Inf Technol Manag (2009) 10:67–79
DOI 10.1007/s10799-008-0047-9

Coordinating randomized policies for increasing security of agent systems

Praveen Paruchuri · Jonathan P. Pearce · Janusz Marecki · Milind Tambe · Fernando Ordóñez · Sarit Kraus

Published online: 24 January 2009

Abstract We consider the problem of providing decision support to a patrolling or security service in an adversarial domain. The idea is to create patrols that can achieve a high level of coverage or reward while taking into account the presence of an adversary. We assume that the adversary can learn or observe the patrolling strategy and use this to its advantage. We follow two different approaches depending on what is known about the adversary. If there is no information about the adversary we use a Markov Decision Process (MDP) to represent patrols and identify randomized solutions that minimize the information available to the adversary. This lead to the development of algorithms CRLP and BRLP, for policy randomization of MDPs. Second, when there is partial information about the adversary we decide on efficient patrols by solving a Bayesian–Stackelberg games. Here, the leader decides first on a patrolling strategy and then an adversary, of possibly many adversary types, selects its best response for the given patrol. We provide two efficient MIP formulations named DOBSS and ASAP to solve this NP-hard problem. Our experimental results show the efficiency of these algorithms and illustrate how these techniques provide optimal and secure patrolling policies. We note that these models have been applied in practice, with DOBSS being at the heart of the ARMOR system that is currently deployed at the Los Angeles International airport (LAX) for randomizing checkpoints on the roadways entering the airport and canine patrol routes within the airport terminals.

Keywords Multiagent systems · Decision theory · Game theory · Security · Randomized policies

P. Paruchuri
Carnegie Mellon University, Pittsburgh, PA 15232, USA
e-mail: paruchur@gmail.com

J. P. Pearce
Knight Capital Group, Jersey city NJ, USA
e-mail: jppearce@usc.edu

J. Marecki
IBM Research, York Town, NY, USA
e-mail: marecki@usc.edu

M. Tambe · F. Ordóñez (✉)
University of Southern California, Los Angeles, CA 90089, USA
e-mail: fordon@usc.edu

M. Tambe
e-mail: tambe@usc.edu

S. Kraus
Bar-Ilan University, Ramat-Gan 52900, Israel
e-mail: sarit@cs.biu.ac.il

S. Kraus
University of Maryland, College Park, MD 20742, USA

1 Introduction

Security, commonly defined as the ability to deal with intentional threats from other agents is a major challenge for agents deployed in adversarial environments [14]. In this paper, we focus on adversarial domains where the agents have limited information about the adversaries. Such adversarial scenarios arise in a wide variety of situations that are becoming increasingly important such as patrol agents providing security for a group of houses or regions [5, 15], UAVs monitoring a humanitarian mission [1, 14], agents assisting in routine security checks at airports [18], agents providing privacy in sensor network routing [12] or agents maintaining anonymity in peer to peer networks [2].

This paper brings together some of our recent work on how to plan for agents acting in uncertain environments in

the presence of adversaries [14, 15, 16]. This research has introduced two very different approaches for increasing security in agent systems and has lead to the Assistant for Randomized Monitoring over Routes (ARMOR) system which has been deployed for security scheduling at the LAX airport since August 2007 [11, 16, 17]. Here we will present the main results and algorithms proposed in these two approaches and highlight the relationship between them. The common assumption in these security domains is that the agent commits to a plan or policy first while the adversary observes the agent's actions and hence knows its plan/policy. The adversary can then exploit the plan or policy the agent committed to. In addition, the agent might have to decide on its strategy having only incomplete information. For example, in a typical security domain such as the patrolling agents example, agents provide security for a group of houses or regions via patrolling. The patrol agents commit to a plan or policy while the adversaries can observe the patrol routes, learn the patrolling pattern and exploit it to their advantage. Furthermore, the agents might not know which adversaries they face or what exactly their objectives are. To solve this problem with incomplete information about the adversaries, we provide efficient algorithms for improving security broadly considering two realistic situations: Firstly, when the agents have no model of their adversaries, our objective is to obtain strategies for a Markov Decision Process (MDP) that balance the agent's reward with the amount of information gained by the adversary about the agent. Secondly, when the agents have partial model of their adversary we use a game theoretic framework to obtain maximal reward strategies taking into account the uncertainty over adversary types.

When the agents have no model of their adversaries, we briefly present efficient algorithms, as introduced in [14], for generating randomized plans or policies for the agents that minimize the information that can be gained by adversaries. Such randomized policies that attempt to minimize the opponent's information gain are referred to as *secure policies*. However, arbitrary randomization can violate quality constraints, such as: increasing resource usage, frequency of patrols in key areas. To that end, we developed algorithms for efficient policy randomization with quality guarantees using MDPs [19]. We measure randomization via an entropy-based metric. In particular, we illustrate that simply maximizing entropy-based metrics introduces a non-linear program that has non-polynomial run-time. Hence, we introduce our Convex combination for Randomization (CRLP) and Binary search for Randomization (BRLP) linear programming (LP) techniques that randomize policies in polynomial time with different tradeoffs as explained later.

When the agents have a partial model of their adversary, we model the security domain as a Bayesian–Stackelberg games [6, 15]. A Bayesian game is a game in which agents may belong to one or more types; the type of an agent determines its possible actions and payoffs. The assumption made here is that the agent knows the adversary's actions and payoffs but does not know which adversary is active at a given time. Usually these games are analyzed according to the concept of Bayes–Nash equilibrium, an extension of Nash equilibrium for Bayesian games in which it is assumed that all the agents choose their strategies simultaneously. However, the main feature of the security games we consider is that one player must commit first to a strategy before the other players choose their strategies. In the patrol domain, the patrol agent commits to a strategy first while the adversaries get to observe the agent's strategy and decide on their choice of action. These scenarios are known as Stackelberg games [9]. More precisely, we model our security domains as Bayesian–Stackelberg games to take into account that the leader must plan for possibly many different types of adversaries. The solution concept for these games is that the security agent has to pick the optimal strategy considering the actions, payoffs and probability distribution over the adversaries. In [15] and [16], we introduced efficient techniques for generating optimal leader strategies with controlled and optimal randomization for Bayesian–Stackelberg games, named as Agent Security via Approximate Policies (ASAP) and Decomposed Optimal Bayesian Stackelberg Solver (DOBSS), respectively. Furthermore, DOBSS is at the heart of the ARMOR [16, 17] system that is currently deployed for security scheduling at the LAX airport, which has been described in popular scientific magazines and news media such as [11].

The ARMOR software is a general-purpose security scheduler built over the DOBSS algorithm. In particular, it is being used for randomizing police checkpoints and canine patrols for improving security at the LAX airport. For example, airports cannot afford having checkpoints on all roads at all times due to limited security personnel. Potential adversaries can monitor the checkpoints regularly and learn weaknesses/patterns. ARMOR accounts for various factors including number of checkpoints, their operation times, traffic patterns, adversary's cost for getting caught, estimated target priority for adversary, etc. to calculate the optimal randomized solution. In most security domains, police/canine units commit first to a security policy while our adversaries observe and exploit the policy committed to. This key observation is mapped to a Bayesian–Stackelberg games and is solved using the DOBSS algorithm.[1]

[1] The ARMOR software has been developed in close interaction with the LAWA (Los Angeles World Airports) police, and has been in use at LAX since Aug'07.

The rest of this paper is organized as follows, in Sect. 2 below we present related work. Section 3 introduces the classic Markov Decision approach for planning and the LP solution to solve it. We then present a non-linear program and two approximate linear programming alternatives called the CRLP and the BRLP algorithms for efficient randomized policy generation in the presence of an unmodeled adversary. Section 4 briefly presents the DOBSS procedure for generating optimal randomized strategies, first for non-Bayesian games, for clarity; then shows how it can be adapted for Bayesian games with partial adversary information. We then provide a brief description of the ASAP algorithm that generates policies with controlled randomization using the framework developed for DOBSS. Section 5 provides experimental results for both the techniques developed in this paper. Section 6 gives the conclusions and discusses policy implications of the methods presented in the paper.

