When Players Affect Target Values: Modeling and Solving Dynamic Partially Observable Security Games

Xinrun Wang[1], Milind Tambe[2], Branislav Bošanský[3], Bo An[1]

[1] Nanyang Technological University, {xwang033,boan}@ntu.edu.sg
[2] Harvard University, milind_tambe@harvard.edu
[3] Czech Technical University in Prague, bosansky@agents.fel.cvut.cz

Abstract. Most of the current security models assume that the values of targets/areas are static or the changes (if any) are scheduled and known to the defender. Unfortunately, such models are not sufficient for many domains, where actions of the players modify the values of the targets. Examples include wildlife scenarios, where the attacker can increase value of targets by secretly building supporting facilities. To address such security game domains with player-affected values, we first propose DPOS3G, a novel partially observable stochastic Stackelberg game where target values are determined by the players' actions; the defender can only partially observe these targets' values, while the attacker can fully observe the targets' values and the defender's strategy. Second, we propose RITA (Reduced game Iterative Transfer Algorithm), which is based on the heuristic search value iteration algorithm for partially observable stochastic game (PG-HSVI) and introduces three key novelties: (a) building a reduced game with only key states (derived from partitioning the state space) to reduce the numbers of states and transitions considered when solving the game; (b) incrementally adding defender's actions to further reduce the number of transitions of the game; (c) providing novel heuristics for lower bound initialization of the algorithm. Third, we conduct extensive experimental evaluations of the algorithms and the results show that RITA significantly outperforms the baseline PG-HSVI algorithm on scalability while allowing for trade off in scalability and solution quality.

1 Introduction

The past decade has witnessed the significant success of applying game theoretic methodologies to various security domains to optimize the allocation of limited resources against the strategic adversary [22, 24, 3, 8]. Examples include ARMOR for airport security [20], PROTECT for coast guard security [21], urban protection [23]. Recent works extend these methodologies to protect forest [14], fish [10] and wildlife [6], where large-scale areas are to be protected by limited resources, e.g., Uganda's Queen Elizabeth Protected Area (QEPA) with 2,520 square kilometers and only 37 patrol posts [15]. Most of previous works in Stackelberg security games assume that the targets are static, i.e., both values and

positions of targets are determined and known by both players [19, 7]. In this case, the defender will commit to an optimal mixed strategy to protect the targets and the pure strategies to be implemented are sampled from the mixed strategy when deployed to the real world.

However, in many wildlife and forest conservation domains, the adversary can build support facilities (e.g., set up camps or build roads to facilitate smuggling) for executing illegal activities to obtain a higher utility, which would change the values of targets in these domains. The values of targets are defined as the payoffs of the adversary when the illegal activities are successfully executed at the targets, e.g., setting up a camp would allow poachers to kill more endangered wildlife in the target zone. However, the defender may not know these changes unless the target is visited by the defender. Some works consider the case where the targets' values are not known by the defender and learning algorithms are proposed [16, 2]. However, in these works, the targets' values are static where the algorithm can learn the optimal defender's strategy offline. There are several works which consider the case that the targets dynamically change their values [27] or positions [5] over a finite time period. Both changes are deterministic and known by the players, however, none of them considers the case where the target can be affected by the players' actions, therefore the previous methods cannot be directly applied to our problem.

To apply the game theoretic methodology to security domains where the targets' values are affected by players' actions, we propose a novel partially observable stochastic Stackelberg security game where the defender can only observe the targets' values being protected and the attacker can fully observe the game. We consider the worst-case scenario where the attacker knows the values of the targets (i.e., the attacker is able to observe the action of the defender) and the defender's strategy. To solve the game, we propose RITA (Reduced game Iterative Transfer Algorithm), which is based on HSVI for partially observable stochastic game proposed in [12]. RITA provides three key novelties: (a) building a reduced game with only key states (derived from partitioning the state space) to reduce the numbers of states and transitions considered when solving the game; (b) incrementally adding defender's actions into the reduced game, instead of considering all defender's actions, to further reduce the number of transitions of the game; (c) providing novel heuristics for lower bound initialization of the algorithm. Finally, we do extensive experimental evaluations of the algorithms and the results show that RITA significantly outperforms the PG-HSVI algorithm on scalability while allowing for trade off in scalability and solution quality.

1.1 Motivation

Our model is quite general and can be easily applied to various security domains, such as the protection against illegal fishing, logging and poaching activities. As a specific motivation scenario of our model, we now briefly present the problem in the context of wildlife protection where rangers patrol target zones against the illegal poaching activities. Specifically, previous works based on Stackelberg

game assume that rangers sample the pure patrol strategy for each daily patrol from a precomputed distribution and only simple poaching activities taking one day to execute (e.g., putting snares) are considered [6]. The algorithms in previous works do not memorize the strategies sampled previously and rangers usually can only patrol a small proportion of zones in a period. Since there can be zones with very small probabilities of being patrolled, it can happen that certain zones are not patrolled for a significant duration of time. Following such a strategy in practice can be exploited by poachers because poachers can change the environment (e.g., set up a big camp in the target zone) when observing that the zone is not protected by the defender and thus increase the damaging impact of their poaching activities. The example in Figure 1 shows that when the targets' values are affected by players' actions, it is necessary for rangers to update their strategies.

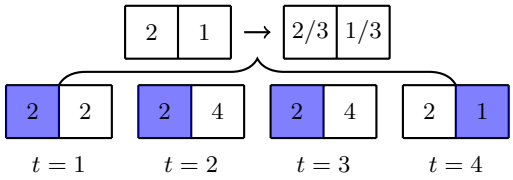


Fig. 1: An illustrative example where the ranger can only patrol one of the two zones in a round. The value of each zone is the attacker's payoff if the attack to this zone is successful. If the attack is intercepted, the attacker's payoff is zero. Suppose that the values of the two zones are 2 (left) and 1 (right), respectively. Therefore, the ranger's optimal patrol strategy is $\langle 2/3, 1/3 \rangle$. For simplicity, we assume the defender knows the zones' values at $t = 0$ in this example. On average, the ranger will patrol the right zone once in three rounds. However, as the daily patrols are randomly sampled, as shown in the figure where blue indicates the zone is patrolled, it is possible that the ranger only patrols the right zone at the forth round. In the first three days, poachers can use two days to build support facilities (e.g., roads) in the right zone (i.e., prepare for the poaching by increasing the zone's value from 1 to 4) and use the third day to poach animals (i.e., collect the reward 4). Though the ranger patrols the right zone at the forth round, he can only destroy all facilities built by the poacher (i.e., reset the zone's value to 1) but does not intercept the poacher.

Therefore, we assume that rangers would update their strategies based on the observations and the history of their actions in this work. Additionally, poachers can execute sophisticated poaching activities, such as building roads to transport poached animals, using support facilities to capture animals. These activities may last for weeks or months and bring much higher payoff to poachers than than the case without these facilities. To address this issue, we consider the attacks with "preparing steps" in this work, which would increase the values of the

areas. The value of an area is the reward for poachers if they successfully attack after preparations. By building support facilities, poachers can obtain a higher reward, i.e., poaching more endangered animals. From the pessimistic perspective, we assume that poachers know the values of all areas and rangers' mixed strategies. Note that we assume the poacher does not know the rangers' pure strategies, i.e., patrol paths, which means rangers and poachers move simultaneously at each round. This is quite reasonable because each round is modeled as a single-shot game. The game is assumed to be zero-sum, where Stackelberg equilibrium, which is the standard solution of previous Stackelberg security game, is equivalent to Nash/minimax equilibrium.

2 Related work

The first line of related works is security games which considers dynamical changes of targets. The work [27] considers the protection of large public event where the targets' values are changed along with the time, which can be fully observed or predicted by the defender. The work [5] considers that targets' positions move along with predefined routes and cannot be changed by any of the players. None of previous works consider the case where the targets can be changed by players' actions, either values or positions, so that the techniques cannot be directly applied to our problem. In this work, we focus on the case that the targets' values are affected by the players' actions.

