Journal of Artificial Intelligence Research 77 (2023) 487-515 Submitted 01/2023; published 06/2023

Stackelberg Security Games with Contagious Attacks on a Network: Reallocation to the Rescue

Rufan Bai YB97439@UM.EDU.MO
State Key Laboratory of Internet of Things for Smart City,
University of Macau, Macau, China

Haoxing Lin HAOXING.LIN@COMP.NUS.EDU.SG
School of Computing,
National University of Singapore, Singapore

Xinyu Yang MB95466@UM.EDU.MO
Xiaowei Wu XIAOWEIWU@UM.EDU.MO
State Key Laboratory of Internet of Things for Smart City,
University of Macau, Macau, China

Minming Li MINMING.LI@CITYU.EDU.HK
Department of Computer Science,
City University of Hong Kong, Hong Kong, China, and
School of Mathematical Sciences,
Ocean University of China, Qingdao, China

Weijia Jia JIAWJ@UIC.EDU.CN
BNU-UIC Institute of Artificial Intelligence and Future Networks,
Beijing Normal University (Zhuhai), Guangdong, China

Abstract

In the classic network security games, the defender distributes defending resources to the nodes of the network, and the attacker attacks a node, with the objective of maximizing the damage caused. In this paper, we consider the network defending problem against contagious attacks, e.g., the attack at a node u spreads to the neighbors of u and can cause damage at multiple nodes. Existing works that study shared resources assume that the resource allocated to a node can be shared or duplicated between neighboring nodes. However, in the real world, sharing resource naturally leads to a decrease in defending power of the source node, especially when defending against contagious attacks. Therefore, 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 the problem of computing optimal defending strategy is NP-hard even for some very special cases. 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.

1. Introduction

Recently, studies on Stackelberg security games have attracted enormous attention in artificial intelligence and combinatorial optimization communities (Letchford et al., 2009; Tambe, 2012; Sinha et al., 2018). Meanwhile, the resulted algorithms have been increasingly deployed to compute the optimal allocation of defending resources in multiple real-world applications, such as patrolling

problems (Yin & Tambe, 2012; Gan et al., 2015), managing energy power system (Paul et al., 2020) and protecting data privacy (Ye et al., 2022). Among these works, a considerable portion focus on games on networks (Assimakopoulos, 1987; Gan et al., 2017; Zhang et al., 2017; Schlenker et al., 2018; Tan & Wang, 2020; Jin et al., 2019), in which a defender needs to allocate resources among nodes on a graph. Each node of the graph has a specific value (loss) and a defending requirement. If the defending resource allocated to the node meets the defending requirement then no loss will incur even if the node is attacked. The defender's goal is to minimize the loss due to any possible attack carried out by an attacker. Motivated by the pandemic, in this paper, we consider the case when the attack is *contagious*, i.e., when the attacker chooses a node of the graph to attack, the neighbors of the chosen node are also under attack. Such contagious attack model can also capture other real-world applications when the attack at a single node may cause loss at multiple (neighboring) nodes, e.g., the spread of virus among computers, or the spread of harmful information in social networks. Below we show an example of such a contagious attack and the defending strategy against it.

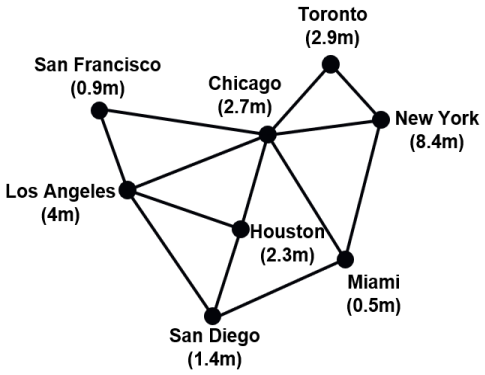


Figure 1: An example of initial graph network (m represents million)

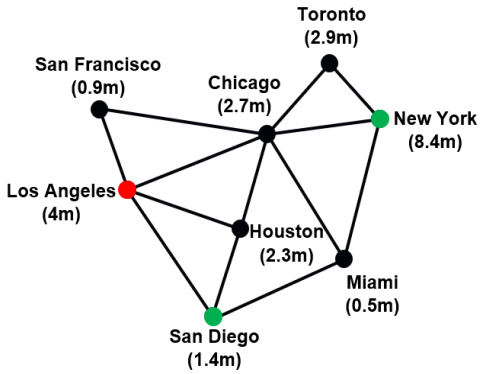


Figure 2: Non-contagious attack

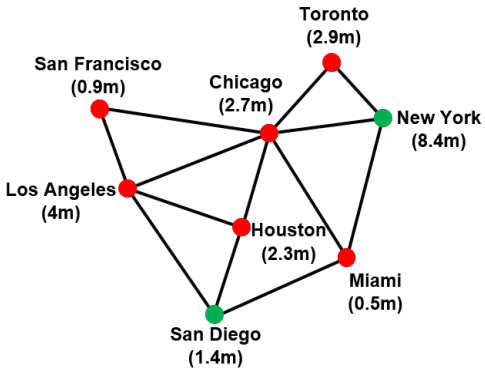


Figure 3: Contagious attack

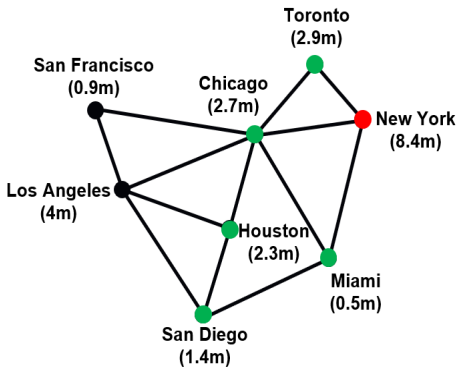


Figure 4: Contagious attack (optimal allocation)

Example 1.1 *Consider the graph shown in Figure 1, where each node represents a city and has a certain population (m for million), which represents its value and defending requirement. Suppose*

that some viral disease will outbreak in one of the cities and we are given limited resources, say, 10 *million masks, to defend against it. When a city is allocated an amount of masks at least that of the population, we assume that the city is well defended. The goal is to allocate the resource to the nodes so that the worst case damage is minimized. To optimize the resource utilization, we always allocate to a node either the resource equal to the defending requirement or* 0 *resource. If the attack is not contagious, the problem can be easily solved by greedily allocating the resource to the nodes with the highest value (see Figure 2 for the optimal defending strategy). In the example, we color a city green if it is well defended; we color a city red if it is not well defended and attacked. If the attack is contagious then the defending strategy shown in Figure 2 might not be optimal. For example, suppose that when a node is attacked, the attack spreads to all its neighbors. Then when Chicago is under attack, the total loss will be very large under the greedy allocation strategy (see Figure 3). Therefore, the optimal (greedy) allocation strategy against non-contagious attack is no longer optimal for contagious attack. For contagious attack, it can be verified (by enumerating all subsets of cities with total threshold at most* $10m$*) that the optimal strategy is to allocate the resources on Toronto, Chicago, Houston, San Diego and Miami (see Figure 4). Under this defending strategy, no matter which city is attacked, the total loss is at most* $8.4m$ *(achieved when New York is under attack).*

From the above example we can see that the optimal defending strategy against contagious attack can be very different from that for non-contagious attack. Moreover, it can be seen that it might be difficult to find the optimal defending strategy. In fact, as we will show in this paper, computing the optimal defending strategy against contagious attack is NP-hard.

For the network security games, 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. (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. (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 combination of the resources allocated to its neighbors. These models are mainly motivated by surveillance or patrolling applications (Vorobeychik et al., 2014; Paruchuri et al., 2007), in which the defending resource, e.g., the patrollers or the surveillance cameras, can often cover a set of (neighboring) positions instead of a single position. Therefore in these models, when a node u shares its resource with its neighbors, the defending power of node u remains the same.

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. For example, consider the outbreak of a highly contagious virus at some city u. At the early stage of the outbreak, it is natural and necessary to borrow medical resources, e.g., doctors and equipment, from neighboring cities of u to improve the defending status of city u. Before long, when the virus breaks out on a large scale to the neighboring cities, the deficiencies of medical resources at these cities are naturally expected. For contagious attack, when evaluating the damage, it is necessary to include damage caused at each of the node the attack spreads to. Consider the virus outbreak at some node u. The virus spreads to the neighbors of u 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 u by taking into account the resources shared from its neighbor v, then naturally, we need to consider the decrease in the defending power of v. Ideally, a node v can only *transfer* (a fraction of) the resource allocated to v to its neighbor u, which increases the

defending power of the receiver u but decreases its own defending power[1]. 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 1.2 *Consider a star graph, with node u at the center, and $v_1, v_2, \ldots, v_{n-1}$ being neighbors of u, where each node requires 1 unit of resource to defend itself. Suppose that 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 well 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 will be nodes that are not well defended.*

In this paper, we consider the problem of defending against contagious attacks, 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, e.g. outbreak of virus, 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. Existing models fail to capture such applications as most of them do not consider the reallocation of defending resources.

