Approximation Approaches for Solving Security Games with Surveillance Cost: A Preliminary Study

Qingyu Guo
Interdisciplinary Graduate School
Nanyang Technological University
Singapore 639798
qguo005@e.ntu.edu.sg

Bo An
School of Computer Eng.
Nanyang Technological University
Singapore 639798
boan@ntu.edu.sg

Andrey Kolobov
Microsoft Research
Redmond, WA, USA
akolobov@microsoft.com

ABSTRACT

Security game models have been deployed to allocate limited security resources for critical infrastructure protection. Much work on this topic assumes that attackers have perfect knowledge of the defender's randomized strategy. However, this assumption is not realistic, considering surveillance cost, since attackers may only have partial knowledge of the defender's strategies, and may dynamically decide whether to attack or conduct more surveillance. Recently, a model called OPTS (OPtimal sTopping Security games) considered surveillance cost and formulated the attacker's optimal decision making problem as a finite state space MDP (Markov Decision Process). However, the known exact algorithms for this MDP cannot scale up. In this paper, we extend several approximation approaches based on the MCTS (Monte-Carlo Tree Search) method and RTDP (Real-Time Dynamic Programming) to MDPs of the OPTS model. We also propose an iterative deepening based approximation algorithm. Experimental results show that these approaches can solve much larger security games with surveillance cost.

General Terms

Algorithms, Performance, Experimentation

Keywords

Security games, Markov Decision Process, Approximation approaches

1. INTRODUCTION

Stackelberg security game models have been successfully deployed for protecting critical infrastructure including LAX Airport, US Coast Guard, and the Federal Air Marshals Service [2, 4, 6, 10, 16, 19, 20]. Most existing work on security games, including deployed applications, assumes that the attacker has perfect knowledge of the defender's strategy or can learn the defender's strategy after conducting a fixed period of surveillance. Although these assumptions are a useful approximation, they are obviously simplistic. In reality, the attacker may have more limited observation capabilities, considering the costs of surveillance and delay of attacks. Attackers may also wish to reduce the number of observations because of the risk of being detected by security forces while conducting surveillance [20]. Therefore, it is essential to consider the attacker's dynamic decision making while conducting limited surveillance.

An *et al.* [1, 3] propose a natural model of limited surveillance OPTS (OPtimal sTopping Security games) in which the attacker dynamically determines whether to make more observations or to attack his best target immediately. The attacker's optimal stopping decision after each observation takes into account both his updated belief based on observed defender actions and surveillance cost. Such an optimal stopping model for limited surveillance does not assume the knowledge about the defender's strategy or the a priori number of observations the attacker will make. The attacker's decision problem is formulated as a discrete state space MDP (Markov Decision Process) and An *et al.*(2013) provide an exact algorithm called *Backward Induction with Forward Search (BI-FS)* for it.

However, *BI-FS* cannot scale up to large games for the following reasons: 1) The size of the state space of MDP in the OPTS model grows exponentially with the number of observations made by the attacker; 2) The size of the defender strategy space also grows exponentially with the problem size. Hence, solving the MDP in the OPTS model has the super-exponential computational complexity with respect to the problem size.

In order to handle the computational complexity, we extend several approximation approaches based on MCTS (Monte-Carlo Tree Search) and RTDP (Real-Time Dynamic Programming) [5], and improve them to be suitable for the MDP in the OPTS model. We also propose an *iterative deepening* based approximation algorithm called *Iterative Deepening Backward Induction (ID-BI)*. Experimental evaluation shows that the improved MC-VOI (Monte-Carlo algorithm for computing Value Of Information) algorithm [14] and *ID-BI* algorithm, significantly improve the scalability of the algorithms to solve the MDP in the OPTS model.

2. THE OPTS MODEL

We first review the OPTS model for solving security games with surveillance cost [1]. A Stackelberg security game has two players: a defender who uses m identical resources to protect a set of targets $T = \{1, 2, ..., n\}$ $(m < n)$, and an attacker who selects a single target to attack. The defender has N pure strategies $\mathcal{A}$, each associated with a coverage

Appears in: *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Bordini, Elkind, Weiss, Yolum (eds.), May, 4–8, 2015, Istanbul, Turkey.*

vector representing which m targets are covered. $A_i = 1$ if target i is covered in strategy $A \in \mathcal{A}$, and $A_i = 0$ otherwise. Each target i is covered by at least one pure strategy. The defender can choose a randomized strategy $\mathbf{x}$, with $x_A \geq 0$ being the probability of playing a pure strategy A. After the defender commits her randomized strategy, the attacker chooses a target to attack. If the attacker attacks target i, there are two cases. If target i is covered, the defender receives a reward R_i^d and the attacker receives a penalty P_i^a. Otherwise, the payoffs for the defender and attacker are P_i^d and R_i^a, respectively. We assume that $R_i^d \geq P_i^d$ and $R_i^a \geq P_i^a$ in order to model that the defender would always prefer the attack to fail, while the attacker would prefer it to succeed.

In an OPTS model the attacker has a prior belief about the defender's strategy, updates this belief upon observing actual defense realizations, and dynamically decides whether to stop surveillance after each observation based on this posterior belief. Suppose that the attacker makes a sequence of observations, such a sequence of observations can be compactly represented by an observation vector $\mathbf{o} = \langle o_A \rangle$ in which o_A is the number of times pure strategy A was observed. Let $\Delta(\mathbf{o}) = \sum_{A\in\mathcal{A}} o_A$ be the length of observation vector $\mathbf{o}$. Additionally, let $\mathbf{o} = \langle o_A = 0 \rangle$ be an "empty" observation vector corresponding to the initial attacker state prior to surveillance activity. The attacker starts the decision problem with a prior belief, which is also known to the defender. The attacker's prior belief is represented as a Dirichlet distribution

$$f(\mathbf{x}) = \frac{\Gamma\left(\sum_{A\in\mathcal{A}} \alpha_A + |\mathcal{A}|\right)}{\prod_{A\in\mathcal{A}} \Gamma(\alpha_A + 1)} \prod_{A\in\mathcal{A}} (x_A)^{\alpha_A} \tag{1}$$

