta^+(U)) - x(\delta^-(U)) = 1$, or $x(\delta^+(U)) = 1 + x(\delta^-(U))$. We have two cases:

- If $a \in \mathcal{F}(P_{r-1}(\epsilon_{r-1}))$ for all $a \in \delta^-(U)$ then $\delta^+(U) \in \mathcal{F}(P_{r-1}(\epsilon_{r-1}))$.
- If there is an arc $\bar{a} \in \delta^-(U)$ with $\bar{a} \notin \mathcal{F}(P_{r-1}(\epsilon_{r-1}))$, then $x(\bar{a}) \geq \epsilon$. Therefore $x(\delta^+(U)) = 1 + x(\delta^-(U)) \geq 1 + \epsilon$. □

The lemma above is key, because it shows that an exponential number of inequalities can be ignored, and only simple inequalities associated with the arcs are needed. Thus to compute the nucleolus, at each step we have to look for the maximum value of λ such that the system below has a solution.

$$x(\delta^-(v)) - x(\delta^+(v)) = \begin{cases} -1 & \text{if } v = s, \\ 0 & \text{if } v \neq s, t, \\ 1 & \text{if } v = t, \end{cases} \tag{20}$$

$$x(u,v) = l(u,v),\ \forall (u,v) \in \mathcal{F}(P_{r-1}(\epsilon_{r-1})), \tag{21}$$

$$x(u,v) \geq l(u,v) + \lambda,$$

$$\forall (u,v) \in A_r = A \setminus \mathcal{F}(P_{r-1}(\epsilon_{r-1})), \tag{22}$$

$$\lambda \geq 0. \tag{23}$$

Here we assume that for $(u,v) \in \mathcal{F}(P_{r-1}(\epsilon_{r-1}))$, $x(u,v) = l(u,v)$. And for $(u,v) \in A_r$, $l(u,v) = \epsilon_{r-1}$.

This could be done with linear programming techniques, however now we show that it can be done in a combinatorial way, namely it reduces to a sequence of minimum cut problems. We define

$$x'(u,v) = x(u,v) - l(u,v) - \lambda, \text{ for } (u,v) \in A_r.$$

Then we have to find the largest value of λ such that the system below has a solution.

$$x'(\delta^-(v)) - x'(\delta^+(v)) = b(v) + \lambda d(v), \text{ for each } v \in V, \tag{24}$$

$$x' \geq 0. \tag{25}$$

Here the arc-set is A_r, and $b(v)$ is the right hand side of (20) minus the sum of the lower bounds $l(u,v)$ for the arcs entering v, plus the sum of the lower bounds $l(v,u)$ for the arcs leaving v. Then $d(v)$ is the number of arcs in A_r leaving v, minus the number of arcs in A_r entering v. Their formal definition is as follows. First we set $b'(s) = -1$, $b'(t) = 1$, and $b'(v) = 0$ if $v \neq s, t$. Then for each $v \in V$,

$$b(v) = b'(v) - \sum_{(u,v) \in A} l(u,v) + \sum_{(v,u) \in A} l(v,u),$$

and

$$d(v) = |\{(v,u) \in A_r\}| - |\{(u,v) \in A_r\}|.$$

3.4. **Parametric Flows.** It only remains to show how to find the maximum value of λ so that (24)-(25) has a solution. For that we use parametric flows as below. First we need the following lemma.

Lemma 7. *Either the system below has a solution,*

$$x(\delta^-(v)) - x(\delta^+(v)) = b(v), \text{ for all } v \in V,$$
$$x(u,v) \geq 0 \quad \text{for all } (u,v) \in A,$$

or there is a set $U \subset V$ with $b(U) < 0$ and $\delta^+(U) = \emptyset$.

Proof. Here we have $b : V \to I\!R$, and we assume that $\sum_v b(v) = 0$. Let $b^+(v) = \max\{b(v), 0\}$, and $b^-(v) = \max\{-b(v), 0\}$, for each $v \in V$. To test if the system above has a solution, we add two nodes s' and t'. Then for each node v with $b(v) < 0$, we add an arc (s', v) with capacity $b^-(v)$, and for each node v with $b(v) > 0$ we add an arc (v, t') with capacity $b^+(v)$. We give infinite capacity to all original arcs. Then we look for a maximum flow from s' to t'.

