A Centrality-Based Security Game for Multi-Hop Networks

Riehl, James Robert; Cao, Ming

Published in:
IEEE Transactions on Control of Network Systems

DOI:
10.1109/TCNS.2017.2728202

Publication date:
2018

A Centrality-Based Security Game for Multi-Hop Networks

James R. Riehl† and Ming Cao†

***Abstract*—We formulate a network security problem as a zero-sum game between an attacker who tries to disrupt a network by disabling one or more nodes, and the nodes of the network who must allocate limited resources in defense of the network. The utility of the zero-sum game can be one of several network performance metrics that correspond to node centrality measures. We first present a fast centralized algorithm that uses a monotone property of the utility function to compute saddle-point equilibrium strategies for the case of single-node attacks and single- or multiple-node defense. We then extend the approach to the distributed setting by computing the necessary quantities using a finite-time distributed averaging algorithm. For simultaneous attacks to multiple nodes the computational complexity becomes quite high, so we propose a method to approximate the saddle-point equilibrium strategies based on a sequential simplification, which performs well in simulations.**

I. Introduction

As society grows more reliant upon networked and cyber-physical systems for communication, transportation, sensing, control, and other applications, these systems occupy larger and more complex networks and are thus increasingly vulnerable to attack by malicious adversaries. With consequences ranging from costly inefficiencies to catastrophic failures, it is critical to understand how to secure these networks against such attacks. The field of research related to the security of such systems has consequently seen rapid growth in recent years [1], [2], and there are many angles from which to approach these problems. The first line of defense against attacks is to make the network as secure and resilient as possible, from physical and cyber security at the node level to robustness and redundancy at the network level [3], [4], [5]. Nevertheless, there will always be some vulnerabilities in large networks and resource limitations that may prevent every node in the network from being defended to the fullest extent. The question then becomes how to optimally distribute limited defense resources around the network to achieve some security objective.

[1]The work was supported in part by the European Research Council (ERC-StG-307207) and the Netherlands Organization for Scientific Research (NWO-vidi-14134).

†Faculty of Science and Engineering, ENTEG, University of Groningen, The Netherlands, {j.r.riehl, m.cao}@rug.nl

In order to answer this question, it is necessary to know how important each node or link is to the network. As a result, the use of *centrality measures*, first introduced to study the most influential people in social networks [6], [7], is starting to gain traction in the communication and control literature. For example, the authors of [8] studied the effects of coordinated attacks to wireless mesh networks and showed that targeting the nodes with the highest betweenness centrality results in a more effective attack than targeting nodes with the highest degree, a result supported by earlier studies on the attack vulnerability of complex networks [9]. However, this strategy might easily be predicted and thwarted by a smart defender. Game theory is ideally suited to such competitive settings and for this reason has become widely adopted in the study of network security [10][11]. Despite these emerging research trends, the use of centrality measures towards the game-theoretic solution of practical network security problems remains largely unexplored.

We specifically consider the case where network performance is of primary concern and the effectiveness of an attack is measured by the resulting change to a given performance metric. Since from the defender's point of view, a worst-case attacker will minimize exactly this performance metric, the problem fits naturally into the framework of zero-sum games, as do many other network security problems [12], [13], [14], [15]. Our approach is further motivated by two key observations. The first is that several important network performance metrics such as throughput, latency, and efficiency, can be decomposed into individual contributions from each node, in the form of a centrality measure. The second observation is that both attack and defense to a network generally require energy or some other finite resource, which is often costly or in short supply. For example, a denial of service (DoS) attack requires energy and bandwidth from one or more attackers who try to deplete the resources of the targeted system, and one way to defend such an attack is to add memory or bandwidth capacity to vulnerable nodes in the network [16]. Since there may not be enough resources to defend every node in a network, the critical decision becomes which nodes to protect, and the complexity of this decision depends on the topology of the underlying network. Moreover,

depending on whether the network has a centralized or distributed implementation, this decision may fall either to a central network administrator or to the individual nodes working in collaboration. We began to address the centralized case in [17]. In this paper we expand on those results and extend to the distributed case for a range of different attack and defense scenarios.

Closely-related to this topic are *interdiction* problems, which involve identifying the most critical nodes or links in a network with respect to various flow or path-distance metrics [18]. In particular, when one party is responsible for choosing the paths and/or flows along which information or products are transported in the presence of full or partial disruptions to the nodes or links, this is commonly referred to as an interdiction game [19], [20]. Typically, interdiction games are multi-stage, with players acting in turn based on observed changes to the network. Here we assume that the attacker knows only the topology of the network and not the level of defense at each node. Conversely, the nodes of the network do not know where the attacker will target, so we effectively have simultaneous play between the attacker and defender. This scenario is both highly plausible and in many cases allows for relatively fast computation of the solution, as we will show later.

Since the problem under consideration is essentially how to allocate defense resources to the nodes in a network, the most relevant literature to our approach involves the application of game theory to resource allocation in the presence of an adversary. One such example is [21], where two teams allocate power between communicating with teammates and jamming the other team's communications. After formulating the problem as a zero-sum differential game, the authors give sufficient conditions on the agent parameters for existence of a pure-strategy Nash equilibrium. Agent-based simulations are used in [22] to determine saddle-point equilibrium strategies in a network security game where both attacker and defender are subject to cost constraints. Coming from a different angle, the optimal design of resource allocation networks subject to attacks is studied in [23], where it is shown that networks with a star topology are often optimal from the perspective of a defender. In a transportation setting, path-planning against adversaries was formulated as a distributed resource allocation problem to which a linear-programming (LP) solution was proposed in [24]. Although our formulation also admits an LP solution in the centralized case, large networks can still pose a computational burden and the combinatorial growth of payoff matrices for multiple-node attack and defense further motivates faster solution algorithms. Moreover, the standard LP solution does not apply to the distributed case.

After summarizing some important background material on centrality and zero-sum game theory in Section II, we state the problem under consideration in Section III. In Section IV, we introduce a linear-time centralized algorithm to compute saddle-point equilibrium strategies for both the attacker and defender in the case of single-node attacks and multiple-node defense. These algorithms provide a network administrator with the probabilities that each node should be protected. We then extend the results for single-node attacks to the distributed setting in Section V, using a finite-time distributed averaging algorithm to compute the same result as in the centralized case. Finally, when multiple nodes are attacked simultaneously, we propose a centralized approximation algorithm in Section VI based on a sequential simplification of the attack strategies, which performs very well in a pair of simulation studies.

II. Preliminaries

Before presenting the main results, we briefly two subjects that are fundamental to understanding the problem and solution approach: node centrality and zero-sum game theory.

