Mixed Strategies for Security Games with General Defending Requirements

Rufan Bai[1], **Haoxing Lin**[2*], **Xinyu Yang**[1], **Xiaowei Wu**[1], **Minming Li**[3], **Weijia Jia**[4 5]
[1] IoTSC, University of Macau
[2] National University of Singapore
[3] City University of Hong Kong
[4] BNU-UIC
[5] Beijing Normal University (Zhuhai)
yb97439@um.edu.mo, haoxing.lin@comp.nus.edu.sg, mb95466@um.edu.mo,
xiaoweiwu@um.edu.mo, minming.li@cityu.edu.hk, jiawj@uic.edu.cn

Abstract

The Stackelberg security game is played between a defender and an attacker, where the defender needs to allocate a limited amount of resources to multiple targets in order to minimize the loss due to adversarial attack by the attacker. While allowing targets to have different values, classic settings often assume uniform requirements to defend the targets. This enables existing results that study mixed strategies (randomized allocation algorithms) to adopt a *compact representation* of the mixed strategies.

In this work, we initiate the study of mixed strategies for the security games in which the targets can have different defending requirements. In contrast to the case of uniform defending requirement, for which an optimal mixed strategy can be computed efficiently, we show that computing the optimal mixed strategy is NP-hard for the general defending requirements setting. However, we show that strong upper and lower bounds for the optimal mixed strategy defending result can be derived. We propose an efficient close-to-optimal Patching algorithm that computes mixed strategies that use only few pure strategies. We also study the setting when the game is played on a network and resource sharing is enabled between neighboring targets. Our experimental results demonstrate the effectiveness of our algorithm in several large real-world datasets.

1 Introduction

Recently, security games have attracted much attention from the game theory society due to its applications in many real world scenarios [Shieh *et al.*, 2012; Conitzer, 2012; Fang *et al.*, 2015]. Classical security games often model the problem as a Stackelberg game [Nguyen *et al.*, 2013; Gan *et al.*, 2019] that is played between two players, the *defender* and the *attacker*, where the defender is the *leader* who commits to a defending strategy before the *follower* (the attacker) observes and responds. In this paper we focus on zero-sum games [Alshamsi *et al.*, 2018; Tsai *et al.*, 2012], in which there are multiple targets to be defended, where each target u has a value α_u that represents the loss if an attack at the target is successful, and a threshold θ_u that represents the resource needed to defend the attack, i.e., the defending requirement. If a target u receives resource at least θ_u, then no loss will occur if u is under attack. In these games, the defender needs to decide an allocation of the limited resources to the targets; and the attacker will choose a target to attack after observing the strategy of the defender. The objective of the defender is to minimize the loss caused by the attack, which we refer to as the defending result of the strategy.

The allocation strategies are often categorized as *pure strategies* and *mixed strategies*. When the allocation is deterministic (resp. randomized), it is called a pure (resp. mixed) strategy. Formally speaking, a mixed strategy is a probability distribution over a set of pure strategies. It has been commonly observed that mixed strategies often achieve defending results that are much better than that of the best pure strategy.

Example 1 *Consider the instance with targets $\{a, b, c, d\}$, each of which has threshold (defending requirement) equals to 1. The values of targets $\{a, b, c\}$ are 3 and the value of target d is 1. Given total resource $R = 2$, every pure strategy has defending result 3 because there always exists a target among $\{a, b, c\}$ with insufficient defending resource. In contrast, the mixed strategy that applies each of the following three pure strategies $(1, 1, 0, 0)$, $(1, 0, 1, 0)$, and $(0, 1, 1, 0)$ with probability $1/3$ achieves a defending result of 1.*

Most existing works that consider mixed strategies for security games assume that the thresholds of targets are uniform [Sinha *et al.*, 2018; Nguyen *et al.*, 2009; Kiekintveld *et al.*, 2009; Korzhyk *et al.*, 2010]. This allows us to represent each mixed strategy by its corresponding *compact representation*, in which the resources allocated to the targets are no longer binary. Instead, when a target receives resources below its threshold, it is assumed that the target is *fractionally* defended when we evaluate the loss due to the attack. It has been shown by [Korzhyk *et al.*, 2010] that it takes polynomial time to translate any compact representation into a mixed strategy that uses $O(n^2)$ pure strategies and achieves the same result as the compact representation, where n is the number of targets.

*Part of this work was done while the author was working as a Research Assistant at the University of Macau.

In this paper, we consider the case when the thresholds of targets are general, i.e., the defending requirements of the targets can be different. For example, while defending against virus, consider the targets as cities and the resources as vaccines. The defending requirements to protect the cities naturally depend on the populations of cities, which can be dramatically different. Therefore, a natural question to ask is whether the compact representation still holds when targets have general thresholds. In particular,

Can every compact representation be transformed into a mixed strategy that achieves the same defending result?

Unfortunately, via the following example, we show that this might not be true when targets have general thresholds.

Example 2 *Consider the instance with three targets* (a, b, c) *with thresholds* $(2, 2, 1)$ *and values* $(3, 3, 1)$*. Given total resource* $R = 4$*, it can be verified that the optimal mixed strategy applies each of the pure strategies* $(3, 0, 1), (0, 3, 1)$ *with probability* $1/2$*, and has defending result* 1*. However, the compact representation* $(15/8, 15/8, 1/4)$ *has defending result* $3/4$ *and clearly, there does not exist any mixed strategy that achieves the same defending result.*

Our Contributions. Since compact representations and mixed strategies are no longer equivalent in the general threshold setting, in the following we refer to each compact representation as a *fractional strategy*. Our work formalizes the mixed strategies and fractional strategies, and studies their connections and differences.