Let $\alpha = \sum_v b^+(v) = \sum_v b^-(v)$. If the flow value is α, then there is a solution, otherwise there is a minimum cut whose capacity is less than α. Now we discuss the structure of this cut. The cut is $\delta^+(U \cup \{s\})$, where $U \subset V$. Its capacity is

$$b^+(U) + b^-(V \setminus U) = b^+(U) + \alpha - b^-(U) =$$
$$= \alpha + b(U) < \alpha.$$

Thus $b(U) < 0$, and since we obtained a minimum cut, we have a set $U \subset V$ with $\delta^+(U) = \emptyset$, (this is because the original arcs have infinite capacity), and such that $b(U)$ is minimum. A maximum flow and a minimum cut can be found in $O(nm \log(n^2/m))$ time, see [15] and [1]. □

Now consider (24)-(25). If the system is infeasible for some value $\bar{\lambda} > 0$, the lemma above shows that there is a set $U \subset V$ with $\delta^+(U) = \emptyset$, $b(U) + \bar{\lambda} d(U) < 0$. Since the system is feasible for $\lambda = 0$, we should have $b(U) \geq 0$, and $d(U) < 0$. To have feasibility, we should impose $b(U) + \lambda d(U) \geq 0$, or $\lambda \leq b(U)/(-d(U))$. Thus

$$\lambda = \min \frac{b(U)}{-d(U)}, \tag{26}$$

where the minimum is taken among all sets $U \subset V$, with $\delta^+(U) = \emptyset$ and $d(U) < 0$. This minimum can be found with Newton's algorithm for minimum ratio problems as below, see [23].

Newton's method

Step 0. Set $\lambda = \lambda_M$.

Step 1. Find $\bar{U} = \text{argmin}\{b(U) + \lambda d(U)\}$, with $U \subset V$, $\delta^+(U) = \emptyset$ and $d(U) < 0$.

Step 2. If $b(\bar{U}) + \lambda d(\bar{U}) < 0$, then update λ as

$$\lambda = \frac{b(\bar{U})}{-d(\bar{U})}$$

and go to Step 1.
Otherwise $b(\bar{U}) + \lambda d(\bar{U}) = 0$, and we stop.

Here λ_M is an upper bound for the value of λ, we can use $\lambda_M = 2$, for instance. The set $\bar{U}$ in Step 1 can be found using a minimum cut algorithm, as seen in Lemma 7.

Let $\tilde{U}$ be a node-set obtained in Step 1 for some value of λ, and $\tilde{\lambda} = b(\tilde{U})/(-d(\tilde{U}))$. Let U' be the node-set obtained in the next iteration. We could have $U' = \tilde{U}$, therefore $b(U') + \tilde{\lambda}d(U') \leq 0$. If $b(U') + \tilde{\lambda}d(U') < 0$, we have $b(U')/(-d(U')) < b(\tilde{U})/(-d(\tilde{U}))$, and we have found improvement. Otherwise $b(U') + \tilde{\lambda}d(U') = 0$ and the set $\tilde{U}$ gives the minimum ratio. Since this is done over a finite number of sets, we have that this method converges in a finite number of steps. Now we use the lemma below to derive a better bound for the number of iterations of Newton's method.

Lemma 8. *Let $\tilde{U}$ be a node-set obtained in Step 1 for some value of λ, and $\tilde{\lambda} = b(\tilde{U})/(-d(\tilde{U}))$. If U' is the node-set obtained in the next iteration and $b(U') + \tilde{\lambda}d(U') < 0$, then $|d(U')| < |d(\tilde{U})|$.*

Proof.

$$\begin{aligned} 0 &> b(U') + \tilde{\lambda}d(U') \\ &= b(U') + \lambda d(U') - \lambda d(U') + \tilde{\lambda}d(U') \\ &\geq b(\tilde{U}) + \lambda d(\tilde{U}) - \lambda d(U') + \tilde{\lambda}d(U') \\ &= (d(\tilde{U}) - d(U'))(\lambda - \tilde{\lambda}). \end{aligned}$$

Thus $|d(U')| < |d(\tilde{U})|$. □

This lemma shows that if $U_1, \ldots, U_k$ is the sequence of sets produced by Newton's method, then $|d(U_i)| > |d(U_{i+1})|$, for $i = 1, \ldots, k-1$. Since $|d(U)| \leq m$, for each set $U \subset V$, and d is integer valued, then Newton's method takes at most m iterations; recall that $m = |A|$.