The second line of related works is security game where the targets' values are static but not known by the defender [16, 17, 2]. They assumed that the defender can sample/query a strategy and observe the attacker's response over many times and learn the optimal defender's strategy to commit. However, in our problem, the targets' values are dynamically influenced by players' actions and the defender cannot observe the attacker's actions, therefore the learning algorithms cannot be applied.

The third line of related works is the partially observable stochastic games (POSG) where one player (the attacker) has perfect information about the course of the game. There are only few works that provide algorithms for solving such games. Examples that provide domain-specific algorithms for different games include patrolling problems [1, 26], or search games [25, 4] that cannot be directly used, as our game model cannot be formulated as a patrolling game or a pursuit-evasion game. Finally, to the best of our knowledge, there is only one domain-independent algorithm that can be used for solving games that correspond to our model, HSVI algorithm for one-sided partially observable stochastic games (denoted PG-HSVI) [12, 11]. We adopt this algorithm and provide significant changes to improve the scalability to solve large games.

3 Model

A defender-sided partially observable stochastic Stackelberg security game (DPOS3G) G is a game between a defender who wants to protect a set of targets $[N]$ using

limited K resources against an attacker over infinite rounds. The game can be represented by a tuple $G = \langle \mathcal{S}, \mathcal{A}_D, \mathcal{A}_A, \mathcal{O}_D, T, R, \gamma \rangle$. $\mathcal{S}$ is the set of states of the game and each state $\mathbf{s} = \langle \mathbf{d}, \mathbf{v} \rangle \in \mathcal{S}$ is a tuple where $\mathbf{d}$ is the defender's action (defined below) in the previous round and $\mathbf{v} = \langle v_i \rangle$ where v_i denotes the value of target i for the attacker, i.e., the payoff obtained by the attacker if being successfully attacked. We note that both $\mathbf{d}$ and the values of targets protected by this action are known by the defender. We say that $\mathbf{s} = \langle \mathbf{d}, \mathbf{v} \rangle$ is a *reachable state* of $\mathbf{d}$, i.e., the state which can be reached at the current round when the defender takes the action $\mathbf{d}$ in the previous round. All reachable states of $\mathbf{d}$ comprise the *reachable state set* $\mathcal{S}_{\mathbf{d}}$. We assume that each of the targets is associated with an initial value v_i^0 and a cap value $\hat{v}_i$ known by both players, which are the minimum and maximum values of the target[4], respectively.

The game is given by an initial action taken by the defender $\mathbf{d}^0$ and the initial state of the game is drawn from a probability distribution over states $b^0 \in \Delta(\mathcal{S}_{\mathbf{d}^0})$, which is also termed as the initial belief, where $\mathcal{S}_{\mathbf{d}^0}$ is the reachable state set $\mathbf{d}^0$ and $\Delta(\cdot)$ denotes the distribution space over a set. Note that from defender's perspective, the game essentially proceeds from a belief over a reachable state set to a belief over another reachable state set, governed by her own action and the attacker's response. $\mathcal{A}_D = \left\{ \mathbf{d} | \sum_{i \in [N]} d_i \leq K, d_i \in \{0, 1\} \right\}$ is the set of actions of the defender where $d_i = 1$ implies target i is protected and $d_i = 0$ otherwise. We assume there are no scheduling constraints on the defender's actions, i.e., constraints for the allocation of resources, and our model can be easily extended to the case with scheduling constraints.

The set $\mathcal{A}_A$ includes three kinds of actions can be taken by the attacker which are i) *preparing action* $a(i, +)$, preparing the attack at target i, which increases the value of target yielding $v_i = f(v_i, a(i, +))$ where $f(\cdot)$ gives the value of the target with the inputs and is upper bounded by $\hat{v}_i$, ii) *collecting action* $a(i, \circ)$, collecting the payoff v_i at target i and iii) *waiting action* $\emptyset$, doing nothing at the current round. Note that the attacker can only obtain payoffs through collecting actions, while preparing actions can increase the potential payoffs obtained in the future. Once the function $f(\cdot)$ is determined, the target can only take a finite number of values, which is denoted by $\mathcal{M}_i$ for simplicity and $M_i = |\mathcal{M}_i|$ is the number of all possible values that target i can take. With a slight abuse of notation, we use a to denote the action taken by the attacker. We assume that the attacker can only execute an action on one target at a round, which can be easily extended to the case where the attacker can take actions on multiple targets at a round.

We assume that the defender's partial observation $\mathbf{o} \in \mathcal{O}_D$ includes the values of targets she protects, i.e., given the states $\mathbf{s} = \langle \mathbf{d}, \mathbf{v} \rangle$ and the actions of both

[4] This assumption is reasonable in wildlife protection because the poacher can obtain a minimal payoff without any support facility and the payoff cannot be arbitrarily increased since it is limited by the natural resources of the area.

players $\langle \mathbf{d}', a \rangle$, the defender's observation $\mathbf{o}$ is

$$o_i = \begin{cases} f(v_i, a(i,+)), & \text{if } d'_i = 1 \ \& \ a = a(i,+); \\ v_i, & \text{if } d'_i = 1 \ \& \ a \neq a(i,+); \\ none, & \text{otherwise.} \end{cases} \tag{1}$$

where $f(v_i, a(i,+))$ implies that the defender will observe the target's value after the successful execution of the preparing action and *none* implies the defender cannot know any information of the target. On the other hand, we assume the attacker knows all targets' values and the defender's action at the previous round, i.e., the attacker can observe the state exactly.

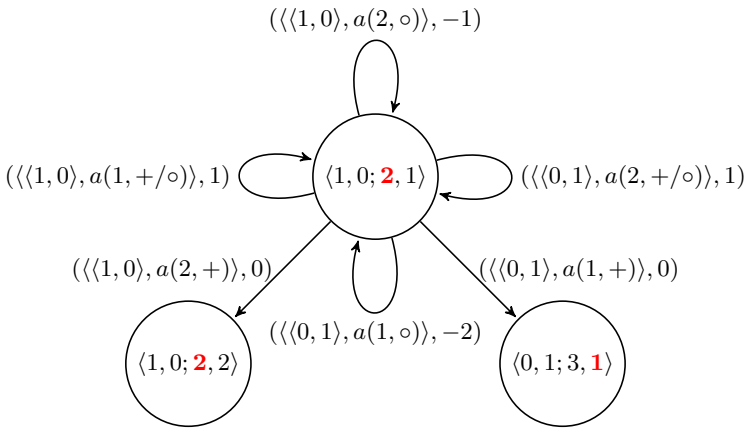


Fig. 2: Part of DPOS3G of example in Figure 1 to illustrate the transitions between states in DPOS3G. The three elements on the arrows are the target being protected by the defender, the action taken by the attacker and the reward to the defender, respectively. Suppose that the game starts with the state $\langle 1,0;\mathbf{2},1\rangle$, i.e., the initial action of the defender is $\langle 1,0\rangle$ and the defender knows the value of target 1, which is highlighted as red. At the first round, if the defend protects the target 1 and the attacker prepares at target 2, the game will transit to the state $\langle 1,0;\mathbf{2},2\rangle$ and the corresponding reward to the defender is 0, i.e., the attacker is not intercepted by the defender and the preparing action will not bring payoff to the attacker. If the defender protects target 1 and the attacker takes collecting action at target 2, the game will stay at $\langle 1,0;\mathbf{2},1\rangle$ and the defender will obtain a reward of -1. If the defender protects target 1 and the attacker takes either preparing action or collecting action at target 1, the game will stay at $\langle 1,0;\mathbf{2},1\rangle$ and the attacker is intercepted by the defender, which brings 1 to the defender.