with a parameter vector $\alpha = \langle \alpha_A \rangle$ with $\alpha_A > -1$ for all $A \in \mathcal{A}$. Having observed $\mathbf{o}$ with an observation length τ, the attacker's posterior belief is

$$f(\mathbf{x}|\mathbf{o}) = \frac{\Gamma\left(\sum_{A\in\mathcal{A}} \alpha_A + |\mathcal{A} + \tau|\right)}{\prod_{A\in\mathcal{A}} \Gamma(\alpha_A + o_A + 1)} \prod_{A\in\mathcal{A}} (x_A)^{\alpha_A + o_A} \tag{2}$$

and the attacker believes that the probability of choosing a pure strategy A is

$$Pr(A|\mathbf{o}) = \frac{\alpha_A + o_A + 1}{\sum_{A'\in\mathcal{A}} \alpha_{A'} + |\mathcal{A}| + \tau} \tag{3}$$

The marginal coverage of target i according to the posterior belief is $c_i^{\mathbf{o}} = \sum_{A\in\mathcal{A}} A_i Pr(A|\mathbf{o})$, denoting the probability with which the attacker believes that the target i is protected by some defender resource. If the attacker chooses to attack target i, his expected utility is $U^a(\mathbf{o}) = c_i^{\mathbf{o}}(P_i^a - R_i^a) + R_i^a$.

In the OPTS model, the attacker's decision problem can be formulated as an MDP in which belief states are the observation vectors $\mathbf{o}$. An observation vector with observation length τ is connected to only $|\mathcal{A}|$ observation vectors with length $\tau + 1$. If the attacker attacks his best target $\psi(\mathbf{o})$ in a state[1] with observation vector $\mathbf{o}$, he will gain an immediate utility

$$W(\mathbf{o}) = U^a(\mathbf{o}) - \lambda \cdot \Delta(\mathbf{o}). \tag{4}$$

Define a value function $V(\mathbf{o})$ for each observation vector $\mathbf{o}$, which represents the attacker's expected utility when his observation vector is $\mathbf{o}$ and he follows the optimal policy afterwards. In every state, the attacker can either attack the best target $\psi(\mathbf{o})$ and gain a utility $W(\mathbf{o})$ or make another observation, reaching state $\mathbf{o}' = \mathbf{o} \cup \{A\}$ with probability $Pr(A|\mathbf{o})$. The value function is

$$V(\mathbf{o}) = \max\{W(\mathbf{o}), \sum_{A\in\mathcal{A}} Pr(A|\mathbf{o}) V(\mathbf{o} \cup \{A\})\} \tag{5}$$

It has been proved [1] that there exists a constant horizon $\tau_{max} = \frac{M}{\lambda} - \sum_{A\in\mathcal{A}} \alpha_A - |\mathcal{A}| - 1$, where $M = \max_{i\in T} R_i^a - P_i^a$, such that $V(\mathbf{o}) = W(\mathbf{o})$, i.e., attacker attacks immediately, when $\Delta(\mathbf{o}) > \tau_{max}$. Thus, the infinite horizon MDP for the attacker's decision-making problem is equivalent to an MDP with a finite state space, and can be solved by an exact algorithm *Backward Induction with Forward Search (BI-FS)*. After solving the attacker's decision problem, the set of observation vectors can be obtained, for each of which the attacker will choose to attack its best target. Based on this set of observation vectors, the OPTS model provides an exact (but nonconvex) mathematical program DF-OPT for computing the defender's optimal strategy $\mathbf{x}$ [1].

However, as every observation vector is connected to $|\mathcal{A}|$ observation vectors, the size of the reachable belief space grows exponentially with τ_{max}. Besides, the number of pure strategies $|\mathcal{A}|$ may also grow nearly exponentially with the problem size, i.e., the number of targets and resources. Since the *BI-FS* algorithm can only scale up to security game instances with 5 targets, 1 resource and a observation cost of 0.2, we provide several approximation approaches based on the MCTS and RTDP in order to handle the computational complexity.

3. APPROXIMATION APPROACHES BASED ON MCTS

MCTS is a popular approach for making near-optimal decisions in many problems such as move planning in combinatorial games. It combines the generality of random simulation with the precision of tree search. The process of MCTS can be broken down into 4 basic steps: *selection*, *expansion*, *simulation* and *backpropagation* [8]. Figure 1 shows an outline of a basic MCTS approach. The selection procedure selects an action at a state of the tree according to the statistics stored, in a way that balances between *exploitation* and *exploration*. Once the algorithm reaches a state not found in the tree, the state is added to the search tree, after which the simulation procedure gets called where actions are randomly selected until the game ended, i.e., the attacker chooses to attack at some state. After reaching the end of the simulated game, each tree state that was traversed during that game is updated by a backpropagation procedure.

We extend three approximation approaches based on MCTS: UCT (UCB applied to Trees) [15], Improved MC-VOI (Monte-Carlo algorithm for computing Value Of Information) [14], and BRUEic [11, 12].

3.1 UCT

UCT (UCB applied to Trees) is an advanced Monte-Carlo tree search algorithm using the UCB1 action selection algorithm, where UCB stands for "Upper Confidence Bound" [15]. In the UCB1 selection procedure, the action with highest upper confidence bound at each state $\mathbf{o}$ is selected. The upper confidence bound of action a can be represented by

[1]For the sake of convenience, we will abuse the terminology throughout the paper and when we talk about *states*, we actually mean *belief states*.

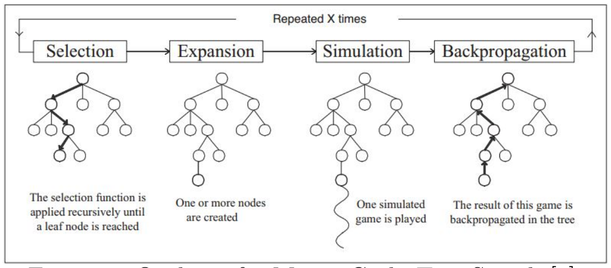

Repeated X times
Selection
Expansion
Simulation
Backpropagation
The selection function is applied recursively until a leaf node is reached
One or more nodes are created
One simulated game is played
The result of this game is backpropagated in the tree

Figure 1: Outline of a Monte-Carlo Tree Search [9].

