Path Disruption Games

Yoram Bachrach
Microsoft Research
Cambridge UK
yobach@microsoft.com

Ely Porat
Department of Computer Science
Bar-Ilan University, Ramat-Gan, Israel
porately@cs.biu.ac.il

ABSTRACT

We propose Path Disruption Games (PDGs), which consider collaboration between agents attempting stop an adversary from travelling from a source node to a target node in a graph. PDGs can model physical or network security domains. The coalition attempts to stop the adversary by placing checkpoints in intermediate nodes in the graph, to make sure the adversary cannot travel through them. Thus, the coalition wins if it controls a node subset whose removal from the graph disconnects the source and target. We analyze this domain from a cooperative game theoretic perspective, and consider how the gains can be distributed between the agents controlling the vertices. We also consider power indices, which express the influence of each checkpoint location on the outcome of the game, and can be used to identify the most critical locations where checkpoints should be placed. We consider both general graphs and the restricted case of trees, and consider both a model with no cost for placing a checkpoint and a model with where each vertex has its own cost for placing a checkpoint.

Categories and Subject Descriptors

F.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity;
I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent Systems*;
J.4 [**Computer Applications**]: Social and Behavioral Sciences—*Economics*

General Terms

Algorithms, Theory, Economics

Keywords

Computational complexity, Vertex Cut, Power indices, Cooperative Game Theory

1. INTRODUCTION

One important facet of multi-agent systems that has been well studied in the past is collaboration. Cooperative game theory considers cooperation among self interested entities, and has been applied to analyze many such domains. Two important application domains for multi-agent systems are physical security and network security. Game theory has already been used to analyze security issues in several domains. One example is [1], which examines the optimal robot patrolling policy. Another example is [19, 18], which attempts improving security at major locations of economic or political importance by selecting the optimal security policy against a sophisticated adversary.

Consider a transportation network and an adversary who is based at a source vertex and wishes to travel to a target vertex. Consider a set of agents, each controlling a certain vertex. Each agent is capable of placing a checkpoint at her controlled vertex, making sure the adversary cannot pass there. Although a single checkpoint is unlikely to stop the adversary from traveling to the target, the agents might be able to place several checkpoints such that any path connecting the source to the target must pass through at least one checkpoint. In some domains, an agent can place a checkpoint on her vertex for free, while in other domains it might be costly. A similar situation can occur in a communication network, where agents attempt to prevent the adversary from sending information from the source to the target. We model this as Path Disruption Games (PDGs).

Given a PDG, several key questions arise. The coalition obtains the reward as a whole. How should this reward be distributed among the coalition members? What is the optimal policy for a coalition of agents? Which of the placed checkpoints is most critical to achieve the task, and which has no influence on the outcome? We analyze PDGs from a game theoretic perspective to answer these questions. Specifically, we consider the core and least-core and the Banzhaf power index. We consider both a model where placing checkpoints is free and a model where there may be a different price for placing different checkpoints. In the no cost model we show computing power indices is generally hard, but tractable for trees. In the model with costs, we show that even finding the optimal checkpoint configuration is hard, but some problems are tractable for trees.

2. PRELIMINARIES

A coalitional game is composed of a set of n agents, $I = (1, \dots, n)$, and a function mapping any subset (coalition) of the agents to a real value $v : 2^I \to \mathbb{R}$. The function v is called the *characteristic function* (or, in some works, *coalitional function*) of the game. In *simple coalitional games*, v only gets values of 0 or 1, so $v : 2^I \to \{0, 1\}$. We say a coalition $C \subseteq I$ *wins* if $v(C) = 1$, and *loses* if $v(C) = 0$. We say

Cite as: Path Disruption Games, Yoram Bachrach, Ely Porat, *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, pp. 1123-1130

an agent i is *critical* in a winning coalition C if the agent's removal from that coalition would makes it coalition lose: $v(C) = 1$ but $v(C \setminus \{i\}) = 0$. We say agent i is a veto agent if no coalition wins without her, so if $v(C) = 1$ then $i \in C$.

In non-simple general coalitional games, the characteristic function may have any value in $\mathbb{R}^+$, which denotes the total gains of a coalition. This function only defines the total utility a *coalition* achieves, but does not define how these gains should be distributed among the agents. An imputation $(p_1, \dots, p_n)$ is a division of the gains of the grand coalition I among the agents, where $p_i \in \mathbb{R}$, and $\sum_{i=1}^{n} p_i = v(I)$. We call p_i the payoff of agent i, and denote the payoff of a coalition C as $p(C) = \sum_{i \in C} p_i$. Obviously, rational agents attempt to maximize their own share of the utility. Cooperative game theory offers several solution concepts, determining which imputations are likely to occur.

A basic requirement is *individual-rationality*: for all agents $i \in C$, we have $p_i \geq v(\{i\})$. If individual rationality does not hold for an agent, she is better off on its own. Similarly, we say a coalition B *blocks* the payoff vector $(p_1, \dots, p_n)$ if $p(B) < v(B)$, as the members of B can drop off from the main coalition and gain more utility without the rest of the agents. Thus, if a blocked imputation is chosen, the coalition is unstable. A solution concept that emphasizes such *stability* is the core [12].

Definition 1. *The core of a coalitional game is the set of all imputations $(p_1, \dots, p_n)$ that are not blocked by any coalition, so that for any coalition C, the following holds: $p(C) \geq v(C)$.*

In some games the core is empty, so we must relax the constrains, based on the assumption that a coalition would only deviate if it gains significantly. The ϵ-core is a solution concept based on relaxing the inequalities of Definition 1.

Definition 2. *The ϵ-core is the set of all imputations $(p_1, \dots, p_n)$ where for any coalition $C \subseteq I$, $p(C) \geq v(C) - \epsilon$.*

We call the difference $v(C) - p(C)$ between a coalition C's value and the payoff the *deficit* of the coalition. Under an imputation in the ϵ-core, the deficit of any coalition is at most ϵ. If ϵ is large enough, the ϵ-core is guaranteed not to be empty. Achieving more stability obviously requires finding the smallest value of ϵ such that the ϵ-core is not empty. This solution concept is called the *least core*. Given a game G we consider $\{\epsilon | \text{The } \epsilon\text{-core of G is not empty}\}$. It is easy to see that this set is compact, and thus has a minimal element ϵ_{min}.