Mixed vs. Fractional. Our first contribution is a theoretical study to establish the connection between mixed and fractional strategies. We let $\mathsf{OPT}_m(R)$ and $\mathsf{OPT}_f(R)$ be the defending result of the optimal mixed and fractional strategy using total resource R, respectively. We first show that computing the optimal mixed strategy is NP-hard, but we always have $\mathsf{OPT}_m(R) \geq \mathsf{OPT}_f(R)$. Since the optimal fractional strategy with defending result $\mathsf{OPT}_f(R)$ can be computed by solving a linear program, we use $\mathsf{OPT}_f(R)$ to lower bound $\mathsf{OPT}_m(R)$. More importantly, we show that given total resource R, we can always find a mixed strategy whose defending result is at most $\mathsf{OPT}_f(R-\theta_{\max})$, where $\theta_{\max}$ is the maximum threshold of the targets. Moreover, we present a polynomial-time algorithm to compute the mixed strategy whose defending result is at most $\mathsf{OPT}_f(R - \theta_{\max})$, and guarantee that $O(n^2)$ pure strategies are used. By proving that the function $\mathsf{OPT}_f(\cdot)$ is convex, we show that when R is much larger than $\theta_{\max}$, $\mathsf{OPT}_m(R)$ and $\mathsf{OPT}_f(R)$ are very close to each other. To the best of our knowledge, we are the first to theoretically establish the almost-equivalence between mixed and fractional strategies for the security game with general defending requirements.

Algorithm with Small Support. For practical use purpose, mixed strategy that uses few pure strategies are often preferred, e.g., it is unrealistic to deploy a mixed strategy that uses $\omega(n)$ pure strategies when the number of targets n is large. Thus we study the computation of mixed strategies that use few pure strategies, i.e., those with small *supports*. Motivated by the Double Oracle algorithm by Jain et al. [Jain *et al.*, 2011] and the column generation techniques [Jain *et al.*, 2013; Wang *et al.*, 2016; Gan *et al.*, 2017], we propose the Patching algorithm that in each iteration finds and includes a new pure strategy to patch the targets that are poorly defended by the current mixed strategy. We show that given a bounded size pure strategy set D, our algorithm takes polynomial time (in $|D|$) to compute an optimal mixed strategy with support D.

Resource Sharing. We also study the setting with resource sharing in the network, which is motivated by patrolling and surveillance camera installation applications [Vorobeychik *et al.*, 2014; Yin *et al.*, 2015; Bai *et al.*, 2021]. In this setting the targets are represented by the nodes of a network, and when a certain target is attacked, some fraction of resources allocated to its neighbors can be shared to the target. Similar to ours, [Li *et al.*, 2020] consider the model with general thresholds, but they only study the computation of pure strategies. When resource sharing is allowed, we show that the gap between $\mathsf{OPT}_m(R)$ and $\mathsf{OPT}_f(R)$ can be arbitrarily large. Therefore, the idea of rounding fractional strategies to get mixed strategies with good approximation guarantee is no longer feasible. However, our Patching algorithm can still be applied to compute mixed strategies efficiently in the resource sharing setting. We show that under certain conditions, the algorithm is able to make progress towards decreasing the defending result.

Experiments. Finally, we conduct extensive experiments on several large real-world datasets to verify our analysis and test our algorithms. The experimental results show that the Patching algorithm efficiently computes mixed strategies with small support, e.g., using 5 pure strategies, whose defending result dramatically improves the optimal pure strategies, and is close-to-optimal in many cases.

The rest of the paper is structured as follows. Section 2 introduces a formal description of the models. Section 3 focuses on the relation between mixed and fractional strategy, where we establish most of our theoretical results. The Patching algorithm is presented in Section 4 and the experimental results are included in Section 5.

Other Related Works. There are works that consider the security game with resource sharing as a dynamic process in which it takes time for the neighboring nodes to share the resources [Basilico *et al.*, 2009; Yin *et al.*, 2015]. Motivated by the applications to stop virus from spreading, network security games with contagious attacks have also received a considerable attention [Aspnes *et al.*, 2006; Kumar *et al.*, 2010; Tsai *et al.*, 2012; Acemoglu *et al.*, 2016; Bai *et al.*, 2021]. In these models the attack at a node can spread to its neighbors, and the loss is evaluated at all nodes under attack. There are also works that consider multi-defender games, where each defender is responsible for one target [Lou and Vorobeychik, 2015; Gan *et al.*, 2018; Gan *et al.*, 2020].

2 Preliminaries

In this section we present the model we study. We define our model in the most general form, i.e., including the network structure and with resource sharing, and consider the model without resource sharing as a restricted setting.

We model the network as an undirected connected graph $G(V, E)$, where each node $u \in V$ has a *threshold* θ_u that represents the defending requirement, and a *value* α_u that

represents the possible damage due to an attack at node u. Each edge $e \in E$ is associated with a weight w_{uv}, which represents the efficiency of resource sharing between the two endpoints. We use $N(u) := \{v \in V : (u, v) \in E\}$ to denote the set of neighbors for node $u \in V$. We use n and m to denote the number of nodes and edges in the graph G, respectively. For any integer i, we use $[i]$ to denote $\{1, 2, \ldots, i\}$.

The defender has a total resource of R that can be distributed to nodes in V. We use r_u to denote the *defending resource*[1] allocated to node u. Thus we require $\sum_{u \in V} r_u \leq R$.

Definition 1 (Pure Strategy) *We use* $\mathbf{r} = \{r_u\}_{u \in V}$ *to denote a* pure strategy *and* $\Omega_p(R) = \{\mathbf{r} \in [0, R]^V : \|\mathbf{r}\| = \sum_{u \in V} r_u \leq R\}$ [2] *to denote the collection of pure strategies using resource* R*. When* R *is clear from the context, we use* Ω_p *to denote* $\Omega_p(R)$*.*

We consider resource sharing in our model. That is, when node u is under attack, it can receive $w_{uv} \cdot r_v$ units of resource shared from each of its neighbors $v \in N(u)$.

