Computing Optimal Mixed Strategies for Security Games with Dynamic Payoffs

Yue Yin[1,2], **Haifeng Xu**[3], **Jiarui Gan**[1,2], **Bo An**[4], **Albert Xin Jiang**[5]
[1]The Key Lab of Intelligent Information Processing, ICT, CAS
[2]University of Chinese Academy of Sciences, Beijing 100190, China
[3]University of Southern California, Los Angeles, CA 90007, USA
[4]School of Computer Engineering, Nanyang Technological University, Singapore 639798
[5]Department of Computer Science, Trinity University, San Antonio, TX 79968, USA
[1]{yiny, ganjr}@ics.ict.ac.cn,[3]haifengx@usc.edu,[4]boan@ntu.edu.sg,[5]xjiang@trinity.edu

Abstract

Security agencies in the real world often need to protect targets with time-dependent values, e.g., tourist sites where the number of travelers changes over time. Since the values of different targets often change asynchronously, the defender can relocate security resources among targets dynamically to make the best use of limited resources. We propose a game-theoretic scheme to develop dynamic, randomized security strategies in consideration of adversary's surveillance capability. This differs from previous studies on security games by considering varying target values and continuous strategy spaces of the security agency and the adversary. The main challenge lies in the computational intensiveness due to the continuous, hence infinite strategy spaces. We propose an optimal algorithm and an arbitrarily near-optimal algorithm to compute security strategies under different conditions. Experimental results show that both algorithms significantly outperform existing approaches.

1 Introduction

Security is a grand challenge faced by the world. Recently, facing rising threats from ISIS, many major cities have boosted security measures to protect potential attack targets, e.g., subway stations, tourist sites, shopping centers [Park, 2014]. Since strategic attackers can take advantage of patterns of security strategies, it is important to develop randomized security strategies considering both different weights of targets and attacker's surveillance capability.

One challenge lies in designing efficient security resource allocation strategies is that the relative importance of different targets can change over time. Thus the limited security resources need to be dynamically allocated for the best use of them. For example, the entrances of museums are usually crowed in the morning, leading to their high weights in security at that time; while the bar streets and entertainment centers usually require more security measures in the evening. Intuitively, to make the best use of limited security resources, the security agency should assign more resources to the museums in the morning, then transfer some resources to the bar streets in the evening. In this work, we aim at computing randomized dynamic defender strategies to protect targets with time-dependent payoffs.

There has been lots of research on game theory based security resource allocation [Tambe, 2011; Yang *et al.*, 2011; An *et al.*, 2013]. Many systems based on defender-attacker security game models have been successfully deployed [Tambe *et al.*, 2014]. However, most work assumes that the payoffs of targets are static over time [Shieh *et al.*, 2012; An *et al.*, 2011; Gan *et al.*, 2015; Vorobeychik *et al.*, 2014]. Some work considers changing target values, but they only allow the security agency or the adversary to move at pre-defined, discretized time points [Varakantham *et al.*, 2013; Fang *et al.*, 2013; Xu *et al.*, 2014] without analyzing how much loss the discretization can lead to when continuous strategy spaces are considered. While Yin et al. [2014] considered varying target values and that the defender can move resources at any time, their algorithms can only compute optimal pure defender strategies, which is unsuitable in domains where the attacker can easily conduct surveillance and recognize patterns of the defender's pure strategy.

We make five key contributions in this work. First, we propose a new Stackelberg game model, in which the targets' weights are time-dependent and both the defender and the attacker can act at any time, i.e., the defender can transfer resources at any time while the attacker can attack at any time. Thus both agents have continuous strategy spaces. Second, we propose COCO (Computing Optimal Coverage withOut transfer time) to compute the optimal mixed strategy for the defender when she can move security resources among targets without time delay. COCO represents mixed strategies in a compact way and avoids traversing the whole strategy space by exploiting the properties of the optimal defender strategies. Our third contribution is an approach to sample pure strategies based on the compact mixed strategy computed by

COCO. The defender can only make a finite number of transfers of resources in a time period in the real world, despite that she can transfer resources at any time point in this period. We sample such 'implementable' pure strategies to realize the compact optimal mixed strategy.

We also consider a more general case in which it takes non-negligible time to move resources among targets. Our forth contribution is an efficient approximation scheme to compute the near optimal defender mixed strategy. We propose an algorithm ADEN (computing Approximately optimal DEfender strategies in games with Nonzero transfer time) which leads to a defender utility at most ϵ less than the optimal defender utility. Our fifth contribution is detailed experimental analysis on the solution quality and computational efficiency of the proposed algorithms. Experimental results show that both COCO and ADEN make the defender better off than algorithms based on the assumptions of static payoffs or discretized strategy spaces.

2 Model

Assume that there are n targets $\mathcal{T} = \{1, \dots, n\}$ and the defender has m identical security resources ($n \geq m$). The defender protects the targets during a time period $[0, t_e]$ (e.g. the operation hours of a day) everyday. The importance of targets changes over time. We use a continuous function $v_i(t)(i \in \mathcal{T}, t \in [0, t_e], v_i(t) > 0)$ to represent the value of a target i at time t and assume $v_i(t)$ to be piecewise linear[1]. As in previous work, we model the problem as a Stackelberg game, where the defender commits to a randomized strategy first, and the attacker responds to this strategy.