Definition 3. *The least-core of the game G is the ϵ_{min}-core of G.*

Imputations in the least-core distribute the gains in a way that minimizes the incentive of a coalition to break away (minimizing the maximal deficit). Under such an imputation, no coalition can gain more than ϵ_{min} by deviating, while for any $\epsilon' < \epsilon_{min}$ it is impossible to distribute the gains so no coalition has a deficit higher than ϵ'.

The definitions of the core and least-core allow us to find stable payoff distributions in PDGs. Another question we address in this paper is measuring the influence of a given checkpoint on the capability to prevent the adversary from traveling between the source and target vertices. A common interpretation for this is the probability that this checkpoint would significantly affect the outcome of the game.

One method of measuring the influence of an agent in cooperative settings is through power indices. One commonly used power index is the Banzhaf power index [6]. It has been widely used for measuring political power in weighted voting systems, but it can be applied to any simple cooperative game. This power index depends on the number of coalitions in which an agent is critical. The Banzhaf index of agent i is the proportion of all winning coalitions where i is critical, out of all coalitions that contain i.

Definition 4. *The Banzhaf index is given by $\beta(v) = (\beta_1(v), \dots, \beta_n(v))$ where*

$$\beta_i(v) = \frac{1}{2^{n-1}} \sum_{S \subseteq N | i \in S} [v(S) - v(S \setminus \{i\})].$$

Consider a situation where the agents are independent in their choices, so every coalition has an equal probability of occurring. Under this setting, the Banzhaf index measures the probability that a certain agent is critical in the occurring coalition. Other power indices reflect other assumptions. For example, the Shapley-Shubik index considers a model where agents are added into a coalition in an order chosen uniformly at random.[1]

Our hardness result for calculating the Banzhaf power index in PDG considers the class #P. The complexity classes #P and #P-complete were introduced by Valiant [23], and are used to examine problems where "counting the number of solutions" is hard.[2]

3. PATH DISRUPTION GAMES

We analyze a domain where a coalition attempts to disrupt connectivity between two vertices, by disallowing the use of certain intermediary vertices. We consider an adversary who attempts to travel between a source and a target in a graph, and a coalition whose goal is to prevent the adversary from doing so. The coalition may place checkpoints on the intermediary vertices to make sure the adversary cannot travel through them, thus eliminating them from the graph. By eliminating vertices, the coalition can disconnect the source and target. Obviously, there may be several possible paths from the source to the target, so the coalition must disrupt all of them to achieve its goal.

Our question is, given our desire to stop the adversary, where should the checkpoints be placed? Given limited security resources, how should they be allocated? On which vertices should we concentrate to make sure the adversary fails? Also, given a certain reward for stooping the adversary, how should this reward be distributed among the agents?

[1]Although the Banzhaf and Shapley-Shubik indices are similar, the Banzhaf index considers all possible *subsets* of agents, whereas the Shapley-Shubik index is defined over all *permutations* of the agents

[2]Informally, NP-hardness describes problems where checking for the *existence* of a solution to a combinatorial problem, whereas #P-hardness describes problems where calculating the *number* of solutions to a combinatorial problem is hard. Obviously, it is harder to count the number of solutions than to find out if at least one solution exists. Thus, #P-complete problems are at least as hard (but possibly harder) than NP-complete problems.

We model the above problem as a cooperative game, called the *Path Disruption Game*. The game is constructed such that the Banzhaf power index in it would provide a good measure of the criticality of each possible vertex checkpoint. We then extend the model to directly model different costs for setting up checkpoints in various locations.

A Path Disruption Game is played over a graph $G = \langle V, E \rangle$ with a source $s \in V$ and target $t \in V$ vertices. In PDGs, each agent controls one of the vertices. A coalition wins if the removal of the vertices owned by its agents disconnects s from t. Consider a set of n agents $I = (1, \ldots, n)$, so that agent i controls vertex $v_i \in V$. Given a coalition $C \subseteq I$ we denote the vertices that C controls as $V(C) = \cup_{i \in C} v_i \subseteq V$. We say $V(C)$ is a vertex $s - t$ cut in G if every path from s to t passes through some $v \in V(C)$.

Definition 5. *A Path Disruption Game (PDG) is a simple coalitional game, where the value of a coalition $C \subseteq I$ is defined as follows:*

$$v(C) = \begin{cases} 1 & \text{if } V(C) \text{ is a vertex } s - t \text{ cut in } G \\ 0 & \text{otherwise} \end{cases}$$

Definition 5 has no *costs* associated with placing a checkpoint. An alternative definition directly takes these costs into account. We associate with each vertex in $v_i \in V$ a cost c_i for placing a checkpoint at that vertex. A coalition C can place a checkpoint at any vertex controlled by a coalition member. Our model assumes there is a reward $r \in \mathbb{R}$ associated for disconnecting s and t. Suppose the coalition decides to place checkpoints at vertices $B \subseteq V(C)$. The cost of this is $\sum_{i|v_i \in B} c_i$. For any $B \subset I$ we denote $w(B) = 1$ if B disconnects s and t, and $w(B) = 0$ otherwise. The optimal checkpoint placement for coalition C has cost $m(C) = min\{c(B)|w(B) = 1\}$ if $w(C) = 1$ (i.e. if C can disconnect s and t), and otherwise we define $m(C) = \infty$.

Definition 6. *A Path Disruption Game with Costs (PDGC) is the game where the value of a coalition $C \subseteq I$ is:*

$$v(C) = \begin{cases} r \text{ - } m(C) & \text{If } m(C) < \infty \\ 0 & \text{otherwise} \end{cases}$$

4. THE NO COSTS MODEL

We first consider the model where a checkpoint may be placed at any vertex controlled by the coalition at no cost. We first consider computing the core in PDGs.

4.1 The Core of PDGs

Consider a PDG where a coalition who succeeds in preventing the adversary from arriving at the destination obtains a certain reward for doing so. This reward is given to the coalition members as a group, and the agents who formed the coalition must then decide how it should be divided among the coalition members. Even if a successful coalition is formed, it may not be stable, as agents who only obtain a small fraction of the reward may try to form a different coalition, in order to increase their share of the reward. The most prominent game theoretic solution concept that focuses on such stability is the core, which can be used to allocate the gains. We now consider the problem of computing the core of PDGs. If the core is non-empty, it contains stable payoff distributions (stable imputations). When the core is empty, any distribution would be somewhat unstable, and the least-core must be used to maximize stability. The core may contain an infinite number of imputations, and there does not always exists a concise representation of the core, but one does exist for PDGs.