A. Node Centrality as Performance Metric

The *centrality* of a node can be thought of as the importance of a node in the context of the network, and there are many different measures of centrality related to various notions of importance. We are particularly interested in centrality measures that represent each node's contribution to a network performance metric. One notable example is *betweenness*, which is the fraction of all shortest paths in the network on which a node lies.

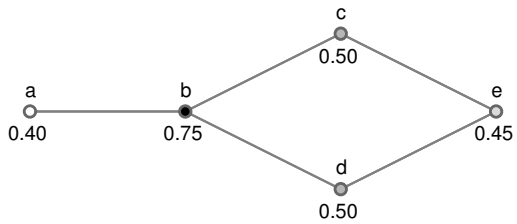

Fig. 1. Betweenness centrality on a small network

Consider the five-node undirected network shown in Fig. 1. There are ten different node pairings in this network but twelve paths that are shortest paths between

pairs of nodes:

$$ab, abc, abd, \{abce, abde\}, bc, bd,$$
$$\{bce, bde\}, \{cbd, ced\}, ce, de, \quad (1)$$

where bracketed sets of paths indicate that there are multiple shortest paths connecting two nodes. The betweenness of each node is obtained by counting the total number of shortest paths in which a node appears, dividing by k when the path is one of k shortest paths connecting two nodes. Finally, we divide by the total number of node pairs in the network so that the result lies between zero and one. Checking each path in (1), we see that node b lies on $1+1+1+(\frac{1}{2}+\frac{1}{2})+1+1+(\frac{1}{2}+\frac{1}{2})+\frac{1}{2}=7.5$ out of the 10 node pairs, or 0.75 of the shortest paths in the network, supporting the intuitive observation that b is an important node in this network. Note that this definition of betweenness actually differs slightly from standard betweenness definitions because we count being an endpoint of a path as being on the path while standard definitions typically do not. The reason is that for our purposes the endpoint nodes are just as important to the communication link as any intermediate node. This also prevents nodes from having a betweenness of zero. Our methods still apply if some nodes have centrality values of zero, but the result is trivially that these nodes should never be attacked or defended since they are negligible to the network with respect to the corresponding performance metric. We do restrict the centrality measures to take non-negative values, but this is quite a mild assumption since the vast majority of standard centrality measures are non-negative by definition.

Suppose now that every pair of nodes in a network exchanges a message using the shortest path between them, and that we measure the performance of the network by the fraction of successfully transmitted messages, *i.e.* the total *throughput* of the network. Betweenness centrality can equivalently be thought of as the fraction of all messages sent through the network that pass through a given node. The removal of this node will cause these transmissions to fail, until the node's removal is detected and the messages can be rerouted. Betweenness thus measures the contribution of each node to the total throughput of a network where traffic between all pairs of nodes is approximately uniform.

In the case that traffic varies significantly between node pairs, one can still define the importance of a node as the fraction of all flow from source to destination that passes through a given node. This is a kind of *flow* centrality. Other notable centrality measures include *degree* centrality, which corresponds to the fraction of all links in the network that connect to a given node, and *closeness* centrality, which is the average distance of a node to all other nodes in the network and is related to how long it takes to broadcast a message to the entire network, alternatively referred to as the *efficiency* of a network.

If multiple performance metrics are of concern, e.g. throughput and latency, these can be weighted and combined into a single performance metric to be used in the security problem, provided that each constituent metric can be decomposed into individual node contributions as described above.

Lastly, when multiple nodes are attacked simultaneously, it will not generally be possible to measure the effect of the attack on network performance from the individual node centrality values. Therefore we need to define the centrality of a *node set*, which we can motivate with an example. Consider two-node attacks to sets $\{a, b\}$ and $\{b, e\}$ in the network of Fig. 1. We want to evaluate the removal of these node pairs from the network, which in the case of betweenness, corresponds to the fraction of shortest paths containing at least one of the removed nodes. For $\{a, b\}$ this comes to 0.75 since every path that contains a also contains b. On the other hand, node e is on every shortest path that does not contain b, so the betweenness of $\{b, e\}$ is 1. Similar modifications can often be made to adapt other single-node centrality measures to node sets.

B. Zero-Sum Game Theory

In a *game*, two or more players choose from a set of strategies in pursuit of different and often competing objectives and receive payoffs that depend on the strategies of all players. A two-player zero-sum matrix game is a game in which one player's gain is equal to the other player's loss and can be modeled using a payoff matrix where one player is the minimizer and the other is the maximizer. A familiar example is the classic Rock-Scissors-Paper game, where the players choose between rock (R), scissors (S), and paper (P), knowing that rock beats scissors, scissors beats paper, paper beats rock, and otherwise there is no winner. A typical payoff matrix is as follows:

$$A = \begin{array}{c} \\ R \\ S \\ P \end{array} \begin{array}{c} \begin{array}{ccc} R & S & P \end{array} \\ \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix} \end{array},$$

where player 1 selects a row and wants to minimize the payoff and player 2 selects a column and wants to maximize the payoff. Players can play deterministically

by selecting a *pure strategy* in the set $\mathcal{S} := \{R, S, P\}$ or they can randomize their strategies using a *mixed strategy* $y := [y_R, y_S, y_P]^T$ where $y_R, y_S, y_P \geq 0$ are the probabilities of playing each respective strategy and thus $y_R + y_S + y_P = 1$. One can compute the expected payoff resulting from any two mixed-strategies y, z by evaluating $v = y^T Az$. Following are three particularly important concepts in game theory:

- a strategy X is a ***best response*** to Y if there is no other available strategy that will improve the payoff when played against strategy Y;
- a strategy is ***strictly dominated*** if it is not a best response to any strategy of the opponent;
- a ***Nash equilibrium*** is a state of a game in which both players are playing best responses to each other's strategy.

Nash equilibria of zero-sum games are commonly referred to as *saddle-point equilibria* since the players share a common utility function. In the Rock-Scissors-Paper example, each pure strategy is a best-response to a different pure strategy, but there is no pair of pure strategies that are best responses to each other. However, there is a saddle-point equilibrium in mixed strategies, when both players play each of the three strategies with one-third probability. We can express this by the mixed-strategy pair $y^* = z^* = [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]^T$. In fact, two-player games with finite strategies always admit a unique saddle-point equilibrium in mixed strategies [25]. This result illustrates another key concept in game theory, which is the *principle of indifference*, the idea being that if I am playing a saddle-point equilibrium mixed strategy, my opponent can pick from several strategies without seeing any change in payoff, and is thus indifferent to which strategy to play.

III. Problem Formulation

Suppose now that an attacker has the ability to disable one or more nodes, effectively removing them from the network, but that a limited number of nodes can be protected and thus immunized from attack. Whether from the perspective of a central administrator or the distributed collective of nodes, the critical question becomes which nodes to protect in order to maximize a particular performance metric in the presence of such an attack. Depending on the specific network application, there will be many different means of attack and potential security measures. The approach we present here is applicable to contexts in which the following assumptions hold:

1) Resource limitations prevent every node from defending itself.
2) The network does not recover fast enough to reroute traffic/information before the performance loss is assessed.
3) The attacker and defender make decisions independently, without knowledge of each other's decisions.

