Connected Subgraph Defense Games*

Eleni C. Akrida[1], Argyrios Deligkas[1,2],
Themistoklis Melissourgos[1], and Paul G. Spirakis[1,3]

[1] Department of Computer Science, University of Liverpool, UK
{E.Akrida,Argyrios.Deligkas,T.Melissourgos,P.Spirakis}@liverpool.ac.uk
[2] Leverhulme Research Centre for Functional Materials Design, Liverpool, UK
[3] Computer Engineering and Informatics Department, University of Patras, Greece

Abstract. We study a security game over a network played between a *defender* and k *attackers*. Every attacker chooses, probabilistically, a node of the network to damage. The defender chooses, probabilistically as well, a connected induced subgraph of the network of λ nodes to scan and clean. Each attacker wishes to maximize the probability of escaping her cleaning by the defender. On the other hand, the goal of the defender is to maximize the expected number of attackers that she catches. This game is a generalization of the model from the seminal paper of Mavronicolas et al. [11]. We are interested in Nash equilibria of this game, as well as in characterizing *defense-optimal* networks which allow for the best *equilibrium defense ratio*, termed *Price of Defense*; this is the ratio of k over the expected number of attackers that the defender catches in equilibrium. We provide characterizations of the Nash equilibria of this game and defense-optimal networks. This allows us to show that the equilibria of the game coincide independently from the coordination or not of the attackers. In addition, we give an algorithm for computing Nash equilibria. Our algorithm requires exponential time in the worst case, but it is polynomial-time for λ constantly close to 1 or n. For the special case of tree-networks, we further refine our characterization which allows us to derive a polynomial-time algorithm for deciding whether a tree is defense-optimal and if this is the case it computes a defense-optimal Nash equilibrium. On the other hand, we prove that it is NP-hard to find a best-defense strategy if the tree is not defense-optimal. We complement this negative result with a polynomial-time constant-approximation algorithm that computes solutions that are close to optimal ones for general graphs. Finally, we provide asymptotically (almost) tight bounds for the Price of Defense for any λ.

Keywords: Defense games · Defense ratio · Defense-optimal.

1 Introduction

With technology becoming a ubiquitous and integral part of our lives, we find ourselves using several different types of "computer" networks. An important issue when dealing with such networks, which are often prone to security breaches [6], is to prevent and monitor unauthorized access and misuse of the network or its accessible resources. Therefore, the study of network security has attracted a lot of attention over the years [18]. Unfortunately, such breaches are often inevitable, since some parts of a large system are expected to have weaknesses that expose them to security attacks; history has indeed shown several successful and highly-publicized such incidents [17]. Therefore, the challenge for someone trying to keep those systems and networks of computers secure is to counteract these attacks as efficiently as possible, once they occur.

To that end, inventing and studying appropriate theoretical models that capture the essence of the problem is an important line of research, ongoing for a few years now [13,14]. Here, extending some known models for very simple cases of attacks and defenses [11,12], we introduce and analyze a more general model for a scenario of network attacks and defenses modeling it as a *defense game*.

The Network Security Game. We follow the terminology established by the seminal paper of Mavronicolas et al. [12]. We consider a network whose nodes are vulnerable to infection by threats

*This work was supported by the NeST initiative of the EEE/CS School of the University of Liverpool and by the EPSRC grant EP/P02002X/1.

called *attackers*; think of those as viruses, worms, Trojan horses or eavesdroppers [7] infecting the components of a computer network. Available to the network is a security software (or firewall), called the *defender*. The defender is only able to "clean" a limited part of the network from threats that occur; the reason for the limited cleaning capacity of the defender may be, for example, the cost of purchasing a global security software. The defender seeks to protect the network as much as possible, and on the other hand, every attacker seeks to increase the likelihood of not being caught. Both the attackers and the defender make individual decisions for their positioning in the network with the aim to maximize their own objectives.

Every attacker targets (and attacks) a node chosen via her own probability distribution over the nodes of the network. The defender cleans a connected induced subgraph of the network with size λ, chosen via her own probability distribution over all connected induced subgraphs of the graph with λ nodes. The attack of a particular attacker is successful unless the node chosen by the attacker is incident to an edge (link) being cleaned by the defender, i.e. to an edge belonging in the induced subgraph chosen by the defender. One could equivalently think of the defender selecting a set of λ connected nodes to defend, and an attacker is successful if and only if she attacks a node that is not being defended. Since attacks and defenses over a large computer network are self-interested procedures that seek to maximize damage and protection, respectively, it is natural to model this network security scenario as a non-cooperative *strategic game* on graphs with two kinds of players: $k \geq 1$ *attackers*, each playing a *vertex* of the graph, and a single *defender* playing a *connected induced subgraph* of the graph. The *(expected) payoff* of an attacker is the probability that she is not caught by the defender; the *(expected) payoff* of the defender is the (expected) number of attackers she catches. We are interested in the Nash equilibria [15, 16] associated with this graph theoretic game, where no player can unilaterally improve her (expected) payoff by switching to another probability distribution. We are also interested in understanding and characterizing the networks that allow for a good *defense ratio*: given a strategy profile, i.e. a combination of strategies for the network entities (attackers and defender), the defense ratio of a network is the ratio of the total number of attackers over the defender's expected payoff in that strategy profile.

1.1 Our results

In this paper we depart from and significantly extend the line of work of Mavronicolas et al. in their seminal paper [12] on defense games in graphs; we term the type of games we consider *CSD games*. In our model the defender is more powerful than in [12], since her power is parameterized by the size, λ, of the defended part of the network. We allow λ to take values from 1 to n, while in [12] only the case where $\lambda = 2$ was studied. We study many questions related to CSD games. We extend the notions of *defense ratio* and *defense-optimal graphs* for CSD games. In fact, the defense ratio of a given graph G and a given strategy profile S of the attackers and the defender is the ratio of the number of attackers, k, over the defender's expected payoff (the number of attackers she catches on expectation). We thoroughly investigate the notion of the defense ratio for Nash equilibria strategy profiles.

Firstly, we precisely characterize the Nash equilibria and defense-optimal graphs in CSD games. This allows us to show that, in equilibrium, the game version of k uncoordinated attackers and a single defender is equivalent to the version in which a single leader coordinates the k attackers, meaning that both versions of the game have the same defense ratio. We present an LP-based algorithm to compute an exact equilibrium of any given CSD game, whose running time is polynomial in $\binom{n}{\lambda}$. Then, we focus on tree-graphs. There, we further refine our equilirbium characterization which allows us to derive a polynomial-time algorithm for deciding whether a tree is defense-optimal and, if this is the case, it computes a defense-optimal Nash equilibrium. A tree is defense-optimal if and only if it can be partitioned into $\frac{n}{\lambda}$ disjoint sub-trees. On the other hand, we prove that it is NP-hard to find a best-defense strategy if the tree is not defense-optimal. We remark that a very crucial parameter for defense-optimality of a graph G is the "best" probability with which any vertex of G is defended in a NE; we call that probability *MaxMin probability* and denote it by $p^*(G)$. Then, for any graph G, the defense ratio in equilibrium is shown to be exactly $\frac{1}{p^*(G)}$. Although it is hard to exactly compute $p^*(G)$ even for trees, we complement this negative result with a polynomial-time constant-approximation algorithm that computes solutions that are close to the optimal ones for any λ, for any given general graph. In particular, we approximate the (best) defense ratio of any

graph within a factor of $2 + \frac{\lambda-3}{n}$. Finally, we provide asymptotically tight bounds for the Price of Defense for any $\lambda \in \omega(1) \cap o(n)$, and almost tight bounds for any other value of λ.