$U(\mathbf{o},a)=\overline{R}(\mathbf{o},a)+C\sqrt{\frac{2lnN(\mathbf{o})}{N(\mathbf{o},a)}}$, where $\overline{R}(\mathbf{o},a)$ is the average long-term reward empirically observed so far by choosing action a in state $\mathbf{o}$, $N(\mathbf{o})$ is the number of visits to state $\mathbf{o}$ made by the tree search procedure, $N(\mathbf{o},a)$ is the number of times that a is selected at state $\mathbf{o}$ during this procedure and C is a tunable parameter. The value of C determines how eager UCT is to *explore* under-sampled state-action pairs instead of *exploiting* those that currently look promising based on their empirical long-term reward $\overline{R}(o,a)$. Since in the OPTS model, there are only two actions: *attacking* and *observing*. Once the *attacking* action is selected, the game ended and an immediate reward $W(\mathbf{o})$ is received.

After a termination criterion, such as a fixed running time, is satisfied, the final attacking policy is obtained in which the action with the highest upper confidence bound is selected at each state. It has been proved that the attacking policy obtained by the UCT algorithm converges to the optimal one when the running time is large enough [15].

3.2 Improved MC-VOI

Without selection and simulation procedures, the MC-VOI mainly consists of two steps: *SampleExecutionPath* and *Evaluate* [14]. The first procedure *SampleExecutionPath*, samples an execution path, a sequence of states from the initial state to a terminal state $\mathbf{o}$ with $\Delta(\mathbf{o})=\tau_{max}$. Since the value of the terminal state is exactly the same as the immediate attacking utility, for each execution path, the second procedure *Evaluate* updates the value $V(\mathbf{o})$ of each state on the execution path, according to the recursive value function in Eq. (5).

It can be regarded as always selecting the "observing" action at each state except the terminal states in the MC-VOI. By always sampling a sequence of states from the initial state to a terminal state, the MC-VOI builds a partial search tree based on the sampled execution paths, and computes the values of states on this partial search tree, according to the recursive value function in Eq. (5).

Compared with the original MC-VOI algorithm, we improve the value updating policy in *Evaluate* procedure and the state selection policy in *SampleExecutionPath* procedure. Instead of selecting the child state $\mathbf{o}'=\mathbf{o}\cup\{A\}$ based on probability $Pr(A|\mathbf{o})$ as in the original *SampleExecutionPath* procedure, we implement the UCB1 selection method into the MC-VOI, and represent the upper bound of confidence for child state $\mathbf{o}'=\mathbf{o}\cup\{A\}$ by $\overline{V}(\mathbf{o}')+C\sqrt{\frac{2lnN}{N(\mathbf{o}')}}$, where $N(\mathbf{o}')$ is the number of times state $\mathbf{o}'$ has been sampled, $\overline{V}(\mathbf{o}')$ is the value of $\mathbf{o}'$ updated based on sampled partial search tree and $N=\sum_{A\in\mathcal{A}}N(\mathbf{o}\cup\{A\})$. The estimated value of state $\mathbf{o}$ in the original MC-VOI algorithm is updated as follows: $\overline{V}(\mathbf{o})=\frac{\sum_{A\in\mathcal{A}}N(\mathbf{o}\cup\{A\})\overline{V}(\mathbf{o}\cup\{A\})}{\sum_{A\in\mathcal{A}}N(\mathbf{o}\cup\{A\})}$. In order to improve the value update, we use a *partial bellman backup* policy to update the value of $\mathbf{o}$, i.e., $\overline{V}(\mathbf{o})=\sum_{A\in\mathcal{A}}Pr(A|\mathbf{o})\cdot\overline{V}(\mathbf{o}\cup\{A\})$.

Algorithm 1: General RTDP algorithm [7]

```
Initialize the heuristic function for all states
RTDP($s$ : $state$)
repeat
    /*Pick best action and update*/
    $a = s.GREEDYACTION()$
    $s.UPDATE()$

    /*Stochastically simulate next state*/
    $s = s.PICKNEXTSTATE(a)$
until termination condition is satisfied;
```

3.3 BRUEic

BRUEic is based on the BRUE algorithm, which is a recently introduced MCTS algorithm guaranteeing an exponential rate of simple regret reduction [11, 12]. In the BRUE algorithm, the sampling path is divided into two parts: the exploration part near the initial state and estimation part near the leaf state. The two parts are connected at a switch point and for each sampling path, only the state at the switch point is updated. This locality of update is required to satisfy the exponential-rate reduction of simple regret over time. BRUEic improves BRUE's short term effectiveness by applying a selective tree expansion policy. For each sampled state $\mathbf{o}$, it treats it as a *forecaster* of two types: T_{OUT} and T_{IN}; For the T_{OUT} forecaster, the value is estimated by $\mathbb{E}_{\pi\sim\mathcal{U}[\Pi]}V^{\pi}(\mathbf{o})$, which is the expected value of a policy sampled from Π, the set of possible policies, uniformly at random, while for the T_{IN} forecaster, the value is estimated based on the recursive function in Eq. (5). Unlike the simulation approach in the UCT algorithm, a leaf state in BRUEic remains a T_{OUT} forecaster for a while until it satisfies $Var[\mathbb{E}[V^{\pi}(\mathbf{o})|\pi]]>\mathbb{E}[Var[V^{\pi}(\mathbf{o})|\pi]]$, at which point it is converted to a T_{IN} forecaster and is no longer treated as a leaf state.

4. APPROXIMATION APPROACHES BASED ON RTDP

RTDP is a well known MDP heuristic search algorithm [5]. Algorithm 1 shows a general framework of RTDP algorithm. The heuristic function for each state is initialized first, then an RTDP trial begins at the initial state of the MDP and explores forward, choosing actions greedily and choosing outcomes stochastically according to their probability. The values of the states on RTDP trail are updated so that the heuristic function converges to the optimal value ultimately [5]. We extend three algorithms based on RTDP: LRTDP (Labeled RTDP) [7], BRTDP (Bounded RTDP) [17] and VPI-RTDP (Value of Perfect Information RTDP) [18].