2 Related work

There are two main methodological directions that are followed in this work: decision theoretic and game theoretic models. Below we point to related work in both these areas and how it pertains to randomizing in security domains.

Decision theoretic frameworks, such as MDPs, are extremely useful and powerful modeling tools that are being increasingly applied to build agents and agent teams that can be deployed in real world. The main advantage of modeling agent and agent teams using these tools is the following:

- Real world is uncertain and the decision theoretic frameworks can model such real-world environmental uncertainty. In particular, the MDP [19] framework can model stochastic actions and hence can handle transition uncertainty.
- Efficient algorithms have been devised for generating optimal plans for agents and agent teams modeled using these frameworks [19].

However, these optimal policy generation algorithms have focused on maximizing the total expected reward while taking the environmental uncertainties into account. Such optimal policies are deterministic and therefore useful when the agents act in environments where acting in a predictable manner is not problematic. As agents get increasingly deployed in real world, they will have to act in adversarial domains often without any adversary model available. Hence, randomization of policies becomes critical. Randomization of policies using decision theoretic frameworks as a goal has received little attention, and is primarily seen as a side-effect in attaining other objectives like in Constrained MDPs [7, 13].

Stackelberg games [20, 23] are commonly used to model attacker–defender scenarios in security domains [3]. In particular [3] develops algorithms to make critical infrastructure more resilient against terrorist attacks by modeling the scenario as a Stackelberg game. However they do not address the issue of incomplete information about adversaries whereas agents acting in the real world quite frequently are uncertain, or do not have complete information, about the adversary faced. Bayesian games have been a popular choice to model such incomplete information games [4, 6] and the solution concept is called the Bayes–Nash equilibrium [9]. The problem of choosing an optimal strategy for the leader to commit to in a Stackelberg game is analyzed in [6] and found to be NP-hard in the case of a Bayesian game with multiple types of followers. Methods for finding optimal leader strategies for non-Bayesian games [6] can be applied to this problem by converting the Bayesian game into a normal-form game by the Harsanyi transformation [10]. However, by transforming the game, the compact structure of the Bayesian game is lost. In addition, the method by [6] (called the Multiple LPs method) requires solving many linear programs, some of which may be infeasible. If, on the other hand, we wish to compute the highest-reward Nash equilibrium, new methods using mixed-integer linear programs (MILPs) [21] may be used, since the highest-reward Bayes–Nash equilibrium is equivalent to the corresponding Nash equilibrium in the transformed game. Furthermore, since the Nash equilibrium assumes a simultaneous choice of strategies, the advantages of being the leader are not considered. Our work proposes an efficient and compact technique for choosing optimal strategies in Bayesian–Stackelberg games.

3 Randomization with no adversary model

In this section, we first describe MDPs, followed by our approaches to randomization of MDP policies. An MDP is a tuple $\langle S, A, P, R\rangle$, that consists of world states $\{s_1,\ldots,s_m\}$, actions $\{a_1,\ldots,a_k\}$, transition function which is a set of tuples $p(s, a, j)$ and immediate reward denoted by tuples $r(s, a)$. If $x(s, a)$ represents the number of times the MDP visits state s and takes action a and α_j represents the number of times the MDP starts in each state $j \in S$, then the optimal policy, maximizing expected reward, is derived via the following linear program [8]:

$$\begin{aligned} \max \quad & \sum_{s\in S}\sum_{a\in A} r(s,a)x(s,a) \\ \text{s.t.} \quad & \sum_{a\in A} x(j,a) - \sum_{s\in S}\sum_{a\in A} p(s,a,j)x(s,a) = \alpha_j, \quad \forall j \in S \\ & x(s,a) \geq 0 \quad \forall_s \in S, a \in A \end{aligned} \tag{1}$$

If x^* is the optimal solution to (1), the optimal policy π^* is given by (2) below, where $\pi^*(s, a)$ is the probability of

taking action a in state s and is deterministic i.e., $\pi^*(s, a)$ has a value of either 0 or 1. However, such deterministic policies are undesirable in security domains.

$$\pi^*(s,a) = \frac{x^*(s,a)}{\sum_{\hat{a}\in A} x^*(s,\hat{a})}. \quad (2)$$

3.1 Maximal entropy solution

We aim to randomize these optimal deterministic policies, where randomness is quantified using some entropy measure. The notion of entropy for probability distributions is introduced by Shannon in [22]. Entropy for a discrete distribution $p_1,\ldots,p_n$ is defined by $H = -\sum_{i=1}^{n} p_i \log p_i$. For a MDP we introduce the *weighted entropy* function, borrowing from the classic entropy definition. The weighted entropy is defined by adding the entropy for the distributions at every state weighted by the likelihood the MDP visits that state, namely

$$\begin{aligned} H_W(x) &= -\sum_{s\in S} \frac{\sum_{\hat{a}\in A} x(s,\hat{a})}{\sum_{j\in S}\alpha_j} \sum_{a\in A} \pi(s,a)\log\pi(s,a) \\ &= -\frac{1}{\sum_{j\in S}\alpha_j} \sum_{s\in S}\sum_{a\in A} x(s,a)\log\left(\frac{x(s,a)}{\sum_{\hat{a}\in A} x(s,\hat{a})}\right). \end{aligned}$$

Note that when all states have equal weight of 1, we call the above function as additive entropy denoted by $H_A(x)$. The maximal entropy solution for MDP can be defined as:

$$\begin{aligned} &\max -\frac{1}{\sum_{j\in S}\alpha_j}\sum_{s\in S}\sum_{a\in A} x(s,a)\log\left(\frac{x(s,\,a)}{\sum_{\hat{a}\in A} x(s,\,\hat{a})}\right) \\ &\text{s.t.}\sum_{a\in A} x(j,a) - \sum_{s\in S}\sum_{a\in A} p(s,a,j)x(s,a) = \alpha_j \quad \forall j\in S \\ &\sum_{s\in S}\sum_{a\in A} r(s,a)x(s,a) \geq E_{\min} \\ &x(s,a)\geq 0 \quad \forall s\in S, a\in A \end{aligned} \quad (3)$$

Here, $E_{\min}$ is the reward threshold and is an input domain parameter. Note that for $E_{\min} = 0$ the above problem finds the maximum weighted entropy policy, and for $E_{\min} = E^*$, Problem (3) returns the maximum expected reward policy with largest entropy, where E^* is the maximum possible expected reward. Unfortunately the function $H_W(x)$ is neither convex nor concave in x, hence there are no complexity guarantees in solving Problem (3), even for local optima. This negative complexity motivates the polynomial methods presented below.

3.2 Efficient single agent randomization

Note that, while entropy calculation is a non-linear function, entropy-maximization is a convex problem. The non-convexity in the functions above arises due to way probabilities are calculated i.e. as a ratio of the flow variables in the (MDP) network. We now present two polynomial time algorithms to obtain policies for an MDP that balance the reward and randomness. The algorithms that we introduce below consider two inputs: a minimal expected reward value $E_{\min}$ and a randomized solution $\bar{x}$ (or policy $\bar{\pi}$). The input $\bar{x}$ can be any solution with high entropy and is used to enforce some level of randomness on the high expected reward output, through linear constraints. One such high entropy input for MDP-based problems is the uniform policy, where $\bar{\pi}(s,a) = 1/|A|$. We enforce the amount of randomness in the high expected reward solution that is output through a parameter $\beta \in [0, 1]$. For a given β and a high entropy solution $\bar{x}$, we output a maximum expected reward solution with a certain level of randomness by solving (4).

$$\begin{aligned} &\max \sum_{s\in S}\sum_{a\in A} r(s,a)x(s,a) \\ &\text{s.t.}\sum_{a\in A} x(j,a) - \sum_{s\in S}\sum_{a\in A} p(s,a,j)x(s,a) = \alpha_j \quad \forall j\in S \\ &x(s,a)\geq \beta\bar{x}(s,a) \quad \forall s\in S, a\in A. \end{aligned} \quad (4)$$

which can be expressed in matrix notation. Let $x = (x(s,a))_{s\in S, a\in A}$ be an $|S||A|$ dimensional variable vector, α a vector in $\Re^{|S|}$, r a vector in $\Re^{|S||A|}$, and M a matrix with $|S|$ rows and $|S||A|$ columns. The matrix shorthand would then be,

$$\begin{aligned} \max \quad & r^T x \\ \text{s.t.} \quad & Mx = \alpha \\ & x \geq \beta\bar{x}. \end{aligned}$$

As the parameter β is increased, the randomness requirements of the solution become stricter and hence the solution to (4) would have smaller expected reward and higher entropy. For $\beta = 0$ the above problem reduces to (1) returning the maximum expected reward solution E^*; and for $\beta = 1$ the problem obtains the maximal expected reward (denoted $\overline{E}$) out of all solutions with as much randomness as $\bar{x}$.

