Defending against Contagious Attacks on a Network with Resource Reallocation*

Rufan Bai[1], **Haoxing Lin**[1], **Xinyu Yang**[1], **Xiaowei Wu**[1], **Minming Li**[†2], **Weijia Jia**[‡3]
[1]IoTSC, University of Macau,
[2]City University of Hong Kong,
[3]BNU (Zhuhai) & UIC,
{yb97439, starklin, mb95466, xiaoweiwu}@um.edu.mo, minming.li@cityu.edu.hk, jiawj@sjtu.edu.cn

Abstract

In classic network security games, the defender distributes defending resources to the nodes of the network, and the attacker attacks a node, with the objective to maximize the damage caused. Existing models assume that the attack at node u causes damage only at u. However, in many real-world security scenarios, the attack at a node u spreads to the neighbors of u and can cause damage at multiple nodes, e.g., for the outbreak of a virus. In this paper, we consider the network defending problem against contagious attacks.

Existing works that study shared resources assume that the resource allocated to a node can be shared or duplicated between neighboring nodes. However, in real world, sharing resource naturally leads to a decrease in defending power of the source node, especially when defending against contagious attacks. To this end, we study the model in which resources allocated to a node can only be transferred to its neighboring nodes, which we refer to as a reallocation process.

We show that this more general model is difficult in two aspects: (1) even for a fixed allocation of resources, we show that computing the optimal reallocation is NP-hard; (2) for the case when reallocation is not allowed, we show that computing the optimal allocation (against contagious attack) is also NP-hard. For positive results, we give a mixed integer linear program formulation for the problem and a bi-criteria approximation algorithm. Our experimental results demonstrate that the allocation and reallocation strategies our algorithm computes perform well in terms of minimizing the damage due to contagious attacks.

Introduction

In recent years, security games have attracted much research attention within the artificial intelligence community and have been widely adopted for the computation of optimal allocation of security resources in many areas of the field (Letchford, Conitzer, and Munagala 2009; Tambe 2012; Yin and Tambe 2012; Sinha et al. 2018). A considerable portion of these works consider the security games played within a network structure, i.e., the network security games (Assimakopoulos 1987; Gan et al. 2017; Zhang et al. 2017; Schlenker et al. 2018). In a network security game, there is an underlying graph, where each node of the graph represents a target with a defending requirement and a value to protect. The game is played between a defender who allocates defensive resources to the nodes of the graph and an attacker who picks a node to attack, depending on how the nodes are defended.

Many existing works consider the setting when the allocated resource can be shared between neighboring nodes (Yin et al. 2015). For example, Gan et al. (Gan, An, and Vorobeychik 2015) considered a network security game in which allocating one unit of resource to some target protects not only the target but also the neighboring targets. Li et al. (Li, Tran-Thanh, and Wu 2020) studied the model in which the defending power of each node u is determined by the resource r_u allocated to u, plus a linear function of the resources allocated to its neighbors. These models are mainly motivated by surveillance or patrolling applications, in which when a node u shares resource with its neighbor, we do not need to worry about the defending power of u.

However, for defending problems in which the attack is contagious, it is necessary to take into account the decrease in the defending power of node u, especially when u is at the risk of being involved in the attack. Consider a contagious attack, e.g., the spread of a virus, on a node v. Suppose the attack spreads to the neighbors of v and can cause damage at each of the nodes the attack spreads to, depending on how well the node is defended. In this case, if we measure the defending power of v by taking into account the resources shared from its neighbor u, then naturally, we need to consider the decrease in the defending power of u.

Ideally, a node u can only *transfer* (a fraction of) the resource it owns to its neighbor v, which increases the defending power of the receiver v but decreases its own defending

*Funded by the Science and Technology Development Fund, Macau SAR (File no. SKLIOTSC-2018-2020), the Start-up Research Grant of University of Macau (File no. SRG2020-00020-IOTSC). This work was supported in part by the Science and Technology Development Fund, Macau SAR under File no. 0060/2019/A1, and in part by Research Grant of University of Macau under Grant MYRG2018-00237-FST.

†City University of Hong Kong Shenzhen Research Institute, Shenzhen, P. R. China. The work described in this paper was partially sponsored by Project 11771365 supported by NSFC.

‡BNU-UIC Institute of Artificial Intelligence and Future Networks, Beijing Normal University (Zhuhai), Guangdong, China. The work was partially supported by Chinese National Research Fund (NSFC) Key Project No. 61532013; NSFC grant No. 61872239; and Guangdong Provincial Key Lab of AI and Multimodal Data Processing at BNU-HKBU UIC.

power. When defending against attacks without spreading effects, this assumption is equivalent to being able to duplicate resources between neighbor nodes, as we can always transfer the maximum possible resources towards the node under attack. However, when the attack can spread to neighbors of the node under attack, this assumption demands a stronger defending requirement. Specifically, the following example shows that when resources can only be transferred (instead of being duplicated), the total resource required to obtain a good defending result can be much larger.

Example 1 *Consider a star graph, with node u in the center, and $v_1, v_2, \ldots, v_{n-1}$ being neighbors of u. Suppose each node requires 1 unit of resource to defend himself. Suppose node u is attacked and the attack spreads to all neighbors of u. When resources can be duplicated, allocating one unit of resource at node u guarantees that every node is sufficiently defended, and thus no loss is incurred. However, when resources can only be transferred, as long as the total resources allocated are less than n units, there always exists at least one insufficiently defended node.*

In the paper, we consider the problem of defending against contagious attack, in which the defending resources can only be transferred between neighboring nodes. Specifically, when the attacker attacks a node u in the network, the attack spreads to neighbors of u and may cause damage at multiple nodes. The defender decides an allocation strategy of defending resources to nodes in the graph before the attack happens, and is allowed to transfer some resources between neighboring nodes (subject to some capacity constraints) when the attack happens. Our model is motivated by real-world applications like defending against virus spreading. In these applications, it is reasonable to assume that we can transfer medical resources or doctors between neighboring cities or countries in order to minimize the damage when the virus breaks out. Unfortunately, existing models fail to capture such applications as most of them do not consider the reallocation of defending resources.