4.1 LRTDP

Our extension of LRTDP, an RTDP variant with convergence detection, keeps an admissible (upper bound) and monotonic heuristic reward function $h^u(\mathbf{o})$ for the *observing* action at each state $\mathbf{o}$ where $h^u(\mathbf{o})=R^a_{max}-\lambda\cdot\Delta(\mathbf{o})$ and

$R^a_{max} = \max_{i \in T} R^a_i$. During the updating procedure of state $\mathbf{o}$, the reward function $h^u(\mathbf{o})$ and the estimated value $V'(\mathbf{o})$ of the state $\mathbf{o}$ on the sampling path is updated according to $h^u(\mathbf{o}) = -\lambda + \sum_{A \in \mathcal{A}} Pr(A|\mathbf{o})V'(\mathbf{o} \cup \{A\})$ and $V'(\mathbf{o}) = \max\{h^u(\mathbf{o}), W(\mathbf{o})\}$.

LRTDP keeps a label SOLVED or NOTSOLVED for each state $\mathbf{o}$, and the termination condition of the algorithm is that the initial state of MDP is labelled as SOLVED. There are three situations where a state $\mathbf{o}$ will be labeled as SOLVED: 1) $h^u(\mathbf{o}) < W(\mathbf{o})$ and the greedy action is *attacking*. Since $h^u(\mathbf{o})$ is the upper bound of reward for *observing* action it decreases monotonically by LRTDP's properties [7], the optimal reward for *observing* action is also smaller than $W(\mathbf{o})$; 2) $\Delta(\mathbf{o}) = \tau_{max}$ and the estimated value is set as the immediate reward of the *attacking* action, i.e., $V'(\mathbf{o}) = W(\mathbf{o})$; 3) All the child states $\mathbf{o} \cup \{A\}$ of $\mathbf{o}$ are labeled as SOLVED.

4.2 BRTDP

Unlike LRTDP, which only keeps an upper bound heuristic reward function $h^u(\mathbf{o})$ for *observing* action, BRTDP algorithm also holds a lower bound $h^l(\mathbf{o})$ for the *observing* action which is $h^l(\mathbf{o}) = R^a_{min} - \lambda \cdot (\Delta(\mathbf{o}) + 1)$, where $R^a_{min} = \min_{i \in T} R^a_i$. With an upper bound and a lower bound for the reward function of *observing* action, there is also an upper bound and a lower bound for the estimated value of state $\mathbf{o}$: $V^u(\mathbf{o}) = \max\{h^u(\mathbf{o}), W(\mathbf{o})\}$ and $V^l(\mathbf{o}) = \max\{h^l(\mathbf{o}), W(\mathbf{o})\}$. Both the upper bound and lower bound of state's value are updated during the updating procedure. The algorithm continues until a state $\mathbf{o}$ with $V^u(\mathbf{o}) - V^l(\mathbf{o}) < \epsilon$, where ϵ is a fixed error tolerance, is sampled.

For each state $\mathbf{o}$, there is a gap between the upper bound and lower bound of the estimated value $V^u(\mathbf{o}) - V^l(\mathbf{o})$. During the procedure of picking next state, the child state with the highest gap is preferred to choose. Similar to LRTDP, the upper bound $V^u(\mathbf{o})$ and lower bound $V^l(\mathbf{o})$ for a state $\mathbf{o}$ on horizon τ_{max} are set to be its known value $V(\mathbf{o}) = W(\mathbf{o})$.

4.3 VPI-RTDP

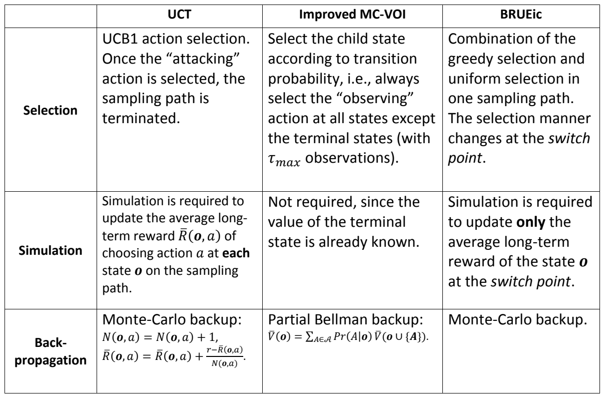
VPI-RTDP also keeps an upper bound and a lower bound similar to BRTDP, however, it prefers to prioritize picking states whose value update may lead to the largest improvement in policy value. Since performing this value of information analysis would be prohibitively expensive if done exactly and non-myopically, VPI-RTDP uses a myopic Bayesian value of information framework [13] to approximate the expected improvement in decision quality resulting from a state's value update.

VPI-RTDP assumes that the value $V(\mathbf{o})$ of state $\mathbf{o}$ is uniformly distributed between $V^l(\mathbf{o})$ and $V^u(\mathbf{o})$, and uses this distribution in a myopic VPI (Value of Perfect Information) [13] framework to approximate the expected improvement in decision quality resulting from the update of a state's value. During the procedure of picking next state, the one with the highest VPI value is preferred to select.

5. COMPARISON OF MCTS AND RTDP BASED APPROACHES

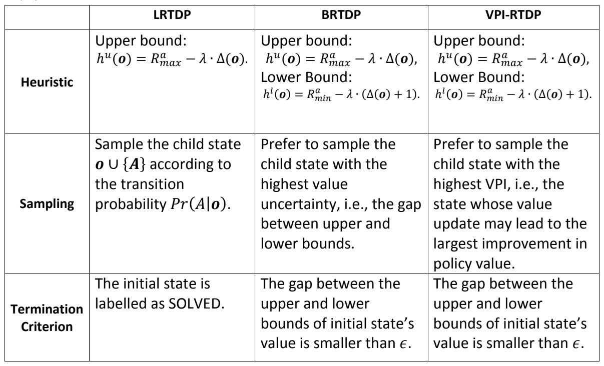
There are at least two important differences between the MCTS based approaches and RTDP based approaches. First, the RTDP based approaches hold the heuristic bound(s) of the state's value, therefore the advance knowledge of the state's value function is required, and the performance relies on the tightness of the heuristic bound(s). While the MCTS based approaches need no prior knowledge of the state's value and are easier to implement. Second, the RTDP based approaches guarantee the convergence of the computed policy and detect the convergence according to the heuristic bound(s). For example, BRTDP and VPI-RTDP ensure the convergence when the gap between the upper bound and lower bound of the state's value $V^u(\mathbf{o}) - V^l(\mathbf{o})$ is smaller than ϵ. However, MCTS based approaches cannot provide a guarantee of the convergence at any time.