Assumption 1 holds particularly true in the case of physical attacks to a network. When security is simply a matter of implementing code, resource constraints would most likely not be the key limiting factor for network security. The second assumption will hold for networks in which speed is critical or recovery is slow. Although the problem is still solvable if the network is allowed to reroute traffic, it becomes significantly more complex, and we leave these dynamic extensions for future research. The third assumption establishes that this is not a sequential game but a simultaneous game; in fact, the sequential game is less interesting in this case, because the choices for either the attacker or defender become quite clear once the opposing player has acted. Mixed strategies therefore capture the players' ability to independently randomize their choices. Since a worst-case attacker wants to minimize the same performance metric that the defender wants to maximize, this problem fits well into the setting of a two-player zero-sum game. In fact we can already express the problem in general form as the following *minimax* problem:

$$v^* = \min_{y \in \Delta_{\mathcal{A}}} \max_{z \in \Delta_{\mathcal{D}}} y^T Az, \tag{2}$$

where y and z are mixed strategies of the attacker and defender, respectively, A is a payoff matrix whose entries represent the performance change in the network resulting from each pair of pure attack and defense strategies, and v^* is the saddle-point equilibrium value of the game. We denote by $\Delta_{\mathcal{A}}$ and $\Delta_{\mathcal{D}}$ the mixed-strategy simplexes over pure strategy attack and defense sets $\mathcal{A}$ and $\mathcal{D}$, which will depend on the number of nodes attacked (α) and defended (δ) as described in the following sections. Finally, we say that a pure strategy is contained in the *support* of a mixed strategy if the corresponding element of the mixed strategy is strictly positive.

A. Single-Node Attack / Single-Node Defense

We begin with the case where both the attacker and defender possess only enough resources to target one node, resulting in pure strategy sets that map directly to the nodes in the network, *i.e.* $\mathcal{A} = \mathcal{D} := \mathcal{V}$, where $\mathcal{V} := \{1, \ldots, n\}$. To construct the payoff matrix, each entry A_{ij} should correspond to the change in network performance when node i is attacked and node j is

defended. Since a defended attack results in no change to the network, A should have zeros on the diagonal. All other entries correspond to an undefended attack on node i, whose removal from the network results in a performance loss exactly equal to its centrality value, which we denote by a_i. The payoff matrix can therefore be expressed as

$$A = \begin{array}{c} \\ att_1 \\ att_2 \\ \vdots \\ att_n \end{array} \begin{array}{c} \begin{array}{cccc} def_1 & def_2 & \cdots & def_n \end{array} \\ \begin{pmatrix} 0 & -a_1 & \cdots & -a_1 \\ -a_2 & 0 & \cdots & -a_2 \\ \vdots & \vdots & \ddots & \vdots \\ -a_n & -a_n & \cdots & 0 \end{pmatrix} \end{array}, \quad (3)$$

where att_i and def_j respectively indicate attack to node i and defense of node j.

B. Single-Node Attack / Multiple-Node Defense

In the previous section we assumed that both the attacker and defender could only target a single node, but it is easy to imagine that the defender might have enough resources to protect $\delta > 1$ nodes. In this case, the attack strategy set $\Delta_{\mathcal{A}}$ remains the same, but the defense strategy set expands to cover all possible combinations of defended nodes. Let $\mathcal{V}_k$ denote the set of all subsets consisting of k distinct nodes in the network. Then we have the pure strategy set $\mathcal{D} := \mathcal{V}_\delta$ and $\Delta_{\mathcal{D}}$ becomes the mixed-strategy simplex over $\mathcal{D}$. Let $\mathcal{D}(j)$ denote the j^{th} set of δ nodes under some enumeration of the set $\mathcal{D}$. Each entry A_{ij} in the payoff matrix now corresponds to the expected change in network performance when node i is attacked and the nodes $\mathcal{D}(j)$ are defended. Therefore $A_{ij} = 0$ whenever $i \in \mathcal{D}(j)$ and $A_{ij} = -a_i$ otherwise.

C. Multiple-Node Attack / Multiple-Node Defense

Next we consider the case where the attacker also has enough resources to target $\alpha > 1$ nodes. Now the pure attack strategy set is $\mathcal{A} := \mathcal{V}_\alpha$, the pure defense strategy set remains $\mathcal{D} := \mathcal{V}_\delta$, and each entry A_{ij} corresponds to the expected network performance change when nodes $\mathcal{A}(i)$ are attacked and nodes $\mathcal{D}(j)$ are defended. Recall that we defined the centrality of a node set in Section II-A as the effect on network performance when that set of nodes is removed from the network. Hence A_{ij} is equal to the centrality of the node set $\mathcal{A}_\alpha(i) - \mathcal{D}_\delta(j)$ whenever this set is nonempty, and 0 otherwise.

IV. Centralized Solution

We begin by considering the problem from the perspective of a network administrator who has access to full information about the network. Problem (2) is in a standard form that admits a linear programming solution, and although there are algorithms that can solve linear programming problems in polynomial time [26], this can still pose a computational burden when working with very large networks. Moreover, the fact that payoff matrices grow combinatorially with the number of attack and defense nodes provides even stronger motivation to develop faster solution algorithms. We present here a simple and linear-time algorithm to compute the saddle-point equilibrium strategies in the case of single-node attacks.

A. Algorithm Design: Single-Node Defense

The key insight behind our approach is that the expected payoff resulting from a mixed attack strategy is monotone in the addition of pure strategies or nodes. In other words, when choosing between two nodes to add to the support of a mixed attack strategy, it is always better to choose the node with the higher centrality. This means we can incrementally construct a mixed defense strategy by adding support nodes in order of decreasing centrality until all remaining pure strategies are dominated. This approach is made possible by the following propositions.

Proposition 1: At a saddle-point equilibrium, all pure strategies not contained in the support are strictly dominated.

Proposition 1 follows directly from the fact that strictly dominated pure strategies always correspond to the zero-entries of equilibrium mixed strategies [25].

Proposition 2: Support nodes are those with the highest centrality.

Proof: Suppose there exist nodes i and j such that $i \in \mathcal{S}$, $j \notin \mathcal{S}$, and $a_j > a_i$. Since $z_j = 0$, this node is undefended and the attacker can achieve a lower payoff with a mixed strategy y' such that $y'_j = y_i$, $y'_i = 0$, and $y'_k = y_k$ for all $k \notin \{i, j\}$. Therefore, node j is not strictly dominated, a contradiction of Proposition 1, which means that the support nodes must be those with the highest centrality. ■

Proposition 3: The principle of indifference applies to support nodes.

