# Unleashing Dec-MDPs in Security Games: Enabling Effective Defender Teamwork

**Eric Shieh**[1], **Albert Xin Jiang**[1], **Amulya Yadav**[1], **Pradeep Varakantham**[2], and **Milind Tambe**[1]

**Abstract.**

Multiagent teamwork and defender-attacker security games are two areas that are currently receiving significant attention within multiagent systems research. Unfortunately, despite the need for effective teamwork among multiple defenders, little has been done to harness the teamwork research in security games. This paper is the first to remedy this situation by integrating the powerful teamwork mechanisms offered by Dec-MDPs into security games. We offer the following novel contributions in this paper: (i) New models of security games where a defender team's pure strategy is defined as a Dec-MDP policy for addressing coordination under uncertainty; (ii) New algorithms based on column generation that enable efficient generation of mixed strategies given this new model; (iii) Handling global events during defender execution for effective teamwork; (iv) Exploration of the robustness of randomized pure strategies. The paper opens the door to a potentially new area combining computational game theory and multiagent teamwork.

## 1 Introduction

Driven by the problem of optimizing team performance in domains with significant uncertainty, research in multiagent teamwork has for the past decade or more focused on fundamental advances in decentralized Markov Decision Problems (Dec-MDPs), providing major algorithmic breakthroughs [3, 6, 11]. On the other hand, security games have recently emerged as a research area in multiagent systems, leading to successful deployments that aid security scheduling at ports, airports and other infrastructure sites, while also aiding in anti-poaching efforts and protection of fisheries [18]. The challenge addressed in security games is optimizing the use of a defender's limited security resources in the presence of an adversary who can conduct surveillance before planning an attack.

This paper focuses on a challenge at the intersection of these two key areas in multiagent systems, potentially opening a fruitful new line of inquiry. In many security environments, teamwork among multiple defender resources of possibly different types (e.g., aerial, canine, motorized vehicles) is important to the overall effectiveness of the defender. However, teamwork is complicated by three factors—(i) requiring defender resources to coordinate under uncertainty; (ii) handling the dynamic inability of a resource to continue teamwork; and (iii) lack of communication—as we explain next.

While the work presented in this paper applies to many of the application domains of security games, including the security of flights, ports and rail [18], we focus on the metro rail domain for a concrete example, given the increasing amount of rail related terrorism threats [15]. The defender resources (i.e., canine, motorized) patrol the stations while the adversary conducts surveillance and may take advantage of the defender's predictability to plan an attack. Defender resources may engage in *teamwork* to patrol certain key areas that may be advantageous in thwarting the adversary compared to individual patrolling. Thus, if the adversary observes a coordinated set of defender resources patrolling a station, he will have to overcome multiple defenders if he decides to attack. Within this metro rail domain, we can see three factors that complicate teamwork. First, while defender resources are on patrol, one or more of them may be forced to deviate from the given patrol due to unforeseen events, such as questioning of suspicious individuals which results in delays in the patrol – but they may still need to continue to coordinate. Second, one of the defender resources may get interrupted to deal with a serious bomb threat – the entire team may be alerted to this threat via an emergency channel and the responsible resources may take over the response, resulting in the resource stopping the patrol and requiring others to fill in any gaps as a team. This type of global event affects the entire team and impacts the coordination among patrol resources. Third, in this rail domain there is often no communication among the defender resources due to various reasons, such as the trains and stations being underground or the use of cell phones or radio giving away the defender's coordinates or information.

Unfortunately, previous work in security games has mostly ignored this challenge of defender teamwork; while deployment of multiple defenders is optimized, most previous research has not focused on coordination among these resources. To handle teamwork of defender resources in security games under uncertainty, our work makes the following contributions. First, this paper provides a new model of a security game where a Dec-MDP policy is used as the defender's pure strategy to handle coordination under uncertainty. Second, we present a new algorithm that uses column generation to efficiently generate Dec-MDP policies as pure strategies used in determining the optimal mixed strategy for the defender team. Third, global events among defender resources are modeled and leveraged in handling teamwork. Fourth, we show heuristics that help scale-up to real-world scenarios. Fifth, while exploring randomized pure strategies previously seen to converge faster, we discovered that they were not as fast but instead were more robust.

## 2 Background: DEC-MDP

To represent the security problems of interest, we employ the well known DEC-MDP model that is defined by the tuple: $\langle Ag, S, A, T, R\rangle$. $Ag = \{1, \ldots, n\}$ represents the set of $n$ defender resources. $S = S_u \times S_1 \times \cdots \times S_n$ is a finite set of world states of the form $s = \langle s_u, s_1, \cdots, s_n\rangle$. Each resource $i$'s local state $s_i$ is a tuple $(t_i, \tau_i)$ where $t_i$ is the target and $\tau_i$ is the time at which resource $i$

[1] University of Southern California, USA
[2] Singapore Management University, Singapore

reaches target $t_i$. Time is discretized and there are $m$ decision epochs $\{1, \ldots, m\}$. $s_u$ is the *unaffected state*, meaning that it is not affected by the resources' actions. It is employed to represent occurrence of global events (bomb threats, increased risk at a location etc.)