1.1 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 et al., 2015; Li et al., 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. A defending strategy needs to compute an allocation strategy before the attack happens, and a reallocation strategy depending on which node is attacked, subject to the capacity constraints associated with the edges. In this work, we study the defending strategy under two forms of attack: adaptive attack and uniform attack. For adaptive attack, the attacker will choose a node to maximize the damage. So the objective of the defender is to minimize the maximum possible damage due to an attack. For uniform attack, the attacker will attack each node with equal probability and thus the objective of the defender is to minimize the expected loss due to the random attack[2].

1. Throughout this paper we assume that the resource transfers happen instantaneously, which is different from the works (e.g., (Yin et al., 2015; Lamballais et al., 2022)) in which the reallocation of resource takes time.
2. It should be noted that under this attack mode, the problem is no longer a Stackelberg Game.

Our work mainly focuses on the case when the attack is adaptive. In Section 3 and 4, we present the hardness and algorithms against adaptive attacks. We show that the 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 (Section 3). Then we show that if no reallocation is allowed, the problem of computing the optimal allocation strategy is also NP-hard (Section 4.1). Regarding positive results, we provide mixed integer linear programs (MILPs) to model the computation of allocation and reallocation strategies (in Section 4.2). We show that the optimal solutions for the MILP 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 (Section 4.3). Then we give a polynomial time bi-criteria $(\frac{1}{1-\epsilon}, \frac{1}{\epsilon})$-approximation algorithm, for any $\epsilon \in (0, 1)$ (see Section 4.4 for a formal definition of bi-criteria approximations). Specifically, for $\epsilon = 0.5$ we have a bi-criteria $(2, 2)$-approximation. Moreover, we show that under the Unique Game Conjecture (Khot & Regev, 2008), there does not exist $(2 - \delta, 2 - \delta)$-approximation, for any constant $\delta > 0$. We consider the uniform attacks in Section 5, in which we show that computing the optimal allocation and reallocation strategy is NP-hard. We also show that the MILP formulation and the approximation algorithm we developed for adaptive attacks can also be applied to defend against uniform attacks.

Finally, we perform an extensive evaluation of our algorithms on synthetic and real-world datasets in Section 6. In the experiments, we evaluate the performance of our approximation algorithm by comparing its effectiveness and efficiency with the optimal solution and other algorithms. By varying the total resources and the contagiousness on different datasets, our experiments demonstrate that our approximation algorithm outperforms other heuristic algorithms and is always able to efficiently compute close-to-optimal defending strategies.

Our contributions can be summarized as follows:

- For adaptive contagious attacks, we prove that both the computation of the optimal allocation and the optimal reallocation strategies are NP-hard.
- We formulate an MILP for computing the optimal defending strategy, based on which we propose polynomial time algorithms for computing the perfect defending strategy and the bi-criteria approximation solutions.
- We show that the problem remains NP-hard under uniform contagious attacks and our bi-criteria approximation algorithm still applies in this setting.
- We perform an extensive evaluation of our algorithms on synthetic and real-world datasets to verify their effectiveness under different settings.

1.2 Other Related Work

There are existing works that consider security game models in which nodes have arbitrary values (Kiekintveld et al., 2009; Korzhyk et al., 2010; Conitzer & Sandholm, 2006). However, it is commonly assumed in these works that the thresholds of nodes are uniform[3] and they do not con-

3. Their works often do not specify the "threshold" explicitly. Instead they assume that a single unit of defending resource suffices to defend one target, which is equivalent to having uniform defending thresholds.

sider resource sharing. To be more specific, the problem is usually modeled by allocating m units of resources to n nodes ($m < n$) in either a deterministic or randomized way. A node is well protected if it receives one unit of resource. In our model, the threshold of each node can be arbitrary. Besides, we also consider resource sharing between neighboring nodes. In contrast, there is a sequence of existing works in the network security game domain that consider resource sharing between nodes. Gan et al. (2015, 2017) consider models in which allocating one unit of defending resource to a node can also protect the neighbors of that node. Yin et al. (2015) also study a model in which the resource can be transferred, and they assume transferring resources takes time. However, these existing models do not consider contagious attacks. There are also works that study the contagion in network security games (Aspnes et al., 2006; Nguyen et al., 2009; Tsai et al., 2012; Bachrach et al., 2013; Goyal & Vigier, 2014; Vorobeychik & Letchford, 2015; Acemoglu et al., 2016; Lou et al., 2017). Nevertheless, 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 et al., 2017; Li et al., 2020). Moreover, Zhao et al. (2019) study algorithms for computing optimal resource allocation on networks in the application of virus propagation. However, resource reallocation is not considered in their model. The problem of efficient resource reallocation has also been studied in other related works (Bondi et al., 2020; Yedidsion, 2012). However their focus are different from ours. For example, Bondi et al. (2020) consider the security games with signaling, with the goal of minimizing the uncertainties in the real-world applications. There are also works that study other game-theoretic models of the security games (Kunreuther & Heal, 2003; Johnson et al., 2010; Chan et al., 2012).

To enable a clear comparison of our model with the most closely related existing models, we summarize the main features of the models in Table 1.

Model	Limited Budget		Contagious Attacks		Resource Sharing	
	Yes	No	Yes	No	Duplicate	Reallocate
(Gan et al., 2015)	✓			✓	✓	
(Li et al., 2020)	✓			✓	✓	
(Yin et al., 2015)	✓			✓		✓
(An et al., 2013)	✓			✓		
(Vorobeychik et al., 2014)	✓			✓		
(Kiekintveld et al., 2009)	✓			✓		
(Korzhyk et al., 2010)	✓			✓		
(Conitzer & Sandholm, 2006)		✓		✓		
(Aspnes et al., 2006)		✓	✓			
Ours	✓		✓			✓

Table 1: Summary of existing models.

2. Model Description

We model the network as an undirected[4] 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

4. While we assume the graph is undirected, all our results extend straightforwardly to directed graphs.

damage due to an attack at node u. 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.

2.1 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*[5] 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 resource to its neighbor v, where $w_{uv} \in [0, 1]$ is the *weight* of edge (u, v), which represents the efficiency (or willingness) when transferring resource between u and v.

Definition 2.1 (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 2.2 (Reallocation Strategy) *We use $t(u, v) \geq 0$ to denote the resource node u transfers to its neighbor v. In general node 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 following conditions:

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

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 allocated to u by the allocation strategy: $\sum_{v \in N(u)} t(u, v) \leq r_u$.

Since the resources can be sent and received, the defending power of a node is not fixed. When evaluating whether a node is well defended or not, we need to look at its defending power after the reallocation. In other words, the initial allocation of defense resources does not decide the final loss due to attack, unless no resource is reallocated in the reallocation strategy. Depending on the attack, the defending power at each node can be adaptive by deciding an appropriate reallocation strategy.

Definition 2.3 (Defending Power) *The defending power of node u is defined as the total resource node u receives 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 takes values in range $[\bar{p}_u, \hat{p}_u]$, where

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

5. Similar to (Li et al., 2020), we assume that the allocated resource can take non-integer values in our model.

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 2.4 (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* $\mathbf{r} = \{r_u\}_{u\in V}$ *and* n *reallocation strategies* $\{\mathbf{t}^u\}_{u\in V}$.

2.2 Loss Due to An Attack

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 2.5 (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.

Remark. 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. Then the problem degenerates to the single-threshold model of (Li et al., 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.

3. 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.

Example 3.1 *Consider the graph given in Figure 5(a), and suppose that node a is under attack. For $k = 1$, the attack spreads to $N_1(a) = \{a, b, d, e\}$. Suppose that (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. Since the defending resource $r_a < \theta_a$ and $r_e < \theta_e$, without any reallocation of resource, we suffer from a total loss of $\alpha_a + \alpha_e$. However, if we are allowed to reallocate resources between neighboring nodes, then we can transfer some resources as shown in Figure 5(b). For example, node f transfers one unit of resource to node b and one unit of resource to node e. Note that the transferred resource along each edge incident to f is upper bounded by 0.5 times r_f, and the total transfer is at most r_f. Hence after the reallocation, all nodes in $N_1(a)$ are well defended, and no loss incurs.*

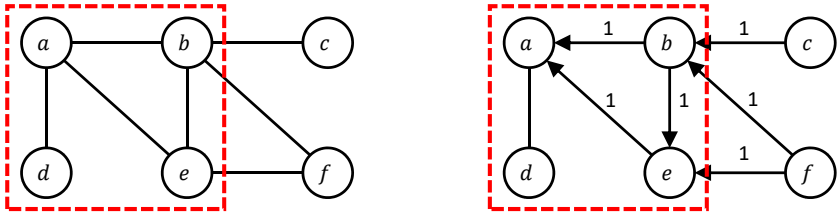


Figure 5: Example of a reallocation strategy, where a directed edge indicates a transfer of resource. For example, the edge from b to a with value 1 indicates that node b transfers $t(b, a) = 1$ unit of resource to node a. While there is an edge between node a and node d, we do not transfer any resource along this edge.

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.