The transition between states is determined by both players' actions. We assume that i) if target i is protected by the defender, the value of the target is initialized to v_i^0 whatever the action taken by the attacker on this target, i.e., the defender will destroy all attacker's preparations at that target when protecting

it, ii) the preparing action at target i can increase the target's value from v_i to $f(v_i, a(i,+))$ if not being intercepted by the defender and iii) the collecting action at target i will not influence the target's value[5]. Formally, the successive state of $\mathbf{s} = \langle \mathbf{d}, \mathbf{v} \rangle$ with actions $\langle \mathbf{d}', a \rangle$ taken by players is $\mathbf{s}' = \langle \mathbf{d}', \mathbf{v}' \rangle$ where

$$v_i' = \begin{cases} v_i^0, & \text{if } d_i' = 1; \\ f(v_i, a(i,+)), & \text{if } d_i' = 0 \ \& \ a = a(i,+); \\ v_i, & \text{otherwise.} \end{cases} \tag{2}$$

For simplicity, we denote the transition by $T_{\mathbf{s},\mathbf{d}',a}(\mathbf{o}, \mathbf{s}')$ where $T_{\mathbf{s},\mathbf{d}',a}(\mathbf{o}, \mathbf{s}') = 1$ if $\mathbf{o}$ and $\mathbf{s}'$ satisfy Eq.(1) and Eq.(2), respectively, and $T_{\mathbf{s},\mathbf{d}',a}(\mathbf{o}, \mathbf{s}') = 0$ otherwise.

We use $R(\mathbf{s}, \mathbf{d}', a)$ to denote the defender's reward at state s with actions $\langle \mathbf{d}', a \rangle$ taken by players, which is defined as

$$R(\mathbf{s}, \mathbf{d}', a) = \begin{cases} c, & \text{if } a = a(i,+/\circ) \ \& \ d_i' = 1; \\ 0, & \text{if } a = a(i,+) \text{ or } a = \emptyset \ \& \ d_i' = 0 \ ; \\ -v_i, & \text{if } a = a(i,\circ) \ \& \ d_i' = 0. \end{cases} \tag{3}$$

where $c > 0$ is the attacker's penalty when intercepted by the defender, which is the increment of the defender's utility. The attacker's reward is $-R(\mathbf{s}, \mathbf{d}', a)$, i.e., the game is a zero-sum game and we assume that players' rewards are discounted over time with the discount factor $\gamma < 1$.

We assume *perfect recall*, hence both players remember their respective history. A history of the defender is formed by actions he played and observations he received, i.e., $(\mathbf{d}', \mathbf{o})^t$ and a history of the attacker is $(\mathbf{s}, \mathbf{d}', a, \mathbf{o})^t$. Therefore, the spaces of the histories of players are $(\mathcal{A}_D, \mathcal{O}_D)^t$ and $(\mathcal{S}, \mathcal{A}_D, \mathcal{A}_A, \mathcal{O})^t$ for the defender and the attacker, respectively. The strategies σ_D, σ_A of players map each of their histories to a distribution over actions. For any given pair of strategies $\langle \sigma_D, \sigma_A \rangle \in \Sigma = \langle \Sigma_D, \Sigma_A \rangle$, we use $u_D(\sigma_D, \sigma_A)$ to denote the expected utility of the defender when players follow the strategies $\langle \sigma_D, \sigma_A \rangle$, respectively, i.e.,

$$u_D(\sigma_D, \sigma_A) = \sum\nolimits_{t=1}^{\infty} \gamma^t \mathbb{E}[R(\mathbf{s}, \mathbf{d}', \mathbf{a})]. \tag{4}$$

A best response of player $\theta \in \{D, A\}$ to the opponent's strategy $\sigma_{-\theta}$ is the strategy $\sigma_\theta^{BR} \in BR(\sigma_{-\theta})$ where $u_\theta(\sigma_\theta^{BR}, \sigma_{-\theta}) \geq u_\theta(\sigma_\theta', \sigma_{-\theta})$ for any other $\sigma_\theta' \in \Sigma_\theta$. We use the Stackelberg equilibrium as our solution concept. A strategy profile $\sigma = \langle \sigma_D, \sigma_A \rangle$ forms a Stackelberg equilibrium if i) $\sigma_A \in BR(\sigma_D)$ and ii) $U_D(\sigma_D, \sigma_A) \geq U_D(\sigma_D', \sigma_A')$ where $\sigma_A' \in BR(\sigma_D')$. With the zero-sum assumption, the Stackelberg equilibrium can be computed by the minimax formulation

$$\max_{\sigma_D \in \Sigma_D} \min_{\sigma_A \in \Sigma_A} u_D(\sigma_D, \sigma_A). \tag{5}$$

4 Applying PG-HSVI to DPOS3G

We present an overview of applying PG-HSVI proposed in [12] to DPOS3G in this section and describe our RITA algorithm in the next section, where RITA

[5] Alternatively, we can assume that the target's value is reduced to v_i^0 when the attack takes the collecting action at target i, which can also be solved by our algorithm.

adopts PG-HSVI as a subroutine in building a strategy. The PG-HSVI algorithm approximates the optimal value function of the infinite horizon DPOS3G by considering value-functions of the game with a restricted horizon. The value function of a defender's strategy σ_D is defined as $V_{\sigma_D} : \Delta(\mathcal{S}) \to \mathbb{R}$ which means given the initial action $\mathbf{d}^0$ and initial belief b^0, the function returns the expected utility of the defender when following the strategy σ_D and the optimal value function of the game is $V^*(b^0) = \sup_{\sigma_D} V_{\sigma_D}(b^0)$. Each iteration of PG-HSVI assumes players will play their Stackelberg equilibrium strategies with subsequent utilities (defined below) given by value functions of previous iterations.

Value backup. PG-HSVI performs a *Value Backup* operation, which is denoted by H, at the belief b to improve the approximation of the value function, which corresponds to solve a *stage game*, i.e., game with one round, denoted as $[HV](b)$ where V is the approximated value function obtained by previous iterations. The defender's and the attacker's strategies for a stage game are denoted as $\pi_D \in \Delta(\mathcal{A}_D)$ and $\pi_A : \mathcal{S} \to \Delta(\mathcal{A}_A)$, respectively. Note that the attacker's strategy specifies his strategy at each state in $\mathcal{S}$ because we assume that the attacker can observe the state explicitly. The utilities of $[HV](b)$ depend on the immediate reward R and the discounted value of the subsequent game, represented by V. The immediate reward of the defender is

$$R^{imm}_{\pi_D,\pi_A} = \sum\nolimits_{\mathbf{s}\in\mathcal{S}} \sum\nolimits_{\mathbf{d}\in\mathcal{A}_D} \sum\nolimits_{a\in\mathcal{A}_A} b(\mathbf{s})\pi_D(\mathbf{d})\pi_A(\mathbf{s},a)R(\mathbf{s},\mathbf{d},a). \tag{6}$$

As the defender knows her action $\mathbf{d}$ and the observation $\mathbf{o}$, she can derive the belief of the subsequent game by

$$b^{\mathbf{d},\mathbf{o}}_{\pi_A}(\mathbf{s}') = \frac{1}{Pr(\mathbf{o}|\mathbf{d},\pi_A)} \sum\nolimits_{\mathbf{s}\in\mathcal{S}} \sum\nolimits_{a\in\mathcal{A}_A} T_{\mathbf{s},\mathbf{d},a}(\mathbf{o},\mathbf{s}')b(s)\pi_A(a). \tag{7}$$

And the subsequent reward of the game is the expected utility over all action-observation pairs $\langle \mathbf{d}, \mathbf{o}\rangle$ of the defender for a game starting with a belief $b^{\mathbf{d},\mathbf{o}}_{\pi_A}$, i.e.,

$$R^{suq}_{\pi_D,\pi_A}(V) = \sum\nolimits_{\mathbf{d}\in\mathcal{A}_D} \sum\nolimits_{\mathbf{o}\in\mathcal{O}} \pi_D(\mathbf{d})Pr(\mathbf{o}|\mathbf{d},\pi_A)V(b^{\mathbf{d},\mathbf{o}}_{\pi_A}(\mathbf{s}')). \tag{8}$$