Definition 2 (Defending Power) *Given pure strategy* $\mathbf{r}$*, the defending power of node* u *is defined as* $\pi_u(\mathbf{r}) = r_u + \sum_{v \in N(u)} w_{uv} \cdot r_v$*. We use* $\boldsymbol{\pi}(\mathbf{r}) = (\pi_u(\mathbf{r}))_{u \in V}$ *to denote defending powers of nodes.*

Definition 3 (Defending Status) *Given a pure strategy* $\mathbf{r}$*, we use* $\mathbf{x}(\mathbf{r}) \in \{0, 1\}^V$ *to denote the defending status of nodes under* $\mathbf{r}$*, where for each node* u *we have* $x_u(\mathbf{r}) = 1$ *if* $\pi_u(\mathbf{r}) \geq \theta_u$*, i.e., node* u *is well defended; and* $x_u(\mathbf{r}) = 0$ *otherwise.*

Each pure strategy $\mathbf{r}$ has a unique defending status $\mathbf{x}(\mathbf{r})$ but different strategies can have the same defending status.

Definition 4 (Defending Result) *Given a pure strategy* $\mathbf{r}$*, when node* $u \in V$ *is under attack, the loss is given by* $L_p(u, \mathbf{r}) = \alpha_u$ *if* $x_u(\mathbf{r}) = 0$*;* $L_p(u, \mathbf{r}) = 0$ *otherwise. The defending result of strategy* $\mathbf{r}$ *is defined as the maximum loss due to an attack:* $L_p(\mathbf{r}) = \max_{u \in V}\{L_p(u, \mathbf{r})\}$.

We use $\mathbf{r}^*$ to denote the optimal pure strategy, i.e., the pure strategy that has the minimum defending result $\mathbf{r}^* = \arg\min_{\mathbf{r} \in \Omega_p}\{L_p(\mathbf{r})\}$. The corresponding defending result is defined as $\mathsf{OPT}_p = L_p(\mathbf{r}^*)$.

Definition 5 (Mixed Strategy) *A mixed strategy is denoted by* $(D, \mathbf{p})$*, where* $D \subseteq \Omega_p$ *is a subset of pure strategies and* $\mathbf{p}$ *is a probability distribution over* D*. For each* $\mathbf{r} \in D$*, we use* $p(\mathbf{r})$ *to denote the probability that pure strategy* $\mathbf{r}$ *is used.*

A mixed strategy is a randomized algorithm that applies pure strategies with certain probabilities. Note that $\sum_{\mathbf{r} \in D} p(\mathbf{r}) = 1$. We can also interpret $\mathbf{p}$ as a $|D|$ dimension vector with $\|\mathbf{p}\| = 1$. We use $\Omega_m(R) = \{(D, \mathbf{p}) : D \subseteq \Omega_p(R), \mathbf{p} \in [0, 1]^{|D|}, \|\mathbf{p}\| = 1\}$ to denote the collection of all mixed strategies using resource R. When R is clear from the context, we use Ω_m to denote $\Omega_m(R)$.

Definition 6 (Defending Status of Mixed Strategy) *Given a mixed strategy* $(D, \mathbf{p})$*, we use* $x_u(D, \mathbf{p}) = \sum_{\mathbf{r} \in D} p(\mathbf{r}) \cdot x_u(\mathbf{r})$ *to denote the defending status of node* $u \in V$ *under* $(D, \mathbf{p})$*. In other words,* $x_u(D, \mathbf{p})$ *is the probability that node* u *is well defended under mixed strategy* $(D, \mathbf{p})$*. We use* $\mathbf{x}(D, \mathbf{p}) = (x_u(D, \mathbf{p}))_{u \in V} \in [0, 1]^V$ *to denote the defending status of* $(D, \mathbf{p})$*.*

Definition 7 (Defending Result of Mixed Strategy) *Given mixed strategy* $(D, \mathbf{p})$*, we use* $L_m(u, (D, \mathbf{p})) = (1 - x_u(D, \mathbf{p})) \cdot \alpha_u$ *to denote the (expected) loss when node* u *is under attack. The defending result is defined as* $L_m(D, \mathbf{p}) = \max_{u \in V}\{L_m(u, (D, \mathbf{p}))\}$.

We use $(D^*, \mathbf{p}^*)$ to denote the optimal mixed strategy, i.e., the mixed strategy with the minimum defending result. The corresponding defending result is defined as $\mathsf{OPT}_m = L_m(D^*, \mathbf{p}^*)$. Next we define the fractional strategies. Technically, a fractional strategy is not a strategy, but instead a pure strategy equipped with a fractional valuation of defending loss. In the remaining of this paper, when a pure strategy is evaluated by its fractional loss, we call it a *fractional strategy*.

Definition 8 (Fractional Loss) *Given a pure strategy* $\mathbf{r} \in \Omega_p$*, we evaluate the fractional loss when node* u *is attacked by* $L_f(u, \mathbf{r}) = (1 - \min\{\pi_u(\mathbf{r})/\theta_u, 1\}) \cdot \alpha_u$*. The fractional defending result is defined as* $L_f(\mathbf{r}) = \max_{u \in V}\{L_f(u, \mathbf{r})\}$.

In a fractional strategy, if a node u has defending power π_u, then we assume that $\min\{\pi_u/\theta_u, 1\}$ fraction of the node is defended. Thus when node u is under attack, the loss is given by $(1 - \min\{\pi_u/\theta_u, 1\}) \cdot \alpha_u$. We use $\tilde{\mathbf{r}}^*$ to denote the optimal fractional strategy, i.e., the strategy with minimum $L_f(\tilde{\mathbf{r}}^*)$. The corresponding defending result is defined as OPT_f. We use $\mathsf{OPT}_p(R), \mathsf{OPT}_m(R)$ and $\mathsf{OPT}_f(R)$ to denote the defending result of the optimal pure, mixed and fractional strategy using total resource R, respectively. When R is clear from the context, we simply use $\mathsf{OPT}_p, \mathsf{OPT}_m$ and OPT_f. The following lemma implies that the optimal fractional strategy has a defending result at most that of the optimal mixed strategy. For space limits, the proof of the lemma (and lemmas to follow) are included in the full version.