Theorem 1 *If $\bar{x}$ is a feasible solution to (1) (that is $M\bar{x} = \alpha, \bar{x}\geq 0$) and$E^*$ is finite, then $\bar{x}$ is an optimal solution to (4) when $\beta = 1$ and $\bar{E} = \sum_{s\in S}\sum_{a\in A} r(s,a)\bar{x}(s,a) = r^T\bar{x}$.*

Proof If E^* is finite then for any x such that $Mx = 0$, $x \geq 0$ we must have that $r^T x \leq 0$. By construction $\bar{x}$ is feasible for (4) with $\beta = 1$. Consider a solution $\tilde{x}$ feasible for (4) with $\beta = 1$. Then $\tilde{x} - \bar{x}\geq 0$ and $M(\tilde{x} - \bar{x}) = 0$, therefore since E^* is finite we have $r^T(\tilde{x} - \bar{x}) \leq 0$, which shows that $\bar{x}$ is optimal for (4). □

Our new algorithm to obtain an efficient solution with a expected reward requirement of $E_{\min}$ is based on the following result which shows that the solution to (4) is a

convex combination of the deterministic and random input solutions.

Theorem 2 ([14], Theorem 1) *Consider a solution $\bar{x}$,which satisfies $M\bar{x} = \alpha$and $\bar{x} \geq 0$.Letx* be the solution to (1) and $\beta \in [0, 1]$. If$x_\beta$is the solution to (1) then $x_\beta = (1-\beta)x^* + \beta\bar{x}$.*

Proof We reformulate problem (4) in terms of the slack $z = x - \beta\bar{x}$ of the solution x over $\beta\bar{x}$ leading to the following problem:

$$\begin{aligned} \beta r^T\bar{x} + \quad & \max \quad r^T z \\ & \text{s.t.} \quad Mz = (1-\beta)\alpha \\ & \qquad z \geq 0, \end{aligned}$$

The above problem is equivalent to (4), where we used the fact that $M\bar{x} = \alpha$. Let z^* be the solution to this problem, which shows that $x_\beta = z^* + \beta\bar{x}$. Dividing the linear equation $Mz = (1-\beta)\alpha$, by $(1-\beta)$ and substituting $u = z/(1-\beta)$ we recover the deterministic problem (1) in terms of u, with u^* as the optimal deterministic solution. Renaming variable u to x, we obtain $\frac{1}{1-\beta}z^* = x^*$, which concludes the proof. □

Since $x_\beta = (1-\beta)x^* + \beta\bar{x}$ we can directly find a randomized solution which obtains a target expected reward of $E_{\min}$. Due to the linearity in relationship between x_β and β, a linear relationship exists between the expected reward obtained by x_β (i.e $r^T x_\beta$) and β. In fact setting $\beta = \frac{r^T x^* - E_{\min}}{r^T x^* - r^T \bar{x}}$ makes $r^T x_\beta = E_{\min}$. Using the theorem, we now present below algorithm CRLP based on the observations made about β and x_β.

Algorithm 1 CRLP($E_{\min}, \bar{x}$)

1. Solve Problem (1), let x^* be the optimal solution
2. Set $\beta = \frac{r^T x^* - E_{\min}}{r^T x^* - r^T \bar{x}}$
3. Set $x_\beta = (1-\beta)x^* + \beta\bar{x}$
4. **return**x_β (expected reward = $E_{\min}$, entropy based on $\beta\bar{x}$)

Algorithm CRLP is based on a linear program and thus obtains, in polynomial time, solutions to problem(4) with expected reward values $E_{\min} \in [\overline{E}, E^*]$. Note that Algorithm CRLP might unnecessarily constrain the solution set as Problem (4) implies that at least $\beta \sum_{a \in A} \bar{x}(s,a)$ flow has to reach each state s. This restriction may negatively impact the entropy it attains, as experimentally verified in Sect. 5. This concern is addressed by a reformulation of Problem (4) replacing the flow constraints by policy constraints at each stage. For a given $\beta \in [0, 1]$ and a solution $\bar{\pi}$ (policy calculated from $\bar{x}$), this replacement leads to the following linear program:

$$\begin{aligned} \max \quad & \sum_{s \in S}\sum_{a \in A} r(s,a)x(s,a) \\ \text{s.t.} \quad & \sum_{a \in A} x(j,a) - \sum_{s \in S}\sum_{a \in A} p(s,a,j)x(s,a) = \alpha_j, \quad \forall j \in S \\ & x(s,a) \geq \beta\bar{\pi}(s,a)\sum_{b \in A} x(s,b), \quad \forall s \in S, a \in A. \end{aligned} \tag{5}$$

For $\beta = 0$ this problem reduces to (1) returning E^*, for $\beta = 1$ it returns a maximal expected reward solution with the same policy as $\bar{\pi}$. This means that for β at values 0 and 1, problems (4) and (5) obtain the same solution if policy $\bar{\pi}$ is the policy obtained from the flow function $\bar{x}$. However, in the intermediate range of 0 to 1 for β, the policy obtained by problems (4) and (5) are different even if $\bar{\pi}$ is obtained from $\bar{x}$. Thus, theorem 2 holds for problem (4) but not for (5). We now present our BRLP algorithm 2.

Given input $\bar{x}$, algorithm BRLP runs in polynomial time, since at each iteration it solves an LP and for tolerance of ε, it takes at most $O\left(\frac{E(0)-E(1)}{\epsilon}\right)$ iterations to converge (E(0) and E(1) expected rewards correspond to 0 and 1 values of β).

Algorithm 2 BRLP($E_{\min}, \bar{x}$)

1. Set $\beta_l = 0$, $\beta_u = 1$, and $\beta = 1/2$.
2. Obtain $\bar{\pi}$ from $\bar{x}$
3. Solve Problem (5), let x_β and $E(\beta)$ be the optimal solution and expected reward value returned
4. **while** $|E(\beta) - E_{\min}| > \epsilon$ **do**
5. **if** $E(\beta) > E_{\min}$**then**
6. Set $\beta_l = \beta$
7. **else**
8. Set $\beta_u = \beta$
9. $\beta = \frac{\beta_u + \beta_l}{2}$
10. Solve Problem (5), let x_β and $E(\beta)$ be the optimal solution and expected reward value returned
11. **return**x_β (expected reward = $E_{\min} \pm \epsilon$,entropy related to $\beta\bar{x}$)

Throughout this paper, we set $\bar{x}$ based on uniform randomization $\bar{\pi} = 1/|A|$. By manipulating $\bar{x}$, we can accommodate the knowledge of the behavior of the adversary. For instance, if the agent knows that a specific state s cannot be targeted by the adversary, then $\bar{x}$ for that state can have all values 0, implying that no entropy constraint is necessary. In such cases, $\bar{x}$ will not be a complete solution for the MDP but rather concentrate on the sets of states and actions that are under risk of attack. For $\bar{x}$ that do not solve the MDP, Theorem 2 does not hold and therefore Algorithm CRLP is not valid. In this case, a high-entropy solution that meets a target expected reward can still be obtained via Algorithm BRLP.