Strategies. A pure defender strategy S includes the initial assignment of resources at time 0 and the subsequent transfers during $[0, t_e]$, i.e., $S = \langle (i, j, t) : i, j \in \mathcal{T}, t \in [0, t_e] \rangle$, where (i, j, t) represents that a resource is transferred from target i to target j at time t. (If a target i is protected at time 0, then $(i, i, 0) \in S$). We denote $D = \langle d_{ij} : i, j \in \mathcal{T} \rangle$ where d_{ij} represents the time needed to transfer resources from target i to target j. We denote $q_i(S, t)$, a 0-1 indicator, to represent whether target i is protected at time t if the defender plays pure strategy S. $q_i(S, t)$ is a piecewise constant function over time $t \in [0, t_e]$ with values in $\{0, 1\}$. Let $\mathcal{S}$ be the defender strategy space. Since the defender can transfer resources at any time during $[0, t_e]$, $\mathcal{S}$ is continuous. A mixed defender strategy can be represented by a distribution $\mathbf{x} = \langle x_S : S \in \mathcal{S} \rangle$ with x_S representing the probability density that pure strategy S is played by the defender. As in previous work, we restrict attacker strategies to pure strategies, denoted as $A = (i, t_a)$, representing that the attacker attacks target i at time t_a.

Utilities and Equilibrium. We assume the game to be zero-sum. If the attacker plays $A = (i, t_a)$ while the defender plays a strategy S with $q_i(S, t_a) = 0$, i.e., the attack succeeds, the attacker will gain the value of target i at time t_a, $v_i(t_a)$, and the defender gains $-v_i(t_a)$. Otherwise, if $q_i(S, t_a) = 1$, the attack fails and both players get utility 0. Given a defender's mixed strategy $\mathbf{x}$ and an attacker's pure strategy $A = (i, t_a)$, the expected attacker utility is

$$U_a(\mathbf{x}, A) = v_i(t_a) \cdot (1 - \int_{\mathcal{S}} x_S \cdot q_i(S, t_a)). \qquad (1)$$

The expected defender utility is $U_d(\mathbf{x}, A) = -U_a(\mathbf{x}, A)$ due to the zero-sum assumption.

The Stackelberg equilibrium is the same as the maximin equilibrium given the zero-sum assumption: the defender maximizes her minimum utility, or equivalently, minimizes the maximum attacker utility, which indicates that the attacker responds the best.

Compact Representation. It is complicated to directly compute $\mathbf{x}$ given its infinite dimensions. We map a mixed defender strategy into a compact form for the ease of computation. Given a mixed strategy $\mathbf{x}$, for each target $i \in \mathcal{T}$, we define a coverage function $c_i(t)$ which describes the probability of target i being covered by a resource at time t.

$$c_i(t) = \int_{\mathcal{S}} x_S q_i(S, t). \qquad (2)$$

$c_i(t)$ is a function ranging in $[0, 1]$ over time $t \in [0, t_e]$. A mixed strategy $\mathbf{x}$ corresponds to a coverage vector, $\mathbf{c}(t) = \langle c_i(t) : i \in \mathcal{T}, t \in [0, t_e] \rangle$. Given $\mathbf{x}$ and $A = (i, t_a)$, the expected attacker utility can also be represented as $U_a(\mathbf{c}(t), A) = v_i(t_a) \cdot (1 - c_i(t_a))$.

3 Negligible Transfer Time

We begin with a simple case in which the transfer time of resources is negligible, i.e., $d_{ij} = 0$. Zero transfer time is a realistic assumption in domains in which targets are close to each other or resources can be transferred quickly (e.g., using helicopters), thus transfer time is negligible compared with the operation time of a pure strategy, i.e., $[0, t_e]$. We will relax this assumption in the next section. In this section, we first propose an approach to compute the optimal coverage vector, then propose an approach to sample pure strategies with finite transfers to produce the optimal coverage vector.

3.1 Computing Optimal Coverage Vector

If the transfer time is negligible, we can treat each time point in $[0, t_e]$ as an independent time point, and just need to compute the optimal coverage vector by computing the 'locally' optimal coverage at each time point. Formally, at a specific time point t_0, $\mathbf{c}(t_0)$ is locally optimal at t_0 if it satisfies that

$$\max_{i \in \mathcal{T}} v_i(t_0)(1 - c_i(t_0)) \leq \max_{i \in \mathcal{T}} v_i(t_0)(1 - c'_i(t_0)), \forall \mathbf{c}'(t_0).$$

This indicates that the locally optimal coverage minimizes the maximum attacker utility if he attacks at t_0. We then explore how to achieve the locally optimal coverage. Let I represent an 'attack set' consisting of targets which lead to the locally optimal attacker utility at t_0 if the coverage of targets at t_0 is $\mathbf{c}(t_0)$. Formally,

$$v_i(t_0)(1 - c_i(t_0)) = v_j(t_0)(1 - c_j(t_0)), \forall i, j \in I; \qquad (3)$$

$$v_i(t_0)(1 - c_i(t_0)) > v_j(t_0)(1 - c_j(t_0)), \forall i \in I, j \in \mathcal{T} \setminus I. \quad (4)$$

We have the following Lemma on the properties of a locally optimal coverage.

[1] Piecewise linear functions are widely used to approximate complex functions, while the accuracy of approximation can be controlled by setting the number of linear segments.

Lemma 1. *A coverage vector $\mathbf{c}(t)$ is locally optimal at t_0 if and only if it satisfies $\sum_{i\in I} c_i(t_0) = m$, hence $c_j(t_0) = 0, \forall j \in \mathcal{T} \setminus I$. m is the number of resources.*

The detailed proofs for lemmas, propositions, and theorems in this paper are available in the online appendix[2]. We now consider computing a coverage vector which is locally optimal at any $t{\in}[0,t_e]$. We first assume that all targets are in the attack set all the time. Such an assumption may enlarge the attack set, and result in infeasible coverage on some targets during some time intervals, i.e., some targets need to be protected with negative probability. We then find these time intervals, revise the attack set and the coverage. We use an example to illustrate the process.