Each time that we find the value of λ in (20)-(23), we obtain a new set of arcs such that their associated variables should remain fixed. Thus it takes at most m times until all variables are fixed. Now we can give a high level description of the algorithm that computes the nucleolus.

Step 0. Build the system (10)-(12). Set $l(u, v) = 0$ for all arcs (u, v). Label as "fixed" all arcs in A_0, and the other arcs as "free." Set $\epsilon_0 = 0$, $r = 1$.

Step 1. Find the maximum value of λ so that the system (20)-(23) has a solution. This done as in Subsection 3.4. Add λ to $l(u, v)$ for all arcs (u, v) that are free. Set $\epsilon_r = \epsilon_{r-1} + \lambda$. The algorithm in Subsection 3.4 also gives a node-set U so that all arcs entering U should become "fixed."

Step 2. If there are free arcs remaining, increase r by one and go to Step 1. Otherwise stop, the system (20)-(23) has a unique solution that is the nucleolus.

This leads to our main result below.

Theorem 9. *Computing the nucleolus of the Network Disconnection Game requires $O(m^3 n \log(n^2/m))$ time.*

Proof. The algorithm above takes at most m iterations. Thus Newton's method has to be applied at most m times, and each time requires at most m min-cut problems. Therefore we have $O(m^2)$ min-cut problems. Since each of them requires $O(nm\log(n^2/m))$ time we obtain the bound. □

4. Two small examples

Now we show a simple example based on the graph in Figure 3 (a). We do not go into the full details of the algorithm, but we just show the most significant steps. We start by building system (10)-(12), and we have $A_0 = \{(s,h),(h,i),(i,c)\}$. We fix to zero the variables corresponding to all arcs in A_0. To compute ϵ_1 we set a lower bound λ for the flow in every arc, and look for the maximum value such that there is a flow. Consider the st-cut depicted in Figure 3 (b), since the sum of the flows across this cut is one, the value of λ cannot be greater than $1/3$. On the other hand in Figure 3 (d) we have a flow that satisfies these lower bounds, so $1/3$ is the right value. With $1/3$ as lower bound, the flow in the arcs across this cut should remain fixed to this value. Then the flow equations imply the flow values of all arcs to the right of node c. The flow values are shown in Figure 3 (c). We call all these arcs "fixed," and $\epsilon_1 = 1/3$. For the arcs that remain free, we try again to increase their lower bound. These are the arcs to the left of node c. Consider the cut depicted in Figure 3 (d), since the sum of the flows across this cut is one, we have that the new value cannot be greater than $1/2$. Once we set the lower bounds across this cut to $1/2$ we obtain the flow shown Figure 3 (e). Now the flow in every arc is fixed, so we have found the nucleolus.

Now let us look at the values of this vector. If we look at all possible shortest st-paths, the arcs (c,e) and (f,t) appear in $2/3$ of them, this is reflected in the values in the nucleolus. Each of the arcs (s,a) and (s,b) belongs to $1/2$ of all possible st-paths, this is consistent with the values in the nucleolus. Similarly we have that each arc with value $1/3$ appears in $1/3$ of all shortest st-paths. Here the arcs (c,e) and (f,t) are the most important to place checkpoints, followed by $(s,a),(a,c),(s,b)$ and (b,c). Therefore if the budget only allows to install three checkpoints, for instance, we should choose (c,e), (f,t) and one of the arcs whose associated value is $1/2$.

A more extreme example is in Figure 4. The values near the arcs describe the nucleolus. Arc (a,t) gets the value 1, because it appears in every st path. On the other hand every other arc gets the value $1/4$ because each of them appears in $1/4$ of all st-paths. This shows their individual contribution to detecting the adversary.

5. Final Remarks

Now we summarize our procedure. We start by assuming that there is a checkpoint on every arc, and compute the nucleolus of the game. The components of the nucleolus reflect the contribution of each arc to the common objective of detecting the adversary. Thus the larger values indicate the most important locations. Since in reality one might not be able to put a checkpoint on every arc, we propose to concentrate on the arcs that have the largest values.