4 Randomization using partial adversary model

In this section, we first describe the Bayesian–Stackelberg games, followed by our efficient approaches to obtain optimal randomized policies. As mentioned in Sect. 1, in the case that the leader has a partial model of the adversary, we use a Bayesian–Stackelberg games to represent the interaction between players. In a Stackelberg game, a leader commits to a strategy first, and then a follower (or group of followers) selfishly optimize their own rewards, *considering the action chosen by the leader*. To see the advantage of being the leader in a Stackelberg game, consider a simple game with the payoff table as shown in the table below. The leader is the row player and the follower is the column player.

	1	2
1	2, 1	4, 0
2	1, 0	3, 2

If we consider the above problem to be a simultaneous move game, then the only pure-strategy Nash equilibrium for this game is when the leader plays 1 and the follower plays 1 which gives the leader a payoff of 2; in fact, for the leader, playing 2 is strictly dominated. However, if the leader can commit to playing 2 before the follower chooses its strategy, then the leader will obtain a payoff of 3, since the follower would then play 2 to ensure a higher payoff for itself. If the leader commits to a uniform mixed strategy of playing 1 and 2 with equal (0.5) probability, then the follower will play 2, leading to a payoff for the leader of 3.5.

4.1 Exact solution: DOBSS

We developed an efficient exact procedure to generate an optimal leader strategy for security domains known as DOBSS. This method has two key advantages. First, it directly searches for an optimal strategy, rather than a Nash (or Bayes–Nash) equilibrium, thus allowing it to find high-reward non-equilibrium strategies. Second, the method expresses the Bayes–Nash game compactly without requiring conversion to a normal-form game.

The DOBSS procedure we propose operates directly on the compact Bayesian representation, without requiring the Harsanyi transformation. This is achieved because the different follower (robber) types are independent of each other. Hence, evaluating the leader strategy against a Harsanyi-transformed game matrix is equivalent to evaluating against each of the game matrices for the individual follower types. This independence property is exploited in DOBSS to yield a decomposition scheme. Also, note that DOBSS requires the solution of one optimization problem, rather than solving a series of problems as in the Multiple LPs method [6].

Note that for a single follower type, we simply take the mixed strategy for the leader that gives the highest payoff when the follower plays a reward-maximizing strategy. We need only to consider the reward-maximizing pure strategies of the followers, since for a given fixed strategy x of the leader, each follower type faces a problem with fixed linear rewards. If a mixed strategy is optimal for the follower, then so are all the pure strategies in the support of that mixed strategy.

We begin with the case of a single follower. Let the leader be the row player and the follower the column player. We denote by x the leader's policy, which consists of a vector of the leader's pure strategies. The value x_i is the proportion of times in which pure strategy i is used in the policy. Similarly, q denotes the vector of strategies of the follower. We also denote X and Q the index sets of the leader and follower's pure strategies, respectively. The payoff matrices R and C are defined such that R_{ij} is the reward of the leader and C_{ij} is the reward of the follower when the leader takes pure strategy i and the follower takes pure strategy j.

We first fix the policy of the leader to some policy x. We formulate the optimization problem the follower solves to find its optimal response to x as the following linear program:

$$\begin{aligned} \max_q \quad & \sum_{j\in Q}\sum_{i\in X} C_{ij}x_i\, q_j \\ \text{s.t.} \quad & \sum_{j\in Q} q_j = 1 \\ & q \geq 0. \end{aligned} \tag{6}$$

Thus, given the leader's strategy x, the follower's optimal response, $q(x)$, satisfies the LP optimality conditions:

$$a \geq \sum_{i\in X} C_{ij}x_i, \quad j \in Q$$

$$q_j\left(a - \sum_{i\in X} C_{ij}x_i\right) = 0 \quad j \in Q$$

$$\sum_{j\in Q} q_j = 1$$

$$q \geq 0.$$

Therefore the leader solves the following integer problem to maximize its own payoff, given the follower's optimal response $q(x)$:

$$\begin{aligned} \max_x \quad & \sum_{i\in X}\sum_{j\in Q} R_{ij}q(x)_j\, x_i \\ \text{s.t.} \quad & \sum_{i\in X} x_i = 1 \\ & x_i \in [0\ldots1]. \end{aligned} \tag{7}$$

Problem (7) maximizes the leader's reward with follower's best response, denoted by vector $q(x)$ for every leader strategy x. We complete this problem by including the characterization of q(x) through linear programming optimality conditions. The leader's problem becomes:

$$\begin{aligned}
\max_{x,q,a} \quad & \sum_{i\in X}\sum_{j\in Q} R_{ij}x_i q_j \\
\text{s.t.} \quad & \sum_i x_i = 1 \\
& \sum_{j\in Q} q_j = 1 \\
& 0 \le (a - \sum_{i\in X} C_{ij}x_i) \le (1-q_j)M \\
& x_i \in [0\ldots 1] \\
& q_j \in \{0,1\} \\
& a \in \Re. \qquad (8)
\end{aligned}$$

Here, M is some large constant and a is the follower's maximum reward value. The first and fourth constraints enforce a feasible mixed policy for the leader, and the second and fifth constraints enforce a feasible pure strategy for the follower. The third constraint enforces dual feasibility of the follower's problem (leftmost inequality) and the complementary slackness constraint for an optimal pure strategy q for the follower (rightmost inequality).

We now show how we can apply our decomposition technique on the MIQP to obtain significant speedups for Bayesian games with multiple follower types. To admit multiple adversaries in our framework, we modify the notation defined in the previous section to reason about multiple follower types. We denote by x the vector of strategies of the leader and q^l the vector of strategies of follower l, with L denoting the index set of follower types. We also denote by X and Q the index sets of leader and follower l's pure strategies, respectively. We also index the payoff matrices on each follower l, considering the matrices R^l and C^l.

Given a priori probabilities p^l, with $l \in L$ of facing each follower, the leader now faces the decomposed problem:

$$\begin{aligned}
\max_{x,q,a} \quad & \textstyle\sum_{i\in X}\sum_{l\in L}\sum_{j\in Q} p^l R^l_{ij} x_i q^l_j \\
\text{s.t.} \quad & \textstyle\sum_i x_i = 1 \\
& \textstyle\sum_{j\in Q} q^l_j = 1 \\
& 0 \le (a^l - \sum_{i\in X} C^l_{ij}x_i) \le (1-q^l_j)M \\
& x_i \in [0\ldots 1] \\
& q^l_j \in \{0,1\} \\
& a \in \Re \qquad (9)
\end{aligned}$$

Proposition 1 *Problem (9) for a Bayesian game with multiple follower types is equivalent to Problem (8) on the payoff matrices given by the Harsanyi transformation.*

Proof To show the equivalence we show that a feasible solution to (9) leads to a feasible solution to (8) of same objective value or better and vice-versa. This implies the equality in optimal objective value and the correspondence between optimal solutions.

Consider x, q^l, a^l with $l \in L$ a feasible solution to Problem (9). We now construct a feasible solution to (8). From its second constraint and integrality of q we have that for every l there is exactly one j_l such that $q^l_{jl} = 1$. Let j be the Harsanyi action that corresponds to $(j_1,\ldots,j_{|L|})$ and let q be its pure strategy (i.e. q is a strategy in the transformed game where $q_j = 1$, and $q_h = 0$ for all other $h \neq j$). We now show that the objective of (9) equals that of (8) exploiting these corresponding actions. In particular:

$$\begin{aligned}
\sum_{i\in X}\sum_{l\in L} p^l x_i \sum_{h\in Q} R^l_{ih} q^l_h &= \sum_{i\in X} x_i \sum_{l\in L} p^l R^l_{ij_l} \\
&= \sum_{i\in X} x_i R_{ij} = \sum_{i\in X}\sum_{h\in Q} x_i R_{ih} q_h
\end{aligned}$$

So now we just have to show that x, q, and $a = \sum_{l\in L} p^l a^l$ is feasible for Problem (8). Constraints 1, 2, 4, and 5 in (8) are easily satisfied by the proposed solution. Constraint 3 in (9) means that $\sum_{i\in X} x_i C^l_{ij_l} \ge \sum_{i\in X} x_i C^l_{ih}$, for every $h \in Q$ and $l \in L$, leading to

$$\sum_{i\in X} x_i C_{ij} = \sum_{l\in L} p^l \sum_{i\in X} x_i C^l_{ij_l} \ge \sum_{l\in L} p^l \sum_{i\in X} x_i C^l_{ih_l} = \sum_{i\in X} x_i C_{ih'},$$

for any pure strategy $h_1,\ldots,h_{|L|}$ for each of the followers and h' its corresponding pure strategy in the Harsanyi game. We conclude this part by showing that

$$\sum_{i\in X} x_i C_{ij} = \sum_{l\in L} p^l \sum_{i\in X} x_i C^l_{ij_l} = \sum_{l\in L} p^l a^l = a.$$