Our Results

We study the problem of computing optimal allocations and reallocations of defending resources. Since our main motivation of the problem is defending against virus spreading and, in real world, the allocation of defending resources is usually public information, we focus only on pure strategies, i.e., deterministic defending algorithms. We propose a mathematical model that generalizes that of (Gan, An, and Vorobeychik 2015; Li, Tran-Thanh, and Wu 2020), and assume that (1) an attack spreads to a subset of nodes and may cause damage at each of them; (2) defending resources can be transferred between neighboring nodes, which we refer to as a reallocation of resources. The objective is to minimize the maximum possible damage due to an attack.

We show that this general model is difficult in two aspects. We first show that even with a given allocation of resources and a node that is attacked, computing the optimal reallocation is NP-hard Then we show that if no reallocation is allowed, the problem of computing the optimal allocation strategy is also NP-hard.

Regarding positive results, we provide mixed integer linear programs (MILPs) to model the computation of allocation and reallocation strategies. We show that the optimal solutions for the MILPs provide optimal allocation and reallocation strategies. Since solving an MILP is not guaranteed to terminate in polynomial time, we also propose polynomial time algorithms for special cases and approximation algorithms. We give a polynomial time algorithm that decides whether there exists a defending strategy in which no loss incurs, and outputs one if it exists. Then we give a polynomial time bi-criteria $(\frac{1}{1-\epsilon}, \frac{1}{\epsilon})$-approximation algorithm, for any $\epsilon \in (0, 1)$ Specifically, for $\epsilon = 0.5$ we have a bi-criteria $(2, 2)$-approximation. Moreover, we show that under the Unique Game Conjecture (Khot and Regev 2008), there does not exist $(2-\delta, 2-\delta)$-approximation, for any constant $\delta > 0$.

Finally, we extensively evaluate our algorithms on synthetic and real-world datasets.

Other Related Work

As mentioned, there is a sequence of existing works in the network security game domain that consider resource sharing between nodes. Gan et al. (Gan, An, and Vorobeychik 2015; Gan et al. 2017) consider models in which allocating a unit of defending resource to a node can also protect the neighbors of that node. Their models only study the binary version of resource allocation, i.e., $r_u \in \{0, 1\}$. Yin et al. (Yin et al. 2015) also study a model in which the resource can be shared, and they assume sharing resources takes time. However, these existing models does not consider the contagious attacks or the resource reallocation.

There are also works that study the contagion in network security games (Nguyen, Alpcan, and Basar 2009; Bachrach, Draief, and Goyal 2013; Vorobeychik and Letchford 2015; Acemoglu, Malekian, and Ozdaglar 2016; Lou, Smith, and Vorobeychik 2017; Goyal and Vigier 2014; Aspnes, Chang, and Yampolskiy 2006). Besides, Tsai et al. (Tsai, Nguyen, and Tambe 2012) study a zero-sum two-player influence blocking maximization game, in which the attacker and the defender try to maximize their influence on a network. However, these works do not model the problem in terms of allocating defending resources to meet defending requirements and minimizing the loss due to attack, and thus are incomparable to our model. There are other works that study contagion of attack by assuming that an insufficiently protected node can affect the defending result of its neighboring nodes (Chan, Ceyko, and Ortiz 2017; Li, Tran-Thanh, and Wu 2020). There are also works that study game-theoretic models of the security games (Kunreuther and Heal 2003; Johnson et al. 2010; Chan, Ceyko, and Ortiz 2012).

Model Description

We model the network as an undirected[1] 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.

[1] While we assume the graph is undirected, it can be verified that all our results extend straightforwardly to directed graphs.

We use $N(u) := \{v : (u,v) \in E\}$ to denote the set of neighbors for node $u \in V$. We use $N_k(u)$ to denote the set of nodes at distance at most k from $u \in V$. By definition we have $N_1(u) = \{u\} \cup N(u)$. We use n and m to denote the number of nodes and edges in the graph G, respectively.

Defending Resource and Defending Power

The defender has a total resource of R that can be distributed to nodes in V, where r_u is the *defending resource*[2] allocated to node u, and $\sum_{u \in V} r_u = R$. Each node u can transfer at most $w_{uv} \cdot r_u$ units of defending resource to each of its neighbor v, where $w_{uv} \in [0,1]$ is the *weight* of edge (u,v), which represents the efficiency (or willingness) when transferring defending resource between u and v.

Definition 1 (Allocation Strategy) *We use $r_u \geq 0$ to denote the resource allocated to node u. We use $\mathbf{r} = \{r_u\}_{u \in V}$ to denote an* allocation strategy.

Definition 2 (Reallocation Strategy) *We use $t(u,v) \geq 0$ to denote the resource u transfers to its neighbor v. In general v can also send resource to node u (which is denoted by $t(v,u) \geq 0$). We use $\mathbf{t} = \{t(u,v), t(v,u)\}_{(u,v) \in E}$ to denote a* reallocation strategy.

The fractions of resource transferred between u and v are upper bounded by the edge weight as follows:

$$t(u,v) \leq w_{uv} \cdot r_u, \qquad t(v,u) \leq w_{uv} \cdot r_v.$$

That is, each node u can transfer at most w_{uv} fraction of the resource r_u to its neighbor v. Additionally, we need to guarantee that the total resources node u sends out is at most the total resource it owns:

$$\textstyle\sum_{v \in N(u)} t(u,v) \leq r_u.$$

Since the resources can be sent and received, the defending power of a node is not fixed. Instead, depending on the attack, the defending power at each node can be adaptive by deciding an appropriate reallocation strategy.