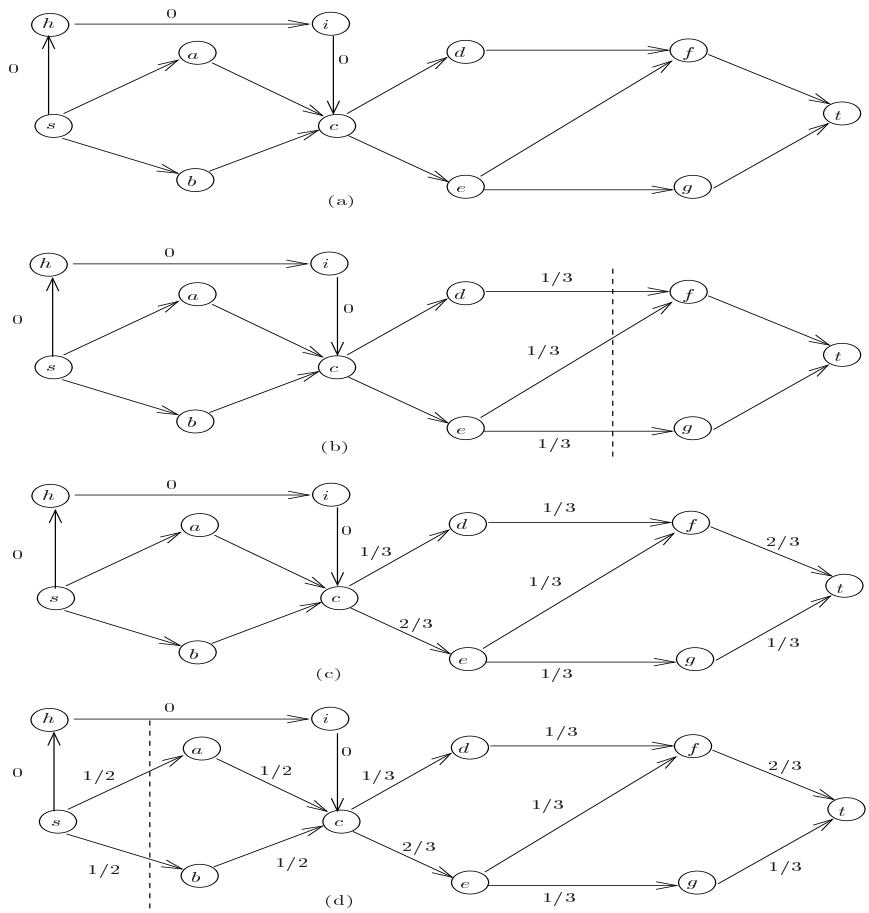

FIGURE 3. A small example. The values in the nucleolus appear in the last picture.

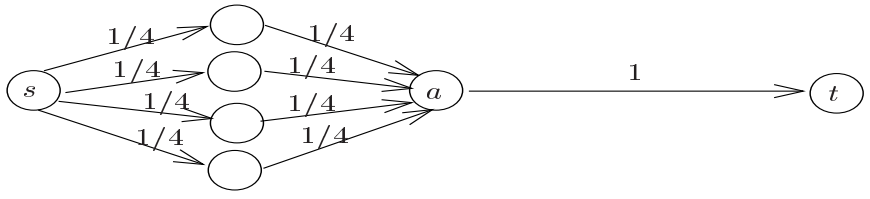

FIGURE 4. A second example. The values in the nucleolus appear in the picture.

We have used network flow techniques to give a combinatorial polynomial algorithm for computing the nucleolus of the Network Disconnection Game. The basic tools are algorithms for shortest paths and for minimum cuts.

Consider now the case when there are k adversaries, each of them trying to travel from s_i to t_i, for $i = 1, \ldots, k$. We propose to compute the nucleolus for each value of i, $i = 1, \ldots, k$, and then add all these vectors. The components of this final vector should be used to decide where to put the checkpoints.

ACKNOWLEDGEMENTS

We are grateful to an anonymous referee whose comments helped us to improve the presentation.

REFERENCES

[1] R. K. AHUJA, T. L. MAGNANTI, AND J. B. ORLIN, *Network flows: Theory, algorithms, and applications*, 1993.

[2] D. S. ALTNER, Ö. ERGUN, AND N. A. UHAN, *The maximum flow network interdiction problem: valid inequalities, integrality gaps, and approximability*, Operations Research Letters, 38 (2010), pp. 33–38.

[3] H. AZIZ AND T. B. SØRENSEN, *Path coalitional games*, arXiv preprint arXiv:1103.3310, (2011).

[4] Y. BACHRACH AND E. PORAT, *Path disruption games*, in Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1, International Foundation for Autonomous Agents and Multiagent Systems, 2010, pp. 1123–1130.

[5] R. L. CHURCH, M. P. SCAPARRA, AND R. S. MIDDLETON, *Identifying critical infrastructure: the median and covering facility interdiction problems*, Annals of the Association of American Geographers, 94 (2004), pp. 491–502.