For detailed proofs and auxiliary figures of the results presented, we refer the reader to the full version of the paper [1].

1.2 Related work

Our graph-theoretic game is a direct generalization of the defense game considered by Mavronicolas et al. [11,12]. In the latter, the authors examined the case where the size of the defended part of the network is $\lambda = 2$, i.e. where the defender "cleans" an edge. This lead to a nice connection between equilibria and (fractional) matchings in the graph [13]. But when λ is greater than 2, one has to investigate (as we shall see here) how to sparsely cover the graph by as small a number as possible of connected induced subgraphs of size λ. This direction can be seen as an extension of fractional matchings to covers of the graph by equisized connected subgraphs. Sparse covering of graphs by connected induced subgraphs (clusters), not necessarily equisized, is a notion known to be useful also for distributed algorithms, since it affects message communication complexity [5].

In another line of work, Kearns and Ortiz [9] study *Interdependent Security games* in which a large number of players must make individual decisions regarding security. Each player's *safety* may depend on the actions of the entire population (in a complex way). The graph-theoretic game that we consider could be seen as a particular instance of such games with some sort of limited interdependence: the actions of the defender and an attacker are interdependent, while the actions of the attackers are not dependent on each other.

Aspnes et al. [4] consider a graph-theoretic game that models containment of the spread of viruses on a network; each node individually must choose to either install anti-virus software at some cost, or risk infection if a virus reaches it without being stopped by some intermediate node with installed anti-virus software. Aspnes et al. [4] prove several algorithmic properties for their graph-theoretic game and establish connections to a certain graph-theoretic problem called *Sum-of-Squares Partition*.

A game on a weighted graph with two players, the *tree player* and the *edge player*, was studied by Alon et al. [2]. At each play, the tree player chooses a spanning tree and the edge player chooses an edge of the graph, and the payoffs of the players depend on whether the chosen edge belongs in the spanning tree. Alon et al. investigate the theoretical aspects of the above game and its connections to the *k-server problem* and *network design*.

Finally, there is a long line of work on security games [3] where many scenarios are modelled using graph theoretic problems [8, 10, 19, 20].

2 Preliminaries

The game. A *Connected-Subgraph Defense (CSD) game* is defined by a graph $G = (V, E)$, a *defender*, $k \geq 1$ *attackers*, and a positive integer λ. Throughout the paper, λ is considered to be a *given* parameter of the game. A pure strategy for the defender is any induced connected subgraph H of G with λ vertices, which we term *λ-subgraph*. For any λ-subgraph H of G we denote $V(H)$ its set of vertices. Since $V(H)$ uniquely defines an induced subgraph of G, we will use the term λ-subgraph to denote either $V(H)$ or H. The *action set* of the defender is $D := \{V(H) | H \text{ is a } \lambda\text{-subgraph of } G\}$ and we will denote its cardinality by θ, i.e. $\theta := |D|$. For ease of presentation, we will also refer to D as $[\theta] := \{1, 2, \ldots, \theta\}$. A pure strategy for each of the attackers is any vertex of G. So, the action set of every attacker is V, the vertex set of G; we denote $n := |V|$ and we similarly refer to V also as $[n]$.

To play the game, the defender chooses a *defense (mixed) strategy*, i.e. a probability distribution over her action set, and each attacker chooses an *attack (mixed) strategy*, i.e. a probability distribution over the vertices of G. We denote a strategy by $s := (s_1, \ldots, s_d) \in \Delta_d$, i.e. by the probability distribution over d enumerated pure strategies, where $\Delta_d := \{x_1, \ldots, x_d \geq 0 | \sum_{i=1}^{d} x_i = 1\}$ is the $(d-1)$-unit simplex. In a defense strategy $q \in \Delta_\theta$ each pure strategy $j \in [\theta]$ is assigned a probability q_j.

We say that a pure strategy of the defender, i.e. a specific λ-subgraph H of G, *covers* a vertex $v \in V$ if $v \in V(H)$. A defense strategy covers a vertex $v \in V$ if it assigns strictly positive probability to at least one λ-subgraph H of G which contains v.

Definition 1 (Vertex Probability). *The* vertex probability p_i *of vertex* $i \in [n]$*, is the probability that* i *will be covered, formally* $p_i := \sum_{j \in [\theta]:\ i \in j} q_j$.

The *support* of a strategy s, denoted by $\text{supp}(s)$, is the subset of the action set that is assigned strictly positive probability.

Payoffs and Strategy profiles. A *strategy profile* is a tuple of strategies $S = (q, t_1, \ldots, t_k)$, where q denotes the defender's strategy and t_j denotes the j-th attacker's strategy, $j \in [k]$. A strategy profile is pure if the support of every strategy has size one. The *payoff* of every attacker is 1 in any pure strategy profile where she does not choose a defended vertex, and 0 in all the rest. The payoff of the defender in a pure strategy profile where she defends $V(H)$, is the number of attackers that choose a vertex in $V(H)$. Under a strategy profile, the *expected payoff* of the defender is the expected number of attackers that she catches, which we call *defense value*, and the expected payoff of the attacker is the probability that she will not get caught. A *best response* strategy for a participant is a strategy that maximizes her expected payoff, given that the strategies of the rest of the participants are fixed. A *Nash equilibrium* is a strategy profile where all the participants are playing a best response strategy. In other words, neither the defender nor any of the attackers can increase their expected payoff by unilaterally changing their strategy.

Definition 2 (Defense Ratio). *For a given graph* G *we define a measure of the quality of a strategy profile* S*, called* defense ratio of G *and denoted* $DR(G, S)$*, as the ratio of the total number of attackers* k *over the defense value.*

In this work we are only interested in the cases where S is an equilibrium. For a given graph, when in equilibrium, the defender's expected payoff is unique (due to Corollary 1 (a)) and achieves the *equilibrium defense ratio* $\text{DR}(G, S^*)$, where S^* is an equilibrium. The defense strategy in S^* which achieves this defense ratio will be termed *best-defense strategy.*

Definition 3 (MaxMin Probability, p^*). *We call* MaxMin Probability *of a graph* G *the maximum, over all defense strategies, minimum vertex probability in* G*, that is:*

$$p^*(G) := \max_{q \in \Delta_\theta} \min_{i \in [n]} p_i.$$

As we will show in Lemma 1, the equilibrium defense ratio of a graph G turns out to be $\text{DR}(G, S^*) = 1/p^*(G)$.

Definition 4 (Price of Defense). *The* Price of Defense, PoD*, for a given parameter* λ *of the game, is the worst defense ratio, over all graphs, achievable in equilibrium, that is:*

$$\text{PoD}(\lambda) = \max_G \text{DR}(G, S^*).$$

Definition 5 (Defense-Optimal Graph). *For a given* λ*, a graph* G^* *that achieves the minimum equilibrium defense ratio over all graphs, i.e.* $G^* \in \arg\min_G \text{DR}(G, S^*)$*, is called* defense-optimal graph.

In the following, for ease of presentation, whenever we refer to defense optimality, we implicitly assume that λ has a fixed value.

3 Nash equilibria

In this section, we provide a characterization of Nash equilibria in CSD games, as well as important properties of their structure which prove useful for the development of our algorithms in the remainder of the paper.