Definition 3 (Defending Power) *The defending power of node u is defined as the total resource node u owns after the reallocation, which is given as follows:*

$$p_u = r_u - \sum_{v \in N(u)} t(u,v) + \sum_{v \in N(u)} t(v,u).$$

We use $\mathbf{p} = \{p_u\}_{u \in V}$ to denote defending powers of nodes.

Depending on the reallocation, the defending power p_u of node u can take values in range $[\bar{p}_u, \hat{p}_u]$, where

$$\bar{p}_u = \max\{1 - \textstyle\sum_{v \in N(u)} w_{uv}, 0\} \cdot r_u,$$
$$\hat{p}_u = r_u + \textstyle\sum_{v \in N(u)} w_{uv} \cdot r_v.$$

Note that the allocation strategy $\mathbf{r}$ (which allocates the defending resources) must be decided before the attack happens. In contrast, the defender can decide the reallocation strategy depending on which node is attacked. Specifically, the defender can define n reallocation strategies $\{\mathbf{t}^u\}_{u \in V}$, one for each node when it is attacked.

Put differently, there are four sequential steps:

(1) the algorithm decides an allocation strategy $\mathbf{r}$, which allocates a total of R resources;

(2) the attacker picks a node u to attack;

(3) the algorithm decides a reallocation strategy $\mathbf{t}^u$ to minimize the loss due to the attack. Note that at this point, the allocation strategy is fixed, but the defending power depends on the reallocation strategy.

(4) the loss due to the attack is evaluated.

Definition 4 (Defending Strategy) *We refer to a solution for the defending problem as a* defending strategy *$(\mathbf{r}, \{\mathbf{t}^u\}_{u \in V})$, which consists of an allocation strategy $\mathbf{r}$ and n reallocation strategies $\{\mathbf{t}^u\}_{u \in V}$.*

Next, we define the loss due to an attack. Let $\mathbf{p} = \{p_u\}_{u \in V}$ be the defending powers of nodes. Suppose u is attacked, the attack spreads to all nodes in $N_k(u)$, where k is a parameter that represents the level of contagiousness of the attack. The loss due to the attack is the total damage caused at nodes in $N_k(u)$, where each node $v \in N_k(u)$ suffers from a damage of α_v if $p_v < \theta_v$. If $p_v \geq \theta_v$, then no damage is caused at v.

Definition 5 (Defending Result) *Given defending strategy $(\mathbf{r}, \{\mathbf{t}^u\}_{u \in V})$, let $\mathsf{Loss}(u)$ be the total damage when u is attacked and the reallocation strategy $\mathbf{t}^u$ is deployed. The defending result is defined as the maximum loss due to an attack, i.e., $\max_{u \in V} \mathsf{Loss}(u)$.*

The objective of the problem is to compute a defending strategy with the minimum defending result. We use OPT to denote the optimal (minimum) defending result. In the remaining part of the paper, we use DCA (Defending against Contagious Attack) to refer to the problem of computing the defending strategy against contagious attack. Note that the decision problem of verifying whether a defending strategy has result at most some value is in NP. Given the defending strategy, the verification can be done by computing $\mathsf{Loss}(u)$ for every node u and taking the maximum, both of which take polynomial time.

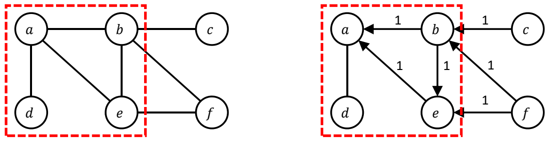
When $k = 0$, there is no spreading effect and we only need to protect the node under attack by borrowing defending resources from its neighbors. Hence in this case we have $p_u = \hat{p}_u$ if node u is attacked. Consequently, the problem degenerates to the single-threshold model of (Li, Tran-Thanh, and Wu 2020), which can be solved in polynomial time. However, in general (when $k \geq 1$), when the attack spreads to multiple nodes, the reallocation must be carefully designed so as to protect multiple nodes, because when a node transfers resource to its neighbors, its own defending power decreases.

Optimal Response to an Attack

As a warm-up towards further analysis, in this section, we first focus on the subproblem of computing optimal reallocations. That is, given a fixed allocation strategy $\mathbf{r} = \{r_u\}_{u \in V}$ and suppose node u is under attack, we compute the reallocation strategy $\mathbf{t}^u$ with which $\mathsf{Loss}(u)$ is minimized. The following example shows how an appropriate reallocation of resources helps reduce the damage due to an attack.

[2] Similar to (Li, Tran-Thanh, and Wu 2020), we assume the resource can be allocated continuously in our model.

Example 2 *Consider the graph given in Figure 1(a), and node a is under attack. Assuming $k = 1$, the attack spreads to $N_1(a) = \{a, b, d, e\}$. Suppose (1) all edges have weight 0.5; (2) $\theta_a = 4, \theta_b = \theta_d = 2$ and $\theta_e = 3$; and (3) all nodes have defending resource 2. Obviously, without any reallocation, we suffer from a total loss of $\alpha_a + \alpha_e$ since only nodes b and d are sufficiently defended. However, if we reallocate the resources as shown in Figure 1(b), then all nodes in $N_1(a)$ are well defended, and no loss incurs.*



(a) (b)

Figure 1: Example of a reallocation strategy, where a directed edge indicates a transfer of resource. For example, the edge from b to a with value 1 indicates that node b transfers $t(b, a) = 1$ unit of resource to node a.