Theorem 3.1 *Unless* P=NP, *for any $k \geq 1$, there does not exist any polynomial time algorithm that given an allocation strategy and a node under attack, computes the optimal reallocation strategy.*

Proof: We prove this by a reduction from the maximum independent set (MIS) problem, which is NP-hard (Berman & Fujito, 1999). In the MIS problem we are given a graph $G_{mis} = (V_{mis}, E_{mis})$ and the problem is to find a maximum size set $S \subseteq V_{mis}$ such that no two vertices in S are adjacent. In the following, we construct an instance of DCA for which the well defended nodes under the

optimal reallocation strategy form a maximum independent set. Intuitively, we would like to design an instance in which all nodes are insufficiently defended initially under the allocation strategy. Moreover, the gap between the defending requirement and defending resource on each node is exactly the degree of the node. By designing the edge weights and the allocation strategy, we can ensure that after any reallocations, any two well defended nodes are not neighbors of each other.

We first initialize the graph structure $G = (V, E)$ to be the same as G_{mis}. We set the parameters as follows. Let $n = |V_{mis}|$ and $d(u) = |N(u)|$ be the degree of node u in G.

- Let $r_v = n$, $\theta_v = n + d(v)$ and $\alpha_v = 1$ for all $v \in V$.
- Let $w_{uv} = \frac{1}{n}$ for all $(u, v) \in E$.

Finally, we insert a new vertex s to V with $r_s = \alpha_s = 0$, $\theta_s = 1$, and let s be connected with all other nodes with edges with weight 0. Note that the final instance of DCA has $n + 1$ nodes and $n + |E_{mis}|$ edges.

Suppose that s is under attack. Since $N_1(s) = V$, the attack spreads to the whole graph G, for $k \geq 1$. Observe that for each node $u \in V_{mis}$, its resource $r_u = n$ is lower than its threshold $\theta_u = n + d(u)$. Moreover, its maximum possible defending power

$$\hat{p}_u = n + \sum_{v \in N(u)} w_{uv} \cdot r_v = n + d(u) = \theta_u$$

can be obtained only by (1) not transferring any resource to its neighbors; (2) receiving $w_{uv} \cdot r_v = 1$ unit of resource from each of its neighbors. Hence under any reallocation strategy, if a node u is well defended ($p_u \geq \theta_u$), then none of its neighbors $v \in N(u)$ is well defended ($p_v < \theta_v$). Let $S^* \subseteq V_{mis}$ be the set of well defended nodes under the optimal reallocation strategy, we have

- S^* is an independent set (by the above argument);
- $\mathsf{Loss}(s) = n - |S^*|$, since each $u \in V_{mis}$ has $\alpha_u = 1$.

Since $\mathsf{Loss}(s)$ is minimized in the optimal reallocation strategy, we know that $|S^*|$ is maximized. In other words, S^* is a maximum size independent set of G_{mis}.

Since the reduction is polynomial time, if there exists a polynomial time algorithm to compute the optimal reallocation strategy, then we can solve the MIS problem in polynomial time, which is a contradiction. Consequently, computing the optimal reallocation strategy is NP-hard. □

We formulate the computation of 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.

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

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 resources 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.

Remark. 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 cannot 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 3.2 (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 $x_u = 1 - \epsilon$.*

Observations. While the integrality gap of the 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. 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 4.3. We also extend this idea in Section 4.4 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.

4. 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 3.1 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 NP-hard.

4.1 Hardness

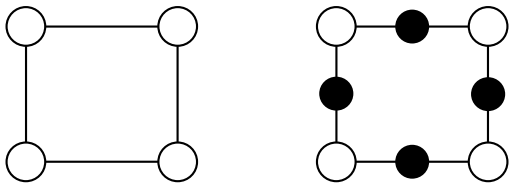
We first define a simple special case of the DCA problem called the *isolated model*, and then show that even for this special case, the problem is NP-hard.

Definition 4.1 (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 maximum value of not-well-defended nodes. However, in contrast to the case when $k = 0$, we show that when $k \geq 1$, the problem even in the isolated model becomes NP-hard.

Theorem 4.1 *Computing the optimal defending strategy is NP-hard when $k \geq 1$, 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 & Chlebíková, 2006). Given an instance $G_{vc} = (V_{vc}, E_{vc})$, the 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 (refer to Fig. 6).



(a) (b)

Figure 6: Example: (a) the VC problem instance G_{vc}; (b) the DCA problem instance G we construct, where solid nodes are splitting nodes and the hollow nodes are original nodes.

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) as 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 resources 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 attacks 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 node, 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 4.1 *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. Here we call a defending strategy a c-approximation if its defending result is at most c times the optimal defending result.*

Proof: Observe that 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 4.1. □

4.2 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 change 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.

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

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 3.2 shows, the integrality gap of LP(R) and MILP(R) is unbounded.

4.3 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 4.2 *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. □

4.4 Bi-criteria Approximation

As Example 3.2 indicates, it is impossible to obtain any bounded approximation of the reallocation by rounding the LP relaxation of MILP(R). Therefore, we consider bi-criteria approximate algorithms, which is a commonly used approximation measurement in the area of scheduling problems (Yedidsion, 2012; Kloh et al., 2012). To be more specific, instead of comparing the defending results of different algorithms using the same amount of resource, we allow an algorithm to augment the amount of resource used, and compare both its defending result and total resource used with the optimal ones. In contrast to the difficulties for approximation algorithms (as Example 3.2 suggests), we show that by augmenting the total resource we use slightly, good approximation solutions (in terms of defending results) can be obtained.

Definition 4.2 (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.

Theorem 4.3 *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.*

Proof: Recall that by the definition of bi-criteria approximations, we need to compute a strategy with the defending result at most $\frac{1}{1-\epsilon} \cdot \mathsf{OPT}$, where OPT is the optimal defending result of defending strategies that use $\epsilon \cdot R$ total defending resource, i.e., OPT is the optimal objective of MILP($\epsilon \cdot R$).

We first run $\mathsf{LP}(\epsilon \cdot R)$ and obtain the optimal (fractional) solution. Note that the optimal objective of $\mathsf{LP}(\epsilon \cdot R)$ is at most OPT, but in the solution each x_v^u can take arbitrary values in $[0, 1]$. In the following, we round the optimal solution $(\mathbf{x}, \mathbf{r}, \mathbf{t})$ of LP$(\epsilon \cdot R)$ and construct a feasible solution $(\hat{\mathbf{x}}, \hat{\mathbf{r}}, \hat{\mathbf{t}})$ for MILP(R). We show that the objective of the solution is at most $\frac{1}{1-\epsilon} \cdot \mathsf{OPT}$. Since each feasible solution for MILP(R) corresponds to a defending strategy, the theorem follows. For each variable $x_v^u \in [0, 1]$, let $\hat{x}_v^u = 1$ if $x_v^u \geq \epsilon$, and let $\hat{x}_v^u = 0$ otherwise. Let $\hat{r}_u = \frac{1}{\epsilon} \cdot r_u$ and $\hat{t}^u(z, v) = \frac{1}{\epsilon} \cdot t^u(z, v)$, for the corresponding variables. We show that the solution $(\hat{\mathbf{x}}, \hat{\mathbf{r}}, \hat{\mathbf{t}})$ we have constructed is feasible for MILP(R). Since originally $\sum_{u \in V} r_u \leq \epsilon \cdot R$, we have $\sum_{u \in V} \hat{r}_u \leq R$, i.e., the first constraint of MILP(R) is satisfied. Constraints (4) in which $\hat{x}_v^u = 0$ are trivially satisfied. For those with $\hat{x}_v^u = 1$, since we increase x_v^u by a factor of at most $\frac{1}{\epsilon}$ and increase all r and t variables by a factor of $\frac{1}{\epsilon}$, Constraints (4) of MILP(R) are satisfied. Since all r and t variables are scaled by the same factor, Constraints (5) (6) of MILP(R) are all satisfied.

Finally, since we decrease each x_v^u (to 0) only if $x_v^u < \epsilon$, by rounding $\mathbf{x}$ into $\hat{\mathbf{x}}$, for each $u \in V$, we have

$$\sum_{v \in N_k(u)} (1 - \hat{x}_v^u)\alpha_v \leq \frac{1}{1-\epsilon} \sum_{v \in N_k(u)} (1 - x_v^u)\alpha_v.$$

Hence the objective of solution $(\hat{\mathbf{x}}, \hat{\mathbf{r}}, \hat{\mathbf{t}})$ for MILP(R), $\mathsf{Loss} = \max_{u \in V}(\sum_{v \in N_k(u)}(1-\hat{x}_v^u)\alpha_v)$, is at most $\frac{1}{1-\epsilon} \cdot \mathsf{OPT}$, as claimed. □

Interestingly, we show that under the Unique Game Conjecture (UGC) (Khot & Regev, 2008), there do not exist strong Pareto improvements over our bi-criteria $(2, 2)$ approximation ratio.