Proposition 3 is a direct consequence of the properties of mixed-strategy equilibria (see for example [27]).

In light of these propositions, the node centralities should first be sorted from highest to lowest in the vector $\tilde{a} := Pa$, where P is the appropriate permutation matrix. Algorithm 1 then proceeds as follows. The counting index k is initialized to 1 and incremented for every node that is added to the mixed strategy, while σ_k is

the negated cumulative sum of reciprocal centralities.[†] These quantities are used to update the expected payoff value v_k, and the loop is repeated until either v_k is less than the next highest negated centrality value $-\tilde{a}_{k+1}$ or $k = n$. Note that if $v_k < -\tilde{a}_{k+1}$, then all remaining pure strategies are strictly dominated, *i.e.* there is no reason for the attacker to consider attacking any of the remaining nodes because it would only decrease the effectiveness of the attack. This value of v_k is in fact the saddle-point equilibrium value, and the corresponding strategies $\tilde{y}$ and $\tilde{z}$ are calculated in steps 9 and 10.

1 $k := 1$
2 $\sigma_k := \frac{1}{-\tilde{a}_1}$
3 **repeat**
4 $\quad k := k + 1$
5 $\quad \sigma_k := \sigma_{k-1} + \frac{1}{-\tilde{a}_k}$
6 $\quad v_k := \frac{k - n_d}{\sigma_k}$
7 **until** $v_k < -a_{k+1}$ *or* $k = n$
8 **foreach** $i \in \{1, \ldots, k\}$ **do**
9 $\quad \tilde{y}_i := \frac{1}{-\tilde{a}_i \sigma_k}$
10 $\quad \tilde{z}_i := 1 - \frac{v_k}{-\tilde{a}_i}$
11 **end**
12 $v^* := v_k$

Algorithm 1: Computes saddle-point equilibrium value and strategies of (2) for the case of single-node attacks.

We denote by n_d the number of nodes simultaneously defended, and the role n_d plays in the algorithm will be made clear in the proof of Theorem 2. The strategies for the original game can then easily be constructed using the inverse permutations $y := P^{-1}\tilde{y}$ and $z := P^{-1}\tilde{z}$. The following theorem confirms that this algorithm achieves the desired result for single-node attack and defense.

Theorem 1: Algorithm 1 computes the unique saddle-point equilibrium value v^* to problem (2) for single-node attack and single-node defense strategies.

Proof: We start by considering Az, which is the vector of expected payoffs for each pure strategy of the attacker. $Az =$

$$\begin{bmatrix} 0 & -a_1 & \cdots & -a_1 \\ -a_2 & 0 & \cdots & -a_2 \\ \vdots & \vdots & \ddots & \vdots \\ -a_n & -a_n & \cdots & 0 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix} = \begin{bmatrix} -a_1(1 - z_1) \\ -a_2(1 - z_2) \\ \vdots \\ -a_n(1 - z_n) \end{bmatrix},$$

where we used the fact that $\sum_{i=1}^{n} z_i = 1$. The saddle-point equilibrium mixed-strategy z^* will have support on a subset $S \subseteq \mathcal{V}$ of pure strategies. Let $\tilde{z} = Pz$ be a reordering of the vector z, where P is a permutation matrix to be determined later such that the support strategies occupy the first k^* entries of $\tilde{z}$. Using the same P matrix, let $\tilde{y} = Py$ and $\tilde{a} = Pa$. By Proposition 3, the expected payoff to the attacker against each of these support strategies must be equal:

$$\tilde{a}_1(1 - \tilde{z}_1) = \tilde{a}_2(1 - \tilde{z}_2) = \cdots = \tilde{a}_{k^*}(1 - \tilde{z}_{k^*}). \quad (4)$$

Since the summation over $\tilde{z}$ must equal one, we have

$$\sum_{i=1}^{k^*} (1 - \tilde{z}_i) = k^* - 1. \quad (5)$$

Solving for $\tilde{z}$ from the linear system of equations (4)-(5) yields:

$$\tilde{z}_i = 1 - (k^* - 1)\frac{1}{-\tilde{a}_i \sigma_{k^*}}, \qquad \text{for } i \in \{1, \ldots, k^*\},$$

where

$$\sigma_{k^*} := \sum_{j=1}^{k^*} \frac{1}{-\tilde{a}_j}.$$

For all pure strategies not in the support ($i > k^*$), $\tilde{z}_i = 0$.

Let us now write the expected payoff for a generic attack strategy $\tilde{y}$:

$$v = \tilde{y}^T \tilde{A}\tilde{z} = \tilde{y}^T \begin{bmatrix} -\tilde{a}_1(1 - \tilde{z}_1) \\ -\tilde{a}_2(1 - \tilde{z}_2) \\ \vdots \\ -\tilde{a}_{k^*}(1 - \tilde{z}_{k^*}) \\ -\tilde{a}_{k^*+1} \\ \vdots \\ -\tilde{a}_n \end{bmatrix} = \tilde{y}^T \begin{bmatrix} \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \frac{k^* - 1}{\sigma_k^*} \\ \begin{bmatrix} -\tilde{a}_{k^*+1} \\ \vdots \\ -\tilde{a}_n \end{bmatrix} \end{bmatrix}$$

By Proposition 1, all pure strategies not belonging to the support are strictly dominated. Therefore, the expected payoff of the game is $\frac{k^* - 1}{\sigma_{k^*}}$, which the attacker can achieve by targeting any combination of the k^* support nodes. Notice that this expected value $\frac{k^* - 1}{\sigma_{k^*}}$ is monotone in the centrality values a_i, following Proposition 2. That is, when choosing between adding two nodes i and j to a mixed-strategy, where $a_i > a_j$, adding node i will always decrease (improve) the expected payoff for the attacker more than node j. Therefore $\tilde{a}$ should contain the values of a sorted in decreasing value, which determines the permutation matrix P. The saddle-point equilibrium strategies for the original network are then given by $y = P^{-1}\tilde{y}$ and $z = P^{-1}\tilde{z}$.

All that remains is to compute k^*, which we can do incrementally by adding nodes in order of decreasing centrality, computing the resulting value and checking

[†]Trivially, there should be at least one node with non-zero centrality ($\tilde{a}_1 > 0$). Otherwise, the problem is degenerate.

to see of the next value is dominated. That is exactly the procedure of Algorithm 1 and thus the proof is completed. ■

B. Algorithm Design: Multiple-Node Defense

It turns out that Algorithm 1 can also be adapted to the case of multiple-node defense with only a small modification. To see how, note that the particular combination of nodes defended does not matter in the single-node attack case. All that does matter is whether a node is defended or not. Recall that the mixed strategy vector z now has length $\binom{n}{\delta}$ corresponding to the pure strategies of defending every possible combination of δ nodes, but let us define a new n-length vector $\bar{z}$, which contains the probability that each node in the network is defended. We will also define $\tilde{z} = P\bar{z}$. Note that $\bar{z}$ is *not* a mixed strategy since its values do not sum to one, but it serves the purpose of the algorithm, and later we can construct a mixed-strategy z by solving the underdetermined system of equations $Az = diag(-a)(\mathbf{1}_n - \bar{z})$.

