# Keep the Adversary Guessing: Agent Security by Policy Randomization

by

Praveen Paruchuri

---

A Dissertation Presented to the
FACULTY OF THE GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements for the Degree
DOCTOR OF PHILOSOPHY
(COMPUTER SCIENCE)

May 2007

## Dedication

This dissertation is dedicated to my parents and my sister.

## Acknowledgements

I would like to thank all the people who have helped me complete my thesis.

First and foremost, I would like to thank my advisor, Milind Tambe for his attention, guidance, and support at every step of this thesis. He served as an excellent role model and source of valuable advice. I would also like to thank Sarit Kraus and Fernando Ordonez for being my collaborators since the time beginning my PhD. Their valuable guidance and support made my journey through the PhD program very memorable.

I wish to thank Gaurav S. Sukhatme, Stacy Marsella and Leana Golubchik for being on my thesis committee. Their valuable insights and comments were instrumental in structuring my dissertation. I am grateful to all the members of the TEAMCORE research group for being my friends and collaborators. Gal Kaminka, David Pynadath, Paul Scerri, Hyuckchul Jung, Jay Modi, Ranjit Nair, Rajiv Maheswaran, Nathan Schurr, Pradeep Varakantham, Jonathan Pearce, Emma Bowring, Janusz Marecki, Tapana Gupta and Zvi Topol have always been very helpful and supportive. I would like to thank all my collaborators for their numerous stimulating discussions that helped in shaping my thesis.

I would also like to thank all my friends and roommates who have made my stay at USC a memorable experience. Lastly and most importantly, I would like to express my gratitude to my

family. In particular, I would like to thank my parents and sister for believing in me and pushing me to get a doctorate degree.

## Abstract

Recent advances in the field of agent/multiagent systems brings us closer to agents acting in real world domains, which can be uncertain and many times adversarial. Security, commonly defined as the ability to deal with intentional threats from other agents is a major challenge for agents or agent-teams deployed in these adversarial domains. Such adversarial scenarios arise in a wide variety of situations that are becoming increasingly important such as agents patrolling to provide perimeter security around critical infrastructure or performing routine security checks. These domains have the following characteristics: (a) The agent or agent-team needs to commit to a security policy while the adversaries may observe and exploit the policy committed to. (b) The agent/agent-team potentially faces different types of adversaries and has varying information available about the adversaries (thus limiting the agents' ability to model its adversaries).

To address security in such domains, I developed two types of algorithms. First, when the agent has no model of its adversaries, my key idea is to randomize agent's policies to minimize the information gained by adversaries. To that end, I developed algorithms for policy randomization for both the Markov Decision Processes (MDPs) and the Decentralized-Partially Observable MDPs (Dec POMDPs). Since arbitrary randomization can violate quality constraints (for example, the resource usage should be below a certain threshold or key areas must be patrolled with a certain frequency), my algorithms guarantee quality constraints on the randomized policies

generated. For efficiency, I provide a novel linear program for randomized policy generation in MDPs, and then build on this program for a heuristic solution for Dec-POMDPs. Second, when the agent has partial model of the adversaries, I model the security domain as a Bayesian Stackelberg game where the agent's model of the adversary includes a probability distribution over possible adversary types. While the optimal policy selection for a Bayesian Stackelberg game is known to be NP-hard, my solution approach based on an efficient Mixed Integer Linear Program (MILP) provides significant speedups over existing approaches while obtaining the optimal solution. The resulting policy randomizes the agent's possible strategies, while taking into account the probability distribution over adversary types. Finally, I provide experimental results for all my algorithms, illustrating the new techniques developed have enabled us to find optimal secure policies efficiently for an increasingly important class of security domains.

# Table of Contents

# List Of Figures

# List Of Tables

# 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 [Sasemas, 2005]. My thesis focuses 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 [Carroll et al., 2005; Billante, 2003; Lewis et al., 2005], UAV's monitoring a humanitarian mission [Beard and McLain, 2003], agents assisting in routine security checks at airports [Poole and Passantino, 2003], agents providing privacy in sensor network routing [Ozturk et al., 2004], realistic game playing bots such as in unreal tournament [Smith et al., 2007] or agents maintaining anonymity in peer to peer networks [Borisov and Waddle, 2005].