Example 1. *Assume that $t_e = 2$. There are 3 targets, 1 resource. The value functions of targets are: $v_1(t) = t, v_2(t) = -t + 10, v_3(t) = -0.5t + 5$. First, assume that all targets are in the attack set all the time. Based on Lemma 1 and the definition of attack set, the coverage functions satisfy that*

$$\begin{cases} \sum_{i\in\{1,2,3\}} c_i(t) = 1 \\ v_1(t)\cdot(1-c_1(t)) = v_2(t)\cdot(1-c_2(t)) = v_3(t)\cdot(1-c_3(t)). \end{cases}$$

Thus we can get coverage functions based on value functions $v_i(t)$. It is easy to compute that when $t \in [0,2], c_1(t) < 0, c_2(t) > 0, c_3(t) > 0$. It indicates that target 1 should not be in the attack set during $[0,2]$. Thus we can set $c_1(t) = 0$ for $t \in [0,2]$, and remove target 1 from the attack set. Similar to the previous step, we now compute the coverage functions for targets 2 and 3 which stay in the attack set:

$$\begin{cases} \sum_{i\in\{2,3\}} c_i(t) = 1 \\ v_2(t)\cdot(1-c_2(t)) = v_3(t)\cdot(1-c_3(t)). \end{cases}$$

The new functions $c_2(t) \geq 0, c_3(t) \geq 0, \forall t \in [0,2]$. Thus the coverage vector $\mathbf{c} = \langle c_i(t) \rangle$ is now locally optimal $\forall t \in [0,2]$ since it satisfies the condition in Lemma 1 everywhere.

We propose COCO (Computing Optimal Coverage withOut transfer time) in Algorithm 1, which computes the optimal coverage vector for targets in set I during time period $[b,e)$ recursively. We call COCO($I = \mathcal{T}, b = 0, e = t_e$) to compute the optimal coverage vector. In COCO, we first assume that the target set I is the attack set during time period $[b,e)$, then compute the corresponding coverage functions of the targets in I (Line 2). E is a vector of all roots of the coverage functions with elements in increasing order, i.e., t_k represents the k^{th} smallest root in time period (b,e) (Line 3). If E is empty, Line 5 checks the sign of coverage functions during (b,e). If there does not exist any function with negative value in this time period, the coverage vector is feasible in $[b,e)$ and the algorithm terminates (Line 6). Otherwise assume the attack set includes all targets with positive coverage during (b,e) and repeat the process (Lines 7-9). If E is not empty, during (t_k, t_{k+1}), no coverage function has values with opposite signs since (t_k, t_{k+1}) is a time period between the adjacent roots of any function. Line 15 uses a set I' to record targets with positive coverage during (t_k, t_{k+1}). Targets in I' are considered as in the attack set in time period $[t_k, t_{k+1})$ while other targets are not (Line 16). The process is repeated for each time period $[t_k, t_{k+1})$ (Line 17) to compute the coverage of targets in I' until the coverage vector at any time is feasible.

Algorithm 1: COCO(I, b, e)

```
1  while true do
2      $c_i(t) \leftarrow 1 - \frac{(n-m)\frac{1}{v_i(t)}}{\sum_{j\in I}\frac{1}{v_j(t)}}, \forall i \in I, t \in [b,e)$;
3      $E \leftarrow \{t_k : c_i(t_k) = 0, t_k \in (b,e), \forall i \in I\}$;
4      if $E = \emptyset$ then
5          $I' \leftarrow \{i : i \in I, c_i(t) \geq 0, \forall t \in (b,e)\}$ ;
6          if $I' = I$ then break;
7          else
8              $c_i(t) = 0, \forall i \in I \setminus I', \forall t \in [b,e)$ ;
9              COCO($I', b, e$);
10     else
11         $E \leftarrow E \cup \{t_0 = b, t_{|E|+1} = e\}$;
12         for $k \in \{0, \ldots, |E|\}$ do
13             $I' \leftarrow \emptyset$;
14             if $c_i(t) \geq 0, \forall i \in I, \forall t \in (t_k, t_{k+1})$ then
15                 $I' \leftarrow I' \cup \{i\}$;
16             else $c_i(t) = 0, \forall t \in [t_k, t_{k+1})$;
17             COCO($I', t_k, t_{k+1}$);
18 return $\mathbf{c} = \{c_i(t) : i \in I, t \in [b,e)\}$;
```

Theorem 2. *COCO computes the optimal coverage vector.*

3.2 Sampling Pure Strategies

Coverage functions computed by COCO can be continuously changing over time. It is computationally infeasible to treat any $t \in [0, t_e]$ as an independent time point and sample an allocation of resources based on the coverage at t. We now propose an approach to sample pure strategies. Given a coverage vector $\mathbf{c}(t)$, we first discretize time period $[0, t_e]$ as a sequence of time points $\theta = \langle \theta_k \rangle$ (k is an index) such that $\forall t \in (\theta_k, \theta_{k+1}), \forall i \in \mathcal{T}$, $c_i(t)$ is differentiable and monotone. Given the piecewise linear assumption of value functions, Line 2 in COCO ensures $c_i(t)$ to be piecewise differentiable functions. Hence the discretization is feasible. The procedure of sampling a pure strategy is as follows.

Step 1: Sample an allocation of resources at θ_k.

Step 2: Given the allocation at θ_k, sample the departure targets and destination targets of transfers during (θ_k, θ_{k+1}).