Lemma 4.1 *Under UGC, there does not exist polynomial time $(2-\delta, 2-\delta)$-approximate algorithm for the DCA problem, for any constant $\delta > 0$.*

Proof: We use the same reduction from the vertex cover problem as in the proof of Theorem 4.1. It is shown in (Khot & Regev, 2008) that under the Unique Game Conjecture (UGC), there does not exist $(2 - \delta)$-approximation for the vertex cover problem, for any constant $\delta > 0$. Suppose there exists a polynomial time $(2 - \delta, 2 - \delta)$-approximate algorithm for the DCA problem, we show that the DCA problem can be transformed into a $(2 - \delta)$-approximation algorithm for the vertex cover problem, which contradicts the UGC.

Given any instance $G_{vc} = (V_{vc}, E_{vc})$ of a VC problem instance, we construct an instance $G = (V, E)$ of DCA as in the proof of Theorem 4.1. Then for each $R = 1, 2, \ldots, |V_{vc}| - 1$, we run the $(2-\delta, 2-\delta)$-approximate algorithm to compute a defending strategy, and let R^* be the smallest such that when $R = R^*$, the defending result is 1.

Suppose $S^* \subseteq V_{vc}$ is the minimum size vertex cover. By the construction of the DCA problem, when $R = |S^*|$, the optimal defending result is $\mathsf{OPT} = 1$. Hence with $R = (2 - \delta)|S^*|$ total defending resource, the $(2 - \delta, 2 - \delta)$-approximate algorithm computes a defending strategy with defending result at most $(2 - \delta) \cdot \mathsf{OPT} = 2 - \delta$. Since defending results are integers, the defending result by the approximation algorithm is 1. Observe that since all nodes have threshold 1 and resource cannot be transferred, a defending strategy with $R = (2 - \delta)|S^*|$ is equivalent to one with $R = \lfloor(2 - \delta)|S^*|\rfloor$. In other words, when $R = \lfloor(2 - \delta)|S^*|\rfloor$, the defending result of the approximation algorithm is 1, which implies $R^* \leq \lfloor(2 - \delta)|S^*|\rfloor$. Moreover, since the defending result is 1, the set of defended nodes is a vertex cover. Hence we have found a vertex cover of size

$$R^* \leq \lfloor(2 - \delta)|S^*|\rfloor \leq (2 - \delta)|S^*|,$$

which gives a $(2-\delta)$-approximation for the VC problem, and contradicts the UGC. □

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 $\tau = \epsilon$.

5. Defending against Uniform Attacks

In the previous sections, we assume that the attacker always chooses the node that causes maximum damage to attack. In other words, the attacker is *adaptive* to our defending strategy. However, as Theorem 3.1 shows, it is NP-hard to compute the node that causes maximum damage. In this section we consider the case when the node under attack is chosen uniformly at random. In other words, each node has an equal probability of being attacked, regardless of how the defending resources are allocated. This is a reasonable case to consider since in real world, the attack, e.g., the virus outbreak, may happen equally likely at all places. In the following, we show that the problem of computing the optimal defending strategy remains NP-hard for uniform attacks.

5.1 Hardness

We prove the following hardness result in this section.

Theorem 5.1 *The problem of computing the optimal defending strategy against uniform attacks is NP-hard, even for $k = 0$.*

Proof: We prove by a reduction from the maximum coverage problem, which is a classic NP-hard problem (Vazirani, 2013). Every instance of the problem includes a set of n elements $\mathcal{U}$, a collection of m subsets of elements $\mathcal{S} \subseteq 2^{\mathcal{U}}$, and a parameter h. The objective of the problem is to find h sets in $\mathcal{S}$ that cover the maximum number of elements, i.e., find $T \subseteq \mathcal{S}$, $|T| = h$ that maximizes $|\bigcup_{s \in T} s|$.

Given an instance $(\mathcal{U}, \mathcal{S}, h)$ of the maximum coverage problem, we construct the instance of the DCA problem as follows. Let $G(U \cup V, E)$ be a bipartite graph, where U and V are the two node sets. Let $|U| = |\mathcal{U}|$ and $|V| = |\mathcal{S}|$. Each node in U corresponds to an element in $\mathcal{U}$. Each node in V corresponds to a set in $\mathcal{S}$. For each set $s \in \mathcal{S}$ and element $e \in s$, we create an edge between the two corresponding nodes in the bipartite graph G (see Fig. 7 for an illustrating example).

We set other parameters of the instance as follows.

- Let $\theta_u = \frac{1}{h+1}$ and $\alpha_u = 1$ for all $u \in U$.
- Let $\theta_v = 1$ and $\alpha_v = n + 1$ for all $v \in V$.

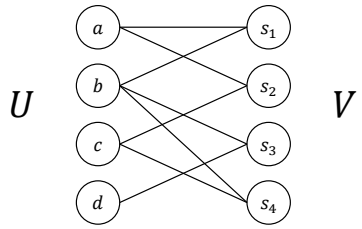


Figure 7: Suppose the given instance of maximum coverage problem contains four elements $\{a, b, c, d\}$ and four sets $s_1 = \{a, b\}$, $s_2 = \{a, c\}$, $s_3 = \{b, d\}$, $s_4 = \{b, c\}$.

- Let $w_{uv} = \frac{1}{h+1}$ for all $(u, v) \in E$.
- Let the total resource $R = h$.

In the following, we show that the optimal defending strategy of the above instance (under uniform attacks) can be translated into the optimal solution to the maximum coverage instance, which proves the NP-hardness. Note that since $k = 0$, any defending strategy only needs to specify the allocation strategy: the reallocation can be trivially done by transferring maximum resource to the node under attack, e.g., as in the resource sharing setting. First, we prove the following useful lemma, which enables us to look at only the canonical allocation strategies.

Lemma 5.1 *For the instance we constructed above, any optimal defending strategy must allocate all resources to nodes in V.*

Proof: Fix any optimal defending strategy. Let h_U (resp. h_V) be the total resource the algorithm allocates to nodes in U (resp. V). Then we have $h = h_U + h_V$, and it remains to show that $h_U = 0$. Recall that a node is well-defended if its defending power is at least its threshold. For each $v \in V$ that is well-defended, we have

$$p_v = r_v + \sum_{u \in N(v)} w_{uv} \cdot r_u = r_v + \frac{\sum_{u \in N(v)} r_u}{h+1} \geq \theta_v = 1.$$

On the other hand, since $\sum_{u \in N(v)} r_u \leq h_U$, we have $p_v \leq r_v + \frac{h_U}{h+1}$. Thus for each well-defended node $v \in V$ we have

$$r_v + \frac{h_U}{h+1} \geq 1. \tag{8}$$

Recall that $\alpha_u = 1$ for all $u \in U$ and $\alpha_v = n + 1$ for all $v \in V$. By picking h arbitrary nodes from V and allocating one unit of resource to each of them, the defending result is at most

$$\frac{1}{m+n} \cdot (n \cdot 1 + (m-h) \cdot (n+1)) < \frac{(m-h+1)(n+1)}{m+n}.$$

Hence in any optimal defending strategy, at least h nodes in V are well-defended, as otherwise its defending result is at least $\frac{(m-h+1)(n+1)}{m+n}$, which is not optimal. Let $T \subseteq V$ be the set of c well-defended nodes in V. By inequality (8), we have

$$\sum_{v \in T} \left(r_v + \frac{h_U}{h+1} \right) \geq |T| = h.$$

On the other hand, we have $\sum_{v \in T} r_v \leq h_V$, which implies

$$h_V + \frac{h}{h+1} \cdot h_U \geq h = h_U + h_V.$$

Observe that the above inequality holds only if $h_U = 0$. □

From Lemma 5.1 we infer that the optimal defending strategy allocates all resources to nodes in V. Moreover, since no two nodes in V are neighbors, the optimal defending strategy must allocate the resources to h nodes, each of which receives one unit of resource. Now the problem is to choose h nodes to allocate the resource, such that a maximum number of nodes in U are well-defended. By the way we set the parameters, if we have $r_v = 1$ for some node $v \in V$, then each of its neighbors $u \in N(v)$ is well-defended since its defending power $p_u \geq w_{vu} \cdot r_v = \frac{1}{h+1} = \theta_u$. Since the defending result is minimized when a maximum number of nodes in U are well-defended (conditioned on the fact that h nodes in V are well-defended), the h nodes in V that are well-defended in the optimal defending strategy correspond to the optimal solution for the maximum coverage problem. Hence the DCA problem is NP-hard under uniform attacks. □

5.2 Lower Bound for the Defending Result

While computing the optimal defending result is NP-hard, we show that we can compute a lower bound for the optimal defending result efficiently via linear programming. Similar to the MILP we have shown in Section 4.2, it is straightforward to show that the computation of optimal defending strategy can be formulated as the following MILP.