Lemma 1 *We have* $\mathsf{OPT}_p \geq \mathsf{OPT}_m \geq \mathsf{OPT}_f$.

3 Computation of Strategies

In this section we consider the computation of the optimal pure, mixed and fractional strategies, and analyze the properties regarding the optimal defending results of different strategies.

We remark that our model is equal to the "single threshold" model of [Li *et al.*, 2020]. We thus use their algorithm (that runs in polynomial time) to compute an optimal pure strategy. Roughly speaking, in their algorithm a target defending result α is fixed and the goal is to decide whether it is possible to defend all nodes $A(\alpha) = \{u \in V : \alpha_u > \alpha\}$ with value larger than α. For every fixed α the above decision problem can be solved by computing a feasibility LP with constraints $\sum_{u \in V} r_u \leq R$ and $r_u + \sum_{v \in N(u)} w_{uv} \cdot r_v \geq \theta_u$ for every $u \in A(\alpha)$. Combining the above sub-routine with a binary search on $\alpha \in \{\alpha_u\}_{u \in V} \cup \{0\}$ yields a polynomial time algorithm for computing the optimal pure strategy, i.e., with the minimum achievable defending result α. The computation of optimal fractional strategy can be done efficiently by solving

[1] As in [Li *et al.*, 2020; Bai *et al.*, 2021], we assume the resource can be allocated arbitrarily in our model.

[2] Throughout this paper we use $\|\cdot\|$ to denote the L_1 norm.

the following linear program $(\mathsf{LP}_f(R))$, where we introduce a variable r_u for each node $u \in V$ that represents the resource u receives, and a variable L for the defending result.

$$\begin{aligned}(\mathsf{LP}_f(R)) \qquad & \text{minimize} && L \\ & \text{subject to} && \textstyle\sum_{u\in V} r_u \le R, \\ & (1-(r_u+\textstyle\sum_{v\in N(u)} w_{uv}\cdot r_v)/\theta_u)\cdot\alpha_u \le L, && \forall u \in V\end{aligned}$$

By solving the above LP we can get the optimal fractional strategy $\tilde{\mathbf{r}}^*$, whose defending result OPT_f is the optimal objective of the LP. From Lemma 1, we have $\mathsf{OPT}_f \le \mathsf{OPT}_m$, i.e., we can use OPT_f as a lower bound for the defending result of the optimal mixed strategy (which is NP-hard to compute, as we will show in the next subsection). In the following we show that the optimal objective of the above LP is a convex function of the total resource R.

Lemma 2 (Convexity) *Given resource R_1 and R_2, we have*

$$\mathit{OPT}_f(R_1) + \mathit{OPT}_f(R_2) \ge 2\cdot \mathit{OPT}_f\left(\tfrac{1}{2}(R_1+R_2)\right).$$

We have shown that the optimal pure and fractional strategies can be computed efficiently. Unfortunately, we show that computing the optimal mixed strategy is NP-hard, even in the *isolated model*, i.e., when $w_{uv} = 0$ for all $(u, v) \in E$.

Theorem 1 *Unless $P = NP$, there does not exist polynomial time algorithm that given a graph G and resource R computes the optimal mixed strategy, even under the isolated model.*

While computing the optimal mixed strategy is NP-hard, we can use $\mathsf{OPT}_f(R)$ to give a lower bound on $\mathsf{OPT}_m(R)$. In other words, if a mixed strategy has a defending result close to $\mathsf{OPT}_f(R)$, then it is close-to-optimal. However, if the lower bound is loose, then no such mixed strategy exists. Therefore, it is crucial to know whether this lower bound is tight. In this section, we show that in the isolated model, we can give a strong upper bound on $\mathsf{OPT}_m(R)$, which shows that $\mathsf{OPT}_f(R)$ is an almost tight lower bound when R is large.

Theorem 2 *In the isolated model, given any instance $G(V, E)$ and a total resource R, we have $\mathit{OPT}_m(R) \le \mathit{OPT}_f(R - \theta_{max})$, where $\theta_{max} = \max_{u\in V}\{\theta_u\}$.*

We remark that by convexity of $\mathsf{OPT}_f(\cdot)$, we have

$$\begin{aligned}&\mathsf{OPT}_f(R-\theta_{\max}) \le \tfrac{R-\theta_{\max}}{R}\cdot\mathsf{OPT}_f(R) + \tfrac{\theta_{\max}}{R}\cdot\mathsf{OPT}_f(0)\\ =&\mathsf{OPT}_f(R)+\tfrac{\theta_{\max}}{R}\cdot(\max\nolimits_{u\in V}\{\alpha_u\}-\mathsf{OPT}_f(R)).\end{aligned}$$

In other words, when $R \gg \theta_{\max}$, $\mathsf{OPT}_f(R)$ and $\mathsf{OPT}_f(R-\theta_{\max})$ have very similar values. Hence combining Lemma 1 and Theorem 2, we have strong upper and lower bounds on $\mathsf{OPT}_m(R)$ when $R \gg \theta_{\max}$. Furthermore, we remark that following our analysis, it can be verified that if $\theta_u = \theta_{\max}$ for all nodes $u \in V$ and R is divisible by $\theta_{\max}$, then we can prove the stronger result $\mathsf{OPT}_m(R) = \mathsf{OPT}_f(R)$. Moreover, there exists a mixed strategy $(D, \mathbf{p})$ with $|D| = O(n^2)$ that achieves this defending result. In other words, our analysis also reproduces the result of [Korzhyk *et al.*, 2010].

We prove Theorem 2 by showing the following lemma, whose proof is included in the full version.