Theorem 1 (Equilibrium characterization). *For a given graph G, in any equilibrium with support $S \subseteq [\theta]$ of the defender and support $T_j \subseteq [n]$ of each attacker $j \in [k]$, the following conditions are necessary and sufficient:*

1. $\min_{i\in[n]} p_i$ *is maximized over all defense strategies, and*
2. $\bigcup_{j\in[k]} T_j \subseteq V^*$, *where* $V^* := \{i \in [n] \mid \min_{i\in[n]} p_i$ *is maximized over all defense strategies*$\}$, *and*
3. *every $s \in S$ has the maximum expected total number of attackers on its vertices over all pure strategies.*

Lemma 1. *For any given graph G, the equilibrium defense ratio is* $\mathrm{DR}(G, S^*) = \frac{1}{p^*(G)}$, *where* $p^*(G) := \max_{q\in\Delta_\theta} \min_{i\in[n]} p_i$ *and S^* is an equilibrium.*

Proof. By Theorem 1, in an equilibrium, every attacker will have in her support only vertices that are defended with probability exactly $p^*(G)$. Therefore, the expected number of attackers that the defender catches is $p^*(G) \cdot k$. By definition of the defense ratio, $\mathrm{DR}(G, S^*) = \frac{k}{p^*(G)\cdot k} = \frac{1}{p^*(G)}$. □

Corollary 1. *The following hold:*

(a) For a given graph G, in any equilibrium, the expected payoff of the defender and each attacker is unique.
(b) For a given graph G, in any equilibrium with support $S \subseteq [\theta]$ of the defender, for every $s \in S$ there exists a vertex $v \in s$ such that $p_v = p^*(G)$.
(c) In any CSD game on a graph G, the problem of finding the equilibrium defense ratio (or equivalently, $p^*(G)$) for $k \geq 2$ attackers reduces to the same problem in the game with $k = 1$ attacker, which is a two-player constant-sum game.

Proof. (a) By Theorem 1, in an equilibrium the defender chooses a strategy that induces probability $p^*(G)$ to some vertex of G (Condition 1). Also, each of the attackers has in her support T only vertices with vertex probability $p^*(G)$. Therefore, all attackers attack only such vertices and the expected payoff of the defender is $k \cdot p^*(G)$. Consider also an attacker with strategy $t = (t_1, t_2, \ldots, t_n)$. Her expected payoff is $\sum_{i\in[n]} t_i(1 - p_i)$, where p_i is the vertex probability of vertex i. This value is equal to $\sum_{i\in T} t_i(1 - p^*(G)) = 1 - p^*(G)$. Since $p^*(G)$ is unique for a graph G, the expected payoffs of the defender and each attacker is unique.

(b) The proof is by contradiction. Consider an equilibrium where the defender's strategy is $q \in [\theta]$ with support S, and there exists a pure strategy $s \in S$ for which every vertex $v \in s$ has $p_v > p^*(G)$. By Condition 2 of Theorem 1, no attacker has in her support a vertex in s. Therefore, the defender can strictly increase her expected payoff by moving all her probability $q_s > 0$ from s to some other pure strategy s' that contains a vertex which is in the support of some attacker.

(c) Observe that for any given graph G, the quantity $p^*(G)$, by definition, only depends on the graph and not the number of attackers k. That is, $p^*(G)$ is the same for every $k \geq 1$. Lemma 1 states that in any equilibrium S^*, it is $\mathrm{DR}(G, S^*) = \frac{1}{p^*(G)}$, therefore the defense ratio in an equilibrium does not depend on k. This means that when we are given G and we are interested in the equilibrium defense ratio, we might as well consider the game with the single defender and a single attacker. By definition of the game (see Section 2) the latter is a two-player constant-sum game. □

The following corollary implies that coordination (resp. individual selfishness) of the attackers cannot increase the attackers' (resp. defender's) expected payoff in equilibrium.

Corollary 2. *Every equilibrium with uncoordinated attackers (i.e. as described in Section 2) is an equilibrium with coordinated (i.e. centrally controlled) attackers, and vice versa.*

The following theorem provides an algorithm for computing an equilibrium for any CSD game, whose running time is polynomial in n when $\lambda = c$ or $\lambda = n - c$, where c is a constant natural.

Theorem 2. *For some given graph G and parameter λ, there is an algorithm that computes $p^*(G)$ and also finds an equilibrium in time polynomial in $\binom{n}{\lambda}$.*

Proof. Given a graph G, the number of attackers $k \geq 1$, and some $\lambda \in \{1, 2, \ldots, n\}$, the action set D of the defender is constructed by the vertex sets of at most $\binom{n}{\lambda}$ λ-subgraphs, so for D's cardinality θ it holds that $\theta \leq \binom{n}{\lambda}$. Consider now the mixed strategy $q \in \Delta_\theta$ of the defender, where each pure strategy $j \in [\theta]$ is assigned probability q_j. Consider also the vertex probability p_i for each vertex $i \in [n]$. According to Corollary 1 (a) and (c), the unique $p^*(G)$ in the case of a single attacker can be used to derive an equilibrium for the case of $k \geq 2$ attackers. Therefore, we will find $p^*(G)$ for a single attacker, find an equilibrium for that case, and then extend this equilibrium to one in the case of $k \geq 2$ attackers. In more detail, after we find the defense strategy q^* that maximizes $\min_{i \in [n]} p_i$ (Condition 1 of Theorem 1), i.e. yields $p^*(G)$ on the set $V^* := \arg\max_{q \in \Delta_\theta} \min_{i \in [n]} p_i$, an equilibrium is achieved if the single attacker assigns probability $1/|V^*|$ to each vertex of V^*; that is because all conditions of Theorem 1 are satisfied. Then, an equilibrium for $k \geq 2$ is achieved if every attacker plays the same strategy as the single attacker; that is because again all conditions of Theorem 1 are satisfied.

The crucial observation that allows us to design such an algorithm is that we can compute $p^*(G)$ via a Linear Program which has $O\left(\binom{n}{\lambda}\right)$ many variables and $O(n)$ constraints, and therefore its running time is in the worst case polynomial in $\binom{n}{\lambda}$, for $\lambda \in \{2, 3, \ldots, n-1\}$. For the trivial cases $\lambda = 1$ and $\lambda = n$, $D = \{\{i\} | i \in V\}$ and $D = V$ respectively, therefore $p^*(G) = 1/n$ and $p^*(G) = 1$ respectively. So in the rest of the proof we will imply that $\lambda \in \{2, 3 \ldots, n-1\}$. It remains to show how $p^*(G)$ is computed.

Let us denote $p^* := p^*(G) := \max_{q \in \Delta_\theta} \min_{i \in [n]} p_i$. The computation of p^* can be done as follows: First, consider each of the $\binom{n}{\lambda}$ subsets of V of size λ, and find if it is a proper λ-subgraphs of G (i.e. connected); this can be done by running a Depth (or Breadth) First Search algorithm for each subset of size λ. If it is not, then continue with the next subset. If it is, we consider it in the action set $[\theta]$, and assign to it a variable q_j which stands for its assigned probability in a general defense strategy. Now, by definition, for some vertex $i \in [n]$, $p_i = \sum_{\substack{j \in [\theta] \\ i \in j}} q_j$. Therefore, we will consider only pure strategies j which are λ-subgraphs to create the p_i's. To compute the minimum p_i over all i's we introduce the variable p' and write the following set of n inequalities as a constraint in our Linear Program:

$$\sum_{\substack{j \in [\theta] \\ i \in j}} q_j \geq p' \quad , \text{ for } i \in \{1, 2, \ldots, n\}.$$

The variable constraints are $p', q_1, q_2, \ldots, q_\theta \geq 0$ and also $\sum_{j=1}^{\theta} q_j = 1$, and all of the aforementioned constraints can be written in canonical form by applying standard transformations. Finally, the objective function of the Linear Program is variable p' and we require its maximization, which is the value p^*. □

3.1 Connections to other types of games

Although CSD games are defined as a normal form game with $k+1$ players, we can observe that there are equivalent to other well-studied types of games: polymatrix games and Stackelberg games.