We first note that PDGs are *simple* coalitional games. The core is very restrictive in simple coalitional games. It is well known that in simple monotone games, the core is closely related to veto agents. The following is a known folklore lemma, proven in [17].

Lemma 1. *The core is non-empty if and only if there is at least one veto player in the game, and if there are veto agents, any imputation that distributes all the gains only to the veto agents (in any way) is in the core*

We now consider computing the core in PDGs. We first show PDGs are monotone.

Lemma 2 (PDGs are monotone). *For all coalitions $A, B \subseteq I$ in a CG we have $v(A \cup B) \geq v(A)$. Alternatively, this can be stated in the following way. Let $W \subseteq I$ be a winning coalition in a PDG, so $v(W) = 1$, and let $C \subseteq I$ be any coalition in that game. Then $W \cup C$ is also a winning coalition, so $v(W \cup C) = 1$.*

Proof. If W is a $s - t$ vertex cut, so the removal of the vertices in W disconnects the s from the t, than removing more vetrices also results in having no path from s to t, so for any C, $W \cup C$ is also an $s - t$ vertex cut. Thus if $v(W) = 1$ than for any C we have $v(C \cup W) = 1$ □

We denote the set of all the agents except i as $I_{-i} = I \backslash \{i\}$. Let G be the PDG graph, and denote by G_{-i} the induced graph when vertex v_i (owned by agent i) is dropped. The graph $G_{-i} = \langle V_{-i}, E_{-i} \rangle$ where $V_{-i} = V \setminus \{v_i\}$ and $E_{-i} = \{(u, v) \in E | u \neq v_i \wedge v \neq v_i\}$. We provide a polynomial algorithm for testing if a player is a veto agent in a PDG.

Lemma 3. *Testing for veto agents in PDGs is in P.*

Proof. First note that I_{-i} is a losing coalition if and only if i is a veto agent. If I_{-i} loses then from monotonicity (Lemma 2) any subset $C \subseteq I_{-i}$, also loses, so any coalition without agent i is losing, and i is a veto player. On the other hand, suppose I_{-i} wins. In this case, I_{-i} is a winning coalition where agent i is not present, so by definition i is not a veto player. Therefore, testing whether i is a veto agent simply requires testing if I_{-i} loses. By Definition 5 of PDGs, checking if I_{-i} wins requires checking if I_{-i} does not contains a path from s to t, which can be performed in polynomial time using a depth first search (DFS), so checking if an agent is a veto agent in a PDG can be done in polynomial time. We also note that I_{-i} induces a graph where all the edges are removed except edges which connect vertices in the set $\{s, t, i\}$. Thus, i is a veto agent if s is directly connected to it, and it is directly connected to t. □

Computing the core in simple monotone games simply requires returning a list of all the veto agents, so we obtain the following corollary.

Corollary 1. *It is possible to compute the core of a PDG in polynomial time.*

Proof. Due to Lemma 3, we can find out who the veto agents are in polynomial time. Due to Lemma 1, if no veto agents exists, the core is empty, and if there is at least one veto agent, any imputation where all the non-veto agents are allocated zero is in the core, since the veto agents share all the gains. □

Although the core of PDGs can be computed in polynomial time, due to the observation in Lemma 3, the cases where there exist veto agents so the core is non-empty are rare degenerate cases. When the core is empty, we must use the more relaxed notion of the least-core. Unfortunately, we show that for general PDGs, computing the least-core is NP-Hard, through the MAX-DEFICIT problem.

Definition 7. *PDG-MAX-DEFICIT Given a PDG and an imputation $p = (p_1, \dots, p_n)$ and a bound d, decide whether $max_{C \subseteq I}(v(C) - p(C)) \leq d$. In other words, decide whether the maximal deficit of any coalitions is d or less.*

Lemma 4. *PDG-MAX-DEFICIT is NP-Complete.*

Proof. PDG-MAX-DEFICIT assigns each vertex v_i a value p_i. Any losing coalition C has value $v(C) = 0$, and so has a negative deficit $v(C) - p(C)$. Winning coalitions have non-negative deficit, as $v(C) = 1$ and $p(C) \leq 1$, so the maximal deficit coalition is a winning one. Thus, finding the maximal deficit coalition requires finding an $s-t$ vertex cut C, that has minimal $p(C)$. If we consider p_i to be the "cost" of vertex v_i, this could be restated as finding the minimal cost vertex $s-t$ cut, a known NP-complete problem [11]. □

Theorem 1. *Testing whether an imputation $p = (p_1, \dots, p_n)$ is in the least-core in PDGs is NP-Hard.*

Proof. We use a Turing reduction from PDG-MAX-DEFICIT to testing whether an imputation is in the least-core. We note that PDGs are simple games, so for $\epsilon = 1$ the ϵ-core is not empty, and for $\epsilon = -1$ the ϵ-core is always empty. Given an input $p = (p_1, \dots, p_n)$ to PDG-MAX-DEFICIT, we can use an oracle for testing whether an imputation is in the least-core, and perform a binary search to find the minimal ϵ for which p is in the ϵ-core. This is, by definition, the maximal deficit of any coalition, and is the answer to PDG-MAX-DEFICIT. □

4.2 Computing the Banzhaf Power Index in PDGs

In the no-cost game we do not directly model costs of placing checkpoints, but rather attempt to identify critical checkpoint locations. We first discuss power indices and their relation to locating the most critical checkpoints to place. Our goal is to identify important locations to secure, and we use the Banzhaf power index in PDGs to do so.

We are interested in preventing the adversary from traveling from the source to the target. Suppose we randomly flip a coin for each vertex to decide whether to place a checkpoint there, giving equal probability to placing a checkpoint or not. This results in a certain probability of preventing the adversary from traveling from the source to the target. Now suppose we can guarantee having a checkpoint at exactly one location, v_i, rather than flipping a coin for that location. The Banzhaf index of v_i measures the probability of the generated checkpoint formation achieving the goal of preventing the adversary from arriving at the destination. Thus, the higher the Banzhaf index of a vertex is, the more important it is to place checkpoint there. Therefore, in order to find important security locations, we can compute the Banzhaf power index, and focus on the vertices with the highest indices. We now consider the complexity of computing the Banzhaf power index in no-cost PDGs.