We compare the three MCTS based approaches with respect to the three procedures: selection, simulation and backpropagation, which is shown in Figure 2a. Figure 2b shows the comparison of the three RTDP based approaches with respect to the heuristic, sampling and termination criterion.



	UCT	Improved MC-VOI	BRUEic
Selection	UCB1 action selection. Once the "attacking" action is selected, the sampling path is terminated.	Select the child state according to transition probability, i.e., always select the "observing" action at all states except the terminal states (with τ_{max} observations).	Combination of the greedy selection and uniform selection in one sampling path. The selection manner changes at the *switch point*.
Simulation	Simulation is required to update the average long-term reward $\bar{R}(\boldsymbol{o}, a)$ of choosing action a at **each** state $\boldsymbol{o}$ on the sampling path.	Not required, since the value of the terminal state is already known.	Simulation is required to update **only** the average long-term reward of the state $\boldsymbol{o}$ at the *switch point*.
Back-propagation	Monte-Carlo backup: $N(\boldsymbol{o}, a) = N(\boldsymbol{o}, a) + 1$, $\bar{R}(\boldsymbol{o}, a) = \bar{R}(\boldsymbol{o}, a) + \frac{r - \bar{R}(\boldsymbol{o},a)}{N(\boldsymbol{o},a)}$.	Partial Bellman backup: $\bar{V}(\boldsymbol{o}) = \sum_{A \in \mathcal{A}} Pr(A|\boldsymbol{o}) \bar{V}(\boldsymbol{o} \cup \{\boldsymbol{A}\})$.	Monte-Carlo backup.

(a) Comparison of UCT, improved MC-VOI and BRUEic



	LRTDP	BRTDP	VPI-RTDP
Heuristic	Upper bound: $h^u(\boldsymbol{o}) = R^a_{max} - \lambda \cdot \Delta(\boldsymbol{o})$.	Upper bound: $h^u(\boldsymbol{o}) = R^a_{max} - \lambda \cdot \Delta(\boldsymbol{o})$, Lower Bound: $h^l(\boldsymbol{o}) = R^a_{min} - \lambda \cdot (\Delta(\boldsymbol{o}) + 1)$.	Upper bound: $h^u(\boldsymbol{o}) = R^a_{max} - \lambda \cdot \Delta(\boldsymbol{o})$, Lower Bound: $h^l(\boldsymbol{o}) = R^a_{min} - \lambda \cdot (\Delta(\boldsymbol{o}) + 1)$.
Sampling	Sample the child state $\boldsymbol{o} \cup \{\boldsymbol{A}\}$ according to the transition probability $Pr(A|\boldsymbol{o})$.	Prefer to sample the child state with the highest value uncertainty, i.e., the gap between upper and lower bounds.	Prefer to sample the child state with the highest VPI, i.e., the state whose value update may lead to the largest improvement in policy value.
Termination Criterion	The initial state is labelled as SOLVED.	The gap between the upper and lower bounds of initial state's value is smaller than ϵ.	The gap between the upper and lower bounds of initial state's value is smaller than ϵ.

(b) Comparison of LRTDP, BRTDP and VPI-RTDP

Figure 2: Comparison of approximation approaches.

6. ID-BI ALGORITHM

Besides the above approximation approaches based on MCTS and RTDP, we also propose an *ID-BI (iterative deepening Backward Induction)* approximation algorithm to compute the approximation attacking policy.

The *ID-BI* algorithm begins with an initial MDP whose horizon is 0, and iteratively increases the horizon by a certain increment, until the optimal value of the initial state $\mathbf{o} = \langle o_A \rangle$ changes by less than ϵ. Since with a larger horizon of MDP, the attacker can obtain more information about

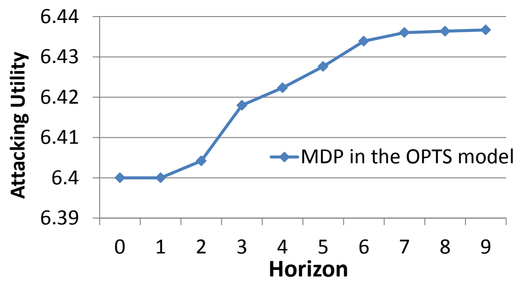


Figure 3: An example of the non-concave attacking utility.

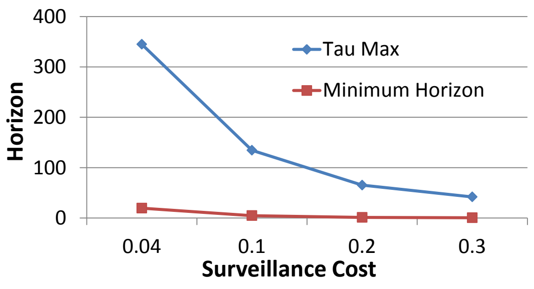


Figure 4: τ_{max} and minimum horizon H_{min}.

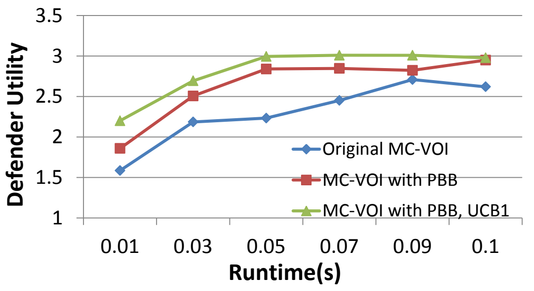


Figure 5: Original MC-VOI and improved MC-VOI.

the defender's strategy, the optimal value of the initial state $\mathbf{o} = \langle o_A \rangle$ increases monotonically with the increasing horizon. Thus, we use the convergence of initial state's value to compute an approximation to the attacker's optimal policy.