$A = A_1 \times \cdots \times A_n$ is a finite set of joint actions $a = \langle a_1, \cdots, a_n \rangle$, where $A_i$ is the set of actions to be performed by resource $i$. $T : S \times A \times S \to \mathbb{R}$ is the transition function where $T(s, a, s')$ represents the probability of the next joint state being $s'$ if the current joint state is $s$ and joint action is $a$. Since transitions between resource $i$'s local states are independent of actions of other resources, we have transition independence. Due to the presence of unaffected states, this notion of transition independence is equivalent to the one employed in Network Distributed POMDPs [13]. Formally, $T(s, a, s') = T_u(s_u, s'_u) \cdot \prod_i T_i(\langle s_u, s_i \rangle, a_i, s'_i)$.

In this paper we are modeling game-theoretic interactions, in which the rewards depend on the strategies of both the defender and the attacker. Therefore standard Dec-MDP reward functions cannot be directly applied. Nevertheless, as part of our algorithm, we will reduce a subproblem to a Dec-MDP problem with a standard Dec-MDP joint reward function of the form $R : S \to \mathbb{R}$, where $R(s)$ represents the reward for reaching joint state $s$. Unlike in the ND-POMDP framework, our reward function is not decomposable.

## 3 Game Formulation

This paper presents a game theoretic model of effective teamwork among multiple defender resources with execution uncertainty by combining security games with Dec-MDPs. A security game [18] is a Stackelberg game with two players, a leader (defender) and a follower (attacker). The attacker is able to observe the mixed strategy of the defender resources, and then chooses a target-time pair $b = (t, \tau)$, where $t$ is the target to attack and $\tau$ is the time point to carry out the attack. In the train domain, targets correspond to stations in the metro system. Let $B$ be the set of target-time pairs. The defender's actions and capabilities (to be explained below) influence the *effectiveness* of coverage on target-time pairs, allowing for partial effectiveness. Each target-time pair $b$ has a payoff associated with it for both the attacker and defender, with $U_d^c(b)$ denoting the payoff for the defender if $b$ is covered (100% effectiveness), and $U_d^u(b)$ denoting the payoff for the defender if $b$ is uncovered (0% effectiveness) — we define defender expected utility under partial effectiveness later. We choose to have payoffs on both the location and time, due to the payoff being dependent on time, e.g., in the train domain, at rush hour the payoffs are larger than in the middle of the night with very few passengers. The payoffs for the attacker are in the same format, $U_a^c(b)$ and $U_a^u(b)$. A common assumption for security games is that $U_d^c(b) > U_d^u(b)$ and $U_a^c(b) < U_a^u(b)$, i.e., when a defender covers $b$, she receives a higher reward while the attacker receives a lower reward [18]. The model allows a non-zero-sum game, where the sum of defender's and attacker's payoff values may be non-zero.

The defender team has a set of $R$ resources. A (naive) patrol schedule for each resource consists of a sequence of commands; each command is of the form: at time $\tau$, the resource should be at target $t$ and execute action $a$. The action of the current command takes the defender resource to the location and time of the next command. In practice, each defender resource faces execution uncertainty, where taking an action might result in the defender resource being at a different location and time than intended. To handle execution uncertainty, we represent the defender's strategy as a joint policy of a transition-independent Dec-MDP, as defined in Section 2. For simplicity of exposition, we first focus on the case with no global events, in which case the unaffected state $s_u$ never changes and can be ignored. (We will consider these global events later in Section 5.) A defender resource $r$'s state $s_r = (t, \tau)$ represents her location (target) and time. Actions at $s_r$ are decisions of which target to visit next. Execution uncertainty is represented by probabilistic transitions. While more complex transitions could be easily modeled, we consider the following simple model of delays that mirror the real-world scenarios of unexpected events: for each action $a_r$ at $s_r$ there are two states $s'_r, s''_r$ with a nonzero transition probability: $s'_r$ is the intended next state and $s''_r$ has the same target as $s_r$ but a later time.

We define $\xi \in [0, 1]$ to be the effectiveness of a single defender resource visiting a target-time pair. $\xi$ can be less than 1 because visiting a target-time pair will not guarantee full protection. For example, if a defender resource visits a station, she will be able to provide some level of effectiveness, however she cannot guarantee that there is no adversary attack. Two or more defender resources visiting the same target-time pair provides an additional effectiveness. Given a global state $s$ of defender resources, let $\texttt{eff}(s, b)$ be the effectiveness of the resources on target-time pair $b$. For concreteness, we define the effectiveness of $k$ resources visiting the same target-time pair to be $1 - (1 - \xi)^k$. This corresponds to the probability of catching the attacker if each resource independently has probability $\xi$ of catching the attacker. Then $\texttt{eff}(s, b) = 1 - (1 - \xi)^{\sum_i I_{s_i = b}}$ where $I_{s_i = b}$ is the indicator function that is 1 when $s_i = b$ and 0 otherwise. Our methods would apply to other models of effectiveness, including when different resources have different capabilities.