Lemma 3 *Given any vector $\mathbf{f} \in [0, 1]^V$ with $\sum_{u\in V} f_u\cdot\theta_u \le R-\theta_{max}$, we can compute in polynomial time a mixed strategy $(D, \mathbf{p}) \in \Omega_m(R)$ with $|D| = O(n^2)$ such that $\mathbf{x}(D, \mathbf{p}) = \mathbf{f}$.*

In particular, let $\tilde{\mathbf{r}}^*$ be the optimal fractional strategy using resource $R - \theta_{\max}$. Note that in the isolated model we have $\pi_u(\tilde{\mathbf{r}}^*) = \tilde{r}^*_u$. Let $\mathbf{f} \in [0, 1]^V$ be defined by $f_u = \min\{\tilde{r}^*_u/\theta_u, 1\}$, for all $u \in V$. That is, f_u is the fraction node u is defended in the fractional strategy. Then $\mathbf{f}$ satisfies the condition of Lemma 3, and hence there exists a mixed strategy $(D, \mathbf{p}) \in \Omega_m(R)$ with $\mathbf{x}(D, \mathbf{p}) = \mathbf{f}$. Hence we have

$$\begin{aligned}&\mathsf{OPT}_m(R) \le L_m(D,\mathbf{p}) = \max_{u\in V}\{(1-x_u(D,\mathbf{p}))\cdot\alpha_u\}\\ =&\max_{u\in V}\{(1-f_u)\cdot\alpha_u\} = L_f(\tilde{\mathbf{r}}^*) = \mathsf{OPT}_f(R-\theta_{\max}).\end{aligned}$$

Mixed Strategy with Resource Sharing. As we have shown above, in the isolated model we can give a strong upper bound $\mathsf{OPT}_m(R)$ by $\mathsf{OPT}_f(R-\theta_{\max})$. It would be natural to ask whether similar upper bounds hold under the non-isolated model, i.e., when defending resource can be shared between neighboring nodes. Unfortunately, we show (in the full version) that when resource sharing is allowed, we do not have such guarantees, even if we allow the mixed strategy to use several times more resource than the fractional strategy.

4 Small Support Mixed Strategies

So far, we evaluate the quality of a mixed strategy only by its defending result without considering its support size $|D|$. In this section, we study the computation of mixed strategies with small support. We propose the Patching algorithm that computes mixed strategies with an upper bound on the support size, but also have good defending results. By Theorem 1, we know that computing the optimal mixed strategy is NP-hard. Moreover, by the reduction we can see that even computing the optimal mixed strategy with $|D| = 2$ is NP-hard. However, we have the following very helpful observations. We show that deciding if a set of nodes can be defended simultaneously using one pure strategy is polynomial-time solvable. Throughout this section we fix $G(V, E)$ to be the graph instance and R to be the total resource.

Lemma 4 *Given a set of nodes $S \subseteq V$, deciding if there exists $\mathbf{r} \in \Omega_p$ with $x_u(\mathbf{r}) = 1$ for all $u \in S$ is polynomial-time solvable. Moreover, if they exist, we can compute one in polynomial time.*

Lemma 5 *Given a set of pure strategies $D \subseteq \Omega_p$, the optimal mixed strategy $(D, \mathbf{p}^*)$ with support D can be computed in time polynomial in $|D|$ and n.*

The Patching Algorithm. Following the above observations, we propose the following algorithm that progressively and efficiently computes a mixed strategy with small support and good defending result. Our algorithm takes as input an iteration bound d, terminates after d search steps and outputs a mixed strategy $(D, \mathbf{p})$ with $|D| \le d$. For convenience of notation we use $L_m(u)$ to denote $L_m(u, (D, \mathbf{p}))$, when the mixed strategy $(D, \mathbf{p})$ is clear from the context.

Intuitively speaking, our algorithm starts from a mixed strategy $(D, \mathbf{p})$ and tries to include a new pure strategy $\mathbf{r}$ into D, so that the optimal mixed strategy with support $D \cup \{\mathbf{r}\}$ is likely to achieve a better defending result. As shown in Lemma 5, as long as $|D|$ is small, computing the optimal mixed strategy with support D can be done efficiently by solving an LP. We

Algorithm 1 Patching

Input: the number of iterations d and optimal pure strategy $\mathbf{r}^*$
Output: mixed strategy set $(D, \mathbf{p})$

for $i = 2, 3, \ldots, d$ **do**
  $\mathbf{p} \leftarrow \mathsf{ProbLP}(D)$
  compute the loss vector L_m of mixed strategy $(D, \mathbf{p})$
  $\mathbf{r} \leftarrow \mathsf{FindR}(L_m)$
  if $\|\mathbf{r}\| \neq 0$ **then**
    $D \leftarrow D \cup \{\mathbf{r}\}$
$\mathbf{p} \leftarrow \mathsf{ProbLP}(D)$
return $(D, \mathbf{p})$

Algorithm 2 FindR

Input: the loss vector L_m with current strategy set D
Output: new pure strategy $\mathbf{r}$

let $M \leftarrow \text{SharedMaxTop}(L_m)$
if $\exists\, \mathbf{r} \in D : x_u(\mathbf{r}) = 1$ for all $u \in M$ **then**
  replace L_m with a random vector in $[0, 1]^V$
  let $M \leftarrow \text{SharedMaxTop}(L_m)$
  if $\exists\, \mathbf{r} \in D : x_u(\mathbf{r}) = 1$ for all $u \in M$ **then**
    return $\{0\}^V$
let $\mathbf{r}$ be the pure strategy that defends all nodes in M
return $\mathbf{r}$

denote this sub-routine by $\mathsf{ProbLP}(D)$. Our main idea is to add the new strategy to patch the poorly defended nodes up based on their current losses. Borrowing some ideas from the proof of Lemma 3, we compute the maximal set of nodes M with largest losses under the current mixed strategy $(D, \mathbf{p})$, and use Lemma 4 to compute a new pure strategy that defends these nodes. As we will show in the next section, as long as the maximum loss of nodes in M is larger than that of nodes not in M, our algorithm can always make progress in decreasing the defending result. Otherwise we randomly permute the nodes in V and try to include a random new pure strategy into D. We introduce the $\mathsf{FindR}(L_m)$ subroutine for the computation of the new pure strategy, for a given loss vector L_m. Note that if it fails to compute a new pure strategy, an all-zero vector will be returned. We summarize the main steps of the Patching algorithm in Algorithm 1. Initially we set the strategy set D to be a singleton containing only the optimal pure strategy and the algorithm terminates after d iterations.