Definition 8. *PDG-BANZHAF: We are given a PDG over the graph $G = \langle V, E \rangle$, with agents $I = (1, \dots, n)$, where agent i controls vertex $v_i \in V$. The game's characteristic function $v : 2^I \rightarrow \{0, 1\}$ is defined as in Definition 5. We are given a target agent i, and are asked to compute its Banzhaf power index, $\beta_i(v)$.*

We now show that in general PDGs, PDG-BANZHAF is #P-complete. We first consider a related problem of testing whether an agent is a dummy agent in a PDG, and show it is NP-Complete. We use #NAE-SAT (Not All Equal SAT), a variant of the famous satisfiability problem.

Definition 9. *#SAT: We are given a propositional formula in CNF (conjunctive normal form), $\phi = c^1 \wedge c^2 \dots \wedge c^n$ (a conjunction of clauses) where $c^i = l^i_1 \vee l^i_2 \vee \dots \vee l^i_n$, and l^i_j is a propositional variable or its negation. We are asked to return the number of different truth assignments satisfying ϕ.*

Definition 10. *Not All Equal SAT (NAE-SAT): We are given a propositional formula in CNF, ϕ, and are asked whether there is a NAE-truth assignment for ϕ. An NAE-assignment for ϕ is a truth assignment satisfying ϕ such that for each clause at least one literal is falsified by the assignment.*

Definition 11. *#NAE-SAT: We are given a propositional formula in CNF (conjunctive normal form), ϕ, and are asked to return the number of different NAE-assignments for ϕ.*

We consider the problem of testing whether an agent is a dummy agent in a PDG.

Definition 12. *PDG-DUMMY: We are given a PDG over the graph $G = \langle V, E \rangle$, with agents $I = (1, \dots, n)$, where agent i controls vertex $v_i \in V$. We are asked whether agent i is a dummy, so for any coalition $C \subset I$ we have $v(C) = v(C \cup \{i\})$.*

Theorem 2. *PDG-DUMMY is coNP-Complete.*

Proof. We reduce a NAE-SAT instance to a PDG-DUMMY instance. Let the NAE-SAT instance be of the formula $\phi = c^1 \wedge \dots \wedge c^m$ over the propositional variables $x_1, \dots, x_n$, so each literal is either x_i or $\neg x_i$. The reduction is as follows. First, we create vertices v_{x_i},$v_{\neg x_i}$,w_{x_i},$w_{\neg x_i}$ for any $i \in \{1, \dots, n\}$. For each clause c_j of q_j literals, where $c^j = l^j_1 \vee l^j_2 \vee \dots \vee l^j_{q_j}$, we create a vertices $v^{c^j}_1, \dots, v^{c^j}_{q_j}$ and vertices $w^{c^j}_1, \dots, w^{c^j}_{q_j}$ so we have a v vertex and a w vertex for every literal in the clause. We call the v vertices the *left* vertices, and the w vertices the *right* vertices. The v_{x_i} and w_{x_i} are called the *positive vertices*, and the $v_{\neg x_i}$ and $w_{\neg x_i}$ are called the *negative vertices*. If $l^j_k = x_p$ (a positive literal) then $v^{c^j}_k, w^{c_j}_k$ are called positive vertices, and if $l^j_k = \neg x_p$ (a negative literal) then $v^{c^j}_k, w^{c_j}_k$ are called negative vertices.

For all i, v_{x_i} is connected to w_{x_i}, and $v_{\neg x_i}$ is connected to $w_{\neg x_i}$. If $l^j_k = x_p$ (a positive literal), we connect v_{x_i} to $w^{c_j}_k$, i.e. connect the left positive vertex and right positive vertex for that proposition. If $l^j_k = \neg x_p$ (a negative literal), we connect $v_{\neg x_i}$ to $w^{c_j}_k$, i.e. connect the left negative vertex and right negative vertex for that proposition The edges connecting left and right vertices are called *shortcut edges*. We also create a vertex v^*, separating the left and right vertices. We create a source vertex s and a target vertex t. We connect s to v_{x_1} and to $v_{\neg x_1}$. We do the following for any $i \in \{1, \ldots, n-1\}$: we connect v_{x_i} to both $v_{x_{i+1}}$ and $v_{\neg x_{i+1}}$, and connect $v_{\neg x_i}$ to both $v_{x_{i+1}}$ and $v_{\neg x_{i+1}}$; Similarly, we connect w_{x_i} to both $w_{x_{i+1}}$ and $w_{\neg x_{i+1}}$, and connect $w_{\neg x_i}$ to both $w_{x_{i+1}}$ and $w_{\neg x_{i+1}}$. We also connect all the vertices of a clause to all the vertices of the next clause, i.e. for any x, $v^{c^j}_x$ is connected to all $v^{c^{j+1}}_y$ and $w^{c^j}_x$ is connected to all $w^{c^{j+1}}_y$, for any y. We connect the vertices of the last left clause to v^*, i.e. for any x, $v^{c^m}_x$ is connected to v^*. We connect v^* to w_{x_1} and to $w_{\neg x_1}$. Finally, all the vertices of the last clause are connected to t, so for any x, $w^{c^m}_x$ is connected to t. Due to this construction, any path from s to t must either pass through v^* or contain a shortcut edge. An example of a construction for the formula $\phi = (x_1 \vee \neg x_2) \wedge (x_2 \vee \neg x_1)$ is given in Figure 1.

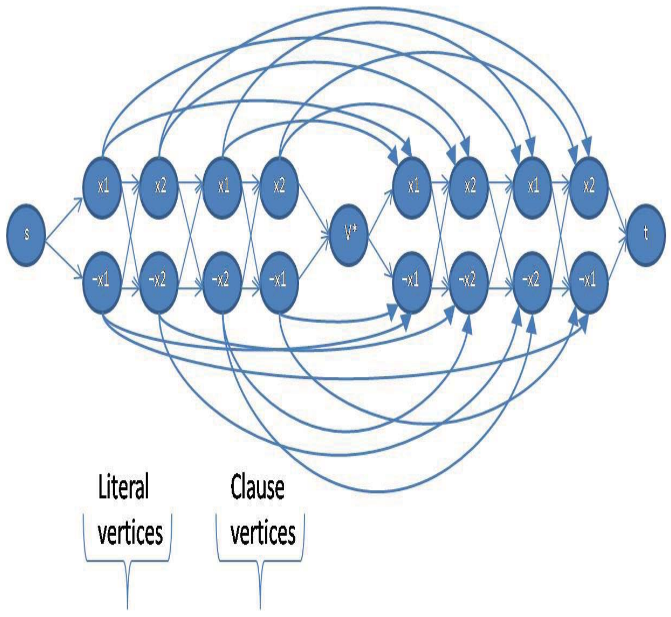

Figure 1: Reduction from NAE-SAT to PDG-DUMMY example.