A polymatrix game is defined by a graph where every vertex represents a player and every edge represents a two-player game played by the endpoints of the edge. Every player has the same set of pure strategies in every game he is involved and to play the game he plays the same (mixed) strategy in every game. The payoff of every player is the sum they get from every two-player game they participate in. In a CSD game we observe the following. Firstly, the payoff of every attacker depends only on the strategy the defender plays, thus every attacker is involved only in one two-player game. In addition, all the attackers have the same set of pure strategies and they share the same payoff matrix. Similarly, the payoff the defender gets from catching an attacker depends only on the strategy the defender and this specific attacker chose. Hence, the payoff of the defender can be decomposed into a sum of payoffs from k two-player games. So, a CSD game can be seen as a polymatrix game where the underlying graph is a star with k leaves that correspond to the attackers and the defender is the center of the star. Although many-player polymatrix games have exponentially smaller representation size compared to the equivalent normal-form representation,

we should note that this polymatrix game is of exponential size in the worst case since the defender can have exponential in n pure strategies to choose from.

A Stackelberg game is an extensive form two-player game. In the first round, one of the players commits to a (mixed) strategy. In the second round, the other player chooses a best response against the committed strategy of her opponent. In a StackeIberg equilirbium the first player is playing a strategy that maximizes her expected payoff, given that the second player plays a best response (mixed strategy). The MaxMin probability $p^*(G)$ for a CSD game on a graph G corresponds to a Stackelberg equilibrium. By Corollary 1(c), any CSD game with $k \geq 1$ attackers has the same p^* as that of the case with $k = 1$. Furthermore, as in a Stackelberg game, in the CSD game with $k = 1$ the defender chooses a mixed strategy that maximizes her expected payoff, given that the attacker plays a best response (mixed strategy). Therefore, when we are interested in the defense-ratio in equilibrium of a CSD game for some arbitrary $k \geq 1$, finding a Stackelberg equilibrium of the corresponding CSD game with $k = 1$ suffices.

4 Defense-Optimal Graphs

We now focus our attention on defense-optimal graphs. We first characterize defense-optimal graphs with respect to the MaxMin probability p^* and then use this characterization to analyze more specific classes of graphs like cycles and trees. We begin by an exact computation of the equilibrium defense ratio of any defense-optimal graph.

Theorem 3. *In any defense-optimal graph G, we have that $DR(G, S^*) = \frac{n}{\lambda}$.*

As an intermediate corollary of Theorem 3 we get the following characterisation of defense-optimal graphs.

Corollary 3. *A graph G is defense-optimal if and only if all of its vertices are defended with probability $\frac{\lambda}{n}$.*

Someone may wonder whether Corollary 3 can be further exploited to prove that, in general, best-defense strategies in defense-optimal graphs are uniform, i.e. every pure strategy s in the support S of the defender is assigned probability $1/|S|$. However, as we demonstrate in Figure 1 this is not the case. On the other hand, this claim is true for cyclic graphs and trees.

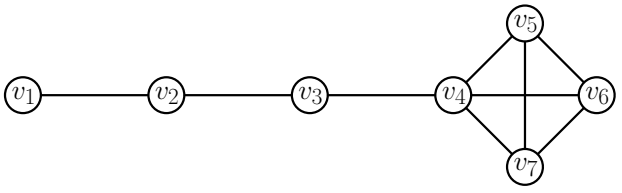

Fig. 1. Here $n = 7$, $\lambda = 3$ and $p^*(G) = 3/7$ is achievable by assigning probability $3/7$ to pure strategy $\{v_1, v_2, v_3\}$ and probability $1/7$ to each of the pure strategies $\{v_4, v_5, v_6\}$, $\{v_4, v_5, v_7\}$, $\{v_4, v_6, v_7\}$, $\{v_5, v_6, v_7\}$, so the graph is defense optimal. However, observe that v_1 cannot participate in more than one pure strategies, so in a uniform defense strategy with support of size r, the vertex probability p_{v_1} has to be $1/r$ (by definition of uniformity), but it also has to be $3/7$. Since $r \in \mathbb{N}$, this is a contradiction.

Observation 1 *All cyclic graphs are defense-optimal.*

Proof. Consider an arbitrary cyclic graph G with n vertices. We will show that the graph can achieve vertex probability $p_i = \frac{\lambda}{n}$ for every $i \in [n]$, thus by Corollary 3 it is defense-optimal. Consider the whole action set D of the defender, i.e. every path starting from a vertex i going clockwise and ending at vertex $i + \lambda - 1$. Observe that there are only n such paths, therefore $\theta := |D| = n$. By assigning probability $\frac{1}{n}$ to each pure strategy $j \in [\theta]$, since each vertex is in exactly λ pure strategies, each vertex $i \in [n]$ has vertex probability $p_i = \lambda \cdot \frac{1}{\theta} = \frac{\lambda}{n}$. □

4.1 Tree Graphs

In this section we focus on the case where the graph is a tree. We first further refine the characterization of defense-optimal graphs for trees. Then, we utilise this characterisation to derive a polynomial-time algorithm that decides in polynomial time whether a given tree is defense-optimal, and if that is the case, it constructs in polynomial time a defense-optimal strategy for it. On the other hand, in the case where the tree is not defense-optimal, we show that it is NP-hard to compute a best-defense strategy for it, namely it is NP-hard to compute $p^*(G)$. We first provide Lemma 2 which will be used in our polynomial-time algorithm for checking defense-optimality on trees. Henceforth, we write that a graph is *covered* by a defense strategy if every vertex of the graph is covered by a λ-subgraph that is in the support of the defense strategy.

Lemma 2. *A tree T is defense-optimal if and only if T can be decomposed into $\frac{n}{\lambda}$ disjoint λ-subgraphs.*

Proof. **(⇒)** Let T be defense-optimal. We will show that the support of any best-defense strategy on T must comprise of pure strategies that are disjoint λ-subgraphs which altogether cover every $v \in V$. Since those are disjoint and cover T, it follows that their number is $\frac{n}{\lambda}$ in total.

If $\lambda = 1$ then the above trivially holds. Assume that $\lambda \geq 2$ and consider a best-defense strategy on T whose support comprises of a collection $\mathcal{L}$ of λ-subgraphs.

Let $u \in V$ be a leaf of T and let $v \in V$ be its parent. Any λ-subgraph in $\mathcal{L}$ covering u must also cover v, since $\lambda \geq 2$. Also, any λ-subgraph in $\mathcal{L}$ covering v must also cover u, otherwise p_v would be greater than p_u. Now, consider the neighbors of v. For those of them that are leaves, the same must hold as holds for u, namely v and its leaf-children must all be covered by the same exact λ-subgraph(s).

Consider the case where there is a leaf $u \in V$, such that a *single* λ-subgraph contains u, its parent v, and all the other leaf-children of v (and, possibly, other vertices connected to v). Then we can remove this λ-subgraph from $\mathcal{L}$ and the corresponding tree from T. This leaves the remainder of T being a forest comprising of trees $T_1, \ldots, T_x$, each of which has a (best-) defense strategy comprising of the corresponding subset of (the remainder of) $\mathcal{L}$ on T_i. Notice that it must be the case that every tree T_i, $i = 1, 2, \ldots, x$, has size at least λ (otherwise the initial collection $\mathcal{L}$ would not have covered T). So, if there is always a leaf u in some tree of the forest, such that a *single* λ-subgraph contains u, its parent v, and all the other leaf-children of v (and, possibly, other vertices connected to v), we can proceed in the same fashion for each of the T_i's, always removing a λ-subgraph from $\mathcal{L}$, and the corresponding vertices from T, until we end up with an empty tree. This means that $\mathcal{L}$ was indeed a collection of disjoint λ-subgraphs covering T.