Step 3: Given the transfers sampled in the previous step, sample the time of each transfer.

Step 4: Go to time point θ_{k+1} and repeat steps 1 - 3.

Next, we describe the implementation of the first three steps. For Step 4, although resources need to be reallocated at θ_{k+1}, the number of transfers at this time point is finite given the finite number of resources.

[2] http://www.ntu.edu.sg/home/boan/papers/IJCAI15_Dynamic_Appendix.pdf

Step 1: Let $L = \langle L_i : i \in \mathcal{T}, L_i \in \{0,1\}, \sum_i L_i = m\rangle$ represent an allocation of resources with L_i indicating whether target i is protected by a resource. Let y_L represent the probability of playing allocation L at time θ_k. Step 1 can be implemented once we get the value of y_L. Given $\mathbf{c}(\theta_k)$, y_L can be computed by approaches used in previous work on security games [Kiekintveld *et al.*, 2009; Letchford and Conitzer, 2013].

Step 2: We first show that a coverage vector can be realized by a specific set of transfers, which is formally shown in Proposition 3. Let $\Lambda = \{u : u \in \mathcal{T}, \frac{dc_u(t)}{dt} < 0, \forall t \in (\theta_k, \theta_{k+1})\}$ be a set of targets whose coverage decreases during (θ_k, θ_{k+1}); let $V = \{v : v \in \mathcal{T}, \frac{dc_v(t)}{dt} > 0, \forall v \in (\theta_k, \theta_{k+1})\}$ be a set of targets whose coverage increases during (θ_k, θ_{k+1}).

Proposition 3. *For $t \in (\theta_k, \theta_{k+1})$, $\mathbf{c}(t)$ can be implemented by sampling from pure strategies in which resources are only transferred from targets $u \in \Lambda$ to targets $v \in V$, and each resource is transferred for at most once.*

Proof Sketch: Assume that the defender uses a pure strategy S_1 with nonzero probability in which a resource is transferred from a target $v \in V$ to another target $i \in \mathcal{T}$ at $t_0 \in (\theta_k, \theta_{k+1})$. Since $c_v(t)$ increases continuously in (θ_k, θ_{k+1}), the defender must also use some other pure strategy S_2 with nonzero probability in which a resource is transferred from some target $j \in \mathcal{T}$ to target v at t_0. Thus the defender can cancel the transfer made from v in S_1 and directly transfer a resource from j to i in strategy S_2. If the defender uses a pure strategy S_3 in which a resource is transferred from some target $i \in \mathcal{T}$ to a target $u \in \Lambda$ at $t_0 \in (\theta_k, \theta_{k+1})$, she can get rid of this transfer in a similar way. Given that resources are only transferred from targets $u \in \Lambda$ to targets $v \in V$, a resource will not be transferred from target $v \in V$ to other targets during $[\theta_k, \theta_{k+1}]$. Thus each resource is transferred for at most once.

We now consider implementing Step 2. Let $\Gamma_L = \{(u,v) : u \in \Lambda, v \in V, L_u = 1, L_v = 0\}$ represent a 'transfer set' of target pairs, i.e., $\forall (u,v) \in \Gamma_L$, a resource is transferred from target u to target v during (θ_k, θ_{k+1}). Step 2 can be done once we get the probability of choosing each Γ_L, i.e., $p(\Gamma_L)$. To compute $p(\Gamma_L)$, we first denote $p((u,v)|L)$ as the probability that a resource is transferred from target u to target v during (θ_k, θ_{k+1}) after L is set as the allocation at θ_k. Once we get the value of $p((u,v)|L)$, $p(\Gamma_L)$ can be computed based on the following class of equations.

$$\sum_{\Gamma_L \ni (u,v)} p(\Gamma_L) = p((u,v)|L), \forall L \tag{5}$$

To compute $p((u,v)|L)$, we let $Z((u,v),t)$ represent the probability that a resource is transferred from target u to target v before time t. For $t \in [\theta_k, \theta_{k+1}]$, $Z((u,v),t)$ is a function with respect to t. If we can get the expression of $Z((u,v),t)$, $p((u,v)|L)$ can be computed based on the following equations.

$$\sum_L y_L{\cdot}L_u{\cdot}(1-L_v)\cdot p((u,v)|L) = Z((u,v),\theta_{k+1}), \forall u,v \tag{6}$$

$$p((u,v)|L) = 0, \forall u \in \Lambda \text{with } L_u = 0, \forall v \in V \text{with } L_v = 1 \tag{7}$$

Eq. 6 is based on the law of total probability. Eq. 7 indicates that a resource can only be transferred from a target which is protected at θ_k to a target which is not protected at θ_k. Our problem now lies in computing the expression of $Z((u,v),t)$.

Computing $Z((u,v),t)$**:** First, a feasible expression of $Z((u,v),t)$ satisfies the following $\forall t \in [\theta_k, \theta_{k+1}]$.

$$\sum_{v \in V} Z((u,v),t) = c_u(\theta_k) - c_u(t), \forall u \in \Lambda \tag{8}$$

$$\sum_{u \in \Lambda} Z((u,v),t) = c_v(t) - c_v(\theta_k), \forall v \in V \tag{9}$$

$$Z((u,v),t) \geq 0, \forall u \in \Lambda, \forall v \in V \tag{10}$$

$$\lim_{\Delta t \to 0} \frac{Z((u,v),t+\Delta t) - Z((u,v),t)}{\Delta t} \geq 0, \forall u \in \Lambda, v \in V \tag{11}$$