Theorem 2: Algorithm 1 computes the unique saddle-point equilibrium value v^* to problem (2) for single-node attack and multiple-node defense strategies.

Proof: We again start by expressing the expected payoff to the attacker:

$$Az = diag(-a)(\mathbf{1}_n - \bar{z}) = \begin{bmatrix} -a_1(1-\bar{z}_1) \\ -a_2(1-\bar{z}_2) \\ \vdots \\ -a_n(1-\bar{z}_n) \end{bmatrix},$$

where $diag(-a)$ is the diagonal $n \times n$ matrix with the entries of $-a$ on the diagonal. Similar to the proof of Theorem 1, let $\tilde{z} = P\bar{z}$ and $\tilde{a} = Pa$. Once again the principle of indifference applies and we have a similar set of equations to solve:

$$\tilde{a}_1(1-\tilde{z}_1) = \tilde{a}_2(1-\tilde{z}_2) = \cdots = \tilde{a}_{k^*}(1-\tilde{z}_{k^*}). \quad (6)$$

However, the final equation given by the summation over the entries of $(1-\tilde{z})$ has changed. The summation now equals

$$\sum_{i=1}^{k^*}(1-\tilde{z}_i) = k^* - n_d, \quad (7)$$

and the resulting solution to the system of equations (6)-(7) is given by

$$\tilde{z}_i = 1 - (k^* - n_d)\frac{1}{\sigma_{k^*}\tilde{a}_i}, \quad \text{for } i \in \{1, \ldots, k^*\},$$

which is exactly what is incrementally computed in step 6 of Algorithm 1. The remainder of the proof follows as in Theorem 1. ■

With the following theorem, we show that the computational complexity of Algorithm 1 is linear in the number of nodes in the network. For comparison purposes, polynomial-time algorithms for linear programming have worst-case computational complexity in the neighborhood of $O(n^{3.5})$ [26]. The reduction in complexity is primarily a result of the monotonicity of the expected payoffs in the pure attack strategies. We also note that the computation of the algorithm itself is likely to run faster than computing and sorting the centrality measures.

Theorem 3: Algorithm 1 requires a maximum of $O(n)$ operations.

Proof: The inside of each loop in Algorithm 1 contains a fixed number of operations and is repeated a maximum of n times resulting in a total number of operations that is in the worst case $O(n)$. ■

V. Distributed Solution

We now turn our attention to the case in which there is no central administrator and the nodes must come to an agreement on the best defense strategy while communicating only with neighbors. In this case, the collection of nodes can be considered as one distributed player whose objective is to implement the saddle-point equilibrium defense policy through coordination of individual decisions. Here we require that the network performance metric be based on a centrality measure that can be computed in a distributed way. This is trivially true of strictly local measures such as degree centrality and flow centrality (local throughput), but it also turns out to be possible for betweenness and closeness centrality, though the communication costs for these will be significantly higher [28]. Since we are not aware of a distributed method for computing the centrality of node sets, we must also restrict our focus in the distributed case to single-node attacks. Finally, we restrict our attention in this section to undirected connected networks.

In this context a collective mixed-strategy can be interpreted in multiple ways. In the first interpretation, a mixed strategy prescribes probabilities that each node should be protected, but since nodes make these decisions independently, it may happen that more than δ nodes request protection, which is a violation of the resource constraint implied by the problem statement. However, since the game is likely to be repeated many times, it is reasonable to relax this constraint to a limit on the average amount of resources consumed per iteration over a large number of repeated games. Since this is a game of complete information, the extension to repeated games has no effect on the saddle-point

equilibrium and the resulting expected payoff of a long-term distributed implementation of a given strategy is equivalent to the one-shot expected payoff in the centralized case. As an alternative interpretation, one could assume a linear relationship between defense resources applied and success probability of the defense. That is, a node that is defended with half resources has a one-half probability of successfully defending an attack and similarly for arbitrary fractions of resources. The result is again mathematically equivalent to the centralized case, although it can be computed in a distributed way.

For both interpretations, the objective is a distributed method for arriving at the solution of (2). We see in Algorithm 1 that the quantities each node needs to compute are $z_i := 1 - \frac{k-n_d}{-a_i \sigma_k}$. Since we assume that n_d is known and that nodes can compute their own centrality values a_i, what remains to be computed are the values of k and σ_k. Since σ_k is simply the sum of reciprocal-centralities of the k support nodes, we propose a distributed averaging algorithm to compute $\frac{\sigma_k}{n}$ for a given set of k support nodes. The question then becomes how to determine via distributed computation which nodes are in the support. Here we can again use the monotone property of the mixed attack strategies. By Proposition 1, we have (i) if after node i computes the expected payoff v_k from σ_k, its negated centrality satisfies $-a_i > v_k$, then the pure attack strategy corresponding to node i is strictly dominated and is thus not in the support of the equilibrium mixed defense strategy, and (ii) at equilibrium there are no nodes such that $-a_i > v_k$. This suggests initializing the algorithm to include all nodes in the support and then letting them drop out of the support if they fail this criteria. The algorithm then iterates until no more nodes drop out at which point the equilibrium has been reached. This is effectively a distributed implementation of a process called *iterated elimination of strictly dominated strategies by a mixed strategy* [29].

In order to know how many nodes have dropped out after each iteration, the nodes must also be able to compute the new value of k. To update k, the nodes can simultaneously compute $\frac{k}{n}$ using distributed averaging, by setting their initial values κ_i to 0 or 1 depending on whether they have dropped out of the support at a given iteration.