However, assume for the sake of contradiction that at some "iteration" the assumption does not hold, namely assume that there is a tree in the forest with no leaf u, such that a single λ-subgraph contains u, its parent v, and all the other leaf-children of v (and, possibly, other vertices connected to v). This means that there are (at least) two λ-subgraphs in $\mathcal{L}$, namely L_1, L_2, that cover u. Due to our initial observations, u, together with its parent v and all of v's leaf-children are contained in both L_1 and L_2. Since those are different λ-subgraphs, there is a vertex z in the tree which belongs to L_2 but does not belong to L_1. Since $p_z = p_v$ (due to the fact that $\mathcal{L}$ is the support of the defense-optimal strategy and Corollary 3), it must hold that there is a different λ-subgraph, L_3, which covers z but does not cover v or any of its leaf-children. If L_3 also covers a vertex in $L_1 \setminus L_2$[4], then there is a cycle in the tree which is a contradiction. So L_3 must not cover vertices in $L_1 \setminus L_2$. Since L_3 is different to L_2, there must be a vertex z' in the tree which belongs in L_3 but not in L_2 (also not in L_1). Since $p_{z'} = p_z$ (due to the fact that $\mathcal{L}$ is the support of the defense-optimal strategy and Corollary 3), it must hold that there is a different λ-subgraph, L_4, which covers z' but does not cover z or any of the vertices in L_2. Similarly to before, if L_4 covers a vertex in $L_1 \setminus L_2$, then there is a cycle in the tree which is a contradiction. So L_4 must not cover vertices in L_1 or in L_2.

Proceeding in the same way, we result in contradiction since the tree has finite number of vertices and there will need to be an overlap in coverage of some L_j with some L_i, $j > i+1$, which would mean that there is a cycle in the tree.

[4]We use $L_i \setminus L_j$ for some λ-subgraphs L_i, L_j to denote the set of vertices which are contained in L_i but not in L_j.

Therefore, there cannot be any overlaps between the λ-subgraphs of $\mathcal{L}$, meaning that $\mathcal{L}$ comprises of $\frac{n}{\lambda}$ disjoint λ-subgraphs which altogether cover T.

($\Leftarrow$) Let $\mathcal{L} = \{L_1, \ldots, L_{\frac{n}{\lambda}}\}$ be a collection of $\frac{n}{\lambda}$ disjoint λ-subgraphs that altogether cover T. Let the defender play each L_i, $i \in \{1, \ldots, \frac{n}{\lambda}\}$, equiprobably, that is, with probability $1/\left(\frac{n}{\lambda}\right) = \frac{\lambda}{n}$. Then every vertex $v \in V$ is covered with probability $p_v = \frac{\lambda}{n} = p^*(G)$, meaning that T is defense-optimal. □

With Lemma 2 in hand we can derive a polynomial-time algorithm that decides if a tree is defense-optimal, and if it is, to produce a best-defense strategy.

Theorem 4. *There exists a polynomial-time algorithm that decides whether a tree is defense-optimal and produces a best-defense strategy for it, or it outputs that the tree is not defense-optimal.*

Proof. The algorithm works as follows. Initially, there is a pointer associated with a counter in every leaf of the tree T that moves "upwards" towards an arbitrary root of the tree. For every move of the pointer the corresponding counter increases by one. The pointer moves until one of the following happens: either the counter is equal to λ, or it reaches a vertex with degree greater of equal to 3 where it "stalls". In the case where the counter is equal to λ, we create a λ-subgraph of T, we delete this λ-subgraph from the tree, we move the pointer one position upwards, and we reset the counter back to zero. If a pointer stalls at a vertex of degree $d \geq 3$, it waits until all $d-1$ pointers reach this vertex. Then, all these pointers are merged to a single one and a new counter is created whose value is equal to the sum of the counters of all d pointers. If this sum is more than λ, then the algorithm returns that the graph is not defense-optimal. If this sum is less than or equal to λ, then we proceed as if there was initially only this pointer with its counter; if the new counter is equal to λ, then we create a λ-subgraph of T and reset the counter to 0; else the pointer moves upwards and the counter increases by one. To see why the algorithm requires polynomial time, observe that we need at most n pointers and n counters and in addition every pointer moves at most n times.

We now argue about the correctness of the algorithm described above. Clearly, if the algorithm does not output that the tree is not defense-optimal, it means that it partitioned T into λ-subgraphs. So, from Lemma 2 we get that T is defense-optimal and the uniform probability distribution over the produced partition covers every vertex with probability $\frac{\lambda}{n}$. It remains to argue that when the algorithm outputs that the graph is not defense-optimal, this is indeed the case. Consider the case where we delete a λ-subgraph of the (remaining) tree. Observe that the λ-subgraph our algorithm deleted should be uniquely covered by this λ-subgraph in any best-defense strategy; any other λ-subgraph would overlap with some other λ-subgraph. Hence, the deletion of such a λ-subgraph was not a "wrong" move of our algorithm and the remaining tree is defense-optimal if and only if the tree before the deletion was defense-optimal. This means that any deletion that occurred by our algorithm did not make the remaining graph non defense-optimal. So, consider the case where after a merge that occurred at vertex v we get that the new counter is $c > \lambda$. Then, we can deduce that all the subtrees rooted at v associated with the counters have strictly less than λ vertices. Hence, in order to cover all the $c > \lambda$ vertices using λ-subgraphs, at least two of these λ-subgraphs cover vertex v. Hence, the condition of Lemma 2 is violated. But since every step of our algorithm so far was correct, it means that v cannot be covered only by one λ-subgraph. Hence, our algorithm correctly outputs that the tree is not defense-optimal. □

In Theorem 4 we showed that it is easy to decide whether a tree is defense-optimal and if this is the case, it is easy to find a best-defense strategy for it. Now we prove that if a tree is not defense-optimal, then it is NP-hard to find a best-defense strategy for it.

Theorem 5. *Finding a best-defense strategy in* CSD *games is* NP*-hard, even if the graph is a tree.*

Proof. We will prove the theorem by reducing from 3-Partition. In an instance of 3-Partition we are given a multiset with n positive integers $a_1, a_2, \ldots, a_n$ where $n = 3m$ for some $m \in \mathbb{N}_{>0}$ and we ask whether it can be partitioned into m triplets $S_1, S_2, \ldots, S_m$ such that the sum of the numbers in each subset is equal. Let $s = \sum_{i=1}^{n} a_i$. Observe then that the problem is equivalent to asking whether there is a partition of the integers to m triplets such that the numbers in every triplet sum up to $\frac{s}{m}$. Without loss of generality we can assume that $a_i < \frac{s}{m}$ for every $i \in [n]$; if this

was not the case, the problem could be trivially answered. So, given an instance of 3-PARTITION, we create a tree $G = (V, E)$ with $s + 1$ vertices and $\lambda = \frac{s}{m} + 1$. The tree is created as follows. For every integer a_i, we create a path with a_i vertices. In addition, we create the vertex v_0 and connect it to one of the two ends of each path. We will ask whether $p^*(G) \geq \frac{1}{m}$.