Eqs. 8 - 9 are based on Proposition 3. Eqs. 10 - 11 are based on the definition of $Z((u,v),t)$. At a time point $t \in [\theta_k, \theta_{k+1}]$, the value of $Z((u,v),t)$ can be computed based on Eqs. 8 - 11. But it is impossible to compute the expression of $Z((u,v),t)$ in this way due to the continuity of time. Note that given Eqs. 8 and 9, the function $Z((u,v),t)$ can be represented as linear combinations of coverage functions ($\forall t \in [\theta_k, \theta_{k+1}]$), i.e.,

$$\begin{aligned} Z((u,v),t) = &\sum_{i \in \Lambda} z^i_{uv}(t)(c_u(\theta_k) - c_u(t)) \\ &+ \sum_{j \in V} z^j_{uv}(t)(c_v(t) - c_v(\theta_k)). \end{aligned} \tag{12}$$

Here $z^i_{uv}(t)$ are parameters depending on t, or 'parameter functions'. We can get $Z((u,v),t)$ by computing these 'parameter functions'. At a specific time $t \in [\theta_k, \theta_{k+1}]$, we can get the value of the parameter functions based on Eqs. 8 - 11. In addition, we have the following proposition which discusses the properties of the parameter functions.

Proposition 4. *There exist parameter functions which lead to a feasible expression of $Z((u,v),t)$ satisfying that each $z^i_{uv}(t)$ is piecewise constant during $[\theta_k, \theta_{k+1}]$.*

The proof of Proposition 4 is based on Lemma 5.

Lemma 5. *If $|\Lambda| > 2$ and $|V| > 2$, there exists feasible $Z((u,v),t)$ satisfying $\lim_{\Delta t \to 0} \frac{Z((u,v),t+\Delta t) - Z((u,v),t)}{\Delta t} > 0, \forall u, \forall v, \forall t \in [\theta_k, \theta_{k+1}]$.*

Based on Proposition 4 and Lemma 5, we propose Algorithm 2 to compute the parameter functions, hence $Z((u,v),t)$. We first compute the value of the parameter functions at a time point t_0, hence we can get an expression of $Z((u,v),t)$ whose parameter functions are constant in $[t_0, \theta_{k+1}]$. Let R be the set of the roots of functions $\frac{\mathrm{d}Z((u,v),t)}{\mathrm{d}t}$ (Line 4). The expression of $Z((u,v),t)$ is feasible from time t_0 to the minimum time point in R (t_m, Line 8) since for $t > t_m$, Eq. 11 is violated. The process is repeated for t_m until we get the feasible expression of $Z((u,v),t)$ for $t \in [\theta,\theta_{k+1}]$.

Theorem 6. *Algorithm 2 computes a feasible expression of $Z((u,v),t), \forall t \in [\theta_k, \theta_{k+1}]$ after finite loops.*

Algorithm 2: Computing $Z((u,v),t)$

```
$t_0 = \theta_k, k_{uv} = 0$;
while $t_m < \theta_{k+1}$ do
    solve $z^i_{uv}(t_0)$ based on Eqs. 8 - 11 (change the ‘0’ in Eq. 10 to $k_{uv}$, change the ‘$\geq$’ in Eq. 11 to ‘>’);
    $R = \langle t : \frac{\mathrm{d}Z((u,v),t)}{\mathrm{d}t} = 0$ and $t \in (t_0, \theta_{k+1})\rangle$;
    if $R = \emptyset$ then
        break;
    else
        $t_m = \min_t R; k_{uv} = Z((u,v), t_0)$;
        $z^i_{uv}(t) = z^i_{uv}(t_0), \forall t \in [t_0, t_m); t_0 = t_m$;
```

Step 3: Let $Z(t|L,(u,v))$ be the cumulative distribution function of t, representing the probability that a resource is transferred from target u to target v before time t ($t \in [\theta_k, \theta_{k+1}]$) after L is set as the allocation at θ_k and step 2 takes a transfer set $\Gamma_L \ni (u,v)$. We can implement Step 3 as long as we can get the expression of $Z(t|L,(u,v))$. Note that the time at which the transfer made from target u to target v occurs is independent of the allocation at θ_k. Thus we have

$$Z(t|L,(u,v)) = Z(t|(u,v)) = \frac{Z((u,v),t)}{Z((u,v),\theta_{k+1})}, \forall t \in [\theta_k, \theta_{k+1}]. \tag{13}$$

Apparently, we can get the expression of $Z(t|L,(u,v))$ based on the expression of $Z((u,v),t)$.

4 Nonzero Transfer Time

In this section, we address general cases with non-negligible transfer time, i.e., $d_{ij} \geq 0$ and propose an efficient approximation scheme to compute the optimal defender's mixed strategy. We first obtain a discretized version of the game by properly partitioning the time space, and more importantly, show that the defender's optimal utility is at most ϵ less than the optimal utility in the original continuous version. We then approach the discrete game with a flow formulation. For constructing the discretized game, we first define an (ϵ,δ)-Mesh.