And the Stackelberg equilibrium of $[HV](b)$ can be computed by

$$\min_{\pi_A} \max_{\pi_D} \left(R^{imm}_{\pi_D,\pi_A} + \gamma R^{suq}_{\pi_D,\pi_A}(V)\right). \tag{9}$$

Point-based update. PG-HSVI performs point-based updates by sampling the belief space to approximate the value function V^*. The lower bound of the value function $\underline{V}$ is represented by a set of α-vectors Γ and the corresponding upper bound is represented as a lower envelope of a set of points Υ. The algorithm is depicted in Algorithm 1. The lower bound of the value function $\underline{V}$ (and Γ) is initialized by the value of the uniform strategy of the defender and the upper bound $\overline{V}$ (and Υ) is initialized by solving a perfect information counterpart of the game. In every iteration, some belief point are sampled by the forward search heuristic (Line 9 in Algorithm 1) which selects the action-observation pair $\langle \mathbf{d}, \mathbf{o}\rangle$

Algorithm 1: PG-HSVI

1 **Input**: $G = \langle \mathcal{S}, \mathcal{A}_D, \mathcal{A}_A, \mathcal{O}_D, T, R, \gamma \rangle$, initial action $\mathbf{d}^0$, initial belief b^0 and desired precision ϵ
2 **Output**: approximated value function $\hat{V}$
3 Initialize $\hat{V} = \{\overline{V}, \underline{V}\}$;
4 **while** $\overline{V}(b^0) - \underline{V}(b^0) > \epsilon$ **do**
5 `Explore`(b^0, ϵ, 0);
6 **procedure** `Explore`(b, ϵ, t)
7 $\pi_A \leftarrow$ optimal strategy of the attacker in $[H\underline{V}](b)$;
8 $\pi_D \leftarrow$ optimal strategy of the defender in $[H\overline{V}](b)$;
9 $\langle \mathbf{d}, \mathbf{o} \rangle \in \arg\max \left\{ \pi_D(\mathbf{d}) \cdot Pr(\mathbf{o}|\mathbf{d}, \pi_A) \cdot \texttt{excess}(b^{\mathbf{d},\mathbf{o}}_{\pi_A}, t+1) \right\}$;
10 **if** *excess* $(b^{\mathbf{d},\mathbf{o}}_{\pi_A}, t+1) > 0$ **then**
11 `Explore` $(b^{\mathbf{d},\mathbf{o}}_{\pi_A}, \epsilon, t+1)$;
12 $\Gamma \leftarrow \Gamma \cup L\Gamma(b)$;
13 $\Upsilon \leftarrow \Upsilon \cup U\Upsilon(b)$;

such that maximizing $\pi_D(\mathbf{d}) \cdot Pr(\mathbf{o}|\mathbf{d}, \pi_A) \cdot \texttt{excess}(b^{\mathbf{d},\mathbf{o}}_{\pi_A}, t+1)$ where π_D is the defender's strategy computed by $[H\overline{V}](b)$ (Line 8 in Algorithm 1). The `excess` is defined as

$$\texttt{excess}(b, t) = \left(\overline{V}(b) - \underline{V}(b)\right) - \rho(t) \tag{10}$$

where $\rho(t) = \epsilon\gamma^{-t} - \sum_{i=1}^{t} R \cdot \gamma^{-i}$ and R is selected to ensure the termination of the exploration. The forward exploration will terminate if the criteria is satisfied (Line 10 in Algorithm 1). After the termination of the forward exploration, PG-HSVI performs updates of $\underline{V}$ and $\overline{V}$ by adding an α-vector $L\Gamma(b)$ into Γ and a point $U\Upsilon(b)$ into Υ, respectively[6]. For more details, please refer to [12].

5 RITA: Algorithm to Improve Scalability

PG-HSVI can be used for solving DPOS3G, however, the scalability of PG-HSVI is limited in this case due to an exponential number of states and transitions (we show the scalability in Section 6). Table 1 shows the number of states and transitions for the number of targets and resources. For example, when $N = 8$, $K = 2$ and $M_i = 3, \forall i \in [N]$, the game has 20,412 states and more than 9,700,000 transitions. PG-HSVI algorithm is not able to handle such large-scale games due to memory requirements (similarly to the original HSVI algorithm for POMDPs).

To this end, we propose RITA, displayed in Algorithm 2, which builds a **R**educed game by considering key states and the associated transitions to reduce the game size, **I**teratively adding the defender's actions into consideration

[6] We remove the fixing process of Lipschitz continuity of $\overline{V}$ to speed up the algorithm. The experiments show that the algorithm converges efficiently, though without the theoretical guarantee of the termination.

Table 1: Numbers of sates and transitions of DPOS3G. For simplicity, all targets have the same number of possible values, i.e., $M_i = M, \forall i$.

	num. of states	num. of transitions
(N, K)	$C_N^K \cdot M^{N-K}$	$(2N+1) \cdot (C_N^K)^2 \cdot M^{N-K}$

and **T**ransferring the α-vectors in the game solved in the previous iteration to the next iteration to improve the initialization of the lower bound. As discussed below, PG-HSVI is adopted as a subroutine of RITA. The general procedure of RITA is as follows. RITA first builds a reduced game which includes all states in $\mathcal{S}_{\mathbf{d}^0}$ and only key states in $\mathcal{S}_{\mathbf{d}}, \mathbf{d} \neq \mathbf{d}^0$ (Line 3 of Algorithm 2 and more details are in Section 5.1). Next, instead of considering all defender's actions, RITA first solves the game with an initial subset of defender's actions (Line 4 of Algorithm 2) and incrementally adds the defender's actions (Line 14 of Algorithm 2) to build the incremental game (Line 7 of Algorithm 2 and more details are in Section 5.2). RITA is terminated when the increment of the defender's utility of one iteration is less than some threshold (Line 9 of Algorithm 2). RITA is guaranteed to give a lower bound of the optimal defender's utility of the original game, providing an efficient way to evaluate the solution quality as shown in Section 6.

Algorithm 2: RITA

```
1  Input: G, d^0, b^0, ε, the minimum incremental gap η
2  Output: approximated value function V̂
3  G' = buildReducedGame(G, d^0);
4  Initialize actions A_D^d ⊆ A_D for each S_d (Algorithm 3);
5  V = −∞, Γ = ∅;
6  while true do
7      G'' = buildIncrementalGame(G', {A_D^d}, Γ) (Algorithm 4);
8      V̂ ← PG-HSVI(G'', d^0, b^0, ε);
9      if V̲(b^0) − V < η then break ;
10     else
11         V = V̲(b^0);
12         α-VectorTransfer(Γ, G'');
13         for d ∈ A_D do
14             d ← actionToAdd(G''), A_D^d = A_D^d ∪ {d};
```

5.1 Building the reduced game

In this section, we propose a flexible reduction scheme to reduce the number of states and transitions in DPOS3G and the game obtained is named as the

reduced game G' (Line 3 in Algorithm 2). The motivation of this scheme is that the states in $\mathcal{S}_{\mathbf{d}}, \mathbf{d} \neq \mathbf{d}^0$ are of less importance, i.e., bring less utility to the defender, compared with the states in $\mathcal{S}_{\mathbf{d}^0}$ due to the discount factor γ. Therefore, RITA keeps all states in the $\mathcal{S}_{\mathbf{d}^0}$ and only a limited number of key states in $\mathcal{S}_{\mathbf{d}}, \mathbf{d} \neq \mathbf{d}^0$. Then, RITA rebuilds the transitions in the reduced game.