Now we start with (x, q, a) feasible for (8). This means that $q_j = 1$ for some pure action j. Let $(j_1,\ldots,j_{|L|})$ be the corresponding actions for each follower l. We show that x, q^l with $q^l_{j_l} = 1$ and $q^l{}_h = 0$ for $h \neq j_l$, and $a^l = \sum_{i\in X} x_i C^l_{ij_l}$ with $l \in L$ is feasible for (9). By construction this solution satisfies constraints 1, 2, 4, 5 and has a matching objective function. We now show that constraint 3 holds by showing that $\sum_{i\in X} x_i C^l_{ij_l} \ge \sum_{i\in X} x_i C^l_{ih}$ for all $h \in Q$ and $l \in L$. Let us assume it does not. That is, there is an $\hat{l} \in L$ and $\hat{h} \in Q$ such that $\sum_{i\in X} x_i C^{\hat{l}}_{ij_{\hat{l}}} < \sum_{i\in X} x_i C^{\hat{l}}_{i\hat{h}}$. Then by multiplying by $p^{\hat{l}}$ and adding $\sum_{l\neq\hat{l}} p^l \sum_{i\in X} x_i C^l_{ij_l}$ to both sides of the inequality we obtain

$$\sum_{i\in X} x_i C_{ij} < \sum_{i\in X} x_i \left(\sum_{l\neq\hat{l}} p^l C^l_{ij_l} + p^{\hat{l}} C^{\hat{l}}_{i\hat{h}} \right).$$

The right hand side equals $\sum_{i\in X} x_i C_{ih}$ for the pure strategy h that corresponds to $(j_1,\ldots,\hat{h},\ldots,j_{|L|})$, which is a contradiction since constraint 3 of (8) implies that $\sum_{i\in X} x_i C_{ij} \ge \sum_{i\in X} x_i C_{ih}$ for all h. □

We can then linearize the quadratic programming problem (9) through the change of variables $z^l_{ij} = x_i q^l_j$, obtaining the following problem:

$$\begin{aligned}
\max_{q,z,a} \quad & \sum_{i\in X}\sum_{l\in L}\sum_{j\in Q} p^l R^l_{ij} z^l_{ij} \\
\text{s.t.} \quad & \sum_{i\in X}\sum_{j\in Q} z^l_{ij} = 1 \\
& \textstyle\sum_{j\in Q} z^l_{ij} \le 1 \\
& q^l_j \le \textstyle\sum_{i\in X} z^l_{ij} \le 1 \\
& \textstyle\sum_{j\in Q} q^l_j = 1 \\
& 0 \le (a^l - \textstyle\sum_{i\in X} C^l_{ij}(\sum_{h\in Q} z^l_{ih})) \le (1 - q^l_j)M \\
& \textstyle\sum_{j\in Q} z^l_{ij} = \sum_{j\in Q} z^1_{ij} \\
& z^l_{ij} \in [0\ldots1] \\
& q^l_j \in \{0,1\} \\
& a \in \Re
\end{aligned} \tag{10}$$

Theorem 3 *Problems (9) and (10) are equivalent.*

Proof Consider x, q^l, a^l with $l \in L$ a feasible solution of (9). We will show that q^l, a^l, $z^l_{ij} = x_i q^l_j$ is a feasible solution of (10) of same objective function value. The equivalence of the objective functions, and constraints 4, 7 and 8 of (10) are satisfied by construction. The fact that $\sum_{j\in Q} z^l_{ij} = x_i$ as $\sum_{j\in Q} q^l_j = 1$ explains constraints 1, 2, 5 and 6 of (10). Constraint 3 of (10) is satisfied because $\sum_{i\in X} z^l_{ij} = q^l_j$.

Lets now consider q^l, z^l, a^l feasible for (10). We will show that q^l, a^l and $x_i = \sum_{j\in Q} z^1_{ij}$ are feasible for (9) with the same objective value. In fact all constraints of (9) are readily satisfied by construction. To see that the objectives match, notice for each l one $q^l{}_j$ must equal 1 and the rest equal 0. Let us say that $q^l{}_{j\_l} = 1$, then the third constraint in (10) implies that $\sum_{i\in X} z^l_{ij_l} = 1$. This condition and the first constraint in (7) give that $z^l{}_{ij} = 0$ for all $i \in X$ and all $j \neq j_l$. In particular this implies that

$$x_i = \sum_{j\in Q} z^1_{ij} = z^1_{ij_1} = z^l_{ij_l},$$

the last equality from constraint 6 of (10). Therefore $x_i q^l_j = z^l_{ij_l} q^l_j = z^l_{ij}$. This last equality is because both are 0 when $j \neq j_l$ (and $q^l{}_j = 1$ when $j = j_l$). This shows that the transformation preserves the objective function value, completing the proof. □

We can therefore solve this equivalent linear integer program with efficient integer programming packages which can handle problems with thousands of integer variables. We implemented the decomposed MILP and the results are shown in the following section.

We now provide a brief intuition into the computational savings provided by our approach. We compare the work done by DOBSS and by the other exact solution approach for Bayesian–Stackelberg games, namely the Multiple-LPs method by [6]. The DOBSS method achieves an exponential reduction in the problem that must be solved over the multiple-LPs approach due to the following reasons: The multiple-LPs method solves an LP over the exponentially blown Harsanyi transformed matrix for each joint strategy of the adversaries (also exponential in number). In contrast, DOBSS solves a problem that has one integer variable per strategy for every adversary.

To be more precise, let X be the number of agent actions, Q the number of adversary actions and L the number of adversary types. The DOBSS procedure solves a MILP with XQL continuous variables, QL binary variables, and 4QL + 2XL + 2L constraints. We note that this MILP has Q^L feasible integer solutions, due to constraint 4 in (10). Solving this problem with a judicious branch and bound procedure will lead in the worst case to a tree with $O(Q^L)$ nodes each requiring the solution of an LP of size $O(XQL)$. Here the size of an LP is the number of variables + number of constraints.

On the other hand the multiple-LPs method needs the Harsanyi transformation. This transformation leads to a game where the agent can take X actions and the joint adversary can take Q^L actions. This method then solves exactly Q^L different LPs, each with X variables and Q^L constraints, i.e. each LP is of size $O(X + Q^L)$.

In summary, both methods require the solution to about $O(Q^L)$ linear programs, however these are of size $O(XQL)$ for DOBSS while they are of size $O(X + Q^L)$ for Multiple LPs. This exponential increase in the problem size would lead to much higher computational burden for Multiple LPs as the number of adversaries increases. We note also that the branch-and-bound procedure seldom explores the entire tree as it uses the bounding procedures to discard sections of the tree which are provably non-optimal. The multiple-LPs method on the other hand must solve all Q^L problems.