However, the *ID-BI* algorithm cannot guarantee the optimality of its solution since it may pre-terminate when the state's value function is non-concave. A security game instance with 5 targets, 1 security resource and a surveillance cost 0.06, whose value function is non-concave with the increasing horizon, is shown in Table 1.

	Target 1	Target 2	Target 3	Target 4	Target 5
R_i^a	5	1	9	6	-4
P_i^a	-7	-1	-4	-4	-4
R_i^d	2	6	2	2	8
P_i^d	-2	-7	-1	-3	-2

Table 1: A security game instance with a non-concave value function.

For the security game instance shown in Table 1, we use the backward induction method to compute the optimal value of the initial state $\mathbf{o} = \langle o_A \rangle$ of MDP with different horizons, and the result is shown in Figure 3. Obviously, the *ID-BI* algorithm terminates when the horizon is 1 and gets an attacking utility of 6.4, while the optimal attacking utility is near 6.44.

7. EXPERIMENTAL EVALUATION

We compare the approximation abilities of different approaches according to their runtimes and defender utilities. We conduct experiments primarily on randomly-generated instances of security games. R_i^a and R_i^d are drawn independently and uniformly from the range [0, 10]. P_i^a and P_i^d are drawn from the range [-10, 0]. All experiments are averaged over 50 sample games. Unless otherwise specified, we use 5 targets, 1 defender resource, $\lambda = 0.2$ as the observation cost, and $\alpha_A = 0$ for every $A \in \mathcal{A}$.

7.1 Structure of MDPs

We conduct experiments to study the structure of MDPs in the OPTS model. In order to do so, we compare the τ_{max} with the minimum horizon H_{min} of horizons where the optimal action for every state is *attacking*. The result is shown in Figure 4, and the data for observation costs 0.1 and 0.04 is obtained using the improved MC-VOI algorithm, since the exact algorithm *BI-FS* does not scale to problems with such small observation costs. States $\mathbf{o}$ with observation length larger than H_{min} are unreachable since there exists no path from the initial state to $\mathbf{o}$ such that the optimal action for every state on the path is *observing*. The MDP with the horizon H_{min} is equivalent with the original MDP with horizon τ_{max} with respect to the optimal attacking policy. As shown in Figure 4, H_{min} is much small compared with the τ_{max}. For example, when the surveillance cost is 0.2, $\tau_{max} = 65.3$ on average, while H_{min} on average is only 1.24.

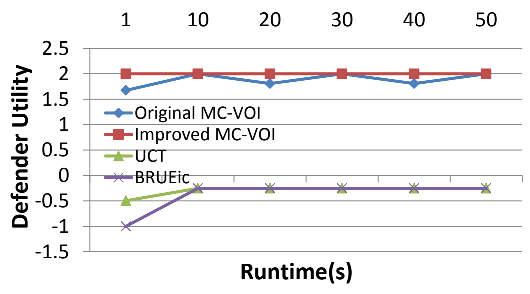
7.2 Improved MC-VOI

As stated in section 3.2, we improve the original MC-VOI method in two aspects: 1) We update the value for each state using the *partial Bellman backup* operator (PBB); 2) We incorporate the UCB1 selection method into the original MC-VOI to select child states at each state on the execution path. Figure 5 shows the performance[2] of three approaches based on MC-VOI: original MC-VOI method [14], MC-VOI with *partial Bellman backup* operator (PBB), and MC-VOI with PBB and UCB1 selection procedure, i.e., the improved MC-VOI. Improved MC-VOI outperforms the other two approaches significantly, and MC-VOI with PBB also outperforms the original MC-VOI method.

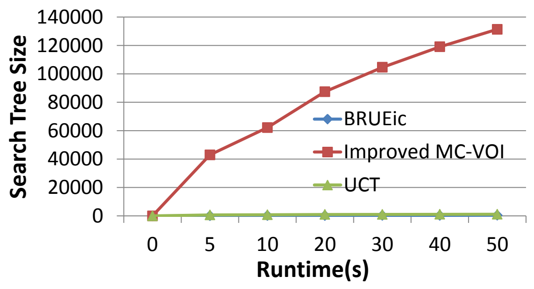
The optimal defender utility computed by the exact algorithm *BI-FS*[3] is 3.005, and takes 54s to calculate. The

[2]The defender strategy obtained by an approximation approach is the optimal one to combat the attacker's near-optimal policy this approach computed, and its defender utility is calculated using the real optimal attacker's policy to attack the defender strategy it obtained.

[3]*BI-FS* can only scale up to security games instances with

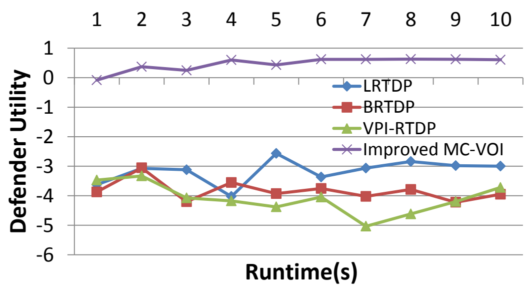


(a) Defender utility

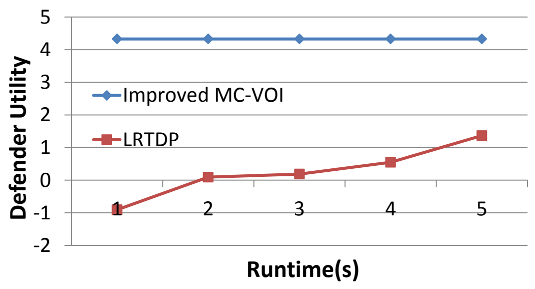


(b) Search tree size

Figure 6: Improved MC-VOI, UCT and BRUEic.



(a) $\lambda = 0.06$



(b) $\lambda = 0.2$

Figure 7: Improved MC-VOI vs. RTDP based approaches.

improved MC-VOI, however, takes only 0.05s to compute the near-optimal defender strategy.