However, in general, we cannot guarantee that there always exists a reallocation strategy under which all nodes under attack are well defended. In this case, we need to compute a reallocation strategy to minimize the total loss. For example, we can choose to protect nodes u with larger value α_u while leaving some nodes v with smaller α_v insufficiently defended. Unfortunately, we show that the problem of computing the optimal reallocation strategy is NP-hard. For space reasons, we move the proof of the following hardness result to the full version of the paper.

Theorem 1 *Unless* P=NP, *there does not exist any polynomial time algorithm that, given an allocation strategy and a node under attack, computes the optimal reallocation strategy, for any $k \geq 1$.*

Next, we formulate the problem of computing the optimal reallocation strategy as a Mixed Integer Linear Program (MILP). Recall that we are given an allocation strategy $\mathbf{r}$ and a node u that is attacked.

$$\begin{aligned}
\text{minimize} \quad & \textstyle\sum_{v \in N_k(u)} (1 - x_v) \cdot \alpha_v \\
\text{subject to} \quad & r_v - \textstyle\sum_{z \in N(v)} t(v, z) + \sum_{z \in N(v)} t(z, v) \\
& \qquad \geq \theta_v \cdot x_v, \quad \forall v \in N_k(u) \qquad (1) \\
& 0 \leq t(v, z) \leq w_{vz} \cdot r_v, \quad \forall z, v \in V \qquad (2) \\
& \textstyle\sum_{z \in N(v)} t(v, z) \leq r_v, \quad \forall v \in V \qquad (3) \\
& x_v \in \{0, 1\}, \quad \forall v \in N_k(u).
\end{aligned}$$

For each node $v \in N_k(u)$ we introduce an integer variable $x_v \in \{0, 1\}$ that indicates whether $p_v \geq \theta_v$. We introduce fractional variables $t(v, z), t(z, v)$ for each $(v, z) \in E$. The objective of the MILP is the total loss due to the attack, which is the sum of values α_v for $v \in N_k(u)$ that is not well defended ($x_v = 0$). Constraints (1) guarantee that if we set $x_v = 1$, then v should be well defended, i.e., $p_v \geq \theta_v$. Constraints (2) and (3) ensure that the transfers of resource between neighboring nodes are feasible.

Note that $\{r_v\}_{v \in V}$ are given and are not variables. The optimal solution $(\mathbf{x}, \mathbf{t})$ for the MILP gives an optimal reallocation $\mathbf{t}$ that minimizes $\mathsf{Loss}(u)$, with the fixed allocation $\mathbf{r}$ and node u that is attacked.

There are redundant variables that can be removed from the MILP. Recall that $N_k(u)$ are the nodes the attack spreads to. For each $v \in V \setminus N_k(u)$, we have no defending requirements and thus do not need to transfer any resources towards these nodes. Consequently, it is unnecessary to introduce variable $t(z, v)$, for any $z \in N(v)$. In other words, we only introduce the variable $t(z, v)$ if $v \in N_k(u)$. With this observation, we can reduce the total number of fractional variables from $|E|$ to $\sum_{v \in N_k(u)} |N(v)|$, which is much smaller when k is small and the graph is sparse.

Note that the MILP can not be solved exactly in time polynomial in $|N_k(u)|$. A natural idea is to relax the integer variables $\mathbf{x}$ to take values in $[0, 1]$. However, the following instance shows that the integrality gap between the MILP and its LP relaxation is unbounded.

Example 3 (Integrality Gap) *Consider the trivial graph with only one node u, where $\theta_u = \alpha_u = 1$. Suppose $R = r_u = 1 - \epsilon$, where $\epsilon > 0$ is arbitrarily small. Obviously we have $\mathsf{Loss}(u) = 1$. However, the optimal objective of the LP relaxation is ϵ, by setting $x_u = 1 - \epsilon$.*

While the integrality gap of MILP and its LP relaxation is unbounded, we still have two useful observations. First, the optimal objective of the LP relaxation provides a lower bound on the optimal objective of the MILP, which will be utilized to do a pruning on the MILP in later sections. Second, for a fixed $\{0, 1\}$-vector $\mathbf{x} \in \{0, 1\}^{N_k(u)}$, the MILP becomes a feasibility LP, which can be solved efficiently. For example, we use this idea to compute defending strategies with defending result 0 in Section 4. We also extend this idea in Section 4 to compute a polynomial time bi-criteria approximation. The idea is to find a vector $\mathbf{x} \in \{0, 1\}^{N_k(u)}$ for which the induced LP is feasible, and the objective $\sum_{v \in V} (1 - x_v) \cdot \alpha_v$ is as small as possible.

Computing the Defending Strategy

In this section, we consider the computation of defending strategies and extend the observations and ideas from the previous section. Recall that the defending result is $\max_{u \in V} \mathsf{Loss}(u)$, and is uniquely determined by the defending strategy $(\mathbf{r}, \{\mathbf{t}^u\}_{u \in V})$. We have shown in Theorem 1 that given a fixed allocation strategy and a node under attack, computing the optimal reallocation strategy is NP-hard. However, the hardness result does not necessarily imply a hardness result for computing the allocation strategy. In the following, we show that computing the allocation strategy is indeed NP-hard.

Hardness

We first define a simple special case of the DCA problem called *isolated model*, and then show that even for this special case, the problem is NP-hard.

Definition 6 (Isolated Model) *We refer to the DCA problem where $w_{uv} = 0$ for all $(u, v) \in E$ as the* isolated model.

Note that in the isolated model, the defending strategy consists of only an allocation strategy since no reallocation is allowed. When $k = 0$, the special case can be solved trivially by greedily allocating resources to the nodes with maximum value, because the defending result is defined by the not-well-defended node with maximum value.

However, in contrast to the case when $k = 0$, we show that when $k \geq 1$, the problem becomes NP-hard.