4.2 Approximate solution: ASAP

We now present our limited randomization approach introduced in [15], where we limit the possible mixed strategies of the leader to select actions with probabilities that are integer multiples of 1/k for a predetermined integer k. One advantage of such strategies is that they are compact to represent (as fractions) and simple to understand; therefore they can potentially be efficiently implemented in real patrolling applications. Thus for example, when k = 3, we can have a mixed strategy where strategy 1 is picked twice i.e., probability = 2/3 and strategy 2 is picked once with probability = 1/3. We now present our ASAP algorithm using mathematical framework developed in the previous section. In particular we start with problem 9 and convert x from continuous to an integer variable that varies between 0 to k; thus obtaining the following problem:

$$
\begin{aligned}
\max_{x,q,a} \quad & \sum_{i\in X}\sum_{l\in L}\sum_{j\in Q}\frac{p^l}{k}R^l_{ij}x_i q^l_j \\
\text{s.t.} \quad & \sum_i x_i = k \\
& \sum_{j\in Q} q^l_j = 1 \\
& 0 \le (a^l - \sum_{i\in X}\tfrac{1}{k}C^l_{ij}x_i) \le (1-q^l_j)M \\
& x_i \in \{0,1,\ldots,k\} \\
& q^l_j \in \{0,1\} \\
& a^l \in \Re
\end{aligned} \tag{11}
$$

We then linearize problem (11) through the change of variables $z^l_{ij} = x_i q^l_j$, obtaining the following equivalent MILP:

$$
\begin{aligned}
\max_{q,z,a} \quad & \sum_{i\in X}\sum_{l\in L}\sum_{j\in Q}\frac{p^l}{k}R^l_{ij}z^l_{ij} \\
\text{s.t.} \quad & \sum_{i\in X}\sum_{j\in Q} z^l_{ij} = k \\
& \sum_{j\in Q} z^l_{ij} \le k \\
& kq^l_j \le \sum_{i\in X} z^l_{ij} \le k \\
& \sum_{j\in Q} q^l_j = 1 \\
& 0 \le \left(a^l - \sum_{i\in X}\tfrac{1}{k}C^l_{ij}\left(\sum_{h\in Q} z^l_{ih}\right)\right) \le (1-q^l_j)M \\
& \sum_{j\in Q} z^l_{ij} = \sum_{j\in Q} z^1_{ij} \\
& z^l_{ij} \in \{0,1,\ldots,k\} \\
& q^l_j \in \{0,1\} \\
& a^l \in \Re
\end{aligned} \tag{12}
$$

Unfortunately, although ASAP was designed to generate simple policies, the fact that it has so many more integer variables makes it a more challenging problem than DOBSS. In fact, as we present in the next section, our computational results show that solution times for DOBSS and ASAP are comparable, when ASAP finds an optimal solution. However, ASAP can experience difficulty in finding a feasible solution for large problems. This added difficulty makes DOBSS the method of choice.

5 Experimental results

5.1 No adversary model

Our first set of experiments examine the tradeoffs in runtime, expected reward and entropy for single-agent problems. Figure 1 shows the results based on generation of MDP policies for 10 MDPs. These experiments compare the performance of our four methods of randomization for single-agent policies. In the figures, *CRLP* refers to Algorithm 1; *BRLP* refers to Algorithm 2; whereas $H_W(x)$ and $H_A(x)$ refer to Problem 3 with these objective functions. The top graph examines the tradeoff between entropy and expected reward thresholds. It shows the average weighted entropy on the y-axis and reward threshold percent on the x-axis. The average maximally obtainable entropy for these MDPs is 8.89 (shown by line on the top) and three of our four methods (except CRLP) attain it at about 50% threshold, i.e. an agent can attain maximum entropy if it is satisfied with 50% of the maximum expected reward. However, if no reward can be sacrificed (100% threshold) the policy returned is deterministic.

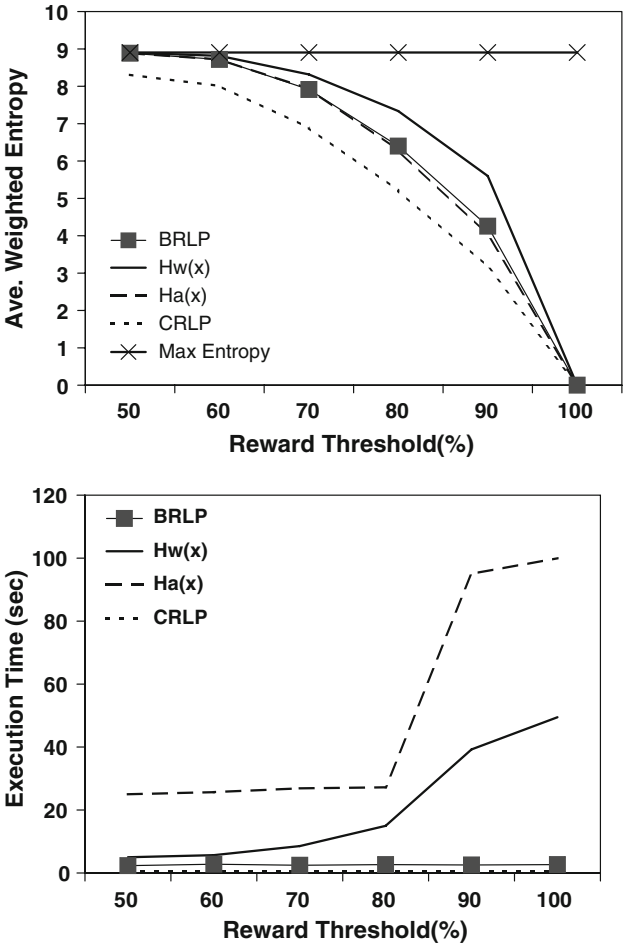

Fig. 1 Comparison of single agent algorithms

Figure 1 also shows the run-times, plotting the execution time in seconds on the y-axis, and expected reward threshold percent on the x-axis. These numbers represent averages over the same 10 MDPs. Algorithm CRLP is the fastest and its runtime is very small and remains constant over the whole range of threshold rewards as seen from the plot. Algorithm BRLP also has a fairly constant runtime and is slightly slower than CRLP. Both CRLP and BRLP are based on linear programs and hence their small and fairly constant runtimes. Problem 3, for both $H_A(x)$ and $H_W(x)$ objectives,

exhibits an increase in the runtime as the expected reward threshold increases. This trend can be attributed to the fact that maximizing a non-concave objective while simultaneously attaining feasibility becomes more difficult as the feasible region shrinks.

We conclude the following from Fig. 1: (i) CRLP is the fastest but provides lowest entropy. (ii) BRLP is significantly faster than Problem [16], providing 7-fold average speedup over the 10 MDPs over the entire threshold range. (iii) Problem 3 with $H_W(x)$ provides highest entropy among our methods, but the average gain in entropy is only 10% over BRLP. (iv) CRLP provides a 4-fold speedup on an average over BRLP but with a significant entropy loss of about 18%. In fact, CRLP is unable to reach maximal possible entropy for the threshold range considered in the plot.

5.2 Partial adversary model

We performed experiments on a patrolling domain where the police patrol various number of houses as presented in [16]. The domain is then modeled as a Bayesian–Stackelberg games consisting of two players: the security agent (i.e. the patrolling robot or the leader) and the robber (the follower) in a world consisting of m houses, $1 \ldots m$. The security agent's set of pure strategies consists of possible routes of d houses to patrol (in some order). The security agent can choose a mixed strategy so that the robber will be unsure of exactly where the security agent may patrol, but the robber will know the mixed strategy the security agent has chosen. With this knowledge, the robber must choose a single house to rob, although the robber generally takes a long time to rob a house. If the house chosen by the robber is not on the security agent's route, then the robber successfully robs the house. Otherwise, if it is on the security agent's route, then the earlier the house is on the route, the easier it is for the security agent to catch the robber before he finishes robbing it.

The payoffs are modeled with the following variables:

- $v_{y,x}$: value of the goods in house y to the security agent.
- $v_{y,q}$: value of the goods in house y to the robber.
- c_x: reward to the security agent of catching the robber.
- c_q: cost to the robber of getting caught.
- p_y: probability that the security agent can catch the robber at the yth house in the patrol $(p_y < p_{y'} \iff y' < y)$.

The security agent's set of possible pure strategies (patrol routes) is denoted by X and includes all d-tuples $i = < w_1, w_2, \ldots, w_d >$. Each of $w_1 \ldots w_d$ may take values 1 through m (different houses), however, no two elements of the d-tuple are allowed to be equal (the agent is not allowed to return to the same house). The robber's set of possible pure strategies (houses to rob) is denoted by Q and includes all integers $j = 1 \ldots m$. The payoffs (security agent, robber) for pure strategies i, j are:

- $-v_{y,x}$, $v_{y,q}$, for $j = l \notin i$.
- $p_y c_x + (1 - p_y)(-v_{y,x})$, $\quad -p_y c_q + (1 - p_y)(v_{y,q})$, $\quad$ for $j = y \in i$.