Denote by $\boldsymbol{\pi}^j$ the defender team's $j^{th}$ pure strategy (joint policy), and $\boldsymbol{\pi}^J$ the set of all defender pure strategies, where $J$ is the corresponding set of indices. Each pure strategy $\boldsymbol{\pi}^j$ induces a distribution over global states visited. Denote by $\Pr(s|\boldsymbol{\pi}^j)$ the probability that global state $s$ is reached given $\boldsymbol{\pi}^j$. The expected effectiveness of target-time pair $b$ from defender pure strategy $\boldsymbol{\pi}^j$, is denoted by $P_b^j$; formally, $P_b^j = \sum_s \Pr(s|\boldsymbol{\pi}^j)\texttt{eff}(s, b)$. Given a defender pure strategy $\boldsymbol{\pi}^j$, and an attacker pure strategy of target-time pair $b$, the expected utility of the defender is $U_d(b, \boldsymbol{\pi}^j) = P_b^j U_d^c(b) + (1 - P_b^j) U_d^u(b)$. The attacker's utility is defined analogously. The defender may also play a mixed strategy $\mathbf{x}$, which is a probability distribution over the set of pure strategies $\boldsymbol{\pi}^J$. Denote by $x_j$ the probability of playing pure strategy $\boldsymbol{\pi}^j$. The players' expected utilities given mixed strategies are then naturally defined as the expectations of their pure-strategy expected utilities. Formally, the defender's expected utility given the defender mixed strategy $\mathbf{x}$ and attacker pure strategy $b$ is $\sum_j x_j U_d(b, \boldsymbol{\pi}^j)$. Let $c_b = \sum_j x_j P_b^j$ be the *marginal* coverage on $b$ by the mixed strategy $\mathbf{x}$ [18], and $\mathbf{c}$ the vector of marginal coverages over target-time pairs. Then this expected utility can be expressed in terms of marginal coverages, as $U_d(b, \mathbf{c}) = c_b U_d^c(b) + (1 - c_b) U_d^u(b)$.

Our model assumes the attacker has a surveillance phase prior to execution of an attack, and that it is difficult for the attacker to observe the defender's pure strategy and conditionally launch different attacks based on different observations. This is based on real-world cases and security experts' feedback [14]. In this Stackelberg game, we assume that the attacker plays a best response against the mixed strategy of the defender, which is a target-time pair $b$ that maximizes the attacker's expected utility given the defender's mixed strategy.

**Problem Statement:** We are interested in computing the *strong Stackelberg equilibrium* (SSE) of the game: the defender commits to the optimal mixed strategy (over a set of joint policies that handle execution uncertainty) that maximizes her expected utility (which requires teamwork among the defender resources), assuming a strategic adversary that best responds to her strategy.

## 4 Approach

To address the problem just outlined, ideally, the goal would be to obtain an optimal solution to the Stackelberg game with Dec-MDPs defining defender strategies. Unfortunately, finding a single optimal policy—a pure strategy—in a transition independent Dec-MDP is itself NP-complete [3]. We thus focus on a heuristic approach.

A standard method for solving Stackelberg games is the Multiple-LP algorithm [5]. It solves $|B|$ linear programs, each corresponding to an attacker pure strategy $b'$. The LP for $b'$, shown in Equations (1) to (5), solves the optimal defender mixed strategy $\mathbf{x}$ to commit to, given that the attacker's best response is to attack $b'$. Then among the $|B|$ solutions, the solution that achieves the best objective (i.e., defender expected utility) is chosen. In more detail, Equation (2) enforces that the best response of the attacker is indeed $b'$. In Equation (3), $\mathbf{P}^j$ is a column vector which gives the values of expected effectiveness $P_b^j$ of each target-time pair $b$ given defender's pure strategy $\boldsymbol{\pi}^j$. An example of a set of column vectors is shown below:

$$\mathbf{P} = \begin{array}{c} \\ b_1 \\ b_2 \\ b_3 \\ b_4 \end{array} \begin{array}{c} \begin{array}{ccc} j_1 & j_2 & j_3 \end{array} \\ \begin{bmatrix} 0.0 & 0.5 & 0.4 \\ 0.2 & 0.7 & 0.0 \\ 0.5 & 0.6 & 0.2 \\ 0.6 & 0.0 & 0.8 \end{bmatrix} \end{array}$$

Column $\mathbf{P}^{j_1} = \langle 0.0, 0.2, 0.5, 0.6 \rangle$ gives the effectiveness $P_{b_i}^{j_1}$ of the defender's pure strategy $\boldsymbol{\pi}^{j_1}$ over each target-time pair $b_i$. For example, policy $\boldsymbol{\pi}^{j_1}$ has an effectiveness of $0.5$ on $b_3$. Thus, Equation (3) enforces that given the probabilities $x_j$ of executing mixed strategies $\pi^j$, $c_b$ is the marginal coverage of $b$.