Theorem 2 *Computing the optimal defending strategy is* NP*-hard for* $k \geq 1$*, even for the isolated model with identical thresholds.*

Proof: We prove by a reduction from the (unweighted) vertex cover (VC) problem, which is known to be NP-hard (Chlebík and Chlebíková 2006). Given an instance $G_{vc} = (V_{vc}, E_{vc})$, the VC problem is to select a minimum size subset $S \subseteq V_{vc}$ such that each edge $(u, v) \in E_{vc}$ has at least one endpoint in S. We construct an instance $G = (V, E)$ of the DCA problem in which $w_{uv} = 0$ for all edges $(u, v) \in E$ and $\theta_u = 1$ for all nodes $u \in V$ as follows. Let the instance G of the DCA problem be obtained by inserting a node for every edge $(u, v) \in E_{vc}$, splitting the edge. Specifically, we first initialize $G = G_{vc}$. Then for each $e = (u, v) \in E_{vc}$, we remove e, insert a new node u_e and two edges $(u, u_e), (u_e, v)$ into E. We refer to these nodes (that split edges) the *splitting nodes*, and the other nodes as *original nodes*. Note that each splitting node has exactly two neighbors, both of which are original nodes. The neighbors of each original node are all splitting nodes. Note that we have $|V| = |V_{vc}| + |E_{vc}|$ and $|E| = 2|E_{vc}|$. Set $\alpha_u = 0$ for splitting nodes, and $\alpha_u = 1$ for original nodes. In other words, only the original nodes are valuable and worth defending. Let $\theta_u = 1$ for all $u \in V$ and $k = 1$.

Observe that since resource cannot be transferred, the optimal allocation strategy assigns resource either 0 or 1 to each original node, and 0 to each splitting node. We call a node u *defended* if $r_u = 1$, *undefended* otherwise. Since $k = 1$, when the attacker chooses to attack an original node u, the total loss is 0 if u is defended, 1 otherwise. However, if the attacker attacks a splitting node, the total loss is the number of undefended neighbors of the splitting nodes, which can be 2. Suppose there exists an allocation strategy using total resource R for which the defending result is at most 1, then there must exist a vertex cover of size at most R for G_{vc}. Specifically, the defended original nodes form a vertex cover for G_{vc} (otherwise, there exists a splitting node whose two neighbors are both undefended). Hence if there exists a polynomial time algorithm for the DCA problem, then we can use binary search on $R \in \{1, 2, \ldots, |V_{vc}| - 1\}$ to identify the minimum R with which the defending result is 1. Consequently, we can compute a minimum vertex cover in polynomial time, which is a contradiction. □

Interestingly, we show that the reduction also implies a hardness of approximation.

Corollary 1 *For any* $c < 2$*, computing a* c*-approximation defending strategy when* $k \geq 1$ *is* NP*-hard, even for the isolated model with identical thresholds.*

Proof: In the above reduction, for any $R < |V_{vc}|$, the defending result is either 1 or 2. Let OPT be the optimal defending result and ALG be that of the c-approximation algorithm, where $c < 2$. Note that both OPT and ALG take values in $\{1, 2\}$. Observe that for $\mathsf{OPT} = 1$, we must have $\mathsf{ALG} = 1$ since otherwise the approximation ratio is 2. Similarly, for $\mathsf{OPT} = 2$, we have $\mathsf{ALG} = 2$. Hence any better-than-2 approximation algorithm is equivalent to an exact algorithm, and the corollary follows from Theorem 2. □

MILP Formulation

Nevertheless, we show that we can formulate the computation of the optimal defending strategy as an MILP as we have done in Section 3. Similar as before, we introduce a set of variables for the case when u is under attack: we introduce an integer variable $x_v^u \in \{0, 1\}$ for each $v \in N_k(u)$, which indicates whether $p_v \geq \theta_v$ when u is under attack; we also introduce a variable $t^u(z, v)$ for each $v \in N_k(u)$ and $z \in N(v)$, which represents the resource z sends to v.

Unlike before, where the allocation strategy is given, here we introduce a variable r_u to denote the resource allocated to node $u \in V$. We also changed the objective from minimizing $\mathsf{Loss}(u)$ to minimizing $\max_{u \in V} \mathsf{Loss}(u)$, by introducing a variable Loss that is at least $\mathsf{Loss}(u) = \sum_{v \in N_k(u)} (1 - x_v^u) \alpha_v$ for all $u \in V$. The computation of the defending strategy is then formulated as follows.

$$\begin{aligned}
\text{minimize} \quad & \mathsf{Loss} \\
\text{subject to} \quad & \textstyle\sum_{u \in V} r_u \leq R, \\
r_v - \textstyle\sum_{z \in N(v) \cap N_k(u)} t^u(v, z) + & \textstyle\sum_{z \in N(v)} t^u(z, v) \\
& \geq \theta_v \cdot x_v^u, \quad \forall u, v && (4) \\
0 \leq t^u(v, z) \leq w_{vz} \cdot r_v, & \quad \forall u, v, z && (5) \\
\textstyle\sum_{z \in N(v) \cap N_k(u)} t^u(v, z) \leq r_v, & \quad \forall u, v, z && (6) \\
\textstyle\sum_{v \in N_k(u)} (1 - x_v^u) \alpha_v \leq \mathsf{Loss}, & \quad \forall u && (7) \\
x_v^u \in \{0, 1\}, & \quad \forall u, v.
\end{aligned}$$

Similar as before, the set of constraints (4) guarantees that the defending power of a node v is at least θ_v when $x_v^u = 1$. Constraints (5) and (6) guarantee feasibility of transfers of resource. Constraints (7) ensure $\mathsf{Loss} = \max_{u \in V} \mathsf{Loss}(u)$ in the optimal solution. As before, we only need to introduce variable $t^u(z, v)$ if $v \in N_k(u)$ and $z \in N(v)$. We use MILP(R) to denote the above program that uses total resource R. Note that in the program r_u's and $t^u(z, v)$'s are fractional variables while x_v^u's are integer variables. We denote by LP(R) the linear program relaxation when we replace each constraint $x_v^u \in \{0, 1\}$ with $x_v^u \in [0, 1]$. As Example 3 shows, the integrality gap of LP(R) and MILP(R) is unbounded. Nevertheless, LP(R) provides a lower bound for MILP(R), which can be used for a pruning on MILP.