With this structure it is possible to model many different types of robbers who have differing motivations; for example, one robber may have a lower cost of getting caught than another, or may value the goods in the various houses differently. We performed our experiments using four methods for generating the security agent's strategy: DOBSS method for finding the optimal solution [16], ASAP procedure that provides best policies with limited randomization [15], the multiple-LPs method presented in [6] that provides optimal policies and the MIP-Nash procedure [21] for finding the best Bayes–Nash equilibrium. The multiple-LPs method and the MIP-Nash procedure require a normal-form game as input, and so the Harsanyi transformation is required as an initial step. We do not record this preprocessing time here thus giving those other methods an advantage.

Figure 2 shows the runtime results for all the four methods for two, three and four houses. Each runtime value in the graph(s) corresponds to an average over twenty randomly generated scenarios. The x-axis shows the number of follower types the leader faces starting from 1 to 14 adversary types and the y-axis of the graph shows the runtime in seconds on log-scale ranging from .01 to 10000 s. The choice of .01 to 10000 is for convenience of representation of log scale(with base 10). All the experiments that were not concluded in 30 min (1800 s) were cut off.

From the runtime graphs we conclude that the DOBSS and ASAP methods outperform the multiple-LPs and MIP-Nash methods with respect to runtime. We modeled a maximum of fourteen adversary types for all our domains. For the domain with two houses, while the MIP-Nash and multiple-LPs method needed about 1000 s for solving the problem with fourteen adversary types, both the DOBSS and ASAP provided solutions in less than 0.1 s. Note that DOBSS provided the optimal solution while ASAP provided the best possible solution with randomization constraints. These randomization constraints also sometimes cause ASAP to *incorrectly* claim solutions to be infeasible, the details of which are presented later on in this section. The runtime for ASAP in all our results is taken as either the time needed to generate an optimal solution or to determine that no feasible solution exists.

The first graph in Fig. 2 shows the trends for all these four methods for the domain with two houses. While the runtimes of DOBSS and ASAP show linear increase in runtimes, the other two show an exponential trend. The

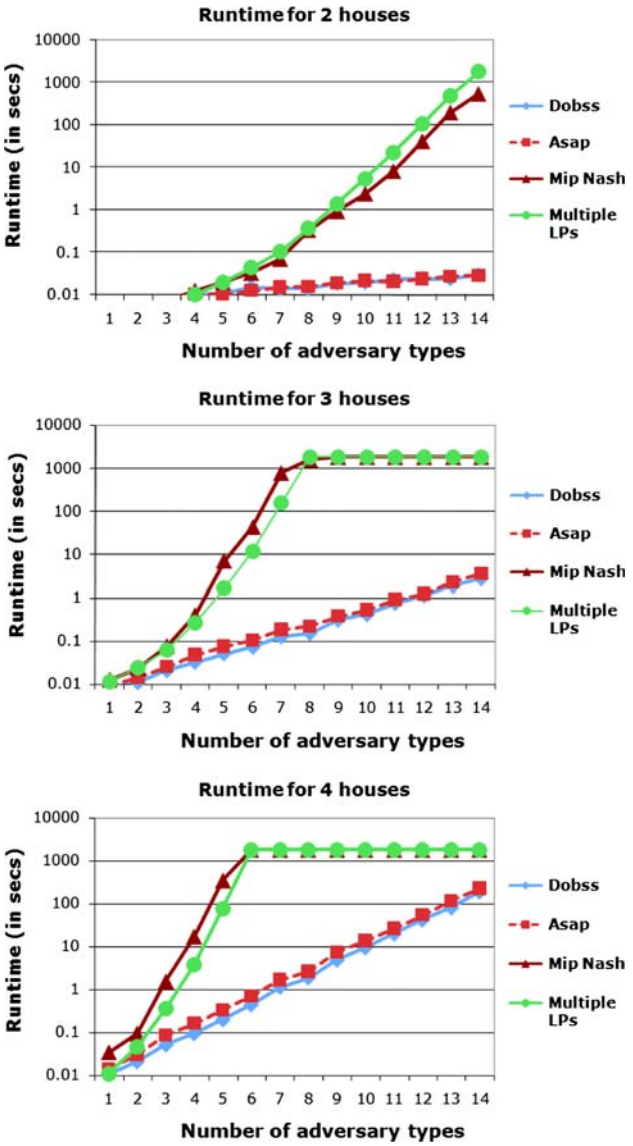

Fig. 2 Runtimes (plotted on log scale): DOBSS, ASAP, MIP-Nash and multiple–LP methods

runtimes of DOBSS and ASAP are themselves exponential since they show a linear increase when plotted on a log-scale graph. Furthermore, they have an exponential speedup over the other two procedures as seen in the graph.

The second graph in Fig. 2 presents results for the domain having three houses. Both the MIP-Nash and multiple-LPs could solve this problem only till seven adversary types within the cutoff time of 1800 s whereas DOBSS and ASAP could solve the problem for all the fourteen adversary types modeled under 10 s. (The cutoff of 1800 s is also the reason MIP-Nash and multiple-LPs appear to have a constant run-time beyond seven adversary types.) Similar trends can be observed in the third graph with a domain of four houses where MIP-Nash and multiple-LPs could solve only till 5 adversary types whereas

	2	3	4	5	6	7
1	0	0	0	0	0	10
2	0	0	0	0	5	0
3	0	5	5	15	5	10
4	0	10	15	20	20	20
5	0	10	10	10	5	10
6	0	10	10	10	15	15
7	0	15	20	15	10	20
8	0	5	15	10	30	45
9	0	5	10	15	20	30*
10	5	10	20	5	35*	35*
11	0	5	20	10	35*	40*
12	0	10	20	10	30*	30*
13	0	20	20	25*	25*	0
14	0	20	20	30*	20*	20*

Fig. 3 Percent of infeasible solutions for ASAP. Rows represent # of adversary types (1–14), columns represent # of houses (2–7)

DOBSS and ASAP could solve till fourteen adversary types within 400 s for DOBSS and 500 s for ASAP. From this set of three graphs, we conclude that DOBSS and ASAP outperform the other two procedures, by an exponential margin.

Between DOBSS and ASAP, DOBSS is our procedure of choice since ASAP suffers from problems of infeasibility. Therefore, we present our second set of experimental results in Fig. 3 to highlight the infeasibility issue of the ASAP procedure. In this experiment, the same settings as described above were used. The number of houses was varied between two to seven (columns in the table) and the number of adversary types was varied between one to fourteen (rows in the table). For each fixed number of houses and follower types, twenty scenarios were randomly generated. We ran the ASAP procedure and presented the number of infeasible solutions obtained, as a percentage of all the scenarios tested for each of the fixed number of houses and adversary types. For example, with the 8th adversary type (row numbered 8) and 4 houses (column numbered 4) scenario, ASAP generates 15% infeasible solutions. Note that for the values marked with a star the percentage presented in the table represents an upper bound on the number of infeasible scenarios. In these starred scenarios the ASAP procedure ran out of time in many instances. When ASAP ran out of time, it either indicated infeasibility, in which case it was classified as infeasible solution making it an upper bound (since there might be feasible solution when sufficient time is provided); or it indicated that there was a feasible solution even though it has not found the optimal yet, in which case it was obviously not marked as infeasible.

We make the following conclusions about ASAP from the table in Fig. 3: (a) In general, given a fixed number of

houses, as the number of adversary types increase (i.e from 1 to 14) the percentage of infeasible solutions increase(as we go down the columns). (b) Given a fixed number of adversary types, as the number of houses increase, the percentage of infeasible solutions increase(as we go across the rows). Although there are exceptions to both the conclusions, the general trend is that as the problem size increases (due to increase in either houses or adversary types or both) ASAP tends to generate more infeasible solutions thus making it unsuitable for bigger problems. From the table we obtain that more than 12.5% of the solutions are infeasible for the five house problem when averaged over all the adversary scenarios. This number increases to as high as 18% and 20% on an average for the six and seven house problems. If we perform similar calculations over the last five adversary scenarios i.e., when the number of adversary types are varied from 10 to 14, we obtain 16%, 29% and 25% respectively for the five, six and seven house scenarios. This shows that the ASAP produces more infeasible solutions as the problem size increases. Furthermore, there is no procedure to determine if ASAP will generate a infeasible solution until runtime, thus making the ASAP approach impractical.