We show v^* is non-dummy if and only if the NAE-SAT is a positive instance. If the NAE-SAT is a positive instance, consider an NAE truth assignment a for ϕ. Denote by T_a the propositions which are assigned True in a, so $T_a = \{x_i | a(x_i) = True\}$, and by F_a the propositions assigned False, so $F_a = \{x_i | a(x_i) = False\}$. Denote the left vertices of literals that are true under the assignment as $L_a = \{v_{x_i} | x_i \in T_a\} \cup \{v_{\neg x_i} | x_i \in F_a\} \cup \{v^{c^j}_i | l^j_i = x_k \wedge x_k \in T_a\} \cup \{v^{c^j}_i | l^j_i = \neg x_k \wedge x_k \in F_a\}$. Denote the right vertices that are false under the assignment as $R_{\neg a} = \{w_{x_i} | x_i \in F_a\} \cup \{w_{\neg x_i} | x_i \in T_a\} \cup \{w^{c^j}_i | l^j_i = \neg x_k \wedge x_k \in T_a\} \cup \{w^{c^j}_i | l^j_i = \neg x_k \wedge x_k \in T_a\}$. We identify the NAE assignment with the coalition $C_a = L_a \cup R_{\neg a}$.

We now note by removing the vertices of C_a from the graph, we eliminate all the shortcut edges: if the left positive vertex of a proposition is not in C_a then right positive vertex of the same proposition is in C_a and vice versa, so C_a eliminates an vertex in one side of each shortcut edge. We now show that even after eliminating C_a, the graph still contains a $s - t$ path. Eliminate C_a from the graph. For any x_i either v_{x_i} or $v_{\neg x_i}$ remains, which allows a path from s to $v^{c^1_i}$ (for any i). Also, for any x_i either w_{x_i} or $w_{\neg x_i}$ remains (w_{x_i} remains iff $v_{\neg x_i}$ remains and $w_{\neg x_i}$ remains iff v_{x_i} remains), which allows a path from v^* to $w^{c^1_i}$ (for any i). Since a is a NAE assignment, each clause c has a literal l^c_x that a satisfied, and a literal l^c_y that a does not satisfy, so v^c_y is not eliminated and w^c_x is not eliminated, so there is a path from either v_{x_n} or $v_{\neg x_n}$ to v^*, and a path from either w_{x_n} or $w_{\neg x_n}$ to t. This means that C_a is a losing coalition, as it does not eliminate all paths from s to t. However, eliminating both C_a and v^* disconnect all $s-t$ paths in the graph, as all such paths must either go through v^* or use a shortcut edge, and eliminating C_a eliminates all shortcut edges. Thus, C_a is losing but $C_a \cup \{v^*\}$ is winning, so v^* is critical in $C_a \cup \{v^*\}$ and is a non-dummy agent.

On the other hand, suppose v^* is a non-dummy agent, so it is critical for a coalition C. We show C encodes an NAE truth assignment for ϕ. Since v^* is critical in C, we know that removing the vertices in C is not enough to disconnect s from t, but that eliminating $C \cup \{v^*\}$ does disconnect s and t. We note that the graph's vertices can be partitioned into layers, where the non-shortcut edges only go between consecutive layers. The layers are a layer including $v_{x_i}, v_{\neg x_i}$, a layer for the v vertices for clause i, including $v^{c^i_j}$ for all j, a layer for v^*, and the equivalent layers for the w vertices. Under these definitions of layers, the only edges that go between non consecutive layers are the shortcut edges. We also note that for any two consecutive layers, all the possible edges between the two layers are in the graph. We now show that eliminating C must eliminate all shortcut edges. Suppose by contradiction that eliminating C results in a certain shortcut edge remaining. Consider the $s - t$ path that remains after C is eliminated. Either this path does not go through v^* or we can construct a path that does not go through v^* by taking this path up to the layer from which the shortcut edge originates, continue through the shortcut edge bypassing v^*, and connect the target edge of the shortcut edge to the rest of the original $s - t$ path. The last connection is possible as the original graph contains all edges between any two consecutive layers. Thus, after eliminating C there remains an $s - t$ path not going through v^*. However, this means that after eliminating $\{v^*\} \cup C$, there still remains an $s-t$ path in contradiction to coalition $C \cup \{v^*\}$ winning. Thus, eliminating C eliminates all shortcut edges.

Since eliminating C eliminates all the shortcut edges and since all the left positive and right positive literals of the same proposition are connected, C must contain one of them to eliminate the shortcut edge connecting them. C cannot contain both v_{x_i} and $v_{\neg x_i}$, as eliminating both of them eliminates all $s - t$ paths. Using C we can thus construct an assignment a: if C contains v_{x_i} then $a(x_i) = False$ and if C contains $v_{\neg x_i}$ then $a(x_i) = True$. We show a is an NAE truth assignment for ϕ. For any clause c^j, C cannot contain $v^{c^j}_i$ for all i, as this means eliminating C disconnects s

and t. Thus there must exist some $v_i^{c^j} \notin C$. If this l_i^j is a positive literal, so $l_i^j = x_i$, C does not contain v_{x_i}, and thus must contain $v_{\neg x_i}$, so $a(x_i) = True$, so c^j is satisfied. If this l_i^j is a negative literal, so $l_i^j = \neg x_i$, C does not contain $v_{\neg x_i}$, and thus must contain v_{x_i}, so $a(x_i) = False$, so c^j is satisfied. Thus a satisfies every clause, and so it satisfies ϕ. One the other hand we can consider the reverse assignment a' where if C contains v_{x_i} then $a'(x_i) = True$ and if C contains $v_{\neg x_i}$ then $a(x_i) = False$. Applying the same arguments regarding the right vertices (the w vertices), a' is also a satisfying assignment. Thus a and its reverse assignment a' both satisfy ϕ, so a is an NAE truth assignment for ϕ. □

We note that the Banzhaf power index of a dummy agent in any game is 0, since Definition 4 simply is a sum of zeros for dummy agents. Thus even testing whether an agent's Banzhaf index is zero or more in a PDG is NP-hard. We now show a stronger result of #P-Completeness.