Selecting key states. To ensure that there always is a state for a transition to transit to, RITA has to keep the *worst* states in $\mathcal{S}_{\mathbf{d}}, \mathbf{d} \neq \mathbf{d}^0$ in the reduced game. The worst state in $\mathcal{S}_{\mathbf{d}}$ is defined as $\mathbf{s}^{\#} = \langle \mathbf{d}, \mathbf{v} \rangle \in \mathcal{S}_{\mathbf{d}}, \mathbf{d} \neq \mathbf{d}^0$ such that $v_i = v_i^0$ if $d_i = 1$ and $v_i = \hat{v}_i$ otherwise, i.e., the unprotected targets of $\mathbf{d}$ are with the highest values. Figure 3 presents the most conservative case of the reduced game for the example in Figure 1 where only the worst state is kept in each $\mathcal{S}_{\mathbf{d}}, \mathbf{d} \neq \mathbf{d}^0$. By including the worst state (together with the definition of transitions), it is guaranteed that we obtain a lower bound to the true value of the game. To improve the bound, additional states can be added to the reduced game without any other requirements. We denote the reachable state set in G' as $\mathcal{S}'_{\mathbf{d}}$.

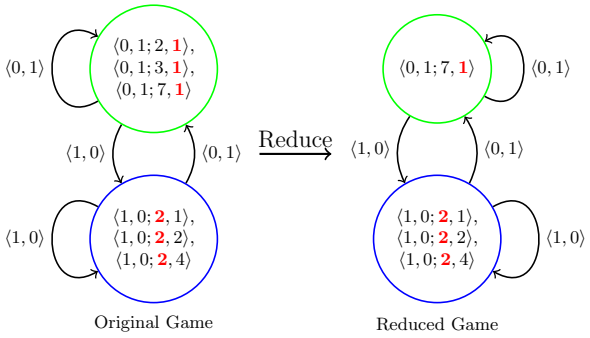


Fig. 3: The reduction of the example in Figure 1 by keeping the worst states in $\mathcal{S}_{\mathbf{d}}, \mathbf{d} \neq \mathbf{d}^0$. The element on the arrow represents the defender's action. The three values of the two targets are $\{2, 3, 7\}$ and $\{1, 2, 4\}$, respectively. Suppose the defender's initial action is $\langle 1, 0 \rangle$ and $\mathcal{S}_{\mathbf{d}^0}$ is depicted by the blue (lower) node where all states are kept in the reduced game. The green (upper) node is the reachable state set of $\langle 0, 1 \rangle$. By building the reduced game, we only keep the worst state, i.e., $\langle 0, 1; 7, \mathbf{1} \rangle$, in the green node. All transitions to states in the green node at the original game will transit to the unique state, i.e., $\langle 0, 1; 7, \mathbf{1} \rangle$ at the reduced game.

Rebuilding the transitions. Reducing the number of states in G' can invalidate transitions of the original game G that lead to these states. Therefore, we need to replace these transitions by defining new transition function. To this end, we introduce a norm to measure the distance from $\mathbf{s}$ to $\mathbf{s}'$:

$$d(\mathbf{s}, \mathbf{s}') = \begin{cases} \sum_{i \in [N]} v'_i - v_i, \text{ if } v'_i \geq v_i, \forall i \in [N], \\ +\infty, \qquad\qquad \text{otherwise.} \end{cases} \tag{11}$$

This norm ensures that if all targets' values in $\mathbf{s}'$ are higher than the values in $\mathbf{s}$, the distance is finite and otherwise the distance is positive infinity. Note that this norm is asymmetric, i.e., the distance from $\mathbf{s}$ to $\mathbf{s}'$ is not equal to the distance from $\mathbf{s}'$ to $\mathbf{s}$. For the rebuilding of the transitions, suppose that $\mathbf{s} \in \mathcal{S}'_{\mathbf{d}}$ is the current state and $\langle \mathbf{d}', a \rangle$ are the actions of both player. If $\mathbf{d}' = \mathbf{d}^0$, then the successive state $\mathbf{s}'$ in the reduced game is the same as the state in the original game, i.e., $T_{\mathbf{s},\mathbf{d}',a}(\mathbf{o}, \mathbf{s}') = 1$. On the other hand, if $\mathbf{d} \neq \mathbf{d}^0$, the successive state will be $\mathbf{s}' \in \arg\min_{\mathbf{s} \in \mathcal{S}'_{\mathbf{d}'}} \{d(\mathbf{s}'', \mathbf{s})\}$ where $\mathbf{s}'' \in \mathcal{S}_{\mathbf{d}'}$ is the successive state in the original game. The basic idea of rebuilding the transitions is that we will use the state in $\mathbf{s} \in \mathcal{S}'_{\mathbf{d}'}$ which is closest to the successive state in the original game as the successive state in the reduced game. Note that as we always keep the worst states in $\mathcal{S}'_{\mathbf{d}}, \mathbf{d} \neq \mathbf{d}^0$, there always exists a states with a finite distance from $\mathbf{s}''$.

For $\mathbf{s} \in \mathcal{S}'_{\mathbf{d}}, \mathbf{d} \in \mathcal{A}_D$, when players take $\langle \mathbf{d}', a \rangle$, the reward for the defender in the reduced game is the same as the reward in the original game, i.e., $R(\mathbf{s}, \mathbf{d}', a)$. After specifying the states, transitions and rewards in the reduced game, RITA applies PG-HSVI to solve the reduced game. Proposition 1 proves that RITA can give a lower bound of the original by solving the reduced game. We note that by keeping more key states in $\mathcal{S}'_{\mathbf{d}}, \mathbf{d} \neq \mathbf{d}^0$, RITA gets a larger reduced game and with a better lower bound to the original game and if all states in $\mathcal{S}_{\mathbf{d}}, \mathbf{d} \neq \mathbf{d}^0$ are kept in $\mathcal{S}'_{\mathbf{d}}$, the reduced is equivalent to the original game. The number of states and transitions of the reduced game is shown in Table 2. Particularly, the number of states and transitions in the most conservative reduced game with $N = 8$, $K = 2$ and $M = 3$ is 756 and 347,004, respectively.

Table 2: Numbers of sates and transitions of reduced games of DPOS3G where Q is the number of states in $\mathcal{S}_{\mathbf{d}}, \mathbf{d} \neq \mathbf{d}_0$.

	num. of states	num. of transitions
(N, K, Q)	$Q \cdot (C_N^K - 1) + M^{N-K}$	$(2N+1)C_N^K \cdot [Q \cdot (C_N^K - 1) + M^{N-K}]$

Proposition 1 *The lower bound obtained by solving the reduced game is also a lower bound of the original game.*

Proof. It can be observed from Eq.(11) that when players take $\langle \mathbf{d}', a \rangle$ at $\mathbf{s} \in \mathcal{S}'_{\mathbf{d}}$ in the reduced game G', the successive state $\mathbf{s}'$ in the reduced game is never better than the successive state $\mathbf{s}''$ in the original game, i.e., $v'_i \geq v''_i, \forall i \in [N]$. Additionally, the rewards in the reduced game are equal to the rewards in the original game. Thus, the optimal defender's utility in the reduced game is never better than the optimal defender's utility in the original game. Therefore, the lower bound obtained by solving the reduced game is a lower bound of the original game.

In essence, by building the reduced game, the states in the reachable state set $\mathcal{S}_{\mathbf{d}}, \mathbf{d} \neq \mathbf{d}^0$ are abstracted into one or more partitions where each partition is

represented by a key state and all transitions to states in partitions will transit to the key states. We emphasize that this reduction scheme is quite flexible which allows the trade-off between the scalability and the solution quality by building different sizes of the reduced games. For the application of our model to the real world, RITA can obtain the defender's strategy for the first round by solving the reduced game because all states in $\mathcal{S}_{\mathbf{d}^0}$ are kept in the reduced game.

5.2 Incrementally adding defender's actions

Although the reduced game can significantly reduce the number of states and transitions, the reduced game is still very large. It is because both defender and the attacker can take all possible actions where the number of the defender's actions is C_N^K, which grows exponentially against the number of targets. A wildly adopted remedy is *strategy generation* [18, 9, 13], which starts by solving a smaller game and incrementally adds actions into consideration.