Firstly, assume that the given instance of 3-PARTITION is satisfiable. Then, given S_j we create a $(\frac{s}{m} + 1)$-subgraph of G as follows. If $a_i \in S_j$, then we add the corresponding path of G to the subgraph. Finally, we add vertex v_0 in our $(\frac{s}{m} + 1)$-subgraph and the resulting subgraph is connected (by the construction of G). Since the sum of a_i's equals $\frac{s}{m}$, the constructed subgraph has $\frac{s}{m} + 1$ vertices. If we assign probability $\frac{1}{m}$ to every $(\frac{s}{m} + 1)$-subgraph we get that $p_v \geq \frac{1}{m}$ for every $v \in V$.

To prove the other direction, assume that $p^*(G) \geq \frac{1}{m}$ and observe the following. Firstly, since as we argued it is $a_i < \frac{s}{m}$ for every $i \in [n]$, it holds that every $(\frac{s}{m} + 1)$-subgraph of G contains vertex v_0. Thus, $p_{v_0} = 1$ and $\sum_{v \neq v_0} p_v \geq \frac{s}{m}$, since there are s vertices other than v_0 and for each one of them holds that $p_v \geq \frac{1}{m}$. In addition, observe that $\sum_{v \in V} p_v = \lambda = \frac{s}{m} + 1$. Hence, we get that $p_v = p^*(G) = \frac{1}{m}$ for every vertex $v \neq v_0$. In addition, observe that every pure defense strategy that covers a leaf of this tree, covers all the vertices of the branch. Hence, for every branch of the tree, all its vertices are covered by the same set of pure strategies; if a vertex u that is closer to v_0 is covered by one strategy that does not cover the whole branch, then the leaf u' of the branch is covered with probability less than u. So, in order for $p_v = p^*(G) = \frac{1}{m}$ for every $v \neq v_0$, it means that there exist a $(\frac{s}{m} + 1)$-subgraph that *exactly* covers a subset of the paths; this means that if a $(\frac{s}{m} + 1)$-subgraph covers a vertex in a path, then it covers every vertex of the path. Hence, by the construction of the graph, we get that this $(\frac{s}{m} + 1)$-subgraph of G corresponds to a subset of integers in the 3-PARTITION instance that sum up to $\frac{s}{m}$. Since, 3-PARTITION is NP-hard, we get that finding a best-defense strategy is NP-hard. □

4.2 General Graphs

We conjecture that contrary to checking defense-optimality of tree graphs and constructing a corresponding defense-optimal strategy in polynomial time, it is NP-hard to even decide whether a given (general) graph is defense-optimal.

Conjecture 1. It is NP-hard to decide whether a graph is defense-optimal.

5 Approximation algorithm for $p^*(G)$

We showed in the previous section that, given a graph G, it is NP-hard to find the best-defense strategy, or equivalently, to compute $p^*(G)$. We also presented in Theorem 2 an algorithm for computing the exact value $p^*(G)$ of a given graph G (and therefore its best defense ratio), but this algorithm has running time polynomial in the size of the input only in the cases $\lambda = c$ or $\lambda = n - c$, where c is a constant natural. On the positive side, we present now a polynomial-time algorithm which, given a graph G of n vertices, returns a defense strategy with defense ratio which is within factor $2 + \frac{\lambda - 3}{n}$ of the best defense ratio for G. In particular, it achieves defense ratio $1/p' \leq \left(2 + \frac{\lambda-3}{n}\right)/p^*(G)$, where $p' = \min_{i \in [n]} p_i$ and every p_i, $i \in [n]$ is the vertex probability determined by the constructed defense strategy. We henceforth write that a collection $\mathcal{L}$ of λ-subgraphs covers a graph $G = (V, E)$, if every vertex of V is covered by some λ-subgraph in $\mathcal{L}$. The algorithm presented in this section returns a collection $\mathcal{L}$ of at most $\frac{2n-3}{\lambda} + 1$ λ-subgraphs that covers G. Therefore, the uniform defense strategy over $\mathcal{L}$ assigns probability at least $1/\left(\frac{2n-3}{\lambda} + 1\right)$ to each λ-subgraph.

For any collection $\mathcal{L}$ of λ-subgraphs and for any $v \in V$, let us denote by $\text{coverage}_{\mathcal{L}}(v)$ the number of λ-subgraphs in $\mathcal{L}$ which v belongs in. Observe that:

$$\sum_{v \in V} \text{coverage}_{\mathcal{L}}(v) = |\mathcal{L}| \cdot \lambda, \tag{1}$$

where $|\mathcal{L}|$ denotes the cardinality of $\mathcal{L}$.

We first prove Lemma 3, to be used in the proof of the main theorem of this Section. We henceforth denote by $V(G)$ and $E(G)$ the vertex set and edge set, respectively, of some graph G.

Lemma 3. *For any tree T of n vertices, and for any $\lambda \leq n$, we can find a collection $\mathcal{L}$ of distinct λ-subgraphs such that for every $v \in V$, it holds that $1 \leq coverage_{\mathcal{L}}(v) \leq degree(v)$, except maybe for (at most) $\lambda - 1$ vertices, where for each of them it holds that $coverage_{\mathcal{L}}(v) = degree(v) + 1$.*

Proof. We will prove the statement of the lemma by providing Algorithm 1 that takes as input T and λ and outputs the requested collection $\mathcal{L}$ of λ-subgraphs.

Algorithm 1 Main Algorithm

Require: A tree graph $T = (V, E)$ of n vertices, and a natural $\lambda \leq n$.
Ensure: A collection $\mathcal{L}$ of distinct λ-subgraphs that satisfies the statement of Lemma 3.

1: i, global variable. *% The index of the λ-subgraph L_i.*
2: *count*, global variable. *% Is 0 until the whole tree is covered, then it becomes 1 to allow for the last λ-subgraph to be completed, if it is not already.*
3: S, global variable. *% The set of vertices already covered by the algorithm.*
4: *vertex*, global variable. *% The vertex considered to be inserted in a λ-subgraph.*

5: $S \leftarrow \emptyset$
6: $i \leftarrow 1$
7: $L_i \leftarrow \emptyset$
8: Pick an arbitrary vertex v of T and consider it the root.
9: $vertex \leftarrow v$
10: $count \leftarrow 0$

11: **while** $count < 2$ **do**
12: **while** $S \neq V$ **do**
13: **while** $vertex \in S$ **do** *% The while-loop to ensure that the first element of L_i is uncovered.*
14: **if** *vertex* has a child $u \notin S$ **then**
15: $vertex \leftarrow u$
16: **else**
17: $vertex \leftarrow$ parent of *vertex*
18: **while** $|L_i| < \lambda$ **do** *% The while-loop that fills in the current λ-subgraph L_i.*
19: $L_i \leftarrow L_i \cup \{vertex\}$
20: $S \leftarrow S \cup \{vertex\}$
21: **if** *vertex* has a child $u \notin S$ **then**
22: $vertex \leftarrow u$
23: **else**
24: $vertex \leftarrow$ parent of *vertex*
25: **if** $count < 1$ **then**
26: $i \leftarrow i + 1$
27: $L_i \leftarrow \emptyset$
28: **else**
29: **break**
30: $S \leftarrow \emptyset$
31: $i \leftarrow i - 1$
32: Pick an arbitrary vertex $v \in L_i$ and consider it the root.
33: $vertex \leftarrow v$
34: $count \leftarrow count + 1$

The algorithm starts by picking an arbitrary vertex v to serve as the root of the tree. Then it performs a Depth-First-Search (DFS) starting from v. We will distinguish between *visiting* a vertex and *covering* a vertex in the following way. We say that DFS visited a vertex if it considered that vertex as a candidate to be inserted to some λ-subgraph, and we say that DFS covered a vertex if it visited *and* inserted the vertex at some λ-subgraph. By definition, DFS visits in a greedy manner first an uncovered child, and only if there is no such child, it visits its parent (lines 14-17, 21-24). The set-variable that keeps track of the covered vertices is S.