Since each column corresponds to a defender pure strategy, this algorithm requires enumerating all possible pure strategies. However, in our game there is an exponential number of possible defender pure strategies, corresponding to joint policies — and thus a massive number of columns that cannot be enumerated in memory — so that the Multiple-LP algorithm cannot be directly applied. For $N$ stations, $T$ time steps, and $R$ defender resources, we may have $(N^T)^R$ policies.

$$\max_{\mathbf{c},\mathbf{x}} \quad U_d(b', \mathbf{c}) \tag{1}$$

$$U_a(b', \mathbf{c}) \geq \quad U_a(b, \mathbf{c}) \quad \forall b \neq b' \tag{2}$$

$$c_b - \sum\nolimits_{j \in J} P_b^j x_j \leq \quad 0 \quad \forall b \in B \tag{3}$$

$$\sum\nolimits_{j \in J} x_j = \quad 1 \tag{4}$$

$$x_j \geq 0 \quad \forall j \in J, \quad c_b \in [0,1] \quad \forall b \in B \tag{5}$$

### 4.1 Column Generation

To deal with this problem, for each of the LPs we apply column generation [1], a method for efficiently solving LPs with large numbers of columns. At a high level, it is an iterative algorithm composed of a master and a slave component; at each iteration the master solves a version of the LP with a subset of columns, and the slave smartly generates a new column (defender pure strategy) to add to the master.

**The master** is an LP of the same form as Equations (1) to (5), except that instead of having all pure strategies, $J$ is now a subset of pure strategies. Pure strategies not in $J$ are assumed to be played with zero probability, and their corresponding columns do not need to be represented. We solve the LP and obtain its optimal dual solution.

**The slave**'s objective is to generate a defender pure strategy $\pi^j$ and add the corresponding column $\mathbf{P}^j$, which specifies the marginal coverages, to the master. We show that the problem of generating a good pure strategy can be reduced to a Dec-MDP problem.

To start, consider the question of whether adding a given pure strategy $\pi^j$ will improve the master LP solution. This can be answered using the concept of the *reduced cost* of a column [1], which intuitively gives the potential change in the master's objective when a candidate pure strategy $\pi^j$ is added. Formally, the reduced cost $\overline{f}_j$ associated with the column $\mathbf{P}^j$ is defined as $\overline{f}_j = \sum_b y_b \cdot P_b^j - z$, where $z$ is the dual variable of (4) and $\{y_b\}$ are the dual variables of Equation family (3). If $\overline{f}_j > 0$ then adding pure strategy $\pi^j$ will improve the master LP solution. When $\overline{f}_j \leq 0$ for all $j$, the current master LP solution is optimal for the full LP.

Thus the slave computes the $\pi^j$ that maximizes $\overline{f}_j$, and adds the corresponding column to the master if $\overline{f}_j > 0$. If $\overline{f}_j \leq 0$ the algorithm terminates and returns the current master LP solution.

### 4.2 Dec-MDP Formulation of Slave

We formulate this problem of finding the pure strategy that maximizes reduced cost as a Dec-MDP. The rewards are defined so that the total expected reward is equal to the reduced cost. The states and actions are defined as before. We can visualize them using *transition graphs*: for each resource $r$, the transition graph $\mathcal{G}_r = (N_r', E_r')$ contains state nodes $s_r = (t, \tau) \in S_r$ for each target and time. In addition, the transition graph also contains action nodes that correspond to the actions that can be performed at each state $s_r$. There exists a single action edge between a state node $s_r$ and each of the action nodes that correspond to the possible actions that can be executed at $s_r$. From each action node $a_r$ from $s_r$, there are multiple outgoing chance edges, to state nodes, with the probability $T_r(s_r, a_r, s_r')$ labeled on the chance edge to $s_r'$. In the illustrative example scenario that we have focused on, with there being delays, each action node has two outgoing chance edges with one chance edge going to the intended next state and another chance edge going to a different state which has the same location as the original node but a later time.

**Example:** Figure 1 shows a sample transition graph showing a subset of the states and actions for resource $i$. Looking at the state node $(t_1, 0)$, assuming target $t_1$ is adjacent to $t_2$ and $t_5$, there are three actions, Stay at $t_1$,Visit $t_2$, or Visit $t_5$. If action, Visit $t_2$ is chosen, then the transition probability is: $T_i((t_1,0), \text{Visit } t_2, (t_2,1)) = 0.9$ and $T_i((t_1,0), \text{Visit } t_2, (t_1,1)) = 0.1$.

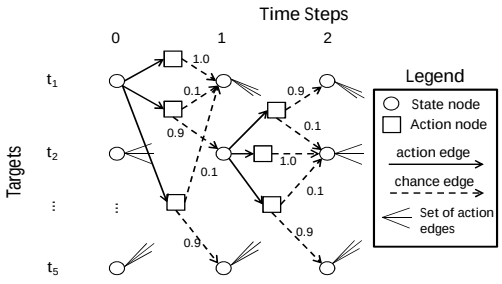



**Figure 1.** Example Transition Graph for 1 defender resource

The reward function $R(s)$ for this slave Dec-MDP — *consisting of multiple such transition graphs* — is dependent on the dual variables, $y_b$, from the master, and the effectiveness $\texttt{eff}(s, b)$ of resources with global state $s$ on target-time pair $b$, as defined in Section 3:

$$R(s) = \sum\nolimits_b y_b \cdot \texttt{eff}(s, b). \tag{6}$$

**Proposition 1.** *Let $\pi^j$ be the optimal solution of the slave Dec-MDP with reward function defined as in* (6)*. Then $\pi^j$ maximizes the reduced cost $\overline{f}_j$ among all pure strategies.*