Although any valid distributed averaging algorithm could be used to compute these values to sufficient accuracy in practice, convergence rates may vary, so we choose to use the finite-time distributed averaging results of [30] to ensure that each node computes the exact average in no greater than n steps. This way, the nodes can ensure the completion of each averaging computation before moving on to the next step in the algorithm.[†]

Let $x := [x_1, \ldots, x_n]^T$ denote the quantities we wish to average. The finite-time averaging procedure begins by computing up to n iterations using a standard distributed averaging algorithm:

$$x_i(t+1) = w_{ii} x_i(t) + \sum_{j \in \mathcal{N}_i} w_{ij} x_j(t), \tag{8}$$

where the weights w_{ij} are elements of a weighting matrix W. The conditions on W to guarantee convergence are that W has one simple eigenvalue at 1 and the rest strictly less than 1, with both left and right eigenvectors equal to $\mathbf{1}_n$ (the vector of length n containing all ones) [31]. It is always possible to find such a matrix W since the underlying network is undirected and connected. One standard choice that meets these requirements is the following [32]:

$$w_{ij} = \begin{cases} \frac{1}{1+\max(d_i, d_j)}, & i \neq j \\ 1 - \sum_{k \in \mathcal{N}_i} w_{ik}, & i = j \end{cases},$$

where d_i denotes the degree of node i. With these weights, each node only needs to know the degree of itself and each neighbor. The finite-time averaging technique is based on the observation that since $x(\tau) = W^\tau x(0)$ and every matrix satisfies its own minimal polynomial, $x(\tau)$ can be expressed in terms of n or fewer of its own previous values. Hence after n iterations, every node has all the information needed to compute the exact average. Let the minimal polynomial of W be given by $W^{D+1} + \alpha_D W^D + \cdots + \alpha_0 I = 0$, where D is the degree of the minimal polynomial. The expression for the exact average is then given by

$$x_i^* = \frac{\begin{bmatrix} x_i(D) & x_i(D-1) & \cdots & x_i(0) \end{bmatrix} s}{\mathbf{1}_{D+1}}, \tag{9}$$

where

$$s := \begin{bmatrix} 1 & 1+\alpha_D & \cdots & 1+\sum_{j=1}^{D} \alpha_j \end{bmatrix}^T.$$

We refer the reader to [30] for a derivation of this result.

The complete distributed computation procedure to be synchronously performed by each node $i \in \mathcal{V}$ such that $a_i > 0$ is shown in Algorithm 2.

Theorem 4: Algorithm 2 computes the unique saddle-point equilibrium value v^* to problem (2) for single-node attack and multiple-node defense strategies.

[†]This method requires that the network topology remains fixed during the period of computation, which is reasonable provided that the communication rates are much faster than the rate of change to the links in the network.

$\kappa_i(0) := 1$
$k_{\text{prev}} := n$
repeat
  $x_i(0) := \frac{1}{-a_i}$
  Compute $x_i(1), \ldots, x_i(D)$ using (8)
  Compute $\kappa_i(1), \ldots, \kappa_i(D)$ using (8)
  Solve for x_i^* and κ_i^* using (9)
  $k := n\kappa_i^*$
  $\sigma_k := nx_i^*$
  $v_k := \frac{k-n_d}{\sigma_k}$
  if $-a_i > v_k$ **then**
    $\kappa_i(0) := 0$
  end
  $\Delta_k := k_{\text{prev}} - k$
  $k_{\text{prev}} := k$
until $\Delta_k = 0$
$y_i := \frac{1}{-a_i \sigma_k}$
$z_i := 1 - \frac{v_k}{-a_i}$
$v^* := v_k$

Algorithm 2: Distributed algorithm to compute saddle-point equilibrium value and strategies of (2).

Proof: Since we already proved in Theorem 2 that y and z are a saddle-point equilibrium strategy pair given the correct value of k^*, it suffices to show that Algorithm 2 indeed computes the correct value of k^*. Since k^* corresponds to the smallest k for which no strategies in the support are strictly dominated, we need only show that for any $i \in \mathcal{V}$ and $k \geq k^*$, if $-a_i > v_k$ then $i \notin \mathcal{S}$. This follows directly from Proposition 1. Now, since we start with $k = n$ and k is monotone decreasing, we will always obtain the correct value of k, and the proof is completed. ∎

Remark 1: In the centralized algorithm, nodes are added to the support set one at a time, while in the decentralized algorithm, multiple nodes can drop out simultaneously. The result is that when there are many nodes in the support set, the decentralized algorithm is likely to terminate in fewer iterations than the centralized algorithm, and indeed that is what we observe in simulation. However, since the decentralized algorithm includes an extra averaging computation for k, it will generally lead to longer total computation times.

VI. Multiple-node attacks

The case of simultaneous attacks to multiple nodes in the network adds additional complexity to the problem because the specific combination of nodes attacked becomes more important. We can no longer consider the removal of nodes from the network in isolation, but rather must capture the inter-dependencies of nodes on network performance through the concept of node-set centrality, introduced in Section II-A. Although the resulting problem (2) remains computable by linear programming, the size of the payoff matrix grows combinatorially in the number of nodes attacked and defended. As an example, the payoff matrix for 5-node attack and defense strategies on a 50 node network contains $\binom{50}{5} \times \binom{50}{5}$ (over 2 million $\times$ 2 million) entries. Unfortunately, we no longer have monotonicity of the mixed-strategies with respect to node-set centrality and therefore we cannot use the methods of Algorithm 1 to simplify and solve this problem exactly.

However, one can approximate the solution to the problem while greatly reducing the computation. The key observation is that similar to the single-node attack case, it is likely that in most practical networks only a few of the large number of possible attack combinations will play a role in the equilibrium solution. If we could somehow identify the most effective attacks without computing the entire payoff matrix, we could save a substantial amount of computation. The method we propose in the next section computes the most effective *sequences* of attacks in order to predict the most effective *parallel* attack strategies from the full payoff matrix.

Remark 2: In general, it may not be known exactly how many nodes will be attacked. If this is the case, the following algorithm can be run several times for different numbers of attack nodes and the results can be weighted based on a probability distribution on the expected number of nodes that will be attacked.

A. Sequential Approximation Algorithm

Let $\mathcal{A}^k \subseteq \mathcal{V}$ denote a set of k nodes targeted by an attack, and let $\bar{\mathcal{A}}^k$ denote a set of k-node attacks. Denote by $\mathcal{S}_\delta(\mathcal{A}^k)$ the set of support nodes resulting from a single-node attack to a network in which the nodes $\mathcal{A}^k$ have been removed, and δ nodes are defended. The sequential approximation (Algorithm 3) works by iteratively building $(k+1)$-node attack sets from k-node attack sets. For each attack $\mathcal{A}^k$ in the k-node attack set $\bar{\mathcal{A}}^k$, the nodes $\mathcal{A}^k$ are removed from the network, and the support nodes for the resulting single-node attack $\mathcal{S}_{\delta-k+b}(\mathcal{A}^k)$ are computed. Then, attacks containing the original nodes $\mathcal{A}^k$ plus the new nodes are added to the $(k+1)$-node attack set $\bar{\mathcal{A}}^{k+1}$. The nonnegative integer b is a free parameter in the algorithm; increasing b may reduce the approximation error by expanding the set of nodes added to the candidate attack sets, but this comes at the cost of increased computation time.