Starting with the root of T, the algorithm simply visits the whole vertex set according to DFS, putting each visited vertex in the same λ-subgraph L_i (starting with $i = 1$) (lines 18-24), and

when $|L_i| = \lambda$, a new empty λ-subgraph L_{i+1} is picked to get filled in with λ vertices (lines 26-27) taking care of one extra thing: The first vertex that the algorithm puts in an empty λ-subgraph L_i, $i \in \{1, 2, \dots\}$ is guaranteed to be one that has not been covered by any other λ-subgraph so far (lines 13-17). This ensures that no two λ-subgraphs will eventually be identical.

The algorithm will not only visit all vertices in T, but also cover them. That is because there is no point where the algorithm checks whether the currently visited vertex is uncovered and then does not cover it. On the contrary, it covers every vertex that it visits, except for some already covered one in case the current λ-subgraph is empty (lines 13-24). And since DFS by construction visits every vertex, we know that at some point the whole vertex set will be covered, or equivalently, $\text{coverage}_{\mathcal{L}}(v) \geq 1, \forall v \in V$. Therefore, the algorithm will eventually exit the while-loop in lines 12-29.

Now we prove that, after the algorithm terminates, every vertex $v \in V$ is covered at most $degree(v)$ times, except for at most $\lambda - 1$ vertices that are covered $degree(v) + 1$ times. Observe that DFS visits every vertex v at most $degree(v)$ times: (a) v will be visited after its parent u only if v is uncovered (lines 14-15, 21-22), v will get covered (lines 19-20), and will not get visited ever again by its parent since it will be covered (lines 16-17, 23-24). (b) v will be visited at most once by each of its children, say w, only if w does not have an uncovered child (lines 16-17, 23-24), and v will not get ever visited by its parent since v will be covered, and also v cannot be visited a second time by any of its children, since they can never be visited again (they can only be visited through v since T is a tree). Therefore, any vertex v will be visited exactly once after its parent is visited, and at most once by each of its children, having a total of at most $degree(v)$ visits. And since, as argued above, the total number of times a vertex will be covered is at most the number of times it will get visited, when DFS terminates (i.e $S = V$), it will be $\text{coverage}_{\mathcal{L}}(v) \leq degree(v)$, for every $v \in V$.

However, note that the last nonempty λ-subgraph L_i might not consist of λ vertices since the entire V was covered and DFS could not proceed further. In this case, the algorithm empties the set S that keeps track of the covered nodes, takes the current L_i and fills it in with exactly another $\lambda - |L_i|$ vertices. This is done by picking an arbitrary vertex from L_i and setting it as the root of T, and performing one last DFS starting from it until L_i has λ vertices in total (lines 30-33). To ensure that the DFS will continue only until it fills in this current L_i, the algorithm counts the number of times that it runs the while-loop of DFS, namely lines 12-29, via the variable *count* (line 34), which escapes the while-loop of DFS in case DFS has filled in L_i (lines 28-29) and terminates. Observe that in the last λ-subgraph L_i, a vertex v inserted in the last iteration of DFS ($count = 1$) and was not inserted in L_i by the first run ($count = 0$) might have been covered by the first run of DFS exactly $degree(v)$ times, therefore when the algorithm terminates it has been covered $degree(v)+1$ times. Since by the end of the first DFS run L_i had at least one vertex, the cardinality of such vertices that are covered more times than their degree are at most $\lambda - 1$. □

We can now prove the following.

Lemma 4. *For any graph G of n vertices, and for any $\lambda \leq n$, there exist (at most) $\frac{2n-3}{\lambda} + 1$ λ-subgaphs of G that cover G.*

Proof. Consider a spanning tree T of G. Then Lemma 3 applies to T. Observe that a collection $\mathcal{L}$ as described in the statement of the aforementioned lemma has the same qualities for G since $V(T) = V(G)$ and $E(T) \subseteq E(G)$. That is, $\mathcal{L}$ is a collection of distinct λ-subgraphs of G, such that for every $v \in V$, it holds that $1 \leq \text{coverage}_{\mathcal{L}}(v) \leq \text{degree}(v)$, except maybe for (at most) $\lambda - 1$ vertices, for each v of which it is $\text{coverage}_{\mathcal{L}}(v) = \text{degree}(v) + 1$, where by $\text{degree}(v)$ we denote the degree of vertex v in T.

Fix a particular value for λ and consider a collection $\mathcal{L}$ of λ-subgraphs as constructed in the proof of Lemma 3. Then, by equation (1),

$$|\mathcal{L}| = \frac{\sum_{v \in V} \text{coverage}_{\mathcal{L}}(v)}{\lambda} \leq \frac{\sum_{v \in V} \text{degree}(v) + (\lambda - 1)}{\lambda} = \frac{2(n-1)}{\lambda} + \frac{\lambda - 1}{\lambda} = \frac{2n-3}{\lambda} + 1.$$

□

We conclude with the simple algorithm that achieves a defense strategy with defense ratio which is within factor $2 + \frac{\lambda - 3}{n}$ of the best defense ratio for G.

Algorithm 2 Approximating the best defense ratio

Require: Graph $G = (V, E)$ of n vertices, a natural $\lambda \leq n$.
Ensure: A defense strategy that satisfies the statement of Theorem 6.

1: Find a spanning tree T of G.
2: Construct a collection $\mathcal{L}$ of λ-subgraphs of T as described in the proof of Lemma 3.
3: Assign probability $q_i = \frac{1}{|\mathcal{L}|}$ to every λ-subgraph in $\mathcal{L}$, $i = 1, 2, \ldots, |\mathcal{L}|$.
4: **return** The above uniform defense strategy over the collection $\mathcal{L}$.

Theorem 6. *Given any graph $G = (V, E)$, Algorithm 2 computes in time $O(|E|)$ a defense strategy such that, for any combination of attack strategies, the resulting strategy profile S yields defense ratio* $\mathrm{DR}(G, S) \leq \left(2 + \frac{\lambda-3}{n}\right) \cdot \mathrm{DR}(G, S^*)$.

Proof. As argued in Lemma 4, there is a collection $\mathcal{L}$ of λ-subgraphs with $|\mathcal{L}| \leq \frac{2n}{\lambda} + 1 - \frac{3}{\lambda}$ which covers G. Therefore, the uniform defense strategy returned by Algorithm 2 (which determines the vertex probability p_i for each vertex i) achieves a minimum vertex probability $p' := \min_{i \in [n]} p_i$ for which it holds that:

$$p' = \frac{1}{|\mathcal{L}|} \geq \frac{1}{\frac{2n}{\lambda} + 1 - \frac{3}{\lambda}} = \frac{\frac{\lambda}{n}}{2 + \frac{\lambda-3}{n}} \geq \frac{1}{2 + \frac{\lambda-3}{n}} \cdot p^*(G),$$

where the first equality is due to the fact that any leaf $v \in V$ of the spanning tree T of G through which $\mathcal{L}$ was created has $\mathrm{coverage}_{\mathcal{L}}(v) = 1$, and therefore there is such a vertex v in G that is covered by exactly one λ-subgraph; and the last inequality is due to the fact that $p^*(G) \leq \lambda/n$ for any graph G (due to Corollary 3), where $p^*(G)$ is the MaxMin probability of G.