*Proof.* The expected reward of the slave Dec-MDP given $\pi^j$ is

$$\sum\nolimits_s \Pr(s|\pi^j)R(s) = \sum\nolimits_b y_b \sum\nolimits_s Pr(s|\pi^j)\texttt{eff}(s,b)$$
$$= \sum\nolimits_b y_b P_b^j = \overline{f}_j + z.$$

Therefore the optimal policy for the Dec-MDP maximizes $\overline{f}_j$. □

### 4.3 Solving the Slave Dec-MDP

One approach to solve the slave Dec-MDP is to use solvers from the MADP toolbox [17] and also the MPS algorithm [6]. Unfortunately, these algorithms are unable to scale up past 4 targets and 4 resources in this problem scenario. Experimental results illustrating this outcome are shown in Section 7.

Our approach, outlined in Algorithm 1, borrows some ideas from the TREMOR algorithm [20], which iteratively and greedily updates the reward function for the individual resources and solves the corresponding MDP. More specifically, in each iteration, this algorithm updates the reward function for the MDP corresponding to resource $r$ and solves the single-agent MDP; the rewards of the MDP are updated so as to reflect the fixed policies of previous resources.

**Algorithm 1** SolveSlave($y_b$, $\boldsymbol{\mathcal{G}}$)

1: Initialize $\pi^j$
2: **for all** $r \in R$ **do**
3: $\quad\boldsymbol{\mu}_r \leftarrow$ ComputeUpdatedReward($\pi^j, y_b, \mathcal{G}_r$)
4: $\quad\pi_r \leftarrow$ SolveSingleMDP($\boldsymbol{\mu}_r, \mathcal{G}_r$)
5: $\quad\pi^j \leftarrow \pi^j \cup \pi_r$
6: $\mathbf{P}^j \leftarrow$ ConvertToColumn($\pi^j$)
7: return $\pi^j, \mathbf{P}^j$

In more detail, this algorithm takes the coefficients $y_b$ (refer Section 4.1) from the master component and $\boldsymbol{\mathcal{G}}$ (which consists of a set of transition graphs $\mathcal{G}_r$ – refer Section 4.2) as input and builds $\pi^j$ iteratively in Lines 2–5. Line 3 computes vector $\boldsymbol{\mu}_r$, the *additional* reward of reaching each of resource $r$'s states. Consider the slave Dec-MDP defined on resources $1, \ldots, r$ (with joint reward function (6)). The additional reward $\mu_r(s_r)$ for state $s_r$ is the marginal contribution of $r$ visiting $s_r$ to this joint reward, given the policies of the $r-1$ resources computed in previous iterations, $\pi^j = \{\pi_1, \ldots, \pi_{r-1}\}$. Specifically, because of transition independence, given $\{\pi_1, \ldots, \pi_{r-1}\}$ we can compute the probability $p_{s_r}(k)$ that $k$ of the first $r-1$ resources has visited the same target and time as $s_r$. Then $\mu_r(s_r) = \sum_{k=0}^{r-1} p_{s_r}(k)(\texttt{eff}(k+1) - \texttt{eff}(k))$, where we slightly abuse notation and define $\texttt{eff}(k) = 1 - (1-\xi)^k$.

Line 4 computes the best individual policy $\pi_r$ for resource $r$'s MDP, with rewards $\boldsymbol{\mu}_r$. We compute $\pi_r$ using value iteration (VI):

$$V(s_r, a_r) = \mu_r(s_r) + \sum\nolimits_{s'_r} T_r(s_r, a_r, s'_r)V(s'_r)$$

where $V(s_r) = \max_{a_r} V(s_r, a_r)$ and $\pi_r(s_r) = \arg\max_{a_r} V(s_r, a_r)$.

In addition to solving the single MDP for each resource by using value iteration, we also solved the MDP using soft-max value iteration (SMVI) [19]. SMVI is similar to VI except that the soft-max function is used instead of max while computing the value function of a state $s$. SMVI generates randomized policies – i.e., randomized pure strategies –, associating probability $\pi_r(s_r, a_r)$ to each action $a_r$ at each state $s_r$. Formally,

$$V(s_r) = \text{softmax}_{a_r} V(s_r, a_r) \equiv \log \sum\nolimits_{a_r} e^{V(s_r, a_r)}$$

$$\pi_r(s_r, a_r) = \frac{e^{V(s_r, a_r)}}{e^{V(s_r)}}$$

SMVI was first explored for its ability to speed up convergence, as in [19]. In our experiments SMVI did not provide significant runtime improvement, however we discovered that the randomized policy obtained from SMVI provides robustness to uncertainty in our estimates of transition probabilities, which is a highly useful feature since this uncertainty often arises in practice. The intuition behind SMVI providing robustness to uncertainty stems from the fact that the SMVI algorithm computes a policy that spreads out the probability of choosing an action at each state, instead of choosing only one action at each state (VI). In the presence of uncertainty, if the action that is chosen by VI is no longer the best action, it will still be chosen with a probability of 1. With soft-max, the probability over the action to take at each state is distributed over the actions based on their values, thereby when noise or uncertainty is added, the randomized policy will still have some probability of choosing the now best action (or a close-to-best action). Such probability will be significant especially when there are many close-to-optimal pure policies.