6 Conclusions and policy implications

In this paper we present our recent work on algorithms for secure patrols in adversarial domains. We follow two different approaches based on what is known about the adversary. When there is no information about the adversary, we provide for random policy generation using MDPs. When there is partial information available about the adversary, we model our domain as a Bayesian–Stackelberg games and provide efficient MIP formulations for it. We also present experimental results to show that our techniques provide optimal, secure policies. Thus, this work represents a significant advance in the state of the art in addressing security domains.

We note that these two types of models are related. In fact, a randomized solution that reduces the information made available to the adversary can be the optimal Stackelberg strategy for the leader given a balanced adversary reward matrix. We illustrate this with the following simple example, where the optimal Stackelberg solution turns out to be the (minimal information) uniformly random solution. Consider a zero-sum square game where the payoff matrix for the agent gives 1 in the diagonal and -1 everywhere else. Note that for this problem, when the agent decides among a set of actions, the maximum entropy solution is also the uniform strategy. Given a strategy x for the leader in this example, the payoff the leader gets if adversary chooses action j is $x_j - (x_1 + \cdots + x_{j-1} + x_{j+1} + \cdots + x_n) = 2\,x_j - 1$ since x adds up to 1. Since the adversary receives a payoff $= 1 - 2x_j$ (a zero sum game) for this action, the adversary will select the action which does the following: $\max_{j=1\ldots n}\ 1 - 2x_j$, giving the leader a reward of $\min_{j=1\ldots n}\ 2x_j - 1$. The leader therefore must maximize $\min_{j=1\ldots n}\ 2x_j - 1$ when selecting x which is done for the uniform x. Thus, we conclude that uniform randomization, which is the maximum entropy solution, is obtained as a Stackelberg game solution with appropriate payoffs for the agent and adversary. In general the uniformly random solution is not the maximum entropy solution, however this simple example helps illustrates the connection between these two types of models.

There are number of different conclusions and policy implications from this work, for example: (a) Randomization decreases predictability and the information given out to the adversary and hence increases security for many problems. The increase security comes at the expense of the efficiency of the solution, where a patrol's efficiency refers to the value of some alternate measure, such as resources consumed, cost, or coverage for example. (b) When there is no model of the adversary or there is great uncertainty about the adversary information, the rational choice for the agent is to patrol following the maximally random solution while ensuring that some efficiency constraints are met. (c) When security planners have a reasonable, possibly partial, model of the adversaries actions and rewards, then it is natural to represent the relation between security and adversaries using game-theoretic formulations. (d) Both the decision-theoretic and game-theoretic approaches return a probability distribution over a set of actions, which can depend on the state in the case of MDPs. Actual patrolling schedules are obtained by sampling from these optimal probability distributions. This procedure on the long run will generate a patrolling policy that approximates the optimal probability distributions over actions. (e) We assume the optimal probability distributions are known for the adversaries because they could observe samples over time to estimate it. In addition we assume that the adversaries respond optimally to this information, either maximizing their reward (in the game-theoretic approach) or by some unspecified means (in the decision-theoretic approach). The realistic situation in which adversaries do not have all the information or do not behave rationally are the topic of current research. (f) The algorithms presented here are general-purpose and can be tailored to improve security at many real-world targets, such as airports, dams, museums, and stadiums. In fact, the DOBSS formulation is used in the real-world security scheduler ARMOR, which is in deployment at the LAX airport since August 2007 [11, 17]. The implementation of this methodology to other domains is the subject of ongoing work.

Acknowledgments This research is supported by the United States Department of Homeland Security through Center for Risk and Economic Analysis of Terrorism Events (CREATE). This work was supported in part by NSF grant no. IIS0705587 and ISF.

References

1. R. Beard, T. McLain, Multiple UAV cooperative search under collision avoidance and limited range communication constraints, in *Proceedings of the 42nd IEEE Conference on Decision and Control*, vol. 1, pp. 25–30 (2003)
2. N. Borisov, J. Waddle, *Anonymity in Structured Peer-to-peer Networks*. University of California, Berkeley, Technical Report No. UCB/CSD-05-1390 (2005)
3. G. Brown, M. Carlyle, J. Salmeron, K. Wood, Defending critical infrastructures. Interfaces **36**(6), 530–544 (2006)
4. J. Brynielsson, S. Arnborg, Bayesian games for threat prediction and situation analysis, in *Proceedings of the Seventh International Conference on Information Fusion*, vol. 2, pp. 1125–1132 (2004)
5. D.M. Carroll, C. Nguyen, H. Everett, B. Frederick, *Development and Testing for Physical Security Robots*. http://www.nosc.mil/robots/pubs/spie5804-63.pdf (2005)
6. V. Conitzer, T. Sandholm, Computing the optimal strategy to commit to, in *Proceedings of the 7th ACM Conference on Electronic Commerce*, pp. 82–90 (2006)
7. D. Dolgov, E. Durfee, Approximating optimal policies for agents with limited execution resources, in *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, AAAI Press, pp. 1107–1112 (2003)
8. D. Dolgov, E. Durfee, *Constructing Optimal Policies for Agents with Constrained Architectures*. Technical report, University of Michigan, 2003.
9. D. Fudenberg, J. Tirole, *Game Theory* (MIT Press, 1991)
10. J.C. Harsanyi, R. Selten, A generalized Nash solution for two-person bargaining games with incomplete information. Manag. Sci. **18**(5), 80–106 (1972)
11. A. Murr, Random checks, in *Newsweek National News*. http://www.newsweek.com/id/43401. Accessed 28 September 2007
12. C. Ozturk, Y. Zhang, W. Trappe, Source-location privacy in energy-constrained sensor network routing, in *Proceedings of the 2nd ACM Workshop on Security of ad hoc and Sensor Networks*, pp. 88–93 (2004)
13. P. Paruchuri, M. Tambe, F. Ordonez, S. Kraus, Towards a formalization of teamwork with resource constraints, in *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 596–603 (2004)
14. P. Paruchuri, M. Tambe, F. Ordonez, S. Kraus, Security in multiagent systems by policy randomization, in *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 273–280 (2006)
15. P. Paruchuri, J.P. Pearce, M. Tambe, F. Ordonez, S. Kraus, An efficient heuristic approach for security against multiple adversaries, in *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multiagent Systems*, Article No. 181 (2007)
16. P. Paruchuri, J.P. Pearce, J. Marecki, M. Tambe, F. Ordonez, S. Kraus, Playing games for security: an efficient exact algorithm for solving Bayesian Stackelberg games, in *Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multiagent Systems*, vol. 2, pp. 895–902 (2008)
17. J. Pita, M. Jain, J. Marecki, F. Ordonez, C. Portway, M. Tambe, C. Western, P. Paruchuri, S. Kraus, Deployed ARMOR protection: the application of a game theoretic model for security at the Los Angeles international airport, in *Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multiagent Systems, Industry Track*, pp. 125–132 (2008)
18. R.W. Poole, G. Passantino, A risk based airport security policy, Policy Study No. 308, Reason Foundation, pp. 20–21 (2003)
19. M. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming* (Wiley, 1994)
20. T. Roughgarden, Stackelberg scheduling strategies, in *Proceedings of the 33rd Annual ACM Symposium on the Theory of Computing*, pp. 104–113 (2001)
21. T. Sandholm, A. Gilpin, V. Conitzer, Mixed-integer programming methods for finding Nash equilibria, in *Proceedings of the 20th National Conference on Artificial Intelligence*, pp. 495–501 (2005)
22. C.E. Shannon, A mathematical theory of communication. Bell Labs Tech. J. **27**, 379–423 and 623–656 (1948)
23. H.V. Stackelberg, *Marketform und Gleichgewicht* (Springer, Vienna, 1934)