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

Note that compared to the MILP we present in Section 4.2, the objective is changed from minimizing $\max_{u \in V} \mathsf{Loss}(u)$[6] to minimizing $\frac{1}{n} \cdot \sum_{u \in V} \mathsf{Loss}(u)$, e.g., minimizing the average damage over all nodes. Given total resource R, we use $\text{MILP}_{\text{uni}}(R)$ to denote the optimal objective of above program. Unfortunately, unlike the adaptive attack case, the computation of $\text{MILP}_{\text{uni}}(R)$ takes much longer time. This is mainly due to the fact that the optimal solution to the above MILP (which corresponds to the optimal defending strategy) needs to compute the optimal reallocation strategy for every node. In contrast, when the attack is adaptive, it is often sufficient to compute optimal reallocations for a few crucial nodes, when computing the optimal strategy. Nevertheless, we can relax the integer constraint $x_v^u \in \{0,1\}$ of $\text{MILP}_{\text{uni}}(R)$ to $x_v^u \in [0,1]$, and obtain the linear program relaxation $\text{LP}_{\text{uni}}(R)$. Since the linear program can be solved efficiently, we use the optimal objective of $\text{LP}_{\text{uni}}(R)$ as a lower bound for the optimal defending result.

6. Recall that $\mathsf{Loss}(u) = \sum_{v \in N_k(u)} (1 - x_v^u)\alpha_v$ is the damage the attack on node u causes.

5.3 Approximation Algorithm

We show that the bi-criteria approximation algorithm we proposed in Section 4.4 applies to the uniform attack setting.

Theorem 5.2 *For any $\epsilon \in (0,1)$, we can compute a $(\frac{1}{1-\epsilon}, \frac{1}{\epsilon})$-approximate defending strategy in polynomial time, when the node under attack is chosen uniformly at random.*

Proof: Since the proof is almost identical to that of Theorem 4.3, here we only state the key steps in the analysis. Recall that we need to compute a strategy with the defending result at most $\frac{1}{1-\epsilon} \cdot \mathsf{OPT}$, where OPT is the optimal objective of $\text{MILP}_{\text{uni}}(\epsilon \cdot R)$. We first run $\mathsf{LP}_{\text{uni}}(\epsilon \cdot R)$ and obtain the optimal (fractional) solution $(\mathbf{x}, \mathbf{r}, \mathbf{t})$. Then we round the solution into a feasible solution $(\hat{\mathbf{x}}, \hat{\mathbf{r}}, \hat{\mathbf{t}})$ for $\text{MILP}_{\text{uni}}(R)$ as follows. For each variable $x_v^u \in [0,1]$, let $\hat{x}_v^u = 1$ if $x_v^u \geq \epsilon$, and let $\hat{x}_v^u = 0$ otherwise. Let $\hat{r}_u = \frac{1}{\epsilon} \cdot r_u$ and $\hat{t}^u(z,v) = \frac{1}{\epsilon} \cdot t^u(z,v)$. The feasibility of $(\hat{\mathbf{x}}, \hat{\mathbf{r}}, \hat{\mathbf{t}})$ for $\text{MILP}_{\text{uni}}(R)$ follows straightforwardly from the proof of Theorem 4.3.

Since we decrease each x_v^u (to 0) only if $x_v^u < \epsilon$, by rounding $\mathbf{x}$ into $\hat{\mathbf{x}}$, for each $u \in V$, we have $\sum_{v \in N_k(u)}(1 - \hat{x}_v^u)\alpha_v \leq \frac{1}{1-\epsilon} \cdot \sum_{v \in N_k(u)}(1 - x_v^u)\alpha_v$. In other words, under solution $(\hat{\mathbf{x}}, \hat{\mathbf{r}}, \hat{\mathbf{t}})$, the loss at every node is at most $\frac{1}{1-\epsilon}$ times its loss under solution $(\mathbf{x}, \mathbf{r}, \mathbf{t})$. Since the objective of $\text{MILP}_{\text{uni}}(R)$ is the average loss over all nodes, the theorem follows immediately. □

As before, we can optimize the bi-criteria approximation algorithm by enumerating different parameters ϵ and τ to run the linear program and do the rounding (see Section 4.4). As we will show by our experiments (see Section 6), the approximation algorithm achieves a defending result that is very close to the lower bound of the optimal defending result, which implies that our algorithm is close to optimal.

6. Experimental Evaluation

In this section, we evaluate the effectiveness and efficiency of our algorithms in both synthetic and real-world datasets. The synthetic datasets include the random graphs (Rand-S, Rand-L) and the power-law distribution graphs (Pow-S, Pow-L). The real-world datasets include the aviation networks (Air-US) and social networks (Email-EU, Facebook, Twitter). We summarize the datasets as follows.

	Rand-S	Rand-L	Pow-S	Pow-L	Air-US	Email-EU	Facebook	Twitter
# Node	200	500	400	700	221	500	600	1000
# Edge	803	2569	1579	2087	2166	2468	4638	13476

- **Random**: We generate the dataset with $(n,p) = (200, 0.04)$ for Rand-S and $(n,p) = (500, 0.02)$ for Rand-L using the algorithm by (Batagelj & Brandes, 2005), where there are n nodes and 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 $[1,9]$. The edge weights w_{uv}'s are uniformly chosen in $[0.3,1]$.

- **Power-Law (Pow)**: We use the graph generator by NetworkX (Hagberg et al., 2008) to generate the power-law distribution graphs, where we set the parameters[7] 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 in Rand-S.

- **Air-US**: We select the flight records in the US from years 2008 to 2010 (Sharma, 2020) 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.

- **Communication and Social Network:** We include networks from Email-EU-core (directed), Facebook (undirected) and Twitter (directed) data to generate our datasets (Leskovec & Krevl, 2014). All of them are extracted from the original data by picking a random node in the network and expand from it by breadth-first-search until the size of the generated network reaches $n = 500$ ($n = 600$ and $n = 1000$). The parameters θ_u's, α_u's are generated randomly with the same setting as in the random graph datasets. In addition, all edge weights in these networks are set to 1.

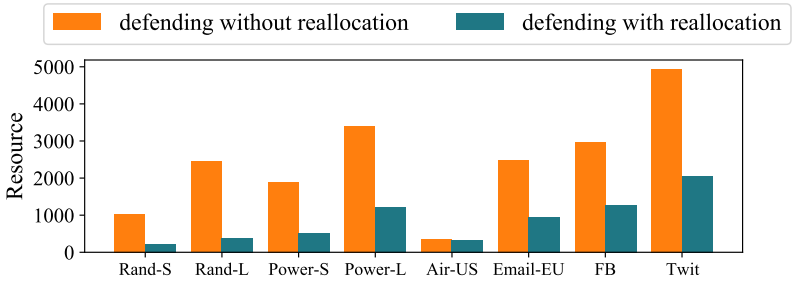
Experiment Environment. We perform our experiments on an AWS Ubuntu 18.04 machine with 32 threads and 128GB RAM without GPU. We use Gurobi optimizer (Gurobi Optimization, 2020) as our solver for the LPs and MILPs.

6.1 Effectiveness of Reallocation

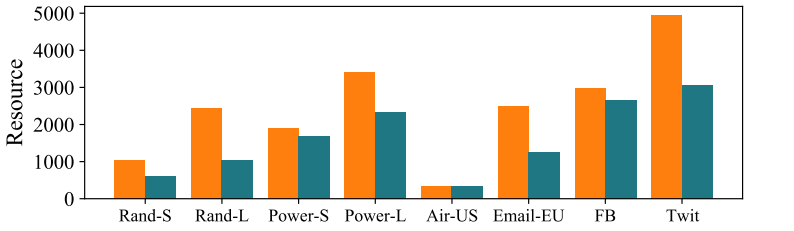
As we have introduced in the previous sections, the effectiveness of our algorithm relies on the reallocation of defending resources among different nodes. Such kind of reallocation strategies allows our algorithm to maximize resource utilization according to the attacks. In this subsection, we compare the defending results between defending strategies with and without resource reallocation to demonstrate the effectiveness of resource reallocation.

We use the algorithm introduced in Section 4.3 to compute for each dataset the minimum total resource required in a perfect defending strategy (a strategy with defending result 0). In the case where the reallocation is not allowed, the resource required for a perfect defending strategy will be exactly the sum of defending requirement of all nodes, i.e., $R = \sum_{u \in V} \theta_u$. In comparison, in the scenario when reallocation is allowed, the total defending resource required is significantly reduced (see Figure 8). In general, we observe that the advantage of reallocation is more obvious when the attack is less contagious, e.g., $k = 1$, and the graph is highly connected. For example, in dataset Rand-L, when $k = 1$, the defending resource required for obtaining a perfect defending result is reduced by about 85% compared to the setting where reallocation is not allowed. When $k = 2$, the reduction is at 60%. This is quite intuitive: since reallocating resources from one node to another decreases the defending power of the source node, when the contagious level is high, e.g., $k = 2$, it is harder to do an effective reallocation, because more nodes are under attack.

7. For the details regarding how the parameters define the graph, please refer to `https://networkx.github.io/documentation/networkx-1.10/reference/generated/networkx.generators.random_graphs.powerlaw_cluster_graph.html`.



(a) k = 1



(b) k = 2

Figure 8: Resource required to achieve defending result 0.

6.2 Comparing Different Algorithms

In this section, we evaluate the effectiveness of the bi-criteria approximation algorithm introduced in Section 4.4, and compare it with the exact solution and different variants of Greedy algorithms.