Prunings. Suppose we have a lower bound l of the optimal defending result OPT, i.e., the optimal objective of MILP(R). Then for every node u with $\sum_{v \in N_k(u)} \alpha_v \leq l$, we can remove all variables with superscript u and all constraints containing such variables. The reason is, when u is attacked, the maximum loss (even if we do not allocate or reallocation any resource) is at most $\sum_{v \in N_k(u)} \alpha_v$. Given $\mathsf{OPT} \geq l$, not defending nodes in $N_k(u)$ does not increase the objective of MILP(R). Note that the optimal solution of LP(R) gives one such lower bound l. The closer the optimal objectives of LP(R) and MILP(R) are, the better the pruning reduces the size of MILP(R).

Existence of Perfect Defending Strategy

While the general problem of computing the optimal allocation strategy is NP-hard, we show in this section that deciding whether there exists a defending strategy with defending result 0 (which we refer to as a *perfect defending strategy*) is polynomial time solvable. Moreover, if they exist, then we can compute one in polynomial time.

Theorem 3 *For every $k \geq 0$, there exists a polynomial time algorithm that computes a perfect defending strategy for the DCA , if perfect defending strategies exist.*

Proof: Recall that MILP(R) computes the optimal defending strategy. If there exist perfect defending strategies, then we have $\mathsf{Loss} = 0$ in the optimal solution for MILP(R). Since $\mathsf{Loss} \geq \sum_{v \in N_k(u)} (1 - x_v^u)\alpha_v$, we must have $x_v^u = 1$ for all integer variables in the optimal solution.

Therefore, by fixing $x_v^u = 1$ for all integer variables, MILP(R) must be feasible. Observe that after fixing an assignment to the integer variables, MILP(R) becomes a feasibility LP, which can be solved exactly in polynomial time. Any feasible solution $(\mathbf{r}, \{\mathbf{t}^u\}_{u \in V})$ for the LP provides a perfect defending strategy, as claimed. □

Bi-criteria Approximation

As Example 3 indicates, it is impossible to obtain any bounded approximation of the reallocation by rounding the LP relaxation of MILP(R). However, we show that by augmenting the total resource we use, good approximation solutions (in terms of defending results) can be obtained.

Definition 7 (Bi-criteria Approximation) *We call a defending strategy (γ, β)-approximate if it uses R total resource and its defending result is at most $\gamma \cdot \mathsf{OPT}$, where OPT is the optimal defending result using R/β resource.*

While it is not possible to obtain bounded (standard) approximations by rounding LP(R), we show that achieving bi-criteria approximations is possible. We defer the proof of the following theorem to the full version of the paper.

Theorem 4 *For any $\epsilon \in (0, 1)$, we can compute a $(\frac{1}{1-\epsilon}, \frac{1}{\epsilon})$-approximate defending strategy in polynomial time. In particular, with $\epsilon = 0.5$ we can compute a $(2, 2)$-approximate solution in polynomial time.*

We show that under the Unique Game Conjecture (UGC) (Khot and Regev 2008), there do not exist strong Pareto improvements over our bi-criteria $(2, 2)$ approximation ratio. The proof is deferred to the full version of paper.

Lemma 1 *Under UGC, there does not exist polynomial time $(2 - \delta, 2 - \delta)$-approximate algorithm for the DCA problem, for any constant $\delta > 0$.*

Implementation. In practice, we can enumerate different $\epsilon \in (0, 1)$ to compute different defending strategies, and then pick the one with the best defending result. In the following, we show that we might be able to improve the defending result further by deploying a more aggressive rounding on $\mathbf{x}$. Specifically, we first solve $\mathsf{LP}(\epsilon \cdot R)$ and get the optimal solution. Then we pick some $\tau \in [0, \epsilon]$, round each x variable that is less than τ to 0, and those at least τ to 1. With the fixed integer variables, we solve MILP(R), which has become a feasibility LP. If the resulting LP is feasible, then we obtain a defending strategy with defending result at most $\frac{1}{1-\tau} \cdot \mathsf{OPT}$, where OPT is the optimal defending result of defending strategies using $\epsilon \cdot R$ resources. Hence the resulting solution is a $(\frac{1}{1-\tau}, \frac{1}{\epsilon})$-approximate defending strategy. For different problem instances, the minimum τ with which the induced LP is feasible can be different. However, the LP must be feasible when $\tau = \epsilon$. As we will show in our experiments, in all datasets we consider, the defending result after optimizing τ is much smaller than using $\tau = \epsilon$.

Experimental Evaluation

In this section, we evaluate the effectiveness and efficiency of our algorithms on synthetic and real-world datasets. Our datasets contain synthetic graphs, including random graphs and power-law distribution graphs, which are well recognized as the best in modeling random networks and social networks. We also consider real-world networks, including aviation networks and social networks, in order to demonstrate the practical performance of our algorithms on defending against contagious attacks in the real world. The datasets are generated as follows. For each dataset, n and m denote the number of nodes and edges, respectively.

	Rand	Pow-S	Pow-L	USAir	FB	Twit
# Node	200	400	700	221	600	1000
# Edge	803	1579	2087	2166	4638	13476

Table 1: Number of nodes and edges of the datasets.