Therefore, instead of considering all defender's actions, RITA selects a subset of the actions initially (Line 4 in Algorithm 2) and incrementally find the defender's actions (Line 14 in Algorithm 2) to add into the incremental game (Line 7 in Algorithm 2). The reduced game where only a subset of defender's actions for each reachable state set is considered is named as *incremental game*.

Algorithm 3: Defender Initial Action selection

Input: the defender previous action $\mathbf{d}$, the targets' possible values $\mathcal{M}_i, \forall i \in [N]$, the number of initial actions Num
Output: $\mathcal{A}_D^{\mathbf{d}}$
for $i \in [N]$ **do**
  if $d_i = 1$ **then** $\tilde{v}_i = v_i^0$;
  else $\tilde{v}_i = \sum_{v \in \mathcal{M}_i} e^{\beta v} v / \sum_{v \in \mathcal{M}_i} e^{\beta v}$;
$r = \min_i\{\tilde{v}_i\} / \max_i\{\tilde{v}_i\}$;
for $1 : Num$ **do**
  $\mathbf{d} = \arg\max_{\mathbf{d} \in \mathcal{A}_D \setminus \mathcal{A}_D^{\mathbf{d}}} \{\sum_i d_i \cdot \tilde{v}_i\}$;
  $\mathcal{A}_D^{\mathbf{d}} = \mathcal{A}_D^{\mathbf{d}} \cup \{\mathbf{d}\}$;
  $v_i = v_i(1 - d_i + r \cdot d_i), \forall i \in [N]$;

For the defender's action $\mathbf{d}$ and its reachable state set $\mathcal{S}_{\mathbf{d}}$, RITA generates the defender's initial subset of the action using Algorithm 3. As for each reachable state set $\mathcal{S}_{\mathbf{d}}$, if a target is not protected by $\mathbf{d}$, i.e., $d_i = 0$, the target can take M_i values. Therefore the importance values of targets $\tilde{\mathbf{v}}$ are computed by

$$\tilde{v}_i = \begin{cases} v_i^0, & \text{if } d_i = 1; \\ \sum_{v \in \mathcal{M}_i} e^{\beta v} v / \sum_{v \in \mathcal{M}_i} e^{\beta v}, & \text{if } d_i = 0. \end{cases} \tag{12}$$

where β is the parameter such that if $\beta \to 0$, the importance value is the average value of the different values the target can be and if $\beta \to +\infty$, the importance

value is the maximum value of the difference that target can be (Lines 3-5 in Algorithm 3).

Then, RITA selects the defender's initial actions for each reachable state set to ensure that i) every target can be protected (i.e., for every target there exists at least one action that protects this target) and ii) there are more actions to protect important targets. To implement this idea, as displayed in Lines 7-10 in Algorithm 3, RITA iteratively selects the action which protect the targets with the highest importance values. After each selection of the action, the targets being protected will multiply a factor r to decrease the importance values (Line 10 in Algorithm 3), which ensures that no target is covered by more than one action until all targets are covered by some action. The factor r is the ratio of the smallest values and the largest importance values of targets (Line 6 in Algorithm 3). After generating $\mathcal{A}_D^{\mathbf{d}}$ for each $\mathcal{S}_{\mathbf{d}}$, RITA generates the game by only keeping the transitions from each state by actions in $\mathcal{A}_D^{\mathbf{d}}$.

Algorithm 4: Build incremental game

```
1 Input: the reduced game G', {A_D^d}, the α-vector set Γ
2 Output: the incremental game G''
3 for T_{s,d',a}(o, s') ∈ G' do
4     if d' ∈ A_D^d then
5         G'' = G'' ∪ {T_{s,d',a}(o, s')};
6 for LΓ(b) ∈ Γ do
7     G'' = G'' ∪ {LΓ(b)};
```

For each iteration, RITA needs to select the actions to be added into $\mathcal{A}_D^{\mathbf{d}}$ (Line 14 in Algorithm 2). As our game is with infinite horizon, it is difficult to find the optimal defender's action for each reachable state set. Therefore, instead of finding the optimal action, for each reachable state set, RITA adds the defender's action which is the best-response to the attacker's strategy computed in Line 7 in Algorithm 1. Note that if a reachable state set is never visited by PG-HSVI during the forward explorations, RITA will not add the new action into the action set. After adding the action into the set $\mathcal{A}_D^{\mathbf{d}}$ for each $\mathcal{S}_{\mathbf{d}}$, RITA includes the transitions associated with these actions into the incremental game (Lines 3-5 in Algorithm 4). RITA terminates when the increment of the lower bound of the defender's utility of an iteration is less than η (Line 9 in Algorithm 2), which means adding the actions cannot increase the defender's utility efficiently. Proposition 2 states that solving the incremental game gives a lower bound of the reduced game.

Proposition 2 *The lower bound obtained by solving the incremental game is also a lower bound of the reduced game.*

Proof. The incremental game has the same states with the reduced game but only consider a subset of the defender's actions in each reachable state set.

Therefore, the optimal defender's strategy obtained by solving the incremental game can also be implemented in the reduced game, i.e., we can obtain a lower bound by solving the reduced game which is at least as good as the lower bound obtained by solving the incremental game.

5.3 Transferring α-vectors

The convergence of PG-HSVI depends on the initialization of both lower and upper bounds. As RITA always adds actions into consideration during the iterations, the α-vectors computed in the j-th iteration can provide a better initialization of the lower bound for the incremental game in the $(j+1)$-th iteration. The correctness of this transferring is proved in Proposition 3. Specifically, after solving an incremental game G'', RITA extracts the α-vectors into Γ. Transferring α-vectors is displayed in Lines 6-7 in Algorithm 4. It is noteworthy that one cannot easily transfer the representation of upper bounds between incremental games because the upper bound from j-th iteration is not guaranteed to be an upper bound for the game in the $(j+1)$-th iteration.

Proposition 3 *The α-vectors transferred to the incremental game provides a lower bound of the incremental game.*

Proof. As we always add the defender's actions into the reachable state set in the incremental game, the forward explorations implemented in the smaller incremental games can also be implemented in the larger one. Therefore, the transferred α-vectors provide a lower bound of the incremental game.

5.4 Early terminating the forward explorations

To further speed up the algorithm and reduce the space used by the algorithm, RITA adopts early termination of the forward explorations. This is motivated by the fact that as RITA only chooses a subset of the defender's actions at the incremental game, the lower bounds would be worse than the case which considers all defender actions. It may take even more iterations to satisfy the termination criteria $\overline{V}(b^0) - \underline{V}(b^0) > \epsilon$, where most of the iterations only update the upper bound and the lower bound dose not change much. Therefore, RITA will terminate the forward exploration when the number of t is larger than $\hat{T}$. We show that this technique can reduce the process memory of RITA in the experiment section (Section 6).

6 Experimental Evaluation

We now present experimental evaluations of RITA and demonstrate the benefits of key aspects of our novel algorithm. The experiments are performed on randomly generated instances of DPOS3G. The initial value v_i^0 of each target is randomly sampled and re-scaled to the range of $[c, c+3]$, where c is the

penalty to the attacker, which is 2 for all experiments in this section. The range would influence the defender's utility, as well as the convergence time and the space needed due to the forward explorations. The cap value of each target is $\hat{v}_i = (v_i^0)^3$ and the function $f(\cdot)$ is $(v_i^0)^k$ where k is the number of preparation actions worked on this target, i.e., $\mathcal{M}_i = \{v_i^0, (v_i^0)^2, \hat{v}_i\}$. We choose the number of initial actions of the defender in each reachable state set as $N/K + 1$ where N and K are the numbers of targets and resources, respectively. The minimal increment of an iteration of RITA is $\eta = 2$ and with a smaller η, RITA can improve the solution quality with loss of the scalability. All computations are performed on a 64-bit PC with 8.0 GB RAM and a 2.3 GHz CPU and all data points are averaged over 30 instances.