- **Greedy:** The algorithm greedily allocates resources to nodes with the maximum value (break ties arbitrarily) until the resources are exhausted, where the resource allocated to a node u is equal to its threshold θ_u.

- **Greedy-R:** Greedy-R is a variant of Greedy that reallocates resources to nodes as in Greedy but also uses resource reallocation. Specifically, when node u is attacked, Greedy-R reallocates resource to each undefended $v \in N_k(u)$ with the highest value (in decreasing order) from v's unattacked neighbor until its defending power is at least θ_v, or when no more resource can be transferred.

- **BA(ϵ):** The approximation algorithm obtained by rounding the optimal solution for LP$(\epsilon \cdot R)$ and optimizing $\epsilon \in (0, 1)$;

- **BA(ϵ, τ):** Compared to BA(ϵ), BA(ϵ, τ) further optimizes $\tau \in (0, \epsilon]$ in the more aggressive rounding (see Section 4.4).

- **Exact:** The baseline algorithm solves the MILP to obtain the exact optimal solution.

In this section, we demonstrate the effectiveness of our bi-criteria approximation algorithm by comparing it with other algorithms. Specifically, we report the defending results of these algorithms under different total resource and contagiousness.

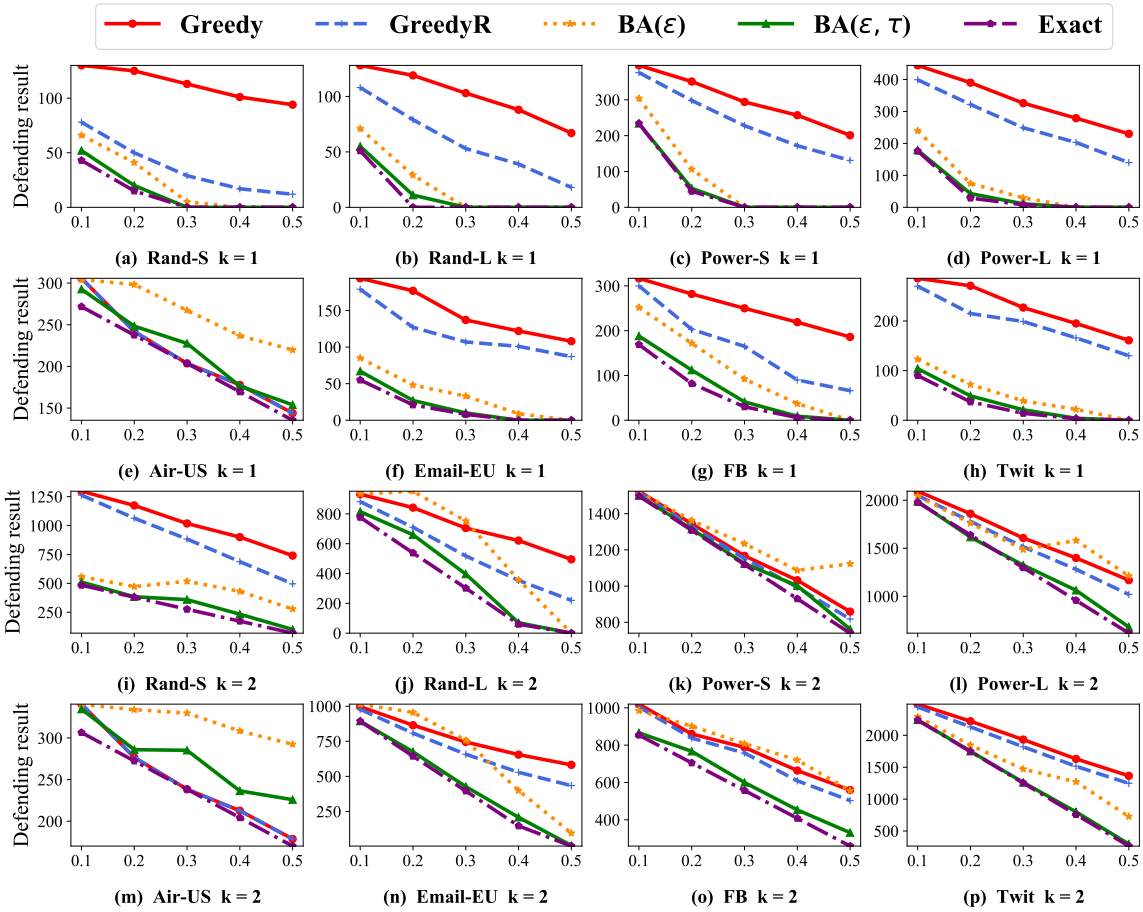


Figure 9: The defending results of different algorithms by varying total resource and contagiousness level. The abscissa value in the figure is η (taking values in $\{0.1, 0.2, 0.3, 0.4, 0.5\}$), which represents the ratio between the total resource and sum-of-threshold.

Varying Total Resource. We denote by η the ratio between the total resource and sum-of-threshold, e.g., $R = \eta \cdot \sum_{u \in V} \theta_u$. In this comparison, we choose η from $\{0.1, 0.2, 0.3, 0.4, 0.5\}$, and compare the defending results of different algorithms under different η. In Figure 9(a) - (h) we report and compare the defending results for $k = 1$; in Figure 9(i) - (p) we report the results when $k = 2$. Note that for any defending algorithm, the defending result is non-increasing with the total resource. However, how fast the defending result decreases with respect to the total resource depends on how efficient the resources are allocated and reallocated in the algorithm. As we can see from Figure 9, our bi-criteria approximation algorithm BA(ϵ, τ) outperforms other algorithms dramatically in almost all datasets[8], for every choice of η. For example, the defending result by BA(ϵ, τ) is 30% - 100% lower than that of Greedy-R for different η in dataset Facebook (see Figure 9(g)) and 60% - 100% lower in dataset Twitter (see Figure 9(h)). Moreover, the defending result BA(ϵ, τ) achieves

8. The only exception is on the dataset Air-US, in which the values of nodes differ drastically and thus the Greedy algorithm is almost optimal.

decreases very fast when η increases, and is always very close to the optimal one. For example, in datasets Facebook and Twitter, BA(ϵ, τ) achieves defending results that are within a 10% difference with the optimal solution. In dataset Power-S and Power-L, the defending results by BA(ϵ, τ) are almost identical to the optimal one.

There are several other interesting observations that are worth noting. For example, BA(ϵ, τ) constantly outperforms BA(ϵ) in all datasets and choices of η. Compared with BA(ϵ), the defending result of BA(ϵ, τ) is 20% - 100% lower, for different η in dataset Facebook (see Fig 9(g)). This demonstrates the usefulness of the more aggressive rounding method (see Section 4.4 for a detailed description), in which the potential of the fractional solution for LP$(\epsilon \cdot R)$ are better explored. We also observe that Greedy-R performs better than Greedy, which again demonstrates the critical role of resource reallocation in the problem.

Varying Contagiousness. Recall that the contagiousness parameter k is used to indicate how far (in level of neighbors) an attack can spread to from the initial node under attack. In this subsection, we perform the experiments on different choices of $k \in \{1, 2\}$ to evaluate how the contagiousness of the attack affects the defending results of different algorithms (see Figure 9). As expected, compared to the case when $k = 1$, the defending results of all algorithms increase by a lot when $k = 2$. In most datasets[9] and $\eta \in \{0.1, 0.2\}$, with a higher contagiousness ($k = 2$), the defending results by the optimal solution and BA(ϵ, τ) are 6 to 25 times higher than the case when $k = 1$.

For example, on dataset Rand-S when $\eta = 0.1$, when k increases from 1 to 2, the defending result of the optimal solution increases from 43 to 484; on dataset Twitter when $\eta = 0.1$, the defending result of the optimal solution increases from below 100 to above 2200. We can also observe that the increase in the defending result (when k increases from 1 to 2) is more dramatic in graphs with higher edge densities, e.g. Rand-L and Twitter.

For different levels of contagiousness, we can observe that the bi-criteria algorithm BA(ϵ, τ) constantly outperforms other algorithms, for different values of η. However, its superiority is less when the contagiousness is higher. For example, in dataset Power-S, while BA(ϵ, τ) achieves defending results that are less than half that of Greedy-R and Greedy when $k = 1$, for the case when $k = 2$, the difference is almost negligible. This is also quite natural: higher contagiousness causes a larger size of nodes under attack, in which case the space for optimization is much smaller.

Increasing Resources vs. Limiting Contagiousness. In order to reduce the defending result, we can either invest more defending resources or lower the level of contagiousness. To summarize the previous results from a different perspective, in the following we compare the effectiveness of these two approaches. Consider the optimal defending result in dataset Rand-L. When $\eta = 0.2$, the defending result is at around 500 when $k = 2$. Observe that with the same total resource but with $k = 1$, the defending result becomes 0. Alternatively, with $k = 2$ and $\eta = 0.5$, we also achieve a 0 defending result. That is to say, lowering the contagiousness is equivalently effective as increasing the total resource by a factor of 2.5. This indicates that limiting the contagiousness usually is much more effective compared with increasing the total resource, which agrees with the real-world situation, especially when it is unrealistic (or impossible) to increase the defending resource, e.g., the medical resources in the current COVID-19 situation. Mapping to the real-world situation, the

9. Again, the only exception is on dataset Air-US (a graph with small diameter), for which the difference between $k = 1, 2$ are very small.

social-distancing and wearing of masks, which limit the contagiousness, have been proved to be the most effective ways as the containment of the pandemic.