- **Random**: We generate the dataset with $n = 200$ and $p = 0.04$ using the algorithm by (Batagelj and Brandes 2005), where there is an edge between each pair of nodes independently with probability p. The thresholds θ_u's and values α_u's are chosen uniformly at random from integers in $[1, 10]$. The edge weights w_{uv}'s are uniformly chosen from $[0.3, 1]$.
- **Power-law distribution graphs (Pow)**: We use the graph generator by NetworkX (Hagberg, Schult, and Swart 2008) to generate the power-law distribution graphs, where we set the parameters[3] to be $(400, 4, 0.5)$ for Pow-S and $(700, 3, 0.5)$ for Pow-L. The parameters θ_u's, α_u's and w_{uv}'s are generated randomly as before (for dataset Rand).

[3] For the details, please refer to https://networkx.github.io/documentation/networkx-1.10/reference/generated/networkx.generators.random_graphs.powerlaw_cluster_graph.html.

- **USAir**: We select the flight records in the US from years 2008 to 2010 to generate a directed graph where each node represents a city. There is a directed edge from city u to city v if the number of flights per week from u to v is at least 25. We set the edge weight as the ratio between the flights-per-week of the edge and the maximum flights-per-week value of all edges. We set θ_u and α_u as the population (in millions) of city u.
- **Social networks**: We use the network of Facebook (undirected) and Twitter (directed) to generate our datasets (McAuley and Leskovec 2012). The dataset FB (resp. Twit) is extracted from the source network by picking a random node in the network and expand using breath-first-search until the size of the dataset reaches $n = 600$ (resp. $n = 1000$). We set $\theta_u = \alpha_u = w_{uv} = 1$ for all nodes and edges.

Experiment Environment. We perform our experiments on an AWS Ubuntu 18.04 machine with 32 threads and 128GB RAM without GPU. We use Gurobi optimizer as our solver for the LPs and MILPs.

We evaluate the effectiveness of our exact and approximation algorithms by comparing the results of defending strategies under different settings and datasets. Throughout all the experiments, we fix the contagiousness parameter $k = 2$.

Effectiveness of Reallocation. One of the main innovations of our work is that we consider the reallocation of defending resources between the nodes. The reallocation allows the algorithm to react adaptively against the attack. In particular, we compare the results of defending strategies with and without reallocation as follows.

We first use the algorithm in Section 4 to compute for each dataset the minimum total resource required in a perfect defending strategy (a strategy with defending result 0). As our experiment (in Table 2) shows, reallocation (see the row with $w \neq 0$) always helps in reducing the requirement on the defending resource, for all datasets. For example, for the first dataset Rand, the resource required in a perfect defending strategy is 40% less than the case when reallocation is not allowed (see the row with $w = 0$).

	Rand	Pow-S	Pow-L	USAir	FB	Twit
$w = 0$	1037	1892	3406	341	600	1000
$w \neq 0$	587	1687	2320	340	524	623

Table 2: Resource required to achieve defending result 0.

Comparing Different Algorithms. We also evaluate the effectiveness of our bi-criteria approximations from Section 4, and compare it with the exact solution and the Greedy algorithms. The results are shown in Table 3, where BA(ϵ) stands for the approximation algorithm by rounding the optimal solution for LP($\epsilon \cdot R$) and optimizing $\epsilon \in (0, 1)$; BA(ϵ, τ) stands for the approximation algorithm that further optimizes $\tau \in (0, \epsilon]$ in the more aggressive rounding. We use Greedy to refer to the algorithm that greedily allocates resources to nodes with the maximum value (break tie arbitrarily) and does not use reallocation; Greedy-R is based on Greedy but uses greedy reallocation. Specifically, when node u is attacked, for each node $v \in N_k(u)$ in decreasing order of their values, the algorithm transfers resource to v until its defending power is at least its threshold, or when no more resource can be transferred from its neighbors. In the experiments, we fix $R = 0.5 \cdot \sum_{u \in V} \theta_u$ for all datasets.

	Rand	Pow-S	Pow-L	USAir	FB	Twit
Greedy	278	859	1168	178.7	188	302
Greedy-R	225	819	1025	178.7	186	281
BA(ϵ)	289	1134	1230	291.8	109	148
BA(ϵ, τ)	107	785	701	204.3	58	55
Exact	69	740	616	170.3	51	53

Table 3: Defending results by the bi-criteria approximations.

From Table 3, we observe that by deploying a more aggressive rounding on the fractional solutions, the approximation solutions by BA(ϵ, τ) outperform BA(ϵ) dramatically, and are very close to the optimal solutions in all datasets. Moreover, in general, BA(ϵ, τ) achieves much smaller defending results when compared with both Greedy approaches in most datasets. The only exception is the dataset USAir, in which the values of nodes differ greatly. Thus, Greedy allocation of resources achieves the almost optimal result. BA(ϵ) does not perform well because the solution is obtained by a very loose rounding on the solution of LP($\epsilon \cdot R$). Observe that with the help of reallocation, Greedy-R obtains advantages over Greedy, which again demonstrates the critical role of reallocation.

Efficiency Evaluation. Finally, we evaluate the efficiency of our algorithms and summarize the running times in Table 4, where No-Prune refers to the algorithm by solving the MILP without using the pruning we mentioned in Section 4; Pruning refers to the one with pruning; BA(ϵ, τ) refers to our bi-criteria approximation algorithm.

	Rand	Pow-S	Pow-L	USAir	FB	Twit
No-Prune	4637	8180	36293	10686	32608	8441
Pruning	4259	3351	8227	2920	18929	2757
BA(ϵ, τ)	246	867	1508	416	2540	1294

Table 4: Running times of different algorithms (in seconds).