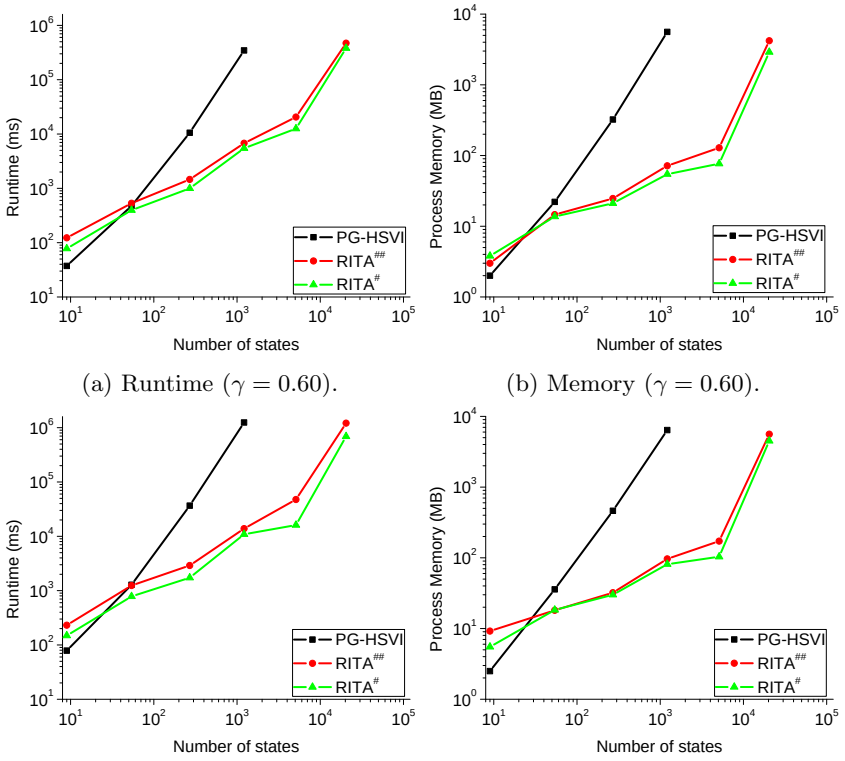
We build the two variants of the reduced game to demonstrate the trade-off between the scalability and the solution quality: i) the most conservative game (Reduced$^{\#}$) and ii) the reduced game (Reduced$^{\#\#}$) which both keeps the worst state and the *second worst* state in $\mathcal{S}'_{\mathbf{d}}, \mathbf{d} \neq \mathbf{d}^0$. The second worst state is defined as $\mathbf{s}^{\#\#} = \langle \mathbf{d}, \mathbf{v} \rangle \in \mathcal{S}_{\mathbf{d}}, \mathbf{d} \neq \mathbf{d}^0$ such that $v_i = v'_i$ if $d_i = 1$ and $v_i = \hat{v}_i$ otherwise, where $f(v'_i, a(i, +)) = \hat{v}_i$, i.e., the unprotected target by $\mathbf{d}$ is with the second highest value of the target.

The six variants of algorithms are tested are 1) PG-HSVI for original game (PG-HSVI), 2) PG-HSVI for the reduced game with both the worst and the second worst states in $\mathcal{S}'_{\mathbf{d}}, \mathbf{d} \neq \mathbf{d}^0$ (PG-HSVI+Reduced$^{\#\#}$), 3) PG-HSVI for the reduced game with the worst states in $\mathcal{S}'_{\mathbf{d}}, \mathbf{d} \neq \mathbf{d}^0$ (PG-HSVI+Reduced$^{\#}$), 4) iteratively solving the incremental games for the original game (Iterative+Original), 5) iteratively solving the incremental games for the reduced game with both the worst and the second worst states in $\mathcal{S}'_{\mathbf{d}}, \mathbf{d} \neq \mathbf{d}^0$ (RITA$^{\#\#}$) and 6) iteratively solving the incremental games for the reduced game with the worst states in $\mathcal{S}'_{\mathbf{d}}, \mathbf{d} \neq \mathbf{d}^0$ (RITA$^{\#}$). The six variants illustrate the influence of different techniques, i.e., building reduced games and incrementally adding defender's actions, on the scalability and the solution quality.

6.1 Scalability

We first investigate the scalability of PG-HSVI and the two variants of RITA. The results about the runtime and the maximum process memory are displayed in Figure 4 with different values of γ. The results show that RITA can be significantly faster than PG-HSVI, especially when the number of states are large. Specifically, with a cap of 1800s, RITA can approximate the game with more than 20000 states and PG-HSVI can only solve the game with less than 2000 states. It can be observed that when γ is larger, both PG-HSVI and RITA need more time and space to solve the game because it may take more forward explorations to meet the termination criteria. Additionally, RITA$^{\#\#}$ takes more runtime and process memory than RITA$^{\#}$ because RITA$^{\#\#}$ has more states and transitions. Another observation is that when the game is small, RITA is not necessarily faster than PG-HSVI because it may take several iterations for RITA before termination, which makes RITA take more time than PG-HSVI.

This is consistent with the performance of strategy-generation algorithms (e.g., double oracle).



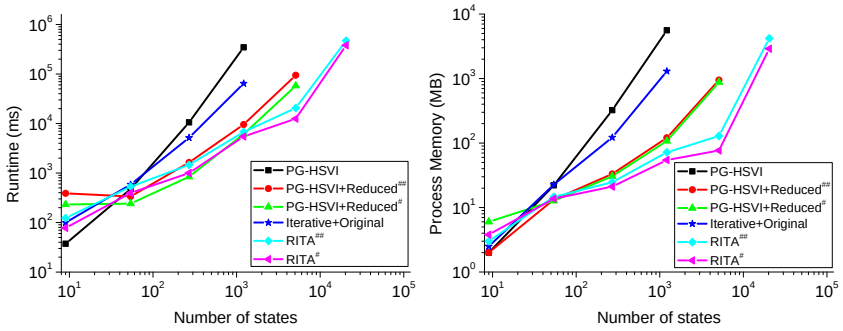
(a) Runtime ($\gamma = 0.60$). (b) Memory ($\gamma = 0.60$).

(c) Runtime ($\gamma = 0.75$). (d) Memory ($\gamma = 0.75$).

Fig. 4: The comparison of runtime and the space for PG-HSVI and RITA. Both axes are plotted in log scale.

We then investigate the efficiency of the two main components of RITA, the reduced game and the iterative method. The results are shown in Figure 5 with $\gamma = 0.6$ for all six variants. The results indicate that building reduced game is more efficient to speed up the algorithm than the iterative method, as well as to reduce the process memory. Because by building reduced games, we ignore most transitions even before solving the game rather than by solving a smaller game and then incrementally increasing the size.

We then investigate the two techniques used in RITA, i) transferring α-vectors and ii) early termination. We test the cases which are i) without transferring α-vectors and without early termination, ii) without transferring α-vectors and with early termination and iii) with transferring α-vectors and with early termination. The results are displayed in Table 3 where both methods can re-



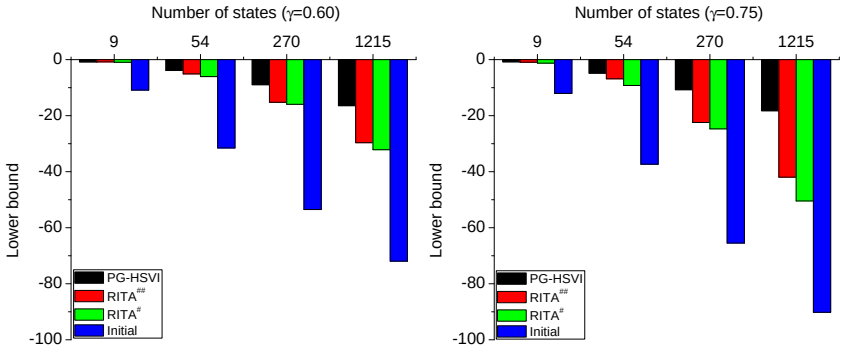
(a) Runtime. (b) Process memory.