Definition 7. *((ϵ,δ)-Mesh) An (ϵ,δ)-mesh is a* uniform *discretization of the time space $[0, t_e]$, denoted as $\zeta = \langle t_0, t_1, ..., t_{|\zeta|}\rangle$ with $t_0 = 0, t_{|\zeta|} = t_e$, such that $\delta = t_k - t_{k-1}$ and $var_i(t_{k-1}, t_k) \leq \epsilon$ for all $k = 1, ..., |\zeta|$, where $var_i(t_{k-1}, t_k) = \max_{t\in[t_{k-1},t_k]} v_i(t) - \min_{t\in[t_{k-1},t_k]} v_i(t)$ is the oscillation of $v_i(t)$ in interval $[t_{k-1}, t_k]$.*

Continuity of value functions guarantees the existence of such an (ϵ,δ)-Mesh, which can be easily found[3]. We now convert a general game G into a discretized game G_d. First, we restrict the defender's resource move can only happen at time points $t_k \in \zeta$ and so is the attacker's attack. Thus the strategy space of G_d is a subset of the strategy space of G. Let $v_i(t)$ represent the value functions in G, we construct the value functions of G_d using the (ϵ,δ)-Mesh: $\forall i \in \mathcal{T}$, let

$$v'_i(t) = \max_{t'\in[t_{k-1},t_k]} v_i(t'), \forall t \in [t_{k-1}, t_k) \tag{14}$$

be the value functions of G_d. Note that $v'_i(t)$ is a constant in $[t_{k-1}, t_k]$ and satisfies $|v'_i(t) - v_i(t)| \leq \epsilon, \forall t$. We also assume that d_{ij} are multipliers of δ, i.e., $d_{ij} = a_{ij}\delta$ with a_{ij} being some integer. This assumption holds if each d_{ij} is given as a rational number and δ is small enough. To show the gap between the optimal defender utility in G_d and the initial game G, we introduce a bridge game G_b, in which the value functions are $v'_i(t)$ as in G_d while the defender and the attacker can act at any time as in G. The following lemma shows the relationship between defender utilities in G_d and G_b.

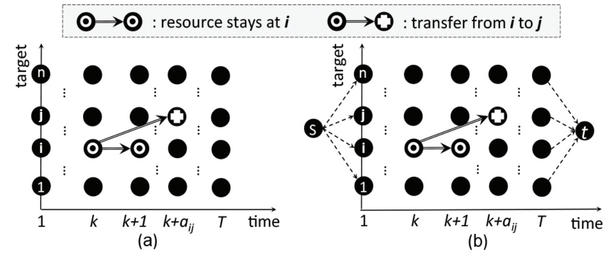


Figure 1: Construction of the discrete game

Lemma 8. *Let $\mathbf{x}$ be a defender's mixed strategy of G_d, then $U^{G_d}_d(\mathbf{x}) = U^{G_b}_d(\mathbf{x})$, where $U^{G_d}_d(\mathbf{x})$ and $U^{G_b}_d(\mathbf{x})$ are the defender's utility in G_d and G_b respectively if the defender plays $\mathbf{x}$ and the attacker responds the best.*

This is because the value functions of G_d and G_b are the same. Furthermore, a deeper conclusion holds.

Lemma 9. *Assume that transfer time between any target pair d_{ij} is multiplier of δ. If $\mathbf{x}$ is an equilibrium strategy of G_d, it is an equilibrium strategy of G_b.*

Note that this does not apply directly from Lemma 8, since the strategy space of G_d is a subset of the strategy space of G_b. Now we are ready to bound the defender's utility gap between G_d and G, as summarized in the following theorem.

Theorem 10. *Let $\mathbf{x}'$ be an equilibrium strategy of G_d and $\mathbf{x}$ be an equilibrium strategy of G, we have*

$$U^G_d(\mathbf{x}') - U^G_d(\mathbf{x}) \geq -\epsilon,$$

where $U^G_d(\mathbf{x}')$ is the defender utility in G if the defender plays $\mathbf{x}'$ and the attacker responds the best, while $U^G_d(\mathbf{x})$ is the optimal defender utility in G.

In addition, we have $U^G_d(\mathbf{x}') = U^{G_d}_d(\mathbf{x}')$ based on Eq. 14. Our task now reduces to computing the equilibrium of G_d. G_d is a discretized spatio-temporal game as is shown in Figure 1(a), in which x-axis is the (ϵ,σ)-Mesh and y-axis indicates the targets, i.e., node (k,i) is target i at time t_k. An attacker's pure strategy is a node in the graph. If we add a departure node and a destination node to the graph as is shown in Figure 1(b) and require that any edge of the graph has capacity 1, the defender's pure strategy is an m-unit $0-1$ integer $s-t$ flow on the graph, i.e., if the defender transfers a resource from target i to target j at time t_k, add an edge between node (k,j) and $(k+a_{ij}, j)$; if the defender does not transfer the resource on target i at time t_k, add an edge between node (k,i) and node $(k+1,i)$. We have the following theorem about the defender's mixed strategies in G_d.

[3] For Lipschitz continuous functions satisfying $|v_i(t) - v_i(t')| \leq K|t - t'|$ for all i, we can simply take $\epsilon = K\delta$.

Theorem 11. *A mixed strategy of G_d is an m-unit fractional flow, and vise versa.*

Let $f(i, j, t_k)$ represent the probability that a resource is transferred from target i to target j at t_k. The optimal defender utility can be computed by the following LP which we call ADEN (computing Approximately optimal DEfender strategies in games with Nonzero transfer time).

$$\textbf{ADEN}: \quad \min U \tag{15}$$

$$\text{s.t. } U \geq v'_i(t_k)(1-f(i,i,t_k)), \forall i \in \mathcal{T}, t_k \in \zeta \tag{16}$$

$$\sum_{i \in \mathcal{T}} f(s, i, t_0) = m, \tag{17}$$

$$f(s, i, t_0) - \sum_{j \in \mathcal{T}} f(i, j, t_0) = 0, \forall i \in \mathcal{T} \tag{18}$$