In my thesis, I address the problem of planning for agents acting in such uncertain environments while facing security challenges. The common assumption in these security domains is that the agent commits to a plan or policy first while the adversary can observe 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 incomplete information about the adversaries. For example, in a typical security domain such as the patrol 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. Alternatively, via eavesdropping or other means, the adversaries may come to know an agent's patrolling plan. Furthermore, the agents might not know which adversaries exist and even if they know, they still have uncertain information about which adversary strikes at what time or region. To solve this problem with incomplete information about the adversaries, I provide efficient security algorithms broadly considering two realistic situations: Firstly, when the agents have no model of their adversaries, we wish to minimize the information the adversary has about the agent by randomizing the agent's policies. Secondly, when the agents have a partial model of their adversaries we maximize the agent's expected rewards while accounting for the uncertainty in the adversary information.

When the agents have no model of their adversaries, I provide efficient algorithms for generating randomized plans or policies for the agents that minimize the information that can be gained by adversaries. In the rest of my thesis I will refer to such randomized policies that attempt to minimize the opponent's information gain as *secure policies*. However, arbitrary randomization can violate quality constraints such as: resource usage should be below a certain threshold or key areas must be patrolled with a certain frequency. To that end, I developed algorithms for efficient policy randomization with quality guarantees. Markovian models such as the Markov Decision Problems (MDPs) [Puterman, 1994], Partially Observable Markov Decision Problems (POMDPs) [Cassandra et al., 1994] and Decentralized POMDPs (Dec-POMDPs) [Pynadath and Tambe, 2002; Nair et al., 2003; Emery-Montemerlo et al., 2004] provide general frameworks for reasoning about the environmental uncertainty while enabling development of mathematical frameworks to quantify the randomization of policies. My contributions for the development of efficient randomized policy generation algorithms using these frameworks are as follows:

- I provide novel techniques that enable policy randomization in single agents, while attaining a certain expected reward threshold. I measure randomization via an entropy-based metric (although my techniques are independent of that metric). In particular, I illustrate that simply maximizing entropy-based metrics introduces a non-linear program that has non-polynomial run-time. Hence, I introduce my CRLP (Convex combination for Randomization) and BRLP (Binary search for Randomization) linear programming (LP) techniques that randomize policies in polynomial time with different tradeoffs as explained later.

- I provide a new algorithm, RDR (Rolling Down Randomization), for generating randomized policies for decentralized POMDPs without communication, given a threshold on the expected team reward loss. RDR starts with a joint deterministic policy for decentralized POMDPs, then iterates, randomizing policies for agents turn-by-turn, keeping policies of all other agents fixed. A key insight in RDR is that given fixed randomized policies for other agents, I can generate a randomized policy via the CRLP or BRLP methods, i.e., my efficient single-agent methods.

When the agents have partial model of their adversaries, I model the security domain as a Bayesian Stackelberg game [Conitzer and Sandholm, 2006]. For example, in some patrol domains, the patrol agents may need to provide security taking into account the partial information available about the adversary. Here, the agents know the adversary's actions and payoffs but does not know which adversary is active at a given time. A common approach for choosing policy of the agents in such scenarios is to model them as Bayesian games [Fudenberg and Tirole, 1991]. 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. 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 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 [Fudenberg and Tirole, 1991] and are commonly used to model attacker-defender scenarios in security domains [Brown et al., 2006]. Thus, given that multiple agent types exist and the notion of a security agent committing first before the adversaries, I model my security domains as Bayesian Stackelberg games. 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. My contributions for the development of efficient solution techniques for Bayesian Stackelberg games are as follows:

- First, I introduced an efficient technique for generating optimal leader strategies with limited randomization for Bayesian Stackelberg games, known as ASAP (Agent Security via Approximate Policies). The advantage of this procedure is that the policies generated are simple and easy to implement in real scenarios since the amount of randomization is controlled. I sometimes refer to this in my thesis as a heuristic to differentiate it from the exact procedure described next.

- Second, I developed an efficient exact procedure, known as DOBSS (Decomposed Optimal Bayesian Stackelberg Solver) using the mathematical framework developed for ASAP.

I developed an efficient Mixed Integer Linear Program (MILP) implementation for both DOBSS and ASAP, along with experimental results illustrating significant speedups and higher rewards over other approaches.