6.3 Efficiency Evaluation

In this subsection, we evaluate the efficiency of our approximation algorithms by comparing their running time with the running time to compute the exact solutions, i.e., by solving the MILP. The results are shown in Figure 10, where Exact refers to the algorithm that solves the MILP and BA(ϵ, τ) refers to our bi-criteria approximation algorithm. In this subsection, we fix the parameter $\eta = 0.5$. To demonstrate how the efficiency of the algorithms are affected by the contagiousness level, we conduct the experiments by varying k from 1 to 3.

Remark. Since solving the MILP can take a very long time when there is a large number of variables, in our experiments we limit the running time to be at most 10 hours (3.6×10^4 seconds). In other words, the algorithm will be aborted if it does not terminate after running for 10 hours.

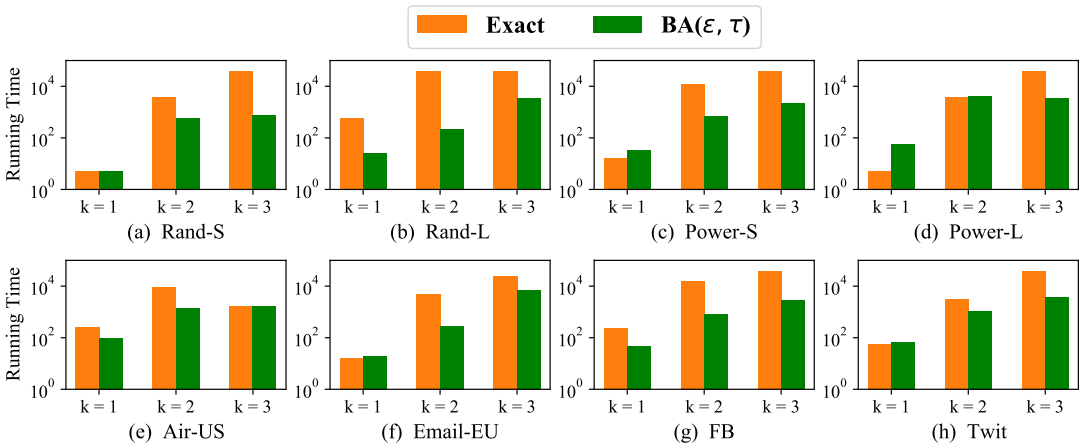


Figure 10: Running times (in seconds) of different algorithms.

Recall that the variables and constraints of the MILP increase exponentially when k grows. As we can see from Fig. 10, in most datasets the running times for computing the optimal solution when $k = 2$ are 100 times larger than the case when $k = 1$. In particular, in dataset Rand-S (see Fig. 10(a)), the running time increases from 10 seconds to 3000 seconds when k increases from 1 to 2. In most datasets, the computation of exact solutions fails to terminate within 10 hours when $k = 3$. In contrast, our algorithm BA(ϵ, τ) is less sensitive to the increase in contagiousness level k and is always able to compute the approximation solution within 10^4 seconds.

When $k = 1$, since the computation of exact solutions is quite efficient, we only observe some slight advantage in running time of our approximation algorithm, e.g., on datasets Rand-L and Facebook. However, with a higher contagiousness level ($k = 2$), our approximation algorithm is often more than 10 times faster than computing the exact solution. For example, in dataset Rand-L when $k = 2$, BA(ϵ, τ) terminates in 100 seconds while the computation of exact solutions fails to finish within the time limit. In other words, the scalability of our algorithm is much better than the optimal solution. Recall from Fig 9 that BA(ϵ, τ) achieves defending results that are very close to the optimal solution, in all datasets and different choices of η and k.

6.4 Network Structures

In this subsection, we study the correlations between the structure of the network and the defending results. We use the graph generator by NetworkX to generate a set of power-law distribution graphs. The parameters of power-law distribution graphs are (n, m, p), where n denotes the number of nodes (that will be added to the graph one-by-one) and m denotes the number of random edges to add when each new node is added to the graph. Therefore, the total number of edges in the graph will be $n \cdot m$. The parameter p denotes the probability of adding a triangle after adding a random edge and we do not concern this parameter. In this experiment, we set p to 0.5. We fix n to 400 and choose m from $\{1, 5, 10, 20, 30\}$, so that we can generate graphs with different densities. We let $k = 1$ and $\eta = 0.3$, where η is the ratio between the total resource and sum-of-thresholds (see Section 6.2). Then we compute the defending result in each graph and show the results in Table 2.

m	1	5	10	15	20	30
# Node	400	400	400	400	400	400
# Edge	399	1963	3854	5689	7461	10834
BA(ϵ, τ)	25	0	144	263	552	661
Exact	20	0	136	256	550	623

Table 2: Defending results under different network structures.

From Table 2, we observe that the defending result grows rapidly with the density of the network. When $m = 1$, the defending result is 25 since the graph is sparse. The defending result soars to 661 when $m = 30$ because when the density of the network is very high, the attack on one node will spread to a large set of neighboring nodes. Interestingly, the defending result is 0 when $m = 5$, which is less than the result when $m = 1$. The reason is that, a higher level of connectivity in the network structure also enables more flexibility for resource reallocations in designing defending strategies. As the results demonstrate, for different network structures, our bi-criteria approximation algorithm can always achieve defending results that are close to the optimal, which shows the robustness of the algorithm against different network structures.

6.5 Uniform Attack vs. Adaptive Attack

Finally, we study the difference between the uniform attack model and the adaptive attack model. Recall that in the uniform attack model, the node attacked is chosen uniformly at random among all nodes, regardless of the defending strategy. We explore how the defending results change when the attacker chooses the node to attack differently under these two models. We also evaluate the performance of our bi-criteria approximation (see Section 5.3) in the uniform attack model.

Recall that since computing the optimal defending result is time consuming, in Section 5.2 we propose a method that gives a lower bound on the optimal defending result. We denote by LB_{uni} this lower bound in this subsection. For a comparison, we use $\text{DR}_{\text{adaptive}}$ to denote the optimal defending result in the adaptive attack model. In addition, we use $\text{BA}_{\text{uni}}(\epsilon, \tau)$ to denote the defending result achieved by the bi-criteria approximation algorithm in the uniform attack model. Throughout the comparisons, we fix $\eta = 0.1$, which means $R = 0.1 \cdot \sum_{u \in V} \theta_u$. For each $k \in \{1, 2\}$, we report the

value of $\text{DR}_{\text{adaptive}}$, LB_{uni} and $\text{BA}_{\text{uni}}(\epsilon, \tau)$ for each dataset in Fig. 11. Note that while we are not able to compute the optimal defending result for the uniform attack model, we do know that its value is between LB_{uni} and $\text{BA}_{\text{uni}}(\epsilon, \tau)$. As we can observe from Fig. 11, in all datasets, $\text{BA}_{\text{uni}}(\epsilon, \tau)$ is very close to LB_{uni}. In other words, the experimental result shows that our approximation solution is also close-to-optimal under the uniform attack model. We also observe that there is a significant gap between the defending result $\text{DR}_{\text{adaptive}}$ under adaptive attacks and the defending results under uniform attacks. Specifically, in dataset Power-S when $k = 1$, $\text{DR}_{\text{adaptive}}$ is above 200, while $\text{BA}_{\text{uni}}(\epsilon, \tau)$ is below 10. This result shows that in the network defending problems, the "worst case" loss due to attack can be much larger than the "average case" loss.

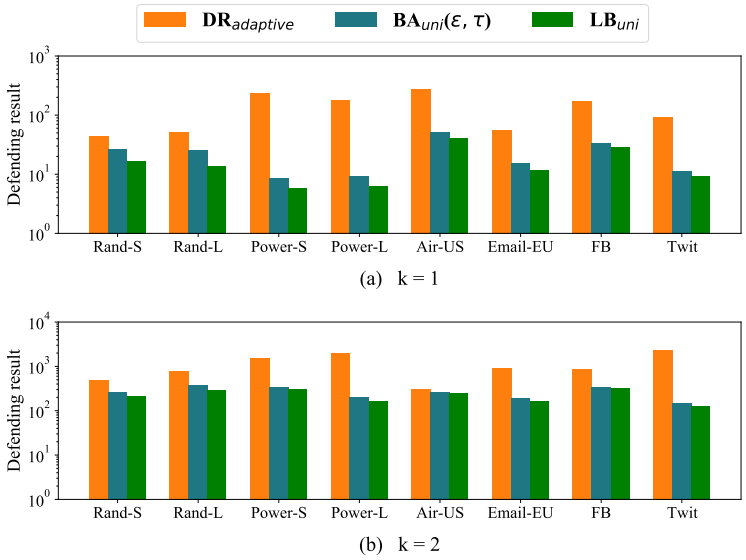


Figure 11: Upper and lower bounds for defending result of uniform attack.