Fig. 5: The comparison of runtime and space for different variants with $\gamma = 0.6$.

duce the runtime of the algorithms and the early termination can also reduce the process memory by avoiding unnecessary forward explorations.

Table 3: Comparisons of runtime and process memory for RITA$^{\#}$.

$\lvert\mathcal{S}\rvert = 20,412, \gamma = 0.75$	runtime	memory
Without trans., without early term.	1783.59s	9.9G
Without trans, with early term.	1062.64s	5.6G
With trans., with early term.	829.56s	5.6G



(a) $\gamma = 0.60$. (b) $\gamma = 0.75$.

Fig. 6: Solution quality where y-axis is the lower bounds of the defender's utility.

6.2 Solution quality

Next we investigate the solution quality of the different methods. We use the initial lower bound of PG-HSVI, which is the defender's utility for the uniform strategy as the baseline and compare the lower bounds returned by PG-HSVI, RITA$^{\#\#}$ and RITA$^{\#}$. Note that PG-HSVI computes the optimal defender's utility. The results are displayed in Figure 6 with $\gamma = 0.6$ and 0.75, respectively.

The results illustrate RITA's ability to trade-off between the solution quality and the scalability. With dramatically improving the scalability, both variants of RITA lose the solution quality compared with PG-HSVI. However, we show that with increasing the runtime and the process memory, RITA$^{\#\#}$ achieves a better solution quality than RITA$^{\#}$; RITA provides the flexibility to maintain more states in the reduced game if the solution quality is more important. The advantage of RITA$^{\#\#}$ over RITA$^{\#}$ is increased when the number of states in the game increases and the value of γ increases. Note that even RITA$^{\#}$ is still far better than the defender's utility of the uniform strategy. Another observation is that when γ is small, RITA can give a better approximation to the optimal defender's utility because that the states in $\mathcal{S}_{\mathbf{d}}, \mathbf{d} \neq \mathbf{d}^0$ are of less importance.

7 Conclusion

In this work, we propose a novel defender-sided partially observable Stochastic Stackelberg security game (DPOS3G) where the targets' values are affected by players' actions and the defender can only partially observe the game. To solve the game, we propose RITA which is based on PG-HSVI and with three key novelties: (a) building a reduced game with only key states; (b) incrementally adding defender's actions to further reduce the number of transitions of the game; (c) providing novel heuristics for lower bound initialization. Finally, we do extensive experimental evaluation of the algorithms and the results shows that RITA significantly outperform the baseline PG-HSVI algorithm on scalability and while allowing for trade off in scalability and solution quality.

8 Acknowledgements

This work was supported by Microsoft AI for Earth, NSF grant CCF-1522054 and the Czech Science Foundation (no. 19-24384Y).

References

1. Basilico, N., Gatti, N., Amigoni, F.: Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In: AAMAS. pp. 57–64 (2009)
2. Blum, A., Haghtalab, N., Procaccia, A.D.: Learning optimal commitment to overcome insecurity. In: NIPS. pp. 1826–1834 (2014)
3. Bucarey, V., Casorrán, C., Figueroa, Ó., Rosas, K., Navarrete, H., Ordóñez, F.: Building real stackelberg security games for border patrols. In: GameSec. pp. 193–212 (2017)
4. Chung, T.H., Hollinger, G.A., Isler, V.: Search and pursuit-evasion in mobile robotics. Autonomous Robots **31**(4), 299–316 (2011)
5. Fang, F., Jiang, A.X., Tambe, M.: Protecting moving targets with multiple mobile resources. JAIR **48**, 583–634 (2013)
6. Fang, F., Nguyen, T.H., Pickles, R., Lam, W.Y., Clements, G.R., An, B., Singh, A., Tambe, M., Lemieux, A., et al.: Deploying PAWS: Field optimization of the protection assistant for wildlife security. In: AAAI. pp. 3966–3973 (2016)

7. Gan, J., An, B., Vorobeychik, Y., Gauch, B.: Security games on a plane. In: AAAI. pp. 530–536 (2017)
8. Gan, J., Elkind, E., Wooldridge, M.: Stackelberg security games with multiple uncoordinated defenders. In: AAMAS. pp. 703–711 (2018)
9. Halvorson, E., Conitzer, V., Parr, R.: Multi-step multi-sensor hider-seeker games. In: IJCAI. pp. 159–166 (2009)
10. Haskell, W.B., Kar, D., Fang, F., Tambe, M., Cheung, S., Denicola, E.: Robust protection of fisheries with compass. In: AAAI. pp. 2978–2983 (2014)
11. Horák, K., Bošanskỳ, B.: A point-based approximate algorithm for one-sided partially observable pursuit-evasion games. In: GameSec. pp. 435–454 (2016)
12. Horák, K., Bosanskỳ, B., Pechoucek, M.: Heuristic search value iteration for one-sided partially observable stochastic games. In: AAAI. pp. 558–564 (2017)
13. Jain, M., Korzhyk, D., Vaněk, O., Conitzer, V., Pěchouček, M., Tambe, M.: A double oracle algorithm for zero-sum security games on graphs. In: AAMAS. pp. 327–334 (2011)
14. Johnson, M.P., Fang, F., Tambe, M.: Patrol strategies to maximize pristine forest area. In: AAAI. pp. 295–301 (2012)
15. Kar, D., Ford, B., Gholami, S., Fang, F., Plumptre, A., Tambe, M., Driciru, M., Wanyama, F., Rwetsiba, A., Nsubaga, M., et al.: Cloudy with a chance of poaching: Adversary behavior modeling and forecasting with real-world poaching data. In: AAMAS. pp. 159–167 (2017)
16. Letchford, J., Conitzer, V., Munagala, K.: Learning and approximating the optimal strategy to commit to. In: International Symposium on Algorithmic Game Theory. pp. 250–262 (2009)
17. Marecki, J., Tesauro, G., Segal, R.: Playing repeated stackelberg games with unknown opponents. In: AAMAS. pp. 821–828 (2012)
18. McMahan, H.B., Gordon, G.J., Blum, A.: Planning in the presence of cost functions controlled by an adversary. In: ICML. pp. 536–543 (2003)
19. Paruchuri, P., Pearce, J.P., Marecki, J., Tambe, M., Ordonez, F., Kraus, S.: Playing games for security: An efficient exact algorithm for solving bayesian stackelberg games. In: AAMAS. pp. 895–902 (2008)
20. Pita, J., Jain, M., Ordónez, F., Portway, C., Tambe, M., Western, C., Paruchuri, P., Kraus, S.: Using game theory for los angeles airport security. AI Magazine **30**(1), 43 (2009)
21. Shieh, E., An, B., Yang, R., Tambe, M., Baldwin, C., DiRenzo, J., Maule, B., Meyer, G.: PROTECT: A deployed game theoretic system to protect the ports of the united states. In: AAMAS. pp. 13–20 (2012)
22. Tambe, M.: Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned. Cambridge University Press (2011)
23. Tsai, J., Yin, Z., Kwak, J.y., Kempe, D., Kiekintveld, C., Tambe, M.: Urban security: Game-theoretic resource allocation in networked physical domains. In: AAAI. pp. 881–886 (2010)
24. Varakantham, P., Lau, H.C., Yuan, Z.: Scalable randomized patrolling for securing rapid transit networks. In: IAAI. pp. 1563–1568 (2013)
25. Vidal, R., Shakernia, O., Kim, H.J., Shim, D.H., Sastry, S.: Probabilistic pursuit-evasion games: theory, implementation, and experimental evaluation. IEEE Transactions on Robotics and Automation **18**(5), 662–669 (2002)
26. Vorobeychik, Y., An, B., Tambe, M., Singh, S.P.: Computing solutions in infinite-horizon discounted adversarial patrolling games. In: ICAPS. pp. 314–322 (2014)
27. Yin, Y., An, B., Jain, M.: Game-theoretic resource allocation for protecting large public events. In: AAAI. pp. 826–834 (2014)