## 5 Global Events

Global events correspond to scenarios such as bomb threats or crime, where a resource must stop patrolling and deal with the unexpected event. Global unaffected state is a vector over different types of events that may be updated at each time step $\tau$. Depending on the type of event, a pre-specified defender resource will be removed from patrolling and allocated to dealing with the event once it occurs.

Transitions associated with global unaffected state, i.e., $T_u(s_u, s'_u)$ could potentially be computed based on the threat/risk levels of various events at the different time steps. The transitions associated with individual defender resources, i.e., $T_i(\langle s_u, s_i\rangle, a_i, s'_i)$ are dependent on whether the defender resource is responsible for handling a global event that has become active in that time step. If $s_u$ indicates that a bomb threat is active and $i$ is the qualified defender resource, then irrespective of the patrolling action, $s'_i$ will correspond to "out of patrolling duty" state, and resource $i$ will remain in that state for the rest of the patrol. Similarly, if $s_u$ indicates a bomb threat and $i$ is not the qualified defender resource, then resource $i$ would transition depending on action $a_i$ and $s_u$ with the knowledge that the qualified defender resource is no longer patrolling. Once this model associated with global events is present, we employ Algorithm 1 to solve the Dec-MDP. It should be noted that once a resource $i$ is out of patrolling duty, the policy of other resources get updated to account for one less resource available for doing patrolling (in "out of patrolling duty" state).

## 6 Improving Runtime

As mentioned earlier, without column generation, our model of Dec-MDPs in security games would be faced with enumerating $(N^T)^R$ columns, making enumeration of defender pure strategies impossible, let alone trying to find a solution. Column generation is thus critical to ensure that our model runs. However, starting each LP with

its own columns (i.e., cold-start) does not scale well. We build on this approach with several heuristics for scale-up:

**Append:** First, we explored reusing the generated defender pure strategies across the multiple LPs. The intuition is that the defender strategies generated by an LP might be useful in solving subsequent LPs, resulting in an overall decrease in the total number of defender pure strategies generated over all the multiple LPs.

**Cutoff:** To further improve the runtime, we explored setting a limit on the number of defender pure strategies generated (i.e., the number of iterations of column generation that is executed) for each LP.

**Ordered:** With this limit on the columns generated, some of the $|B|$ LPs return low-quality solutions, or are even infeasible, due to not having enough columns. Combined with reusing columns across LPs, the LPs that are solved earlier will have fewer columns. Since we only need a high-quality solution for the LP with the best objective, we would like to solve the most promising LPs last, so that these LPs will have a larger set of defender pure strategies to use. While we do not know apriori which LP has the highest value, one heuristic that turns out to work well in practice is to sort the LPs in increasing order of $U_a^u(b)$, the uncovered payoff of the attacker strategies (target-time pairs) chosen; i.e., to solve the LPs that correspond to attack strategies that are less attractive to the attacker first, and LPs (attack strategies) that are more attractive to the attacker later.

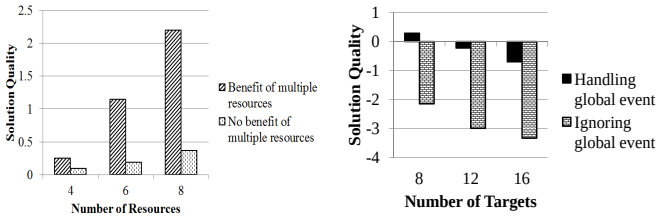
## 7 Evaluation

The experiments detailed in this section were performed on a quad core Linux machine with 12 GB of RAM and 2.3 GHz processor speed. The test results were averaged over 30 game instances, with each game having random payoffs in the range [-10,10]. Unless otherwise stated, the scenarios are run over 8 targets, 8 time steps and 4 resources, with 5% probability of delay and 5% probability of a global events, using VI with append + cutoff + ordering. The graphs of the scenarios are formed by connecting targets together in lines of length 5, and then randomly adding $\frac{|T|}{2}$ edges between targets, to resemble train systems in the real world with complex loops. *All key comparisons where we assert superiority of particular techniques, e.g., as in Figure 5(b), are statistically significant with* $p < 0.01$.

Figure 2(a) shows the benefit of our model's ability to handle teamwork. More specifically, it shows the difference in solution quality between our model where there is a benefit to having multiple resources covering the same target-time pair, $\texttt{eff}(s,b) = 1-(1-\xi)^{\sum_i I_{s_i=b}}$, and the case where there is no such additional effectiveness, $\texttt{eff}(s,b) = \xi \cdot I_{b \in s}$ (i.e., it is $\xi$ as long as at least one resource covers $b$). As the number of defender resources increases, the solution quality for when there is a benefit to having multiple resources increases at a faster rate than when there is no benefit of multiple resources visiting the same state (no teamwork).

Figure 2(b) further illustrates the expressiveness of our teamwork model. It compares the solution quality when we consider global events versus solving under the assumption of no global events. In the latter case, the system solves the model under the assumption that there is no global event, and we compute the defender expected utility if there is a 5% probability of global events at each time step.