[6] V. CHVATAL, *Linear programming*, Macmillan, 1983.

[7] I. CURIEL, *Cooperative combinatorial games*, in Pareto optimality, game theory and equilibria, Springer, 2008, pp. 131–157.

[8] X. DENG AND Q. FANG, *Algorithmic cooperative game theory*, in Pareto Optimality, Game Theory And Equilibria, Springer, 2008, pp. 159–185.

[9] X. DENG, Q. FANG, AND X. SUN, *Finding nucleolus of flow game*, Journal of combinatorial optimization, 18 (2009), pp. 64–86.

[10] X. DENG AND C. H. PAPADIMITRIOU, *On the complexity of cooperative solution concepts*, Mathematics of Operations Research, 19 (1994), pp. 257–266.

[11] E. W. DIJKSTRA, *A note on two problems in connexion with graphs*, Numerische mathematik, 1 (1959), pp. 269–271.

[12] U. FAIGLE, W. KERN, AND J. KUIPERS, *Note computing the nucleolus of min-cost spanning tree games is np-hard*, International Journal of Game Theory, 27 (1998), pp. 443–450.

[13] L. R. FORD AND D. R. FULKERSON, *Maximal flow through a network*, in Classic papers in combinatorics, Springer, 2009, pp. 243–248.

[14] D. B. GILLIES, *Solutions to general non-zero-sum games*, Contributions to the Theory of Games, 4 (1959), pp. 47–85.

[15] A. V. GOLDBERG AND R. E. TARJAN, *A new approach to the maximum-flow problem*, Journal of the ACM (JACM), 35 (1988), pp. 921–940.

[16] D. GRANOT, M. MASCHLER, G. OWEN, AND W. R. ZHU, *The kernel/nucleolus of a standard tree game*, International Journal of Game Theory, 25 (1996), pp. 219–244.

[17] E. ISRAELI AND R. K. WOOD, *Shortest-path network interdiction*, Networks, 40 (2002), pp. 97–111.

[18] E. KALAI AND E. ZEMEL, *Totally balanced games and games of flow*, Mathematics of Operations Research, 7 (1982), pp. 476–478.

[19] W. KERN AND D. PAULUSMA, *Matching games: the least core and the nucleolus*, Mathematics of Operations Research, 28 (2003), pp. 294–308.

[20] A. KOPELOWITZ, *Computation of the kernels of simple games and the nucleolus of n-person games.*, tech. rep., DTIC Document, 1967.
[21] N. MEGIDDO, *Computational complexity of the game theory approach to cost allocation for a tree*, Mathematics of Operations Research, 3 (1978), pp. 189–196.
[22] J. POTTERS, H. REIJNIERSE, AND A. BISWAS, *The nucleolus of balanced simple flow networks*, Games and Economic Behavior, 54 (2006), pp. 205–225.
[23] T. RADZIK, *Fractional combinatorial optimization*, in Handbook of combinatorial optimization, Springer, 1999, pp. 429–478.
[24] D. SCHMEIDLER, *The nucleolus of a characteristic function game*, SIAM Journal on applied mathematics, 17 (1969), pp. 1163–1170.
[25] L. S. SHAPLEY, *Cores of convex games*, International Journal of Game Theory, 1 (1971), pp. 11–26.
[26] T. SOLYMOSI AND T. E. RAGHAVAN, *An algorithm for finding the nucleolus of assignment games*, International Journal of Game Theory, 23 (1994), pp. 119–143.
[27] J. TSAI, Z. YIN, J.-Y. KWAK, D. KEMPE, C. KIEKINTVELD, AND M. TAMBE, *Urban security: Game-theoretic resource allocation in networked physical domains*, in National Conference on Artificial Intelligence (AAAI), 2010.
[28] R. K. WOOD, *Deterministic network interdiction*, Mathematical and Computer Modelling, 17 (1993), pp. 1–18.

(M. Baïou) CNRS, AND UNIVERSITÉ CLERMONT II, CAMPUS DES CÉZEAUX BP 125, 63173 AUBIÈRE CEDEX, FRANCE.

E-mail address, M. Baïou: baiou@isima.fr

(F. Barahona) IBM T. J. WATSON RESEARCH CENTER, YORKTOWN HEIGHTS, NY 10589, USA.

E-mail address, F. Barahona: barahon@us.ibm.com