THEOREM 3. *PDG-BANZHAF is #P-Complete.*

PROOF. PDG-BANZHAF is *in* #P since is is possible to test whether a coalition is winning or losing in polynomial time, so it is possible to construct a Turing machine which non-deterministically chooses a coalition that contains i and accepts if the i is critical in that coalition. The number of accepting paths for that machine would be the number of coalitions where i is critical, c_i. The Banzhaf index of i, β_i, is simply c_i times a constant, as $\beta_i = \frac{1}{2^{n-1}} \cdot c_i$, PDG-BANZHAF is in #P.

We now note that #NAE-SAT is #P-complete, as there exists a parsimonious reduction from SAT to NAE-SAT [11]. Consider a #NAE-SAT instance and the PDG constructed from it using the construction of Theorem 2. We note any coalition C where v is critical encodes a distinct NAE truth assignment. Denote by q the total number of NAE truth assignments. Thus, $\beta_i = \frac{1}{2^{n-1}} \cdot q$, so $q = \beta_i \cdot 2^{n-1}$, so computing the Banzhaf power index in PDG allows us to solve #NAE-SAT. □

4.3 PDGs on Trees

The above discussion shows that certain problems are hard for general graphs in the no costs model. Specifically, although the core is computable in polynomial time, checking ϵ-core membership and computing the least-core are hard, testing for dummy agents is coNP-complete, and computing the Banzhaf index is #P-complete. We show these problems become tractable for the restricted class of PDGs where the underlying graph is a tree.

We first consider testing membership in the ϵ-core and computing the least core.

THEOREM 4. *Testing whether an imputation $p = (p_1, \ldots, p_n)$ is in the ϵ-core in tree PDGs is in P.*

PROOF. We first note that in tree PDGs, there is only one path between s and t. Denote the vertices on this path by V^r. Any coalition containing any vertex in V^r wins, and any coalition containing none of the vertices in V^r loses. In order for p to be in the ϵ-core, the following constraint must hold for any coalition C: $p(C) \geq v(C) - \epsilon$. For any coalition C, $p(C) \geq 0$. Thus, the constraint holds for any losing coalition. Let i be a vertex on the $s - t$ path, so $i \in V^r$. In order for p to be in the ϵ-core, the following must hold: $p_i \geq 1 - \epsilon$. Also note that if the constraint $p_i \geq 1 - \epsilon$ holds, the constraint $p(C) \geq v(C) - \epsilon$ holds for any C such that $i \in C$, since all the p_j's are positive. However, any winning coalition must contain some vertex $i \in V^r$, so if the condition $p_i \geq 1 - \epsilon$ holds for all $i \in V^r$, the condition $p(C) \geq v(C) - \epsilon$ holds for any coalition C. Thus, p is in the ϵ-core if and only if $p_i \geq 1 - \epsilon$ for all $i \in V^r$.

We can easily compute V^r in polynomial time, by running a DFS from s to t, and check if $p_i \geq 1 - \epsilon$ for all $i \in V^r$, so we can test ϵ-core membership in polynomial time. □

THEOREM 5. *Finding the least-core in tree PDGs is in P.*

PROOF. Due to Theorem 4, an imputation $(p_1, \ldots, p_n)$ is in the ϵ-core iff for all $i \in V^r$ we have $p_i \geq 1 - \epsilon$. Denote $p(V^r) = \sum_{i \in V^r} p_i$, and $k = |V^r|$. Summing the constraint across all $i \in V^r$ we get the following constraint: $p(V^r) \geq k(1 - \epsilon)$. Equivalently $\epsilon \geq 1 - \frac{p(V^r)}{k}$. The minimal bound for ϵ is obtained when $p(V^r) = 1$ (as $p(V^r) \leq v(I) = 1$), so we denote $\epsilon_m = 1 - \frac{1}{k}$. Thus, the least-core value must be at least $\epsilon_m = 1 - \frac{1}{k}$. We show that this value is indeed enough. Suppose we split the payoff equally between the vertices in V^r, and give nothing to other vertices, so for any $i \in V^r$ we have $p_i = 1 - \frac{1}{k}$. We note that all the constrains $p_i \geq 1 - \epsilon$ hold for $\epsilon = \epsilon_m = 1 - \frac{1}{k}$. We also note that setting any p_i to a value greater than $1 - \frac{1}{k}$ means that for at least one vertex v_j we would have $p_j < 1 - \frac{1}{k}$ (as $\sum_{i \in I} p_i = 1$, so we have a total of a single unit to distribute), so one of the constraints would not hold.

Thus, the least core is the single imputation $p = (\frac{1}{k}, \ldots, \frac{1}{k})$ and $\epsilon_{min} = 1 - \frac{1}{k}$ for tree PDGs [3]. □

We now consider the Banzhaf index in tree PDGs.

THEOREM 6. *Testing for dummy agents and computing the Banzhaf power index in tree PDGs are in P.*

PROOF. We note any vertex $i \in V^r$ (i.e. a vertex on the $s - t$ path) is critical for the coalition $\{i\}$, since $\{i\}$ wins and the empty coalition loses, so i is not a dummy agent. On the other hand, any vertex $j \notin V^r$ is a dummy, as a coalition wins iff it contains some vertex $i \in V^r$, and this characteristic of a coalition does not change when removing some $j \notin V^r$ from the coalition. Thus, a vertex i is a dummy if and only if $i \notin V^r$, which can be tested in polynomial time, since V^r can be computed in polynomial time.

We now note that by definition the Banzhaf index of i is $\beta_i = \frac{C_i}{2^{n-1}}$ where C_i is the number of coalitions where i is critical in. As noted above, if $i \notin V^r$, she is a dummy agent and has a Banzhaf index of 0. We now consider a vertex $i \in V^r$, on the $s - t$ path. Consider a tree with k vertices on the $s - t$ path so $|V^r| = k$, and with m vertices not on the $s - t$ path, so $n = k + m$. We note i is critical in any coalition C that contains i and no other vertex $j \in V^r$ on the $s - t$ path. There are 2^m such coalitions, so $\beta_i = \frac{2^m}{2^{m+k-1}} = \frac{1}{2^{k-1}}$, so we have a closed form formula for β_i, as $k = |V^r|$ can be computed in polynomial time. □

5. THE VARIOUS COSTS MODEL

We now consider the model where placing a checkpoint is a costly action, and consider the PDGC (Path Disruption

[3]Note that the same argument holds for any monotone simple game where there is a set S of agents where any winning coalition contains at least one member of S and where any member of S wins.