We explain the good performance of the (improved) MC-VOI intuitively as follows: 1) H_{min} can be much smaller than the theoretical bound τ_{max}, although the MC-VOI cannot sample all the states, it still samples almost all the states with length no larger than H_{min}; 2) For a state with more observations, its optimal action is more likely to be *attacking*, and the improvement on attacking utility by one observation is small. In this case, a sample of single execution path is enough to provide necessary information for the states on larger horizons.

7.3 Approximations based on MCTS

We compare the four approaches based on MCTS: the original MC-VOI method, improved MC-VOI, UCT algorithm and BRUEic algorithm. As shown in Figure 6a, compared with the two advanced approaches based on MCTS: UCT and BRUEic, MC-VOI and the improved MC-VOI perform much better.

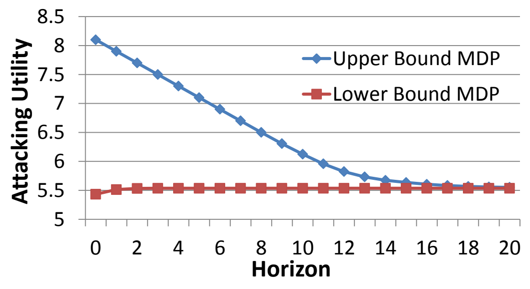
One reason for the better performance of MC-VOI is that it selects "observing" action all the time in the *SampleExecutionPath* procedure. Since the *attacking* action leads to the termination of the game immediately and the reward $W(\mathbf{o})$ of taking *attacking* action is fixed, the traditional MCTS approaches waste much time selecting *attacking* action and pre-terminating of the sampling path. Our analysis is confirmed by Figure 6b, where the search tree size is the number of sampled sates, UCT and BRUEic spend too much time in selecting *attacking* action. Hence, the search trees of these two advanced MCTS approaches consist of states on quite small horizons. MC-VOI, on the other hand, mostly chooses the *observing* action and aims at expanding the search tree as much as possible.

Another reason for the better performance of MC-VOI is that its value update policy, i.e., *partial Bellman backup*, outperforms the traditional value update policy in MCTS. In the traditional MCTS approach, the value update is done by receiving a reward of the a simulation process, which is a random action sequence. In contrast, in MC-VOI, an execution path is sampled from the initial state $\mathbf{o} = \langle o_A = 0 \rangle$ to a state $\mathbf{o}'$ on horizon τ_{max}, whose value $W(\mathbf{o}')$ is fixed and is easily computed, and the value update is done according to the recursive value function in Eq. (5), which is more accurate than the value update based on the reward of a random simulation process.

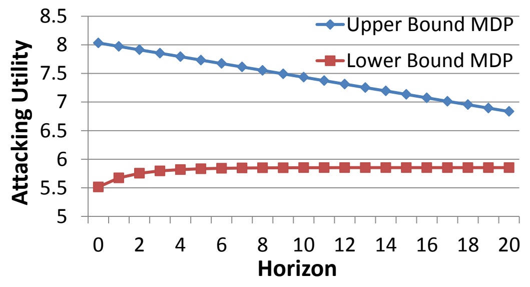
7.4 Approximations based on RTDP

We also compare the improved MC-VOI method with three approaches based on RTDP: LRTDP, BRTDP and VPI-RTDP, on security game instances with surveillance cost 0.06. As Figure 7a shows, improved MC-VOI approach also outperforms RTDP-based approaches a lot. The reason is that although RTDP-based approaches keep a heuristic reward function for *observing* action at each state and guarantee convergence to the optimal attacking policy, this heuristic upper bound or lower bound is too loose, so it takes a long time for the algorithms to reach convergence. Notice that although the attacker's utility of the attacking policy computed by RTDP based algorithms converges to the optimal utility monotonically as runtime increases, the defender

observation cost no smaller than 0.2, 5 targets and 1 defender resource.



(a) $\lambda = 0.2$



(b) $\lambda = 0.06$

Figure 8: Convergence of the lower bound MDP and the upper bound MDP.

utility of the optimal defender strategy for the computed attacking policy, may exhibit no monotonicity with runtime, as shown in Figure 7a.

Figure 7b shows the comparison of the improved MC-VOI and LRTDP for security games instances with surveillance cost 0.2. It takes quite a long time for LRTDP's heuristic upper reward function for *observing* action to converge to the optimal one. While improved MC-VOI has computed the optimal attacking policy within 1s, the attacking policy computed by LRTDP is still far away from optimality at 5s.

7.5 ID-BI Algorithm

Table 2: Attacking policy of the *ID-BI* algorithm.

Algorithm	λ	Runtime(s)	Accuracy Rate
BI-FS	0.2	69.64	100%
ID-BI	0.2	0.0007	100%
BI-FS	0.06	58.9	100%
ID-BI	0.06	2.51	96%

As shown in Figure 4, τ_{max} is much larger than H_{min}. In order to shorten the horizon of MDP, *BI-FS* algorithm solves two MDPs, an upper bound MDP and a lower bound MDP. In the upper bound MDP, for a leaf state, the value is $R^a_{max} - \lambda \cdot \Delta(\mathbf{o})$ where $R^a_{max} = \max_{i \in T} R^a_i$; In the lower bound MDP, the value is $W(\mathbf{o})$ for a leaf state, and for all other states, the values are defined according to the recursive value function in Eq. (5). It has been proved [1] that the attacking utility of the upper bound MDP with any horizon is an upper bound of the optimal attacking utility, and attacking utility of the lower bound MDP with any horizon is a lower bound of the optimal attacking utility. Besides, the attacking utilities of two MDPs converge to the optimal attacking utility as the horizon increases. *BI-FS* algorithm uses the convergence of the attacking utilities of the upper bound and lower bound MDPs to each other to check when an optimal solution has been reached.