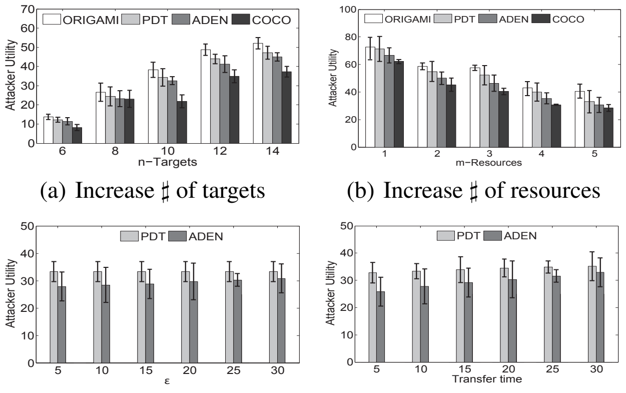
$$\sum_{j \in \mathcal{T}} f(j,i,t_{k-a_{ji}}) - \sum_{j \in \mathcal{T}} f(i,j,t_k) = 0, \forall i \in \mathcal{T}, t_{k-a_{ji}}, t_k \in \zeta \tag{19}$$

Note that the probability that a target i is covered by a resource at time t_k is equal to the probability that a resource stays at target i during the time period after t_k, i.e., $c_i(t_k) = f(i, i, t_k)$. Thus we have the objective and Eq. 16 together to guarantee the optimal defender utility. Eqs. 17 - 19 are for the properties of m-units $s - t$ flow. The coverage vector $\mathbf{c} = \langle f(i, i, t_k) : i \in \mathcal{T}, t_k \in \zeta \rangle$ computed by ADEN can be directly mapped back to a mixed strategy with a finite support, i.e., pure strategies which transfer resources at $t_k(\forall t_k \in \zeta)$. Technically, for a resource at any target i and time t_k, we can sample its next destination j with probability propositional to $f(i, j, t_k)$ ($j = i$ means staying at i). If there is a move, this resource will be in transition from t_k to $t_{k+a_{ij}}$. The initial assignment of resources can be sampled based on $f(s, j, t_0)$. This LP has size $poly(\frac{t_e}{\delta} n^2)$. When the value functions are all Lipschitz continuous functions satisfying $|v_i(t) - v_i(t')| \leq K|t - t'|$ for all i, we take $\epsilon = K\delta$ and generate an LP with size $poly(\frac{Kt_e}{\epsilon} n^2)$, which gives an ϵ-approximation to the original problem.

5 Experimental Evaluation

We evaluate the performance of the proposed algorithms in terms of attacker utility. A lower attacker utility indicates a higher defender utility given the zero-sum assumption. We assume $t_e = 100$. To illustrate the solution quality of the proposed algorithms, we consider two baseline strategies. The first one is the ORIGAMI algorithm [Kiekintveld *et al.*, 2009] assuming static payoffs. The second one, which we call PDT, is based on the assumption that both the defender and the attacker can act at pre-defined, discretized time points $\Phi = \{\frac{n \cdot t_e}{7} : n = 0, 1, \cdots, 7\}$ as in Xu et al. [Xu *et al.*, 2014]. We also evaluate the scalability of the proposed algorithms.

In each game, we sample piecewise linear value functions with at most two sections, i.e., the number of sections is randomly sampled in $\{1, 2\}$. If a value function is determined to have two linear sections, the discontinuity point will be randomly chosen in $(0, t_e)$, then the values of the function at time 0, t_e and the discontinuity point will be randomly chosen in $(0, 100]$, thus the value of a value function ranges in $(0, 100]$. We perform 4 experiment sets to show the performance of the algorithms. In the first experiment set, we restrict transfer time to be zero, and test the solution quality of the algorithms against increasing (1) number of targets; (2) number of resources. We then test the effect of changing ϵ and changing transfer time on the solution quality of ADEN. The third experiment set tests the scalability of the proposed algorithms. We also test the runtime of the sample procedure.



(a) Increase ♯ of targets

(b) Increase ♯ of resources

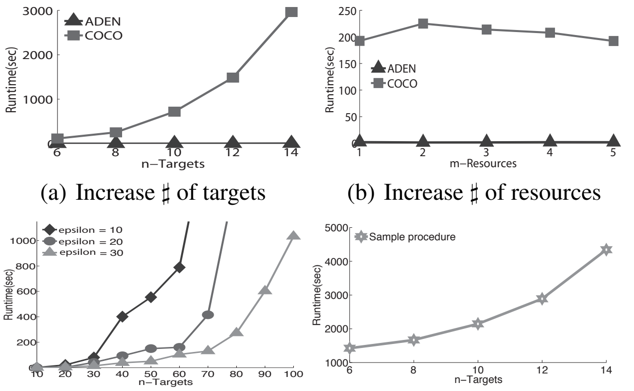
(c) Increase ϵ

(d) Increase d_{ij}

Figure 2: Attacker utility

We first evaluate the solution quality of the algorithms. In Figure 2, the y-axis of all subfigures indicate the optimal attacker utility. Figures 2(a) and 2(b) consider zero transfer time. In Figures 2(a), the x-axis indicates the number of targets. In Figure 2(b), the x-axis indicates the number of resources. These two figures show that despite the number of targets or resources, ADEN and COCO significantly perform better than the baseline strategies. COCO achieves lower attacker utilities than ADEN since it can compute the optimal defender strategies. Figures 2(c) and 2(d) further evaluate the solution quality of ADEN. Figure 2(c) evaluates the effect of increasing ϵ on ADEN. As the value of ϵ increases, the optimal attacker utility computed by ADEN increases gently. Since ϵ is an upper bound of the defender's loss, the defender's loss can be far less than ϵ in reality. Solution quality of PDT is not affected by the value of ϵ, but ADEN outperforms PDT even when $\epsilon = 30$. Figure 2(d) evaluates the effect of increasing transfer time on ADEN and PDT (ϵ is fixed at 10). The x-axis indicates different levels of transfer time, i.e., for $x = 10$, transfer time of resources is randomly generated in $[\frac{10}{2}, 10]$. Figure 2(d) shows that as transfer time increases, both ADEN and PDT performs worse. While the effect of increasing transfer time is more obvious on ADEN than it is on PDT. ADEN outperforms PDT even when the maximum transfer time reaches 30. If the transfer time is very large, i.e., larger than the operation time of pure strategies, then the defender's optimal strategy is static and ADEN and PDT will perform the same as ORIGAMI.