The above inequality implies that if the defender chooses the prescribed strategy the minimum defense ratio cannot be too bad. That is because in the worst case for the defender, each and every attacker will choose a vertex v' on which the aforementioned strategy of the defender results to vertex probability p' (so that the attacker is caught with minimum probability). As a result, the defender will have the minimum possible expected payoff which is $p' \cdot k$. Thus, for the constructed defend strategy and any combination of attack strategies, the resulting strategy profile S yields defense ratio:

$$\mathrm{DR}(G, S) \leq \frac{k}{p' \cdot k} \leq \left(2 + \frac{\lambda - 3}{n}\right) \cdot \frac{1}{p^*(G)} = \left(2 + \frac{\lambda - 3}{n}\right) \cdot \mathrm{DR}(G, S^*),$$

where the last equality is due to Lemma 1.

With respect to the running time, notice that Step 1 of Algorithm 2 can be executed in time $O(|V| + |E(G)|) = O(|E(G)|)$. Step 2 can be executed in time $O(|V| + |E(T)|) = O(|V|)$. Finally, Step 3 can be executed in constant time. Therefore, the total running time of Algorithm 2 is $O(|E(G)|)$. □

Corollary 4. *For any graph G there is a polynomial (in both n and λ) time approximation algorithm (Algorithm 2) with approximation factor $1/\left(2 + \frac{\lambda-3}{n}\right)$ for the computation of $p^*(G)$.*

The merit of finding a probability p' that approximates (from below) $p^*(G)$ for a given graph G through an algorithm such as Algorithm 2 is in guaranteeing to the defender that, no matter what the attackers play, she always "catches" at least a portion p' of them in expectation, where the best portion is $p^*(G)$ in an equilibrium. Algorithm 2 guarantees that the defender catches at least $1/\left(2 + \frac{\lambda-3}{n}\right)$ of the attackers in expectation.

6 Bounds on the Price of Defense

In the following theorem we give a lower bound on the PoD for any given n and $2 \leq \lambda \leq n - 1$ by constructing a graph G with particular (very small) $p^*(G)$ (which, by Lemma 1 implies great best defense ratio). Due to lack of space the construction is omitted and can be found in the full version of the paper [1].

Theorem 7. *The* PoD(λ) *is lower bounded by* $\left\lfloor \frac{2(n-1)}{\lambda} \right\rfloor$ *and* $\left\lfloor \frac{2(n-1)}{\lambda+1} \right\rfloor$ *for* λ *even and odd respectively, when* $\lambda \in \{2, 3, \ldots, n-1\}$.

Corollary 5. *For any given* n *and* $2 \leq \lambda \leq n-1$*, it holds that* $\left\lfloor \frac{2(n-1)}{\lambda+1} \right\rfloor \leq \text{PoD}(\lambda) \leq \frac{2(n-1)+\lambda-1}{\lambda}$. *Furthermore, for the trivial cases* $\lambda \in \{1, n\}$ *it is* $\text{PoD}(1) = n$ *and* $\text{PoD}(n) = 1$.

Proof. The lower bound is established by Theorem 7. The upper bound is due to Theorem 6. For the cases $\lambda = 1$ and $\lambda = n$, observe that the defender's action set is $D = \{\{i\} | i \in V\}$ and $D = \{V\}$ respectively, therefore $p^*(G) = 1/n$ and $p^*(G) = 1$ respectively, and again from Lemma 1 we get the values in the statement of the corollary. □

References

1. Akrida, E.C., Deligkas, A., Melissourgos, T., Spirakis, P.G.: Connected subgraph defense games. CoRR **abs/1906.02774** (2019), http://arxiv.org/abs/1906.02774
2. Alon, N., Karp, R.M., Peleg, D., West, D.B.: A graph-theoretic game and its application to the k-server problem. SIAM J. Comput. **24**(1), 78–100 (1995)
3. An, B., Pita, J., Shieh, E., Tambe, M., Kiekintveld, C., Marecki, J.: Guards and protect: Next generation applications of security games. ACM SIGecom Exchanges **10**(1), 31–34 (2011)
4. Aspnes, J., Chang, K.L., Yampolskiy, A.: Inoculation strategies for victims of viruses and the sum-of-squares partition problem. J. Comput. Syst. Sci. **72**(6), 1077–1093 (2006)
5. Attiya, H., Welch, J.: Distributed Computing: Fundamentals, Simulations and Advanced Topics. John Wiley & Sons, Inc. (2004)
6. Cheswick, W.R., Bellovin, S.M., Rubin, A.D.: Firewalls and Internet Security: Repelling the Wily Hacker. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2 edn. (2003)
7. Franklin, M.K., Galil, Z., Yung, M.: Eavesdropping games: a graph-theoretic approach to privacy in distributed systems. J. ACM **47**(2), 225–243 (2000)
8. Jain, M., Conitzer, V., Tambe, M.: Security scheduling for real-world networks. In: Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems. pp. 215–222. International Foundation for Autonomous Agents and Multiagent Systems (2013)
9. Kearns, M.J., Ortiz, L.E.: Algorithms for interdependent security games. In: Advances in Neural Information Processing Systems 16 [Neural Information Processing Systems, NIPS]. pp. 561–568 (2003)
10. Letchford, J., Conitzer, V.: Solving security games on graphs via marginal probabilities. In: Twenty-Seventh AAAI Conference on Artificial Intelligence (2013)
11. Mavronicolas, M., Michael, L., Papadopoulou, V.G., Philippou, A., Spirakis, P.G.: The price of defense. In: Mathematical Foundations of Computer Science 2006, 31st International Symposium, MFCS. pp. 717–728 (2006)
12. Mavronicolas, M., Papadopoulou, V., Philippou, A., Spirakis, P.G.: A network game with attackers and a defender. Algorithmica **51**(3), 315–341 (2008)
13. Mavronicolas, M., Papadopoulou, V.G., Persiano, G., Philippou, A., Spirakis, P.G.: The price of defense and fractional matchings. In: Distributed Computing and Networking, 8th International Conference, ICDCN. pp. 115–126 (2006)
14. Mavronicolas, M., Papadopoulou, V.G., Philippou, A., Spirakis, P.G.: A graph-theoretic network security game. In: Internet and Network Economics, First International Workshop, WINE. pp. 969–978 (2005)
15. Nash, J.F.: Equilibrium points in n-person games. Proceedings of the National Academy of Sciences **36**(1), 48–49 (1950)
16. Nash, J.: Non-cooperative games. Annals of Mathematics **54**(2), 286–295 (1951)
17. Spafford, E.H.: The internet worm: Crisis and aftermath. Commun. ACM **32**(6), 678–687 (1989)
18. Stallings, W.: Cryptography and network security - principles and practice (3. ed.). Prentice Hall (2003)
19. Vaněk, O., Yin, Z., Jain, M., Bošanskỳ, B., Tambe, M., Pěchouček, M.: Game-theoretic resource allocation for malicious packet detection in computer networks. In: Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 2. pp. 905–912. International Foundation for Autonomous Agents and Multiagent Systems (2012)
20. Xu, H.: The mysteries of security games: Equilibrium computation becomes combinatorial algorithm design. In: Proceedings of the 2016 ACM Conference on Economics and Computation. pp. 497–514. EC '16, ACM, New York, NY, USA (2016). https://doi.org/10.1145/2940716.2940796, http://doi.acm.org/10.1145/2940716.2940796