Game with Costs) of Definition 6. This model associates a cost c_i for placing a checkpoint in vertex v_i, and assumes that a coalition which disrupts the $s-t$ path obtains a reward r for doing so. A coalition C can place checkpoints only in its vertices, but there may be different configurations of checkpoints that disrupt all $s-t$ paths in the graph. Given a subset of coalition vertices $S \subseteq C$, we denote the cost of S as $\sum_{i|v_i \in S} c_i$. As in Section 3, for any $S \subset I$ we denote $w(S) = 1$ if S disconnects s and t, and $w(S) = 0$ otherwise. In Definition 6, given a coalition C, we have used the *lowest cost* placements of checkpoints for C that disrupts all $s-t$ paths in the graph, i.e. a configuration with cost $m(C) = min\{c(B)|B \subseteq C, w(B) = 1\}$. The PDGC is well defined mathematically, and has a very simple representation, which includes a representation of the graph, the reward r and the costs c_i. We show that in PDGCs even computing the value of a coalition is NP-Hard.

LEMMA 5. *Computing the value of a coalition, $v(C)$, in PDGCs is NP-Hard.*

PROOF. We reduce an Min Cost Vertex $s-t$ Cut (MCVC) instance (see Lemma 4) to computing $v(C)$ in PDGCs. We copy the MCVC graph, source s and target t. An MCVC instance has a vertex cost p_i for each vertex, which are the vertex costs c_i in the reduced PDGCs instance. We set the reward $r = \sum_{v_i \in V} c_i$. Consider the grand coalition I. The value of I in the reduced PDGC instance is defined as $v(I) = r - m(C)$, so given $v(I)$ we can compute $m(C) = r - v(I)$. However, $m(C)$ is the cost of the optimal checkpoint placement which eliminates all $s-t$ paths, or in other words, the cost of the minimal cost $s-t$ vertex cut. This is exactly the solution to the MCVC instance. □

Due to Lemma 5, the above PDGC representation is concise, but hard to handle computationally. It is unlikely that significant game theoretic concepts could be computed in polynomial time when even computing the value of a coalition is computationally hard. In order to use such a representation, we must thus restrict the represented domain. Section 4.3 discusses some problems that are computationally hard in the no-costs model that become tractable when considering trees. We show that some problems also become tractable for the model with costs when the graphs are restricted to be trees.

THEOREM 7. *For tree PDGCs, computing a coalition's value, $v(C)$, and testing for dummy agents are in P.*

PROOF. Similarly to the results in Section 4.3, we use the fact that in trees there is a single $s-t$ path, and denote the vertices along this path as $V^r \subseteq V$. We assume that the reward r is greater than the cost of any vertex c_i, since if for some i we have $r < c_i$ then i is a useless agent - placing a checkpoint at that location means the coalition would have a negative value. Thus, we can eliminate from the game any agent with a cost $c_i > r$.

In tree PDGCs, a coalition C can only win if it contains some vertex in V^r so $C \cap V^r \neq \emptyset$. If $C \cap V^R = \emptyset$ then $v(C) = 0$. If $C \cap V^R \neq \emptyset$, placing a checkpoint at any location $i \in V^r$ results in winning. We denote $C^r = C \cap V^R$. The optimal checkpoint location for C is in the minimal cost vertex in C^r, $v_i = argmin_{i \in C^r} c_i$. Thus, the value of a coalition C, $r - min_{i \in C^r} c_i$, is computable in polynomial time. Similarly to Theorem 6, any vertex in $i \in V^r$ is non-dummy (unless $c_i > r$, in which case it is a dummy vertex), and any vertex $i \notin V^r$ is a dummy agent. □

THEOREM 8. *For PDGCs over trees, computing the least-core is in P.*

PROOF. We first show that testing whether an imputation $p = (p_1, \ldots, p_n)$ is in the ϵ-core is in P. In order for p to be in the ϵ-core, the following must hold for every $i \in V^r$: $p_i \geq r - c_i - \epsilon$, since for any $i \in V^r$, $v(\{i\}) = r - c_i$. For any coalition C, if $C \cap V^r = \emptyset$ then $v(C) = 0$, and the constraint $p(C) \geq v(C) - \epsilon$ holds. For any coalition C such that $C \cap V^r \neq \emptyset$, the ϵ-core constraint is $p(C) \geq v(C) - \epsilon = r - min_{i \in C^r} - \epsilon$. Thus, if the constraint $p_i \geq r - c_i - \epsilon$ holds for all $i \in V^r$, the constraint $p(C) \geq r - min_{i \in C^r} - \epsilon$ holds for any C such that $C \cap V^r \neq \emptyset$. Thus, an imputation $p = (p_1, \ldots, p_n)$ is in the ϵ-core iff for all $i \in V^r$ the constraint $p(C) \geq r - min_{i \in C^r} - \epsilon$ holds, which is testable in polynomial time.

We apply a process similar to Theorem 5. Denote $t_i = r - c_i$, $|V^r| = k$, $c(V^r) = \sum_{i \in V^r} c_i$ and $c_{min} = min_{i \in V^r} c_i$. The total value to distribute is $v(I) = r - c_{min}$. The constraint $p_i \geq t_i - \epsilon$ must hold for every $i \in V^r$. Thus, the minimal ϵ must be one such that $\sum_{i \in V^r} t_i - k\epsilon = r - c_{min}$ or equivalently $\epsilon = \frac{1}{k}(\sum_{i \in V^r} t_i - r + c_{min})$. For such ϵ, we can distribute $v(I)$ by having $p_j = 0$ for any $j \notin V^r$ and setting $p_i \geq t_i - \epsilon$ for any $i \in V^r$, so the constraint $p(C) \geq r - min_{i \in C^r} - \epsilon$ holds for any C such that $C \cap V^r \neq \emptyset$. Thus, $\epsilon_{min} = \frac{1}{k}(\sum_{i \in V^r} t_i - r + c_{min})$ is the value of the least-core. These payments are the only imputation in the least-core, as increasing any p_i forces decreasing some other p_j's below the threshold for the constraint $p_j \geq t_j - \epsilon$. □

6. RELATED WORK

We analyzed physical and network security through a cooperative game theoretic prism. We considered the core, the ϵ-core and least-core, and the Banzhaf index. The core was introduced in [12]. The Banzhaf index we have considered is a power index. Power indices were introduced as a measure of the influence players in a game have on determining its outcome. The two most popular indices are the Banzhaf index [6] and the Shapley-Shubik index [22]. Several solution concepts have been suggested as an alternative to the core, when the core is empty. Such relaxed concepts include the nucleolus [21], and the ϵ-core and least-core, discussed in [14]. Power indices were first applied in political science to measure influence in Weighted Voting Games (WVGs).