In Figure 3 we compare the runtime of the VI-based slave for one iteration (no column generation) with other algorithms for Dec-MDPs such as MPS [6], JESP [12] and DICEPS [9]—this is the only figure in this section that focuses only on the slave and not on the master-slave algorithm in full. We show the number of targets along the x-axis and execution time(seconds) along the y axis. We see that JESP and DICEPS run out of memory for more than 2 targets, while MPS runs out of memory for more than 4 targets—thereby suggesting that security games require a new family of fast approximate Dec-MDP algorithms, such as our VI-based slave, providing a new fertile ground for further Dec-MDP research.




(a) Teamwork Effectiveness (b) Global Event

**Figure 2.** Teamwork and Global Event

Figure 4(a) and 4(b), shed light on the run-time and solution quality tradeoff of the various improvements made to the algorithm as mentioned in Section 6. The x-axis is the number of targets, and the y-axis is the runtime in minutes (Figure 4(a)), or solution quality (Figure 4(b)). Figure 4(a) demonstrates the 10 fold speedup resulting from append and 30 fold speedup using append, cutoff and ordering. Figure 4(b) demonstrates that when achieving our maximum speed improvement, our loss in solution quality is less than 3%.

Figure 4(c) shows the effectiveness of append + cutoff + ordering in scale up. We show that it is easily possible to scale up to 35 targets. This figure also compares VI versus SMVI and shows that SMVI does not provide runtime improvement over VI. Figure 4(d) shows the runtime for scaling up the number of defender resources. The cold start approach takes over 20 minutes to run past 4 resources while the append approach takes over 20 minutes to run for more than 6 resources. Using cutoff, the algorithm is able to handle over 10 defender resources.

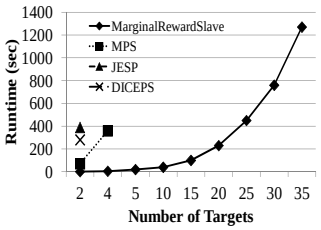
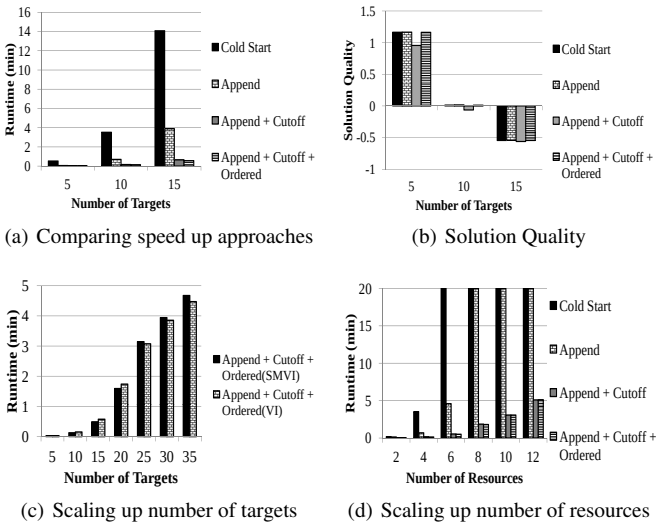



**Figure 3.** Comparison of various Dec-MDP solvers

Figure 5(a) shows the solution quality of our algorithms using SMVI and VI versus the uniform random strategy. The difference in solution quality between our algorithms (VI and SMVI) and Uniform Random suggests that the problem at hand is not trivial. Figure 5(b) shows the difference in solution quality of soft-max value iteration (SMVI) versus value iteration (VI) in the presence of uncertainty in transition probability for zero-sum games. The x-axis is the number of targets and the y-axis is the solution quality. The uncertainty that is added corresponds to the probability of the transition uncertainty being different than the initial assumed value. In this scenario, SMVI and VI obtain Dec-MDP based pure strategies with the assumption that the probability of delay of 5%. However, if the probability of delay was actually 10% while the algorithms assumed a delay of 5%, we look at how the solution quality is impacted. SMVI leads to less than 10% degradation but VI leads to more than 30% degradation due to uncertainty. This shows that without any uncertainty SMVI performs worse than VI, but with uncertainty in the transition probability, SMVI gives a higher solution quality than VI. Thus, SMVI is

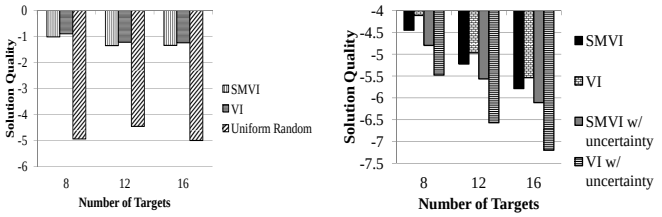



(a) Comparing speed up approaches (b) Solution Quality

(c) Scaling up number of targets (d) Scaling up number of resources

**Figure 4.** Runtime (a, c, d) and solution quality (b)

a more favorable option given uncertainty in transition probability.




(a) Solution Quality Comparison (b) Uncertainty

**Figure 5.** Comparison of different slaves and Uncertainty

## 8 Conclusion and Related Work

The key contribution of this paper is opening up a fruitful new area of research at the intersection of security games and multiagent teamwork. We present a novel game theoretic model that for the first time utilizes Dec-MDPs in addressing teamwork under uncertainty for security games. Handling the well-known computational challenges of Dec-MDPs requires leveraging column generation and further heuristics. Additionally, we handle global events and demonstrate the robustness of using randomized pure strategies.