Having outlined the motivation for my work and key ideas in this section, the rest of my thesis is organized as follows. In the next chapter I describe motivating domains for my work both for single and the multiagent cases. Chapter 3 is devoted to the single agent case. I first introduce the Markov Decision Problem and a LP solution to solve it. I then develop a non-linear program and two approximate linear programming alternatives called the CRLP and the BRLP algorithms for efficient randomized policy generation. Lastly, I provide experimental results illustrating the runtime-reward tradeoffs of the various approaches developed. Chapter 4 introduces the decentralized POMDP framework to model the multi-agent team domains. It then introduces a new iterative solution called the RDR algorithm for generating randomized policies for agent teams. Finally, it provides an experimental validation of the RDR algorithm and also provides an evaluation of the policies obtained earlier for both the single and multiagent cases against a specific adversary. Chapter 6 introduces the ASAP procedure for generating strategies with limited randomization, first for non-Bayesian games, for clarity; then shows how it can be adapted for Bayesian games with uncertain adversaries. Chapter 7 provides an efficient exact optimal policy calculation procedure for Bayesian Stackelberg games based on the ideas used to develop ASAP in the previous chapter. It then provides experimental evaluation of the ASAP and the DOBSS methods against previously existing procedures. Chapter 8 gives an overview of the work done previously related to my own work and chapter 9 provides a brief summary of my work described in this thesis.

# Domains

The three sections in this section describe in detail the UAV and the police patrolling domains introduced in the previous chapter. In the first section, I describe the UAV domain for the single agent case. The next section describes the multi-UAV version of the same problem. In both these cases I assume that no model of the adversary is available and hence randomization of polices plays a key role in providing security. The third section describes the police patrolling domain with the assumption that the police have partial model of the robbers. Hence, the police compute their optimal patrolling policy using the available robber information.

## 2.1 Single agent UAV Patrolling Example

Figure 2.1 shows the UAV patrolling example for the single agent case. Consider a UAV agent that is monitoring a humanitarian mission [Twist, 2005]. A typical humanitarian mission can have various activities going on simultaneously such as providing shelter for refugees, providing food for the refugees, transportation of food to the various camps in the mission and many other such activities. Unfortunately these refugee camps become targets of various harmful activities mainly because they are vulnerable. Some of these harmful activities by an adversary/adversaries can include looting food supplies, stealing equipment, harming refugees themselves and other such activities which are unpredictable. Our worst case assumption about the adversary is as stated in

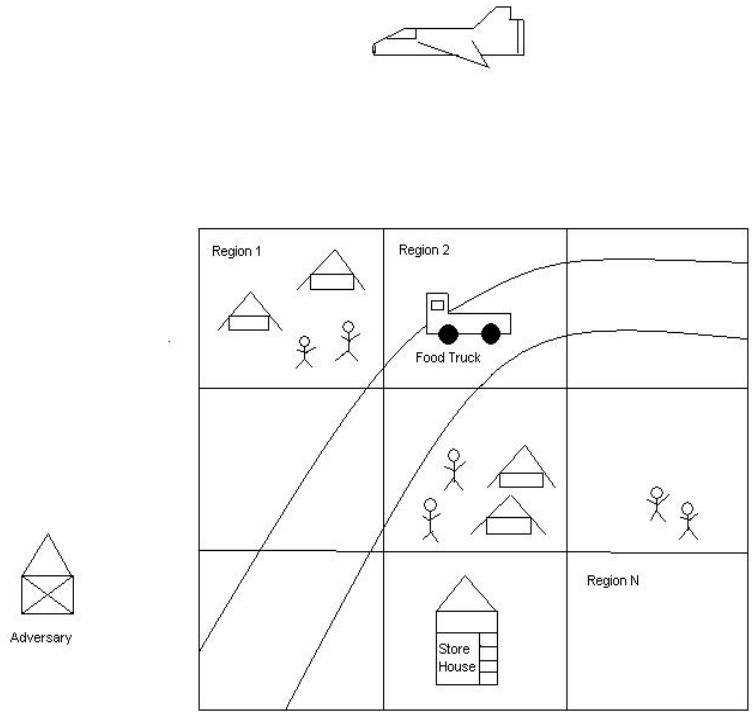



Figure 2.1: Single Agent

the Introduction: (a) the adversary has access to UAV policies due to learning or espionage (b) the adversary eavesdrop or estimates the UAV observations.