Next we introduce the details of the sub-routine FindR. As discussed, given the loss vector L_m, the idea is to first locate the nodes with large losses and then generate a new pure strategy that enhances the defending statuses of these poorly defended nodes. We thus use similar ideas as in the proof of Lemma 3 to compute the maximal set of nodes to be defended. However, since resource sharing is considered, the procedure is slightly more complicated. Given any integer $k > 0$, we can identify the Top-k nodes S with maximal losses in L_m, and check whether it is possible to defend all nodes in S by solving an LP (see Lemma 4). Using a binary search on k we can identify the maximum k for which the corresponding set of nodes S can be defended. Let $\text{SharedMaxTop}(L_m)$ be these nodes (the details of the algorithm can be found in the full version). The sub-routine $\mathsf{FindR}(L_m)$ first computes $M \leftarrow \text{SharedMaxTop}(L_m)$ and tries to include the pure strategy $\mathbf{r}$ that defends all nodes in M. If there already exists a strategy in D that defends all nodes in M, then it is unnecessary to include $\mathbf{r}$ because its inclusion will not help in decreasing the defending result. In such case we do a random permutation on V (by replacing L_m with a random vector in $[0, 1]^V$), and compute another pure strategy. As mentioned, if we fail to find a new pure strategy after the random permutation, then the sub-routine returns the trivial defending strategy $\{0\}^V$. We summarize the steps of $\mathsf{FindR}(L_m)$ in Algorithm 2.

5 Experimental Evaluation

In this section we perform the experimental evaluation of our algorithms on several real-world graph datasets whose sizes range from 1000 nodes to 260k nodes (see the following table). All the datasets are downloaded from SNAP by Stanford [Leskovec and Krevl, 2014]. Unless otherwise specified, we set the parameters of instances as follows.[3] For each of these datasets, we set the value α_u of each node u to be an independent random integer chosen uniformly at random from $[1, 9]$. We set the threshold θ_u of each node u to be an independent random real number chosen uniformly at random from $[1, 10]$. We set the weight w_{uv} of each edge $(u, v) \in E$ to be an independent random real number chosen from $[0, 1]$. We set the total resource $R = 0.2 \cdot \sum_{u \in V} \theta_u$.

Dataset	Email-S	Facebook	Ca-AstroPh	Email-L	Twitter	Amazon
# Nodes	1,005	4,039	18,772	36,692	81,306	262,111
# Edges	27,551	88,234	198,110	367,662	1,768,149	1,234,877

In the experiments we mainly evaluate the effectiveness of the Patching algorithm. Additionally we test and report the mixed strategies we stated in Theorem 2, and compare their defending results and support sizes with that of the mixed strategies returned by Patching. As we have shown in Section 3, for the same problem instance we always have $\mathsf{OPT}_m \geq \mathsf{OPT}_f$. Thus in our experiments we mainly use OPT_f as the baseline to evaluate the performance of the mixed strategies.

Uniform Threshold in the Isolated Model. We first consider the most basic setting with uniform threshold and without resource sharing. The same setting was also considered in [Kiekintveld *et al.*, 2009; Korzhyk *et al.*, 2010]. That is, we set $\theta_u = 1$ for all nodes $u \in V$ and $w_{uv} = 0$ for all edges $(u, v) \in E$, in this subsection. Under this setting it can be shown that $\mathsf{OPT}_m = \mathsf{OPT}_f$ for all instances of the problem [Kiekintveld *et al.*, 2009; Korzhyk *et al.*, 2010]. Thus we measure the effectiveness of the Patching algorithm by comparing the defending result of the returned mixed strategy with OPT_f. For each dataset, we report the defending result of the mixed strategy returned by $\mathsf{Patching}(d)$, for increasing values of $d \in [1, 30]$. Note that for $d = 1$, the mixed strategy uses the optimal pure strategy $\mathbf{r}^*$ with probability 1. For general d, the returned mixed strategy uses $|D| \leq d$ pure strategies. The results are presented as Figure 1.

[3] We remark that for different settings of the parameters, e.g., wider ranges for the values and thresholds, or smaller values of R, the experimental results are very similar.

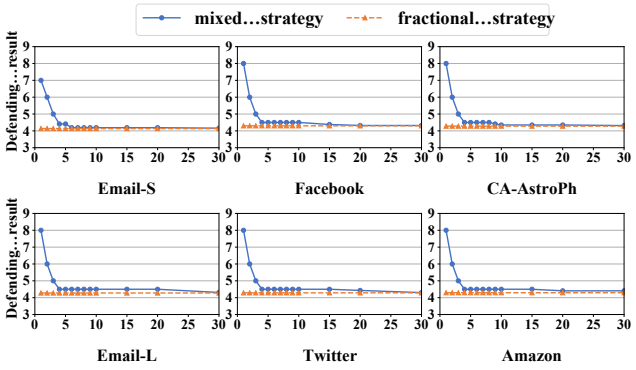

Figure 1: Defending Results of Mixed Strategies by Patching.

From Figure 1, we observe that the defending result rapidly decreases in the first few iterations of the algorithm, and gradually converges to OPT_f. Specifically, within the first 5 iterations, the defending result of the mixed strategy is already within a 5% difference with the optimal one in most datasets. After 30 iterations, the defending result is almost identical to the optimal one in all datasets. The experiment demonstrates the effectiveness of our algorithm on computing mixed strategies that have small support sizes and are close-to-optimal.