However, these two MDPs do not converge to the optimal attacking utility at the same horizon. Figure 8 shows the convergence of attacking utilities of the upper bound and lower bound MDPs with the increasing horizons, and we can see that the attacking utility of the lower bound MDP converges to the optimal attacking utility faster than that of the upper bound MDP. That is the intuitive idea of the *ID-BI* algorithm, where we use the convergence of the attacking utility of the lower bound MDP to compute an approximation to the optimal attacker's policy. However, the attacking utility of the lower bound MDP may not be concave, which implies that when the attacking utility of the lower bound MDP converges, it may not be the optimal attacking utility. Figure 3 shows an example of the non-concave attacking utility of lower bound MDP. Thus, the *ID-BI* algorithm is not an exact algorithm, but it can obtain a good approximation for most cases, as shown in the following experiment.

We compare the performance of the *ID-BI* algorithm with the *BI-FS* algorithm, the exact algorithm for solving security games with surveillance cost [1], based on the attacking policy they obtained. The experiment is averaged over 100 security games instances. The accuracy rate is defined as the percentage of instances on which *ID-BI* computes the same attacker's policy as *BI-FS*[4]. As shown in Table 2, the *ID-BI* algorithm takes a small amount of time to compute a very accurate attacking policy.

8. CONCLUSION

In this paper, we implement several approximation approaches based on the MCTS and RTDP to improve the state of the art in the scalability of the MDP in the OPTS model. For the MC-VOI algorithm based on MCTS, we improve it by incorporating the *partial Bellman backup* policy to update value and UCB1 algorithm to select child states. We also propose an *ID-BI* approximation algorithm. According to our experimental evaluation, the improved MC-VOI not only outperforms the exact algorithm *BI-FS* in scalability for solving security games with surveillance cost, but also outperforms other approximation approaches based on MCTS and RTDP both in scalability and optimality. The *ID-BI* algorithm also outperforms the *BI-FS* algorithm for solving security games with surveillance cost in terms of scalability, and computes near-optimal solutions.

Acknowledgments

This work is supported by Singapore MOE AcRF Tier 1 grant MOE RG33/13. This research is also 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.

[4]Since the *BI-FS* algorithm cannot scale to instances with such a small surveillance cost 0.06, we run the *Backward Induction* algorithm on an MDP with the largest computational reachable horizon 24 to get an near-exact result instead.

REFERENCES

[1] B. An, M. Brown, Y. Vorobeychik, and M. Tambe. Security games with surveillance cost and optimal timing of attack execution. In *Proceedings of the 12th International conference on Autonomous Agents and Multi-Agent Systems (AAMAS'13)*, pages 223–230, 2013.

[2] B. An, M. Jain, M. Tambe, and C. Kiekintveld. Mixed-initiative optimization in security games: A preliminary report. In *Help Me Help You: Bridging the Gaps in Human-Agent Collaboration, Papers from the 2011 AAAI Spring Symposium, Technical Report SS-11-05*, 2011.

[3] B. An, D. Kempe, C. Kiekintveld, E. Shieh, S. P. Singh, M. Tambe, and Y. Vorobeychik. Security games with limited surveillance. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI'12)*, pages 1241–1248, 2012.

[4] B. An, M. Tambe, F. Ordóñez, E. A. Shieh, and C. Kiekintveld. Refinement of strong stackelberg equilibria in security games. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence (AAAI'11)*, pages 587–593, 2011.

[5] A. G. Barto, S. J. Bradtke, and S. P. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72(1):81–138, 1995.

[6] N. Basilico, N. Gatti, and F. Amigoni. Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *Proceedings of the 8th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'09)*, pages 57–64, 2009.

[7] B. Bonet and H. Geffner. Labeled RTDP: improving the convergence of real-time dynamic programming. In *Proceedings of the Thirteenth International Conference on Automated Planning and Scheduling (ICAPS'03)*, pages 12–21, 2003.

[8] G. Chaslot, S. Bakkes, I. Szita, and P. Spronck. Monte-carlo tree search: A new framework for game AI. In *Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE'08)*, 2008.

[9] G. Chaslot, S. Bakkes, I. Szita, and P. Spronck. Monte-carlo tree search: A new framework for game AI. In *Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE'08)*, 2008.

[10] J. P. Dickerson, G. I. Simari, V. S. Subrahmanian, and S. Kraus. A graph-theoretic approach to protect static and moving targets from adversaries. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'10)*, pages 299–306, 2010.

[11] Z. Feldman and C. Domshlak. Monte-carlo planning: Theoretically fast convergence meets practical efficiency. In *Uncertainty in Artificial Intelligence*, page 212. Citeseer, 2013.

[12] Z. Feldman and C. Domshlak. Simple regret optimization in online planning for markov decision processes. *J. Artif. Intell. Res. (JAIR)*, 51:165–205, 2014.

[13] R. A. Howard. Information value theory. *IEEE Trans. Systems Science and Cybernetics*, 2(1):22–26, 1966.

[14] E. Kamar and E. Horvitz. Light at the end of the tunnel: a monte carlo approach to computing value of information. In *Proceedings of the 12th International conference on Autonomous Agents and Multi-Agent Systems (AAMAS'13)*, pages 571–578, 2013.

[15] L. Kocsis and C. Szepesvári. Bandit based monte-carlo planning. In *Proceedings of the 17th European Conference on Machine Learning (ECML'06)*, pages 282–293, 2006.

[16] D. Korzhyk, V. Conitzer, and R. Parr. Complexity of computing optimal stackelberg strategies in security resource allocation games. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI'10)*, pages 805–810, 2010.

[17] H. B. McMahan, M. Likhachev, and G. J. Gordon. Bounded real-time dynamic programming: RTDP with monotone upper bounds and performance guarantees. In *Proceedings of the Twenty-Second International Conference on Machine Learning (ICML'05)*, pages 569–576, 2005.

[18] S. Sanner, R. Goetschalckx, K. Driessens, and G. Shani. Bayesian real-time dynamic programming. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI'09)*, pages 1784–1789, 2009.

[19] E. A. Shieh, B. An, R. Yang, M. Tambe, C. Baldwin, J. DiRenzo, B. Maule, and G. Meyer. PROTECT: an application of computational game theory for the security of the ports of the united states. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI'12)*, 2012.

[20] M. Tambe. *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned.* Cambridge University Press, 1st edition, 2011.