We then evaluate the scalability of the proposed algorithms. In Figure 3, the y-axis indicates runtime in seconds. In Figure 3(a), the x-axis indicates the number of targets. Runtime of COCO shows obvious increasing trend while that of ADEN does not change much as the number of targets



(a) Increase ♯ of targets

(b) Increase ♯ of resources

(c) ADEN

(d) Sample procedure

Figure 3: Runtime

ranges from 6 to 14. In Figure 3(b), the x-axis indicates the number of resources, which does not affect the runtime of COCO or ADEN much as is shown. Figure 3(c) extends the number of targets to 100 and shows the scalability of ADEN against different values of ϵ. Smaller ϵ leads to longer runtime and more obvious increasing trend in runtime. When $\epsilon = 30$, ADEN can solve games with 100 targets in 20 minutes. We did not show the runtime of COCO in Figure 3(c) since it runs out of memory and fails to return a solution when the number of targets $n > 15$. Figure 3(d) shows the runtime of the sample procedure for sampling pure defender strategies based on the compact optimal mixed strategy computed by COCO.

6 Conclusion

This paper makes four key contributions in computing optimal mixed strategies to protect targets with dynamic payoffs: (1) A Stackelberg game model in which both agents have continuous strategy spaces and the payoffs are dynamic. (2) COCO for computing the optimal mixed defender strategy in a compact form when the defender can move resources from one target to another immediately. (3) A sample procedure for sampling implementable pure strategies based on the compact optimal mixed strategy computed by COCO. (4) ADEN for computing an approximately optimal mixed strategy when transfer time of resources is non-negligible. We also provide extensive experimental analysis that shows the solution quality and scalability of the proposed algorithms.

7 Acknowledgements

This work is supported by NSFC grant No. 61202212 and Singapore MOE AcRF Tier 1 grant MOE RG33/13. This research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its IDM Futures Funding Initiative and administered by the Interactive and Digital Media Programme Office.

References

[An *et al.*, 2011] Bo An, Tambe Milind, Fernando Ordonez, Eric Shieh, and Christopher Kiekintveld. Refinement of strong Stackelberg equilibria in security games. In *AAAI*, pages 587–593, 2011.

[An *et al.*, 2013] Bo An, Matthew Brown, Yevgeniy Vorobeychik, and Milind Tambe. Security games with surveillance cost and optimal timing of attack execution. In *AAMAS*, pages 223–230, 2013.

[Fang *et al.*, 2013] Fei Fang, Albert Xin Jiang, and Milind Tambe. Optimal patrol strategy for protecting moving targets with multiple mobile resources. In *AAMAS*, pages 957–964, 2013.

[Gan *et al.*, 2015] Jiarui Gan, Bo An, and Yevgeniy Vorobeychik. Security games with protection externality. In *AAAI*, pages 914–920, 2015.

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

[Letchford and Conitzer, 2013] Joshua Letchford and Vincent Conitzer. Solving security games on graphs via marginal probabilities. In *AAAI*, pages 591–597, 2013.

[Park, 2014] Christine Park. Nyu reacts: Isis security threat. http://www.nyunews.com/2014/09/30/isis-2/, 2014.

[Shieh *et al.*, 2012] Eric Shieh, Bo An, Rong Yang, Milind Tambe, Craig Baldwin, Joseph DiRenzo, Ben Maule, and Garrett Meyer. PROTECT: An application of computational game theory for the security of the ports of the United States. In *AAAI*, pages 2173–2179, 2012.

[Tambe *et al.*, 2014] Milind Tambe, Albert Xin Jiang, Bo An, and Manish Jain. Computational game theory for security: Progress and challenges. In *AAAI Spring Symposium on Applied Computational Game Theory*, 2014.

[Tambe, 2011] Milind Tambe. *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned*. Cambridge University Press, 2011.

[Varakantham *et al.*, 2013] Pradeep Varakantham, Hoong Chuin Lau, and Zhi Yuan. Scalable randomized patrolling for securing rapid transit networks. In *IAAI*, pages 1563–1568, 2013.

[Vorobeychik *et al.*, 2014] Yevgeniy Vorobeychik, Bo An, Milind Tambe, and Satinder Singh. Computing solutions in infinite-horizon discounted adversarial patrolling games. In *ICAPS*, pages 314–322, 2014.

[Xu *et al.*, 2014] Haifeng Xu, Fei Fang, Albert Xin Jiang, Vincent Conitzer, Shaddin Dughmi, and Milind Tambe. Solving zero-sum security games in discretized spatio-temporal domains. In *AAAI*, pages 1500–1506, 2014.

[Yang *et al.*, 2011] Rong Yang, Christopher Kiekintveld, Fernando Ordonez, Milind Tambe, and Richard John. Improving resource allocation strategy against human adversaries in security games. In *IJCAI*, pages 458–464, 2011.

[Yin *et al.*, 2014] Yue Yin, Bo An, and Manish Jain. Game-theoretic resource allocation for protecting large public events. In *AAAI*, pages 826–833, 2014.