General Thresholds in the Isolated Model. Next we consider the more general setting with non-uniform thresholds, e.g., $\theta_u \in [1, 10]$ is chosen uniformly at random for each node $u \in V$. As we have shown in Theorem 1, computing the optimal mixed strategy in this case is NP-hard. On the other hand, we have shown in Theorem 2 that for any instance we always have $\mathsf{OPT}_f(R) \le \mathsf{OPT}_m(R) \le \mathsf{OPT}_f(R - \theta_{\max})$, where the maximum threshold $\theta_{\max} \le 10$ in our experiments. The experimental results are reported in Figure 2.

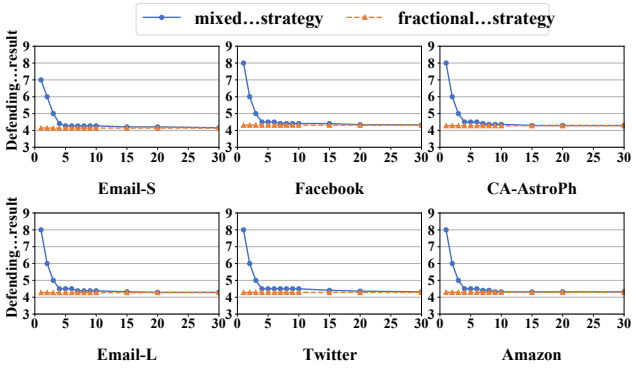

Figure 2: Defending Results of Mixed Strategies by Patching.

As we can observe from Figure 2, the result is very similar to the uniform threshold case we have considered in the previous experiments. This also confirms our theoretical analysis in Theorem 2: when R is sufficiently large (compared with $\theta_{\max}$), the optimal mixed strategy should have defending result close to $\mathsf{OPT}_f(R)$. Furthermore, the experiment demonstrates that even for general thresholds, Patching returns mixed strategies that are close-to-optimal, and uses very few pure strategies.

Recall that in Theorem 2 we show that there exists a mixed strategy $(D, \mathbf{p}) \in \Omega_m(R)$ whose defending result $L_m(D, \mathbf{p}) = \mathsf{OPT}_f(R - \theta_{\max})$. In our experiments, we implement the algorithm and report the defending result and the support size to verify its correctness. The results are presented in Table 1, where $(D, \mathbf{p})$ is the mixed strategy our algorithm computes, and $(D, \mathbf{p}^*)$ is the optimal mixed strategy with support D (which can be computed by solving an LP, as we have shown in Lemma 5). Note that due to long computation time, we did not finish the computing of $L_m(D, \mathbf{p}^*)$ for the Twitter and Amazon datasets (too many variables). In the table, we also compare them with the mixed strategies returned by Patching(d) with $d \in \{5, 30\}$. Consistent with our theoretical analysis, for all datasets we have $\mathsf{OPT}_f(R - \theta_{\max}) = L_m(D, \mathbf{p}) \ge L_m(D, \mathbf{p}^*) \ge \mathsf{OPT}_f(R)$. From the above results we can see that compared to $(D, \mathbf{p})$, the Patching algorithm is able to compute mixed strategies with very small support, e.g., 30 vs. 2500+ for large datasets, while guaranteeing a defending result that is very close.

	Email-S	Facebook	Ca-AstroPh	Email-L	Twitter	Amazon
$\mathsf{OPT}_f(R - \theta_{\max})$	4.161	4.32	4.281	4.273	4.285	4.293
$L_m(D, \mathbf{p})$	4.161	4.32	4.281	4.273	4.285	4.293
$L_m(D, \mathbf{p}^*)$	4.147	4.316	4.28	4.273	–	–
$\lvert D \rvert$	268	671	1900	2592	6052	9957
$\mathsf{OPT}_f(R)$	4.139	4.314	4.28	4.273	4.285	4.293
Patching(5)	4.41	4.5	4.5	4.5	4.5	4.5
Patching(30)	4.161	4.326	4.29	4.291	4.324	4.319

Table 1: Different Mixed Strategies in Different Datasets

We also compare the time to compute $L_m(D, \mathbf{p})$ and the running time (in seconds) of Patching in Table 2. As we can observe from the table, compared to the computation of $L_m(D, \mathbf{p})$, the running time of Patching is less sensitive to the size of the network. Consequently for large datasets the Patching algorithm runs several times faster than computing $L_m(D, \mathbf{p})$. In conclusion, the Patching algorithm computes mixed strategies that use way fewer pure strategies than $(D, \mathbf{p})$ while having similar defending results. Furthermore, its running time is also much smaller in large datasets.

	Email-S	Facebook	Ca-AstroPh	Email-L	Twitter	Amazon
$L_m(D, \mathbf{p})$	0.2	3.2	57	214	1027	9370
Patching(5)	0.3	0.8	3.3	6.5	14	45
Patching(30)	2.3	8.2	37	73	165	523

Table 2: Running Time Comparison (in seconds)

For space limits, we include the experimental results regarding the resource sharing model in the full version.

Acknowledgments

Xiaowei Wu is funded by FDCT (0143/2020/A3, SKL-IOTSC-2021-2023), the SRG of University of Macau (SRG2020-00020-IOTSC) and GDST (2020B1212030003). Weijia Jia is supported by Guangdong Key Lab of AI and Multi-modal Data Processing, BNU-HKBU United International College (UIC), Zhuhai, No. 2020KSYS007; Chinese National Research Fund (NSFC), No. 61872239; Zhuhai Science-Tech Innovation Bureau, Nos. ZH22017001210119PWC and 28712217900001 and Guangdong Engineering Center for Artificial Intelligence

and Future Education, Beijing Normal University, Zhuhai, Guangdong, China.

References

[Acemoglu *et al.*, 2016] Daron Acemoglu, Azarakhsh Malekian, and Asuman E. Ozdaglar. Network security and contagion. *J. Econ. Theory*, 166:536–585, 2016.

[Alshamsi *et al.*, 2018] Aamena Alshamsi, Flávio L Pinheiro, and Cesar A Hidalgo. Optimal diversification strategies in the networks of related products and of related research areas. *Nature communications*, 9(1):1328, 2018.