While there has been significant research in security games including defending mobile targets [4], patrolling in extensive-form infinite-horizon games [2], and simulations and tools for maritime security [8], there has been limited work on coordination among defender resources [16]. However, these algorithms do not handle both execution uncertainty and teamwork among defender resources. The entire issue of planning based on Dec-MDPs in security games is a novel contribution of this work, not discussed in previous research.

Dec-MDPs are a popular framework for multiagent planning and coordination under uncertainty, with work ranging from a simplified model for transition independent Dec-MDPs [3], a toolbox for multiagent planning solvers [17], the use of heuristic search and constraint optimization [6], to multi-robot exploration [10]. A major difference in this paper is the addition of an adversarial agent that is able to respond to the joint policy of the Dec-MDP. Partially Observable Stochastic Games [7] can be used to model an adversarial agent along with cooperative defender agents, however in our domain, the attacker has a more simple problem that does not require such a generalized model allowing us to exploit the specialization for speed-up.

**Acknowledgments:** This research was supported by the United States Department of Homeland Security through the National Center for Risk and Economic Analysis of Terrorism Events (CREATE) under award number 2010-ST-061-RE0001 and MURI grant W911NF-11-1-0332.

## REFERENCES

[1] C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh, and P.H. Vance, 'Branch and price: Column generation for solving huge integer programs', in *Operations Research*, (1994).

[2] Nicola Basilico, Nicola Gatti, and Francesco Amigoni, 'Patrolling security games: Definition and algorithms for solving large instances with single patroller and single intruder', *Artificial Intelligence*, (2012).

[3] Raphen Becker, Shlomo Zilberstein, Victor Lesser, and Claudia V Goldman, 'Solving transition-independent decentralized markov decision processes', in *JAIR*, (2004).

[4] Branislav Bošanskỳ, Viliam Lisỳ, Michal Jakob, and Michal Pěchouček, 'Computing time-dependent policies for patrolling games with mobile targets', in *AAMAS*, (2011).

[5] Vincent Conitzer and Tuomas Sandholm, 'Computing the optimal strategy to commit to', in *ACM EC-06*, pp. 82–90, (2006).

[6] Jilles S Dibangoye, Christopher Amato, and Arnoud Doniec, 'Scaling up decentralized MDPs through heuristic search', in *UAI*, (2012).

[7] Eric A Hansen, Daniel S Bernstein, and Shlomo Zilberstein, 'Dynamic programming for partially observable stochastic games', in *AAAI*, (2004).

[8] Michal Jakob, Ondřej Vaněk, Ondřej Hrstka, and Michal Pěchouček, 'Agents vs. pirates: multi-agent simulation and optimization to fight maritime piracy', in *AAMAS*, (2012).

[9] Shie Mannor, Reuven Y Rubinstein, and Yohai Gat, 'The cross entropy method for fast policy search', in *ICML*, (2003).

[10] Laëtitia Matignon, Laurent Jeanpierre, and Abdel-Illah Mouaddib, 'Coordinated multi-robot exploration under communication constraints using decentralized markov decision processes.', in *AAAI*, (2012).

[11] Francisco S Melo and Manuela Veloso, 'Decentralized MDPs with sparse interactions', *Artificial Intelligence*, (2011).

[12] Ranjit Nair, Milind Tambe, Makoto Yokoo, David Pynadath, and Stacy Marsella, 'Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings', in *IJCAI*, (2003).

[13] Ranjit Nair, Pradeep Varakantham, Milind Tambe, and Makoto Yokoo, 'Networked distributed POMDPs: A synthesis of distributed constraint optimization and POMDPs', in *AAAI*, (2005).

[14] James Pita, Manish Jain, Fernando Ordóñez, Christopher Portway, Milind Tambe, Craig Western, Praveen Paruchuri, and Sarit Kraus, 'Using Game Theory for Los Angeles Airport Security', *AI Magazine*, (2009).

[15] Reuters, 'Al Qaeda planning attacks on high-speed trains in Europe: newspaper', (2013). Retrieved Oct 3, 2013 from http://www.reuters.com/article/2013/08/19/us-germany-security-qaeda-idUSBRE97I0IN20130819.

[16] Eric Shieh, Manish Jain, Albert Xin Jiang, and Milind Tambe, 'Efficiently solving joint activity based security games', in *IJCAI*, (2013).

[17] Matthijs TJ Spaan and Frans A Oliehoek, 'The multiagent decision process toolbox: Software for decision-theoretic planning in multiagent-systems', in *IFAAMAS*, (2008).

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

[19] Pradeep Varakantham, Asrar Ahmed, and Shih-Fen Cheng, 'Decision support for assorted populations in uncertain and congested environments', *In submission to JAIR*, (2013).

[20] Pradeep Varakantham, Junyoung Kwak, Matthew Taylor, Janusz Marecki, Paul Scerri, and Milind Tambe, 'Exploiting coordination locales in distributed POMDPs via social model shaping', in *ICAPS*, (2009).