7. Conclusion

In this work we study the problem of defending against contagious attacks in a network. Different from existing works that consider resource sharing (which might not be applicable against contagious attacks), we consider the model in which the resource can be reallocated between neighboring nodes. We prove that the problem of computing the optimal strategy is NP-hard, and we formulate the computation as an MILP. We further propose a polynomial time bi-criteria approximation algorithm, which we show, both theoretically and experimentally, is close to optimal.

Our work leaves many interesting problems open. For example, our work assumes that the contagious level is fixed regardless of the defending strategy, while might not be able to model applications in which well defended nodes can stop the attack from spreading. It would be interesting to see how our results adapt in this model. In this paper, we focus on the case where only a single node is attacked. It would also be interesting to consider the scenario in which multiple nodes may be attacked simultaneously, or when there are multiple independent attackers. However, as our hardness results suggest, getting non-trivial algorithmic results (even approximately) in these models can be quite challenging.

Acknowledgements

This work was supported by the Science and Technology Development Fund (FDCT) Macau SAR (file no. 0014/2022/AFJ, 0085/2022/A, 0143/2020/A3, SKL-IOTSC-2021-2023). Experiments were conducted at SICC supported by SKL-IOTSC, University of Macau. Minming Li was supported by the Fundamental Research Funds for the Central Universities. Weijia Jia's work was supported in part by the Guangdong Key Lab of AI and Multi-modal Data Processing, United International College (UIC), Zhuhai under Grant 2020KSYS007 sponsored by Guangdong Provincial Department of Education; in part by the Chinese National Research Fund (NSFC) under Grant 62272050; in part by Institute of Artificial Intelligence and Future Networks (BNU-Zhuhai) and Engineering Center of AI and Future Education, Guangdong Provincial Department of Science and Technology, China; Zhuhai Science-Tech Innovation Bureau under Grants ZH22017001210119PWC and 28712217900001, and in part by the Interdisciplinary Intelligence SuperComputer Center of Beijing Normal University (Zhuhai).

References

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

An, B., et al. (2013). Security games with surveillance cost and optimal timing of attack execution. In *AAMAS*, pp. 223–230. IFAAMAS.

Aspnes, J., et al. (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., & Goyal, S. (2013). Contagion and observability in security domains. In *Allerton*, pp. 1364–1371. IEEE.

Batagelj, V., & Brandes, U. (2005). Efficient generation of large random networks. *Phys. Rev. E*, *71*, 036113.

Berman, P., & Fujito, T. (1999). On approximation properties of the independent set problem for low degree graphs. *Theory Comput. Syst.*, *32*(2), 115–132.

Bondi, E., Oh, H., Xu, H., Fang, F., Dilkina, B., & Tambe, M. (2020). To signal or not to signal: Exploiting uncertain real-time information in signaling games for security and sustainability. In *AAAI*, pp. 1369–1377. AAAI Press.

Chan, H., Ceyko, M., & Ortiz, L. E. (2012). Interdependent defense games: Modeling interdependent security under deliberate attacks. In *UAI*, pp. 152–162. AUAI Press.

Chan, H., Ceyko, M., & Ortiz, L. E. (2017). Interdependent defense games with applications to internet security at the level of autonomous systems. *Games*, *8*(1), 13.

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

Conitzer, V., & Sandholm, T. (2006). Computing the optimal strategy to commit to. In *Proceedings of the 7th ACM conference on Electronic commerce*, pp. 82–90.

Gan, J., An, B., & Vorobeychik, Y. (2015). Security games with protection externalities. In *AAAI*, pp. 914–920. AAAI Press.

Gan, J., An, B., Vorobeychik, Y., & Gauch, B. (2017). Security games on a plane. In *AAAI*, pp. 530–536. AAAI Press.

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

Gurobi Optimization, L. (2020). Gurobi optimizer reference manual..

Hagberg, A. A., Schult, D. A., & Swart, P. J. (2008). Exploring network structure, dynamics, and function using networkx. In Varoquaux, G., Vaught, T., & Millman, J. (Eds.), *Proceedings of the 7th Python in Science Conference*, pp. 11 – 15, Pasadena, CA USA.

Jin, R., He, X., & Dai, H. (2019). On the security-privacy tradeoff in collaborative security: A quantitative information flow game perspective. *IEEE Trans. Inf. Forensics Secur.*, *14*(12), 3273–3286.

Johnson, B., Grossklags, J., Christin, N., & Chuang, J. (2010). Uncertainty in interdependent security games. In *GameSec*, Vol. 6442 of *Lecture Notes in Computer Science*, pp. 234–244. Springer.

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

Kiekintveld, et al. (2009). Computing optimal randomized resource allocations for massive security games. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pp. 689–696.

Kloh, H., Schulze, B., Pinto, R. C. G., & Mury, A. R. (2012). A bi-criteria scheduling process with cos support on grids and clouds. *Concurr. Comput. Pract. Exp.*, *24*(13), 1443–1460.

Korzhyk, et al. (2010). Complexity of computing optimal stackelberg strategies in security resource allocation games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 24.

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

Lamballais, T., et al. (2022). Dynamic policies for resource reallocation in a robotic mobile fulfillment system with time-varying demand. *Eur. J. Oper. Res.*, *300*(3), 937–952.

Leskovec, J., & Krevl, A. (2014). SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`.

Letchford, J., Conitzer, V., & Munagala, K. (2009). Learning and approximating the optimal strategy to commit to. In *SAGT*, Vol. 5814 of *Lecture Notes in Computer Science*, pp. 250–262. Springer.

Li, M., Tran-Thanh, L., & Wu, X. (2020). Defending with shared resources on a network. In *AAAI*, pp. 2111–2118. AAAI Press.

Lou, J., Smith, A. M., & Vorobeychik, Y. (2017). Multidefender security games. *IEEE Intell. Syst.*, *32*(1), 50–60.

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

Paruchuri, P., Pearce, J. P., Tambe, M., Ordóñez, F., & Kraus, S. (2007). An efficient heuristic for security against multiple adversaries in stackelberg games. In *AAAI Spring Symposium: Game Theoretic and Decision Theoretic Agents*, pp. 38–46. AAAI.

Paul, S., Ni, Z., & Mu, C. (2020). A learning-based solution for an adversarial repeated game in cyber-physical power systems. *IEEE Trans. Neural Networks Learn. Syst.*, *31*(11), 4512–4523.

Schlenker, A., Thakoor, O., Xu, H., Fang, F., Tambe, M., Tran-Thanh, L., Vayanos, P., & Vorobeychik, Y. (2018). Deceiving cyber adversaries: A game theoretic approach. In *AAMAS*, pp. 892–900.

Sharma, A. (2020). Usa airport dataset..

Sinha, A., Fang, F., An, B., Kiekintveld, C., & Tambe, M. (2018). Stackelberg security games: Looking beyond a decade of success. In *IJCAI*, pp. 5494–5501. ijcai.org.

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

Tan, S., & Wang, Y. (2020). Graphical nash equilibria and replicator dynamics on complex networks. *IEEE Trans. Neural Networks Learn. Syst.*, *31*(6), 1831–1842.

Tsai, J., Nguyen, T. H., & Tambe, M. (2012). Security games for controlling contagion. In *AAAI*. AAAI Press.

Vazirani, V. V. (2013). *Approximation algorithms*. Springer Science & Business Media.

Vorobeychik, Y., An, B., Tambe, M., & Singh, S. P. (2014). Computing solutions in infinite-horizon discounted adversarial patrolling games. In *ICAPS*. AAAI.

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

Ye, D., Shen, S., Zhu, T., Liu, B., & Zhou, W. (2022). One parameter defense - defending against data inference attacks via differential privacy. *IEEE Trans. Inf. Forensics Secur.*, *17*, 1466–1480.

Yedidsion, L. (2012). Bi-criteria and tri-criteria analysis to minimize maximum lateness makespan and resource consumption for scheduling a single machine. *J. Sched.*, *15*(6), 665–679.

Yin, Y., Xu, H., Gan, J., An, B., & Jiang, A. X. (2015). Computing optimal mixed strategies for security games with dynamic payoffs. In *IJCAI*, pp. 681–688. AAAI Press.

Yin, Z., & Tambe, M. (2012). A unified method for handling discrete and continuous uncertainty in bayesian stackelberg games. In *AAMAS*, pp. 855–862. IFAAMAS.

Zhang, Y., An, B., Tran-Thanh, L., Wang, Z., Gan, J., & Jennings, N. R. (2017). Optimal escape interdiction on transportation networks. In *IJCAI*, pp. 3936–3944. ijcai.org.

Zhao, D., Wang, L., Wang, Z., & Xiao, G. (2019). Virus propagation and patch distribution in multiplex networks: Modeling, analysis, and optimal allocation. *IEEE Trans. Inf. Forensics Secur.*, *14*(7), 1755–1767.