[Aspnes *et al.*, 2006] James Aspnes, Kevin L. Chang, and Aleksandr Yampolskiy. Inoculation strategies for victims of viruses and the sum-of-squares partition problem. *J. Comput. Syst. Sci.*, 72(6):1077–1093, 2006.

[Bai *et al.*, 2021] Rufan Bai, Haoxing Lin, Xinyu Yang, Xiaowei Wu, Minming Li, and Weijia Jia. Defending against contagious attacks on a network with resource reallocation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 2021.

[Basilico *et al.*, 2009] Nicola Basilico, Nicola Gatti, and Francesco Amigoni. Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *AAMAS (1)*, pages 57–64. IFAAMAS, 2009.

[Conitzer, 2012] Vincent Conitzer. Computing game-theoretic solutions and applications to security. In *AAAI*. AAAI Press, 2012.

[Fang *et al.*, 2015] Fei Fang, Peter Stone, and Milind Tambe. When security games go green: Designing defender strategies to prevent poaching and illegal fishing. In *IJCAI*, pages 2589–2595. AAAI Press, 2015.

[Gan *et al.*, 2017] Jiarui Gan, Bo An, Yevgeniy Vorobeychik, and Brian Gauch. Security games on a plane. In *AAAI*, pages 530–536. AAAI Press, 2017.

[Gan *et al.*, 2018] Jiarui Gan, Edith Elkind, and Michael J. Wooldridge. Stackelberg security games with multiple uncoordinated defenders. In *AAMAS*, pages 703–711. International Foundation for Autonomous Agents and Multiagent Systems Richland, SC, USA / ACM, 2018.

[Gan *et al.*, 2019] Jiarui Gan, Haifeng Xu, Qingyu Guo, Long Tran-Thanh, Zinovi Rabinovich, and Michael J. Wooldridge. Imitative follower deception in stackelberg games. In *EC*, pages 639–657. ACM, 2019.

[Gan *et al.*, 2020] Jiarui Gan, Edith Elkind, Sarit Kraus, and Michael J. Wooldridge. Mechanism design for defense coordination in security games. In *AAMAS*, pages 402–410. International Foundation for Autonomous Agents and Multiagent Systems, 2020.

[Jain *et al.*, 2011] Manish Jain, Dmytro Korzhyk, Ondrej Vanek, Vincent Conitzer, Michal Pechoucek, and Milind Tambe. A double oracle algorithm for zero-sum security games on graphs. In *AAMAS*, pages 327–334, 2011.

[Jain *et al.*, 2013] Manish Jain, Vincent Conitzer, and Milind Tambe. Security scheduling for real-world networks. In *AAMAS*, pages 215–222. IFAAMAS, 2013.

[Kiekintveld *et al.*, 2009] Christopher Kiekintveld, Manish Jain, Jason Tsai, James Pita, Fernando Ordóñez, and Milind Tambe. Computing optimal randomized resource allocations for massive security games. In *AAMAS (1)*, pages 689–696. IFAAMAS, 2009.

[Korzhyk *et al.*, 2010] Dmytro Korzhyk, Vincent Conitzer, and Ronald Parr. Complexity of computing optimal stackelberg strategies in security resource allocation games. In *AAAI*. AAAI Press, 2010.

[Kumar *et al.*, 2010] V. S. Anil Kumar, Rajmohan Rajaraman, Zhifeng Sun, and Ravi Sundaram. Existence theorems and approximation algorithms for generalized network security games. In *ICDCS*, pages 348–357. IEEE Computer Society, 2010.

[Leskovec and Krevl, 2014] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014. Accessed: 2022-01-01.

[Li *et al.*, 2020] Minming Li, Long Tran-Thanh, and Xiaowei Wu. Defending with shared resources on a network. In *AAAI*, pages 2111–2118. AAAI Press, 2020.

[Lou and Vorobeychik, 2015] Jian Lou and Yevgeniy Vorobeychik. Equilibrium analysis of multi-defender security games. In *IJCAI*, pages 596–602. AAAI Press, 2015.

[Nguyen *et al.*, 2009] Kien C. Nguyen, Tansu Alpcan, and Tamer Basar. Security games with incomplete information. In *ICC*, pages 1–6. IEEE, 2009.

[Nguyen *et al.*, 2013] Thanh Hong Nguyen, Rong Yang, Amos Azaria, Sarit Kraus, and Milind Tambe. Analyzing the effectiveness of adversary modeling in security games. In *AAAI*. AAAI Press, 2013.

[Shieh *et al.*, 2012] Eric Shieh, Bo An, Rong Yang, Milind Tambe, Craig Baldwin, Joseph DiRenzo, Ben Maule, and Garrett Meyer. PROTECT: a deployed game theoretic system to protect the ports of the united states. In *AAMAS*, pages 13–20. IFAAMAS, 2012.

[Sinha *et al.*, 2018] Arunesh Sinha, Fei Fang, Bo An, Christopher Kiekintveld, and Milind Tambe. Stackelberg security games: Looking beyond a decade of success. In *IJCAI*, pages 5494–5501. ijcai.org, 2018.

[Tsai *et al.*, 2012] Jason Tsai, Thanh Hong Nguyen, and Milind Tambe. Security games for controlling contagion. In *AAAI*. AAAI Press, 2012.

[Vorobeychik *et al.*, 2014] Yevgeniy Vorobeychik, Bo An, Milind Tambe, and Satinder P. Singh. Computing solutions in infinite-horizon discounted adversarial patrolling games. In *ICAPS*. AAAI, 2014.

[Wang *et al.*, 2016] Zhen Wang, Yue Yin, and Bo An. Computing optimal monitoring strategy for detecting terrorist plots. In *AAAI*, pages 637–643. AAAI Press, 2016.

[Yin *et al.*, 2015] Yue Yin, Haifeng Xu, Jiarui Gan, Bo An, and Albert Xin Jiang. Computing optimal mixed strategies for security games with dynamic payoffs. In *IJCAI*, pages 681–688. AAAI Press, 2015.