The Banzhaf index can be computed in any simple game, with the complexity depending on the game's representation. If the game is only defined through an oracle for computing coalition values, calculating the Banzhaf index is costly. A naive implementation must enumerate over all coalitions, which is exponential in the number of agents. Matsui [15] showed that computing the Banzhaf or Shapley index is NP-complete in WVGs [16], and [7] shows that computing the Shapley index in WVGs is #P-complete. Otherr works [3, 8] show computing such indices in cooperative skill based and contribution based games is also hard.

Using power indices for voting domains is well studied, but applying them in non-voting domains is much less frequent. Few examples of this are [10, 8, 4, 5]. The models in [4, 5] consider network reliability domains where agents control parts of the network, and use game theory to improve the *reliability* of the network. Another work [20] considers external subsidies guaranteeing a stable distribution of gains in cooperative network flow games. We deal with a very different problem, of improving physical or network *security*, and our agents cooperate in an attempt to prevent the adversary from reaching a target in the network. In this, our models are much more similar to [1, 18, 19] which use game theory in order to improve physical security, by choosing a proper physical security policy. Our work is also very different from those papers, as our model is cooperative in nature — we use coalitional game theory, rather than the non-cooperative game theoretic concepts used in those papers.

Using the Banzhaf index to identify critical security locations requires computing it in large domains. One way of circumventing this problem is by considering restricted domains, so we have considered tree PDGs. A different approach is using approximation algorithms. Several approximation algorithms for computing power indices exist [13, 15]. Two newer methods are [9] which approximates the Shapley value in weighted voting games, and [2], which approximates both the Banzhaf index and Shapley value in general cooperative games. Such methods allow using the models of this work despite the computational obstacles.

7. CONCLUSIONS

We considered a game theoretic approach to physical and network security, based on coalitional game theory. We modeled cooperation between agents wishing to stop an adversary from traveling or communicating between two points, as the Path Disruption Game. We discussed both stable payment schemes and power indices reflecting the relative importance of different locations.

Several directions remain open for future research. First, our security model was simple, of a single adversary wishing to travel between two specific points. It would be interesting to generalize this and model uncertainty about the adversary's origin or target, and about the costs of the agents. Second, it would be interesting to see whether similar approaches work for domains other than physical security. Finally, it would be interesting to see if other game theoretic notions can be computed in this or similar domains.

8. REFERENCES

[1] N. Agmon, V. Sadov, G. A. Kaminka, and S. Kraus. The impact of adversarial knowledge on adversarial planning in perimeter patrol. In *AAMAS (1)*, pages 55–62, 2008.

[2] Y. Bachrach, E. Markakis, A. D. Procaccia, J. S. Rosenschein, and A. Saberi. Approximating power indices. In *AAMAS (2)*, pages 943–950, 2008.

[3] Y. Bachrach and J. S. Rosenschein. Coalitional skill games. In *AAMAS 2008*, pages 1023–1030, Estoril, Portugal, May 2008.

[4] Y. Bachrach and J. S. Rosenschein. Power in threshold network flow games. *Autonomous Agents and Multi-Agent Systems*, 18(1):106–132, 2009.

[5] Y. Bachrach, J. S. Rosenschein, and E. Porat. Power and stability in connectivity games. In *The Seventh International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, pages 999–1006, Estoril, Portugal, May 2008.

[6] J. F. Banzhaf. Weighted voting doesn't work: a mathematical analysis. *Rutgers Law Review*, 19:317–343, 1965.

[7] X. Deng and C. H. Papadimitriou. On the complexity of cooperative solution concepts. *Math. Oper. Res.*, 19(2):257–266, 1994.

[8] E. Elkind, L. A. Goldberg, P. W. Goldberg, and M. Wooldridge. A tractable and expressive class of marginal contribution nets and its applications. In *AAMAS 2008*. IFAAMAS, 2008.

[9] S. S. Fatima, M. Wooldridge, and N. R. Jennings. A linear approximation method for the shapley value. *Artif. Intell.*, 172(14):1673–1699, 2008.

[10] J. Feigenbaum, C. H. Papadimitriou, and S. Shenker. Sharing the cost of multicast transmissions. *Journal of Computer and System Sciences*, 63(1):21–41, 2001.

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

[12] D. B. Gillies. *Some theorems on n-person games*. PhD thesis, Princeton University, 1953.

[13] I. Mann and L. S. Shapley. Values of large games, IV: Evaluating the electoral college by Monte-Carlo techniques. Technical report, The Rand Corporation, Santa Monica, CA, 1960.

[14] M. Maschler, B. Peleg, and L. S. Shapley. Geometric properties of the kernel, nucleolus, and related solution concepts. *Mathematics of Operations Research*, 4(4):303–338, 1979.

[15] Y. Matsui and T. Matsui. A survey of algorithms for calculating power indices of weighted majority games. *J. Operations Research Society of Japan*, 43, 2000.

[16] Y. Matsui and T. Matsui. NP-completeness for calculating power indices of weighted majority games. *Theoretical Computer Science*, 263(1–2), 2001.

[17] M. Osborne and A. Rubinstein. *A course in game theory*. MIT press, 1999.

[18] J. Pita, M. Jain, J. Marecki, F. Ordóñez, C. Portway, M. Tambe, C. Western, P. Paruchuri, and S. Kraus. Deployed armor protection: the application of a game theoretic model for security at the los angeles international airport. In *AAMAS (Industry Track)*, pages 125–132, 2008.

[19] J. Pita, M. Jain, F. Ordóñez, C. Portway, M. Tambe, C. Western, P. Paruchuri, and S. Kraus. Armor security for los angeles international airport. In *AAAI*, pages 1884–1885, 2008.

[20] E. Resnick, Y. Bachrach, R. Meir, and J. Rosenschein. The cost of stability in network flow games. In *Proceedings of the 34th International Symposium on Mathematical Foundations of Computer Science 2009*, page 650. Springer, 2009.

[21] D. Schmeidler. The nucleolus of a characteristic function game. *SIAM Journal on Applied Mathematics*, pages 1163–1170, 1969.

[22] L. S. Shapley and M. Shubik. A method for evaluating the distribution of power in a committee system. *American Political Science Review*, 48:787–792, 1954.

[23] L. G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8:410–421, 1979.