$\bar{\mathcal{A}}^0 := \{\emptyset\}$
for $k := 0$ *to* $\alpha - 1$ **do**
  foreach $\mathcal{A}^k \in \bar{\mathcal{A}}^k$ **do**
    Compute $\mathcal{S}_{\delta-k+b}(\mathcal{A}^k)$ using Algorithm 1
    foreach $v \in \mathcal{S}_{\delta-k+b}(\mathcal{A}^k)$ **do**
      $\bar{\mathcal{A}}^{k+1} := \bar{\mathcal{A}}^{k+1} \cup \{\mathcal{A}^k \cup \{v\}\}$
    end
  end
end
$\hat{v} := \min_{y \in \Delta_{\mathcal{A}^\alpha}} \max_{z \in \Delta_{\mathcal{A}^\alpha}} y^T A z$

Algorithm 3: Algorithm to approximate saddle-point equilibrium value and strategies of (2) for the case of multiple-node attacks based on a sequential simplification.

B. Simulations

It is quite difficult to derive analytical measures for how closely Algorithm 3 approximates the saddle-point equilibrium defense strategy in general, but we can test it on a model of a real network as well as a range of randomly generated networks and compare to other potential defense strategies, including the exact solution of (2) provided the networks are small enough. We begin by testing the approach on connectivity data of the UCSB MeshNet, a real experimental wireless mesh network dataset that was also used in [8].

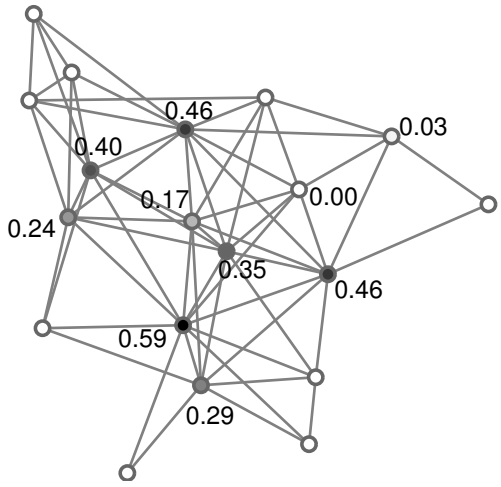

Fig. 2. Results of Algorithm 3 applied to the UCSB MeshNet data from a particular instant of time. The shading of the nodes is proportional to the adjacent numbers which are the probabilities that each node will be defended as part of a three-node mixed defense strategy against a three-node attack.

Fig. 2 shows the network along with the saddle-point-equilibrium mixed-defense strategy for three-node defense against three-node attack. We compared the sequential approximation to two other defense strategies: (1) a *pure strategy*, which defends the three nodes having the highest betweenness centrality (equivalent to the analysis in [8]), and (2) *random sampling*, which samples the payoff matrix by randomly choosing 20% of the possible node combinations and solving the game on the resulting reduced payoff matrix. The random sampling approach is inspired by [33] and has the advantage of approximation guarantees assuming the attacker is also using random sampling. However, in order to make an unbiased comparison between strategies, we assume the attacker plays the saddle-point equilibrium strategy against whatever strategy the defender is using. For the pure strategy, this means the attacker knows the defender will defend the three most central nodes and thus will not attack those nodes, while for the sequential approximation, this means the attacker knows which columns of the payoff matrix have been selected by the defender. Table I shows the expected payoffs for each of these defense strategies as well as the sequential approximation and the true saddle-point equilibrium strategy. Notice that the attacker playing against the sequential approximation can still not improve on the saddle-point equilibrium payoff despite having a significant advantage.

TABLE I
MULTIPLE-NODE ATTACKS TO UCSB MESHNET

Defense strategy	Nodes attacked/defended		
	1	2	3
Saddle-Point Equil.	-0.171	-0.273	-0.342
Sequential Approx.	-0.171	-0.273	-0.342
Random Sampling	-0.193	-0.279	-0.349
Pure Strategy	-0.232	-0.402	-0.539

Next, we tested the accuracy of Algorithm 3 against the two other approaches for a two-node attack and defense scenario on 100 random geometric networks of increasing size, generated by randomly placing n nodes the unit square and connecting any pair of nodes that are closer than a distance threshold of 0.4 from each other. Fig. 3 shows the expected fraction of performance loss ($\frac{\hat{v}-v^*}{v^*}$) of the three approaches compared to the equilibrium payoff, and Fig. 4 compares the average time required to compute the sequential approximation to that for the full payoff matrix and linear programming (LP) solution. We see that the sequential approximation comes quite close to the exact solution with a relatively flat computation time compared to the LP.

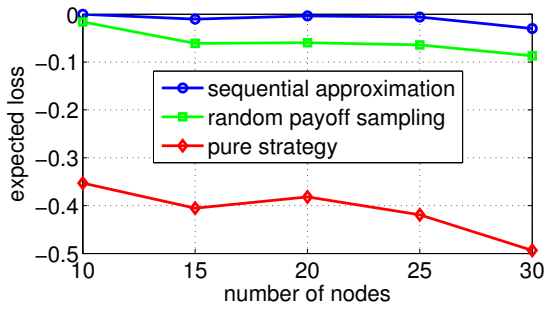

Fig. 3. Mean fraction of expected performance loss of the three approximation algorithms on five sizes of random geometric networks.

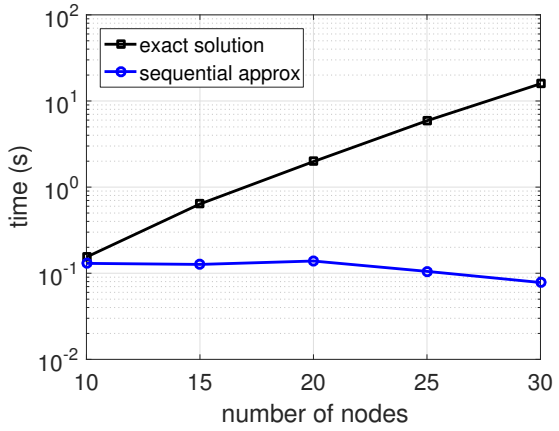

Fig. 4. Mean computation times of the sequential approximation compared to solving the exact solution via linear programming.

VII. Conclusions and Future Work

The main contributions of this work are fast algorithms for allocating defense resources to the nodes of a network with the goal of maximizing one of several suitable centrality-based performance metrics in the presence of attacks. For single-node attacks, we presented both centralized and distributed algorithms to compute the saddle-point equilibrium defense policies against worst-case attacks. The case of simultaneous attacks to multiple nodes is computationally complex, so we proposed a centralized approximation algorithm based on a sequential simplification. Simulations demonstrated that this approach very closely approximates the equilibrium strategies for a variety of cases at greatly reduced computation. Interesting directions for future research include extending the results to the case when links are attacked in place of or in addition to nodes, developing distributed solutions to the multiple-node attack case, and investigating the effect of adversarial nodes in the distributed case.