As we can see from the Table 4, the running times for solving MILPs are obviously improved after pruning, which shows the effectiveness of removing redundant variables. In particular, for the dataset Pow-L, which admits the long tail phenomenon, there is a 77% improvement on the running time after pruning. Our approximation algorithm improves the running time even further, e.g., is several times faster than that of Pruning in all datasets, which demonstrates its great efficiency in practical use.

References

Acemoglu, D.; Malekian, A.; and Ozdaglar, A. E. 2016. Network security and contagion. *J. Econ. Theory* 166: 536–585.

Aspnes, J.; Chang, K. L.; and Yampolskiy, A. 2006. Inoculation strategies for victims of viruses and the sum-of-squares partition problem. *J. Comput. Syst. Sci.* 72(6): 1077–1093.

Assimakopoulos, N. 1987. A network interdiction model for hospital infection control. *Computers in biology and medicine* 17(6): 413–422.

Bachrach, Y.; Draief, M.; and Goyal, S. 2013. Contagion and observability in security domains. In *Allerton*, 1364–1371. IEEE.

Batagelj, V.; and Brandes, U. 2005. Efficient generation of large random networks. *Phys. Rev. E* 71: 036113. doi: 10.1103/PhysRevE.71.036113. URL https://link.aps.org/doi/10.1103/PhysRevE.71.036113.

Chan, H.; Ceyko, M.; and Ortiz, L. E. 2012. Interdependent Defense Games: Modeling Interdependent Security under Deliberate Attacks. In *UAI*, 152–162. AUAI Press.

Chan, H.; Ceyko, M.; and Ortiz, L. E. 2017. Interdependent Defense Games with Applications to Internet Security at the Level of Autonomous Systems. *Games* 8(1): 13. doi:10.3390/g8010013. URL https://doi.org/10.3390/g8010013.

Chlebík, M.; and Chlebíková, J. 2006. Complexity of approximating bounded variants of optimization problems. *Theor. Comput. Sci.* 354(3): 320–338.

Gan, J.; An, B.; and Vorobeychik, Y. 2015. Security Games with Protection Externalities. In *AAAI*, 914–920. AAAI Press.

Gan, J.; An, B.; Vorobeychik, Y.; and Gauch, B. 2017. Security Games on a Plane. In *AAAI*, 530–536. AAAI Press.

Goyal, S.; and Vigier, A. 2014. Attack, defence, and contagion in networks. *The Review of Economic Studies* 81(4): 1518–1542.

Hagberg, A. A.; Schult, D. A.; and Swart, P. J. 2008. Exploring Network Structure, Dynamics, and Function using NetworkX. In Varoquaux, G.; Vaught, T.; and Millman, J., eds., *Proceedings of the 7th Python in Science Conference*, 11 – 15. Pasadena, CA USA.

Johnson, B.; Grossklags, J.; Christin, N.; and Chuang, J. 2010. Uncertainty in Interdependent Security Games. In *GameSec*, volume 6442 of *Lecture Notes in Computer Science*, 234–244. Springer.

Khot, S.; and Regev, O. 2008. Vertex cover might be hard to approximate to within 2-epsilon. *J. Comput. Syst. Sci.* 74(3): 335–349.

Kunreuther, H.; and Heal, G. 2003. Interdependent security. *Journal of risk and uncertainty* 26(2-3): 231–249.

Letchford, J.; Conitzer, V.; and Munagala, K. 2009. Learning and Approximating the Optimal Strategy to Commit To. In *SAGT*, volume 5814 of *Lecture Notes in Computer Science*, 250–262. Springer.

Li, M.; Tran-Thanh, L.; and Wu, X. 2020. Defending with Shared Resources on a Network. In *AAAI*, 2111–2118. AAAI Press.

Lou, J.; Smith, A. M.; and Vorobeychik, Y. 2017. Multidefender Security Games. *IEEE Intell. Syst.* 32(1): 50–60.

McAuley, J. J.; and Leskovec, J. 2012. Learning to Discover Social Circles in Ego Networks. In *NIPS*, 548–556.

Nguyen, K. C.; Alpcan, T.; and Basar, T. 2009. Stochastic games for security in networks with interdependent nodes. In *GAMENETS*, 697–703. IEEE.

Schlenker, A.; Thakoor, O.; Xu, H.; Fang, F.; Tambe, M.; Tran-Thanh, L.; Vayanos, P.; and Vorobeychik, Y. 2018. Deceiving Cyber Adversaries: A Game Theoretic Approach. In *AAMAS*, 892–900.

Sinha, A.; Fang, F.; An, B.; Kiekintveld, C.; and Tambe, M. 2018. Stackelberg Security Games: Looking Beyond a Decade of Success. In *IJCAI*, 5494–5501. ijcai.org.

Tambe, M. 2012. *Security and Game Theory - Algorithms, Deployed Systems, Lessons Learned*. Cambridge University Press.

Tsai, J.; Nguyen, T. H.; and Tambe, M. 2012. Security Games for Controlling Contagion. In *AAAI*. AAAI Press.

Vorobeychik, Y.; and Letchford, J. 2015. Securing interdependent assets. *Auton. Agents Multi Agent Syst.* 29(2): 305–333.

Yin, Y.; Xu, H.; Gan, J.; An, B.; and Jiang, A. X. 2015. Computing Optimal Mixed Strategies for Security Games with Dynamic Payoffs. In *IJCAI*, 681–688. AAAI Press.

Yin, Z.; and Tambe, M. 2012. A unified method for handling discrete and continuous uncertainty in Bayesian Stackelberg games. In *AAMAS*, 855–862. IFAAMAS.

Zhang, Y.; An, B.; Tran-Thanh, L.; Wang, Z.; Gan, J.; and Jennings, N. R. 2017. Optimal Escape Interdiction on Transportation Networks. In *IJCAI*, 3936–3944. ijcai.org.