One way of reducing the vulnerability of such humanitarian missions is to have continuous monitoring activity which would deter the adversaries from performing such criminal acts. In practice, such monitoring activities can be handled by using UAVs [Twist, 2005]. To start with, we assume a single UAV is monitoring such a humanitarian mission. For expository purposes, we further assume that the mission is divided into n regions say region 1, region 2, ...., region n. If the surveillance policy of UAV is deterministic, e.g. everyday at 9am the UAV takes action survey region 1, 10am survey region 2 etc, its quite easy for the adversaries to know exactly where the UAV will be at some given time without actually seeing the UAV, allowing the adversary ample time

to plan and carry out sabotage. On the other hand, if the UAV patrolling policy was randomized, e.g. UAV surveys region 1 with 60% probability, region 2 with 40% probablity, ...., etc it would then be difficult for the adversary to predict the UAV action at a particular time without actually seeing the UAV at that instant. The effect of randomization is that the there is minimum amount of information available to the adversary about the agent's actions even though the exact policy the agent follows is known. This uncetainty in agent's actions deters the performance of adversarial actions, in effect, increasing the security of the humanitarian mission. Thus, if the policy is not randomized, the adversary may exploit UAV action predictability in some unknown way such as jamming UAV sensors, shooting down UAVs or attacking the food convoys, etc. Since little is known about the adversary's ability to cause sabotage, the UAV must maximize the adversary's uncertainty via policy randomization, while ensuring that quality constraints like resource or frequency constraints are met which are expressed as a reward function in our domains.

Different regions in the humanitarian mission can have different activities going on. Some of these activities could be critical such as saving human lives whereas some other activities might be less important. Hence, it is a necessity for the UAV to visit different regions with different frequencies. We achieve this in our domain, by assigning a specific weight or a reward function with each patrol action. This would mean that the UAV gets higher reward by taking patrol action to some regions rather than the others. The problem for the UAV would then be to have a patrolling policy that maximizes the policy randomization while ensuring that a threshold reward is maintained. This threshold reward would be set based on the amount of reward the UAV can sacrifice for increasing the security of the humanitarian mission.

## 2.2 Multi-agent UAV Patrolling example

For the multiagent case, we extend the single agent case to 2 UAVs. In particular, we introduce a simple UAV team domain that is analogous to the illustrative multiagent tiger domain [Nair et al., 2003] except for an adversary – indeed, to enable replicable experiments, rewards, transition and observation probabilities from [Nair et al., 2003] are used, the details of which we provide in the experiments section. We assume the adversary is just like the one introduced in the single agent case.

Consider a region in a humanitarian crisis, where two UAVs execute daily patrols to monitor safe food convoys. However, these food convoys may be disrupted by landmines placed in their route. The convoys pass over two regions: Left and Right. For simplicity, we assume that only one such landmine may be placed at any point in time, and it may be placed in any of the two regions with equal probability. The UAVs must destroy the landmine to get a high positive reward whereas trying to destroy a region without a landmine disrupts transportation and creates a high negative reward; but the UAV team is unaware of which region has the landmine. The UAVs can perform three actions *Shoot-left, Sense* and *Shoot-right* but they cannot communicate with each other. We assume that both UAVs are observed with equal probability by some unknown adversary with unknown capabilities, who wishes to cause sabotage. We make the following worst case assumptions about the adversary (as stated earlier): (a) the adversary has access to UAV policies due to learning or espionage (b) the adversary eavesdrop or estimates the UAV observations.

When an individual UAV takes action *Sense*, it leaves the state unchanged, but provides a noisy observation OR or OL, to indicate whether the landmine is to the left or right. The *Shoot-left* and *Shoot-right* actions are used to destroy the landmine, but the landmine is destroyed only if both UAVs simultaneously take either *Shoot-left* or *Shoot-right* actions. Unfortunately, if agents miscoordinate and one takes a *Shoot-left* and the other *Shoot-right* they incur a very high negative reward as the landmine is not destroyed but the food-convoy route is damaged. Once the shoot action occurs, the problem is restarted (the UAVs face a landmine the next day).

## 2.3 The Police Patrolling Domain

Given our assumption that the police know the partial model of the robber we formulate the patrolling domain as a game. The most basic version of this game consists of two players: the security agent (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 an 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. For example, the robber can observe over time how often the security agent patrols each area. With this knowledge, the robber must choose a single house to rob. We assume that 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.

We model the payoffs for this game with the following variables:

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

The security agent's set of possible pure strategies (patrol routes) is denoted by $X$ and includes all $d$-tuples $i =< w_1, w_2, ..., w_d >$ with $w_1 \dots w_d = 1 \dots m$. where no two elements are 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 $j = 1 \dots m$. The payoffs (security agent, robber) for pure strategies $i, j$ are:

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

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. If the distribution of different robber types is known or inferred from historical data, then the game can be modeled as a Bayesian game [Fudenberg and Tirole, 1991]. Table 6.2 provides an example of a bayesian game derived using the notation described above. More details about Bayesian games are provided in later chapters.

## Single Agent Security Problem

Markovian models such as the Markov Decision Problems (MDPs), Partially Observable Markov Decision Problems (POMDPs) and Decentralized POMDPs (Dec-POMDPs) are now popular frameworks for building agent and agent teams [Pynadath and Tambe, 2002; Cassandra et al., 1994; Paquet et al., 2005; Emery-Montemerlo et al., 2004]. The basic assumptions of these models are that the agent/agent-teams are acting in accessible or inaccessible stochastic environments with a known transition model. There are many domains in the real world where such agent/agent-team have to act in adversarial environments. In these adversarial domains, agent and agent teams based on single-agent or decentralized (PO)MDPs should randomize policies in order to avoid action predictability.

Such policy randomization is crucial for security in domains where we cannot explicitly model our adversary's actions and capabilities or its payoffs, but the adversary observes our agents' actions and exploits any action predictability in some unknown fashion. Consider agents that schedule security inspections, maintenance or refueling at seaports or airports. Adversaries may be unobserved terrorists with unknown capabilities and actions, who can learn the schedule from observations. If the schedule is deterministic, then these adversaries may exploit schedule predictability to intrude or attack and cause tremendous unanticipated sabotage. Alternatively, as mentioned in the previous chapter, consider a team of UAVs (Unmanned Aerial Vehicles) [Beard

and McLain, 2003] monitoring a region undergoing a humanitarian crisis. Adversaries may be humans intent on causing some significant unanticipated harm — e.g. disrupting food convoys, harming refugees or shooting down the UAVs — the adversary's capabilities, actions or payoffs are unknown and difficult to model explicitly. However, the adversaries can observe the UAVs and exploit any predictability in UAV surveillance, e.g. engage in unknown harmful actions by avoiding the UAVs' route. Therefore, such patrolling UAV agent/agent-team need to randomize their patrol paths to avoid action predictability [Lewis et al., 2005].

One feature of the above mentioned domains is that the actions the agents take can be stochastic and hence contribute towards randomization. For example, in the airport scenario, airplanes can get delayed with a small probability. Common activities like refueling or maintainance can take more time than expected. This is also true of the patrolling scenario where a patrol agent can end up patrolling a certain house at a given time rather than an intended house due to real world uncertainties. However given that we intend to develop secure policies, we randomize the actions of the agents explicitly rather than rely on the randomization inherently present within any action like the delay in schedule of airplanes. Thus, our key assumption is that predictability of actions is itself inherently dangerous (because the adversary can exploit this predictability), regardless of the uncertainty in the outcome of the actions. (This is indeed why for example patrols in the real-world may be randomized, as discussed in chapter 8.)

While we cannot explicitly model the adversary's actions, capabilities or payoffs, in order to ensure security of the agent/agent-team we make two worst case assumptions about the adversary. (We show later that a weaker adversary, i.e. one who fails to satisfy these assumptions, will in general only lead to enhanced security.) The first assumption is that the adversary can estimate the agent's state or belief state. In fully observable domains, the adversary estimates the agent's state

to be the current world state which both can observe fully. If the domain is partially observable, we assume that the adversary estimates the agent's belief states, because: (i) the adversary eavesdrops or spies on the agent's sensors such as sonar or radar (e.g., UAV/robot domains); or (ii) the adversary estimates the most likely observations based on its model of the agent's sensors; or (iii) the adversary is co-located and equipped with similar sensors. The second assumption is that the adversary knows the agents' policy, which it may do by learning over repeated observations or obtaining this policy via espionage or other exploitation.

Thus, we assume that the adversary may have enough information to predict the agents' actions with certainty if the agents followed a deterministic policy. Hence, our work maximizes policy randomization to thwart the adversary's prediction of the agent's actions based on the agent's state and minimize adversary's ability to cause harm. Unfortunately, while randomized policies are created as a side effect [Altman, 1999] and turn out to be optimal in some stochastic games [Littman, 1994; Koller and Pfeffer, 1995], little attention has been paid to intentionally maximizing randomization of agents' policies even for single agents. Obviously, simply randomizing an MDP/POMDP policy can degrade an agent's expected rewards, and thus we face a randomization-reward tradeoff problem: how to randomize policies with only a limited loss in expected rewards. Indeed, gaining an explicit understanding of the randomization-reward tradeoff requires new techniques for policy generation rather than the traditional single-objective maximization techniques. However, generating policies that provide appropriate randomization-reward tradeoffs is difficult, a difficulty that is exacerbated in agent teams based on decentralized MDPs/POMDPs, as randomization may create miscoordination.

In particular, my thesis provides two key contributions to generate randomized policies: (1) I provide novel techniques that enable policy randomization in single agents, while attaining a certain expected reward threshold. I measure randomization via an entropy-based metric (although our techniques are not dependent on that metric). In particular, I illustrate that simply maximizing entropy-based metrics introduces a non-linear program that does not guarantee polynomial run-time. Hence, I introduce my CRLP (Convex combination for Randomization) and BRLP (Binary search for Randomization) linear programming (LP) techniques that randomize policies in polynomial time with different tradeoffs as explained later. (2) I provide a new algorithm, RDR (Rolling Down Randomization), for generating randomized policies for decentralized POMDPs without communication, given a threshold on the expected team reward loss. RDR starts with a joint deterministic policy for decentralized POMDPs, then iterates, randomizing policies for agents turn-by-turn, keeping policies of all other agents fixed. A key insight in RDR is that given fixed randomized policies for other agents, I can generate a randomized policy via CRLP or BRLP, i.e., my efficient single-agent methods.

The motivation for use of entropy-based metrics to randomize our agents' policies stems from information theory. It is well known that the expected number of probes (e.g., observations) needed to identify the outcome of a distribution is bounded below by the entropy of that distribution [Shannon, 1948; Wen, 2005]. Thus, by increasing policy entropy, we force the opponent to execute more probes to identify the outcome of our known policy, making it more difficult for the opponent to anticipate our agent's actions and cause harm. In particular, in our (PO)MDP setting, the conflict between the agents and the adversary can be interpreted as a game, in which the agents generate a randomized policy above a given expected reward threshold; the adversary knows the agent's policy and the adversary's action is to guess the exact action of the agent/agent-team by

probing. For example, in the UAV setting, given our agent's randomized policy, the adversary generates probes to determine the direction our UAV is headed from a given state. Thus, in the absence of specific knowledge of the adversary, we can be sure to increase the average number of probes the adversary uses by increasing the lower bound given by the entropy of the policy distribution at every state.

In the rest of this chapter we focus on randomizing single agent MDP policies, i.e., a single MDP-based UAV agent is monitoring a troubled region, where the UAV gets rewards for surveying various areas of the region, but as discussed above, security requires it to randomize its monitoring strategies to avoid predictability. The case of multi-UAV teams will be discussed in the next chapter.

## 3.1 Markov Decision Problems(MDP)

An MDP is a model of an agent interacting with a world. As shown in Figure 3.1, the agent takes as input the state of the world and generates as output actions, which themselves affect the state of the world. In the MDP framework, it is assumed that, although there may be great deal of uncertainty about the effects of an agent's actions, there is never any uncertainty about the agent's current state – it has complete and perfect perceptual abilities.

### 3.1.1 Basic Framework

An MDP is denoted as a tuple $\langle S, A, P, R \rangle$, where

$S$ is a set of world states $\{s_1, \ldots, s_m\}$;

$A$ the set of actions $\{a_1, \ldots, a_k\}$;

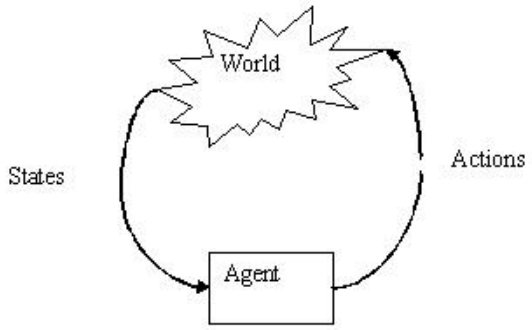



Figure 3.1: Markov Decision Process

P the set of tuples $p(s, a, j)$ representing the transition function and

R the set of tuples $r(s, a)$ denoting the immediate reward for taking action a in state s.

If $x(s, a)$ represents the number of times the MDP visits state $s$ and takes action $a$ and $\alpha_j$ represents the number of times that the MDP starts in each state $j \in S$, then the optimal policy, maximizing expected reward, is derived via the following linear program [Dolgov and Durfee, 2003b]:

$$\begin{aligned} \max \quad & \sum_{s \in S} \sum_{a \in A} r(s, a) x(s, a) \\ \text{s.t.} \quad & \sum_{a \in A} x(j, a) - \sum_{s \in S} \sum_{a \in A} p(s, a, j) x(s, a) = \alpha_j \\ & \qquad\qquad\qquad\qquad\qquad\qquad \forall j \in S \\ & x(s, a) \geq 0 \qquad\qquad\qquad\qquad \forall s \in S, a \in A \end{aligned} \tag{3.1}$$

If $x^*$ is the optimal solution to (3.1), optimal policy $\pi^*$ is given by (3.2) below, where $\pi^*(s, a)$ is the probability of taking action a in state s. It turns out that $\pi^*$ is deterministic and uniformly optimal regardless of the initial distribution $\{\alpha_j\}_{j \in S}$ [Dolgov and Durfee, 2003b] i.e., $\pi^*(s, a)$

has a value of either 0 or 1. However, such deterministic policies are undesirable in domains like our UAV example.

$$\pi^*(s,a) = \frac{x^*(s,a)}{\sum_{\hat{a}\in A} x^*(s,\hat{a})} \ . \tag{3.2}$$

## 3.2 Randomness of a policy

We borrow from information theory the concept of entropy of a set of probability distributions to quantify the randomness, or information content, in a policy of the MDP. For a discrete probability distribution $p_1, \ldots, p_n$ the only function, up to a multiplicative constant, that captures the randomness is the entropy, given by the formula $H = -\sum_{i=1}^{n} p_i \log p_i$ [Shannon, 1948]. We now introduce the *weighted entropy* function as borrowed from [Shannon, 1948] defined for a Markoff process, the equivalent term for the Markov process we define here. The *weighted entropy* function is used to quantify the randomness in a policy $\pi$ of an MDP and express it in terms of the underlying frequency $x$. Note from the definition of a policy $\pi$ in (3.2) that for each state $s$ the policy defines a probability distribution over actions. 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

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

We note that the randomization approach we propose works for alternative functions of the randomness yielding similar results. For example we can define an *additive entropy* taking a simple sum of the individual state entropies as follows:

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

We now present three algorithms to obtain random solutions that maintain an expected reward of at least $E_{\min}$ (a certain fraction of the maximal expected reward $E^*$ obtained solving (3.1)). These algorithms result in policies that, in our UAV-type domains, enable an agent to get a sufficiently high expected reward, e.g. surveying enough area, using randomized flying routes to avoid predictability.

### 3.2.1 Maximal entropy solution

We can obtain policies with maximal entropy but a threshold expected reward by replacing the objective of Problem (3.1) with the definition of the weighted entropy $H_W(x)$. The reduction in expected reward can be controlled by enforcing that feasible solutions achieve at least a certain expected reward $E_{\min}$. The following problem maximizes the weighted entropy while maintaining the expected reward above $E_{\min}$:

$$
\begin{aligned}
\max \quad & -\tfrac{1}{\sum_{j\in S}\alpha_j}\sum_{s\in S}\sum_{a\in A} x(s,a)\log\left(\frac{x(s,a)}{\sum_{\hat{a}\in A} x(s,\hat{a})}\right) \\
\text{s.t.} \quad & \sum_{a\in A} x(j,a) - \sum_{s\in S}\sum_{a\in A} p(s,a,j)x(s,a) = \alpha_j \\
& \qquad\qquad\qquad\qquad\qquad\qquad \forall j \in S \\
& \sum_{s\in S}\sum_{a\in A} r(s,a)x(s,a) \geq E_{\min} \\
& x(s,a) \geq 0 \qquad\qquad\qquad\qquad \forall s \in S, a \in A
\end{aligned}
\tag{3.3}
$$

$E_{\min}$ is an input domain parameter (e.g. UAV mission specification). Alternatively, if $E^*$ denotes the maximum expected reward from (1), then by varying the expected reward threshold $E_{\min} \in [0, E^*]$ we can explore the tradeoff between the achievable expected reward and entropy, and then select the appropriate $E_{\min}$. Note that for $E_{\min} = 0$ the above problem finds the maximum weighted entropy policy, and for $E_{\min} = E^*$, Problem (3.3) returns the maximum expected reward policy with largest entropy. Solving Problem (3.3) is our first algorithm to obtain a randomized policy that achieves at least $E_{\min}$ expected reward (Algorithm 1).

**Algorithm 1** MAX-ENTROPY($E_{\min}$)

1: Solve Problem (3.3) with $E_{\min}$, let $x_{E_{\min}}$ be optimal solution
2: **return** $x_{E_{\min}}$ (maximal entropy, expected reward $\geq E_{\min}$)

Unfortunately entropy-based functions like $H_W(x)$ are neither convex nor concave in $x$, hence there are no complexity guarantees in solving Problem (3.3), even for local optima [Vavasis, 1991]. This negative complexity motivates the polynomial methods presented next.

## 3.3 Efficient single agent randomization

The idea behind these polynomial algorithms is to efficiently solve problems that obtain policies with a high expected reward while maintaining some level of randomness. (A very high level of randomness implies a uniform probability distribution over the set of actions out of a state, whereas a low level would mean deterministic action being taken from a state). We then obtain a solution that meets a given minimal expected reward value by adjusting the level of randomness in the policy. The algorithms that we introduce in this section 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. For example, one such high entropy input for MDP-based problems is the uniform policy, where $\bar{\pi}(s, a) = 1/|A|$. Note that uniform policies need not always lead to highest entropy policies for an MDP. A simple example can be an MDP where every state has two actions say left and right. For the start state we assume that action left leads to end state but action right leads to a state which has two more actions and the tree is symmetric from thereafter. In this case a totally uniform policy gives lower entropy than a policy which deterministically chooses action right in the start state and is uniform thereafter. 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 $\beta$ and a high entropy solution $\bar{x}$, we output a maximum expected reward solution with a certain level of randomness by solving (3.4).

$$
\begin{aligned}
\max \quad & \sum_{s\in S}\sum_{a\in A} r(s,a)x(s,a) \\
\text{s.t.} \quad & \sum_{a\in A} x(j,a) - \sum_{s\in S}\sum_{a\in A} p(s,a,j)x(s,a) = \alpha_j \qquad \forall j \in S \\
& x(s,a) \geq \beta\bar{x}(s,a) \qquad\qquad \forall s \in S, a \in A\ .
\end{aligned}
\tag{3.4}
$$

which can be referred to in matrix shorthand as

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

As the parameter $\beta$ is increased, the randomness requirements of the solution become stricter and hence the solution to (3.4) would have smaller expected reward and higher entropy. For $\beta = 0$ the above problem reduces to (3.1) 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 $\bar{x}$. If $E^*$ is finite, then Problem (3.4) returns $\bar{x}$ for $\beta = 1$ and $\overline{E} = \sum_{s\in S}\sum_{a\in A} r(s,a)\bar{x}(s,a)$.

Our second 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 (3.4) is a convex combination of the deterministic and highly random input solutions.

**Theorem 1** *Consider a solution $\bar{x}$, which satisfies $A\bar{x} = \alpha$ and $\bar{x} \geq 0$. Let $x^*$ be the solution to (3.1) and $\beta \in [0,1]$. If $x_\beta$ is the solution to (3.4) then $x_\beta = (1-\beta)x^* + \beta\bar{x}$.*

**proof:** We reformulate problem (3.4) in terms of the slack $z = x - \beta\bar{x}$ of the solution $x$ over $\beta\bar{x}$ leading to the following problem :

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

The above problem is equivalent to (3.4), where we used the fact that $A\bar{x} = \alpha$. Let $z^*$ be the solution to this problem, which shows that $x_\beta = z^* + \beta\bar{x}$. Dividing the linear equation $Az = (1-\beta)\alpha$, by $(1-\beta)$ and substituting $u = z/(1-\beta)$ we recover the deterministic problem (3.1) 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 $x_\beta$ and $\beta$, a linear relationship exists between the expected reward obtained by $x_\beta$ (i.e $r^T x_\beta$) and $\beta$. In fact setting $\beta = \frac{r^T x^* - E_{\min}}{r^T x^* - r^T \bar{x}}$ makes $r^T x_\beta = E_{\min}$. We now present below algorithm CRLP based on the observations made about $\beta$ and $x_\beta$.

**Algorithm 2** $\mathrm{CRLP}(E_{\min}, \bar{x})$

1: Solve Problem (3.1), 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** $x_\beta$ (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(3.4) with expected reward values $E_{\min} \in [\overline{E}, E^*]$. Note that Algorithm CRLP might

unnecessarily constrain the solution set as Problem (3.4) 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 Section 5. This concern is addressed by a reformulation of Problem (3.4) 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

$$\begin{aligned} \max \quad & \sum_{s \in S} \sum_{a \in A} r(s,a) x(s,a) \\ \text{s.t.} \quad & \sum_{a \in A} x(j,a) - \sum_{s \in S} \sum_{a \in A} p(s,a,j) x(s,a) = \alpha_j, \quad \forall j \in S \\ & x(s,a) \geq \beta \bar{\pi}(s,a) \sum_{b \in A} x(s,b), \qquad \forall s \in S, a \in