References

[1] A. A. Cardenas, S. Amin, and S. Sastry, "Secure control: Towards survivable cyber-physical systems," in *The 28th International Conference on Distributed Computing Systems Workshops*. IEEE, 2008, pp. 495–500.

[2] A. Teixeira, I. Shames, H. Sandberg, and K. H. Johansson, "A secure control framework for resource-limited adversaries," *Automatica*, vol. 51, pp. 135–148, 2015.

[3] J. P. Sterbenz, D. Hutchison, E. K. Çetinkaya, A. Jabbar, J. P. Rohrer, M. Schöller, and P. Smith, "Resilience and survivability in communication networks: Strategies, principles, and survey of disciplines," *Computer Networks*, vol. 54, no. 8, pp. 1245–1265, 2010.

[4] P. Smith, D. Hutchison, J. P. Sterbenz, M. Schöller, A. Fessi, M. Karaliopoulos, C. Lac, and B. Plattner, "Network resilience: a systematic approach," *IEEE Communications Magazine*, vol. 49, no. 7, pp. 88–97, 2011.

[5] G. Como, K. Savla, D. Acemoglu, M. A. Dahleh, and E. Frazzoli, "Robust distributed routing in dynamical networkspart i: Locally responsive policies and weak resilience," *IEEE Transactions on Automatic Control*, vol. 58, no. 2, pp. 317–332, 2013.

[6] A. Bavelas, "A mathematical model for group structures," *Human organization*, vol. 7, no. 3, pp. 16–30, 1948.

[7] L. C. Freeman, "Centrality in social networks conceptual clarification," *Social networks*, vol. 1, no. 3, pp. 215–239, 1978.

[8] M. Kas, S. Appala, C. Wang, K. M. Carley, L. R. Carley, and O. K. Tonguz, "What if wireless routers were social?" *Wireless Communications, IEEE*, vol. 19, no. 6, pp. 36–43, 2012.

[9] P. Holme, B. J. Kim, C. N. Yoon, and S. K. Han, "Attack vulnerability of complex networks," *Physical Review E*, vol. 65, no. 5, p. 056109, 2002.

[10] Z. Han, D. Niyato, W. Saad, and A. Hjørungnes, *Game theory in wireless and communication networks: theory, models, and applications*. Cambridge University Press, 2011.

[11] S. Roy, C. Ellis, S. Shiva, D. Dasgupta, V. Shandilya, and Q. Wu, "A survey of game theory as applied to network security," in *System Sciences (HICSS), 2010 43rd Hawaii International Conference on*. IEEE, 2010, pp. 1–10.

[12] K. G. Vamvoudakis, J. P. Hespanha, B. Sinopoli, and Y. Mo, "Adversarial detection as a zero-sum game," in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*. IEEE, 2012, pp. 7133–7138.

[13] E. Altman, K. Avrachenkov, and A. Garnaev, "Jamming in wireless networks: The case of several jammers," in *Game Theory for Networks, 2009. GameNets' 09. International Conference on*. IEEE, 2009, pp. 585–592.

[14] T. Basar, "The gaussian test channel with an intelligent jammer," *IEEE Transactions on Information Theory*, vol. 29, no. 1, pp. 152–157, 1983.

[15] A. Kashyap, T. Basar, and R. Srikant, "Correlated jamming on mimo gaussian fading channels," *IEEE Transactions on Information Theory*, vol. 50, no. 9, pp. 2119–2123, 2004.

[16] S. Arunmozhi and Y. Venkataramani, "A new defense scheme against DDoS attack in mobile ad hoc networks," in *Advanced Computing*. Springer, 2011, pp. 210–216.

[17] J. R. Riehl and M. Cao, "Optimal defensive resource allocation for a centrality-based security game on multi-hop networks," in *Decision and Control (CDC), 2015 IEEE 54th Annual Conference on*. IEEE, 2015, pp. 6257–6262.

[18] 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*, vol. 94, no. 3, pp. 491–502, 2004.

[19] J. C. Smith and C. Lim, "Algorithms for network interdiction and fortification games," in *Pareto optimality, game theory and equilibria*. Springer, 2008, pp. 609–644.

[20] H. Sreekumaran, A. R. Hota, A. L. Liu, N. A. Uhan, and S. Sundaram, "Multi-agent decentralized network interdiction games," *arXiv preprint arXiv:1503.01100*, 2015.
[21] S. Bhattacharya, A. Khanafer, and T. Başar, "Power allocation in team jamming games in wireless ad hoc networks," in *Proceedings of the 5th International ICST Conference on Performance Evaluation Methodologies and Tools*. ICST, 2011, pp. 515–524.
[22] A. Nochenson and C. L. Heimann, "Simulation and game-theoretic analysis of an attacker-defender game," in *Decision and Game Theory for Security*. Springer, 2012, pp. 138–151.
[23] S. Goyal and A. Vigier, "Attack, defence, and contagion in networks," *The Review of Economic Studies*, vol. 81, no. 4, pp. 1518–1542, 2014.
[24] G. C. Chasparis and J. S. Shamma, "Linear-programming-based multi-vehicle path planning with adversaries," in *American Control Conference*. IEEE, 2005, pp. 1072–1077.
[25] T. Başar and G. J. Olsder, *Dynamic Noncooperative Game Theory, 2nd Edition*. SIAM, 1999.
[26] G. Strang, "Karmarkars algorithm and its place in applied mathematics," *The Mathematical Intelligencer*, vol. 9, no. 2, pp. 4–10, 1987.
[27] T. S. Ferguson. Game theory, second edition. [Online]. Available: http://www.math.ucla.edu/tom/Game_Theory/Contents.html
[28] K. A. Lehmann and M. Kaufmann, "Decentralized algorithms for evaluating centrality in complex networks," Tech. Rep. WSI-2003-10, October 2007.
[29] T. Fujiwara-Greve, *Non-Cooperative Game Theory*. Springer, 2015, vol. 1.
[30] S. Sundaram and C. N. Hadjicostis, "Finite-time distributed consensus in graphs with time-invariant topologies," in *American Control Conference*. IEEE, 2007, pp. 711–716.
[31] P. Denantes, F. Bénézit, P. Thiran, and M. Vetterli, "Which distributed averaging algorithm should I choose for my sensor network?" in *INFOCOM 2008: The 27th Conference on Computer Communications*. IEEE, 2008.
[32] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The journal of chemical physics*, vol. 21, no. 6, pp. 1087–1092, 1953.
[33] S. D. Bopardikar, A. Borri, J. P. Hespanha, M. Prandini, and M. D. Di Benedetto, "Randomized sampling for large zero-sum games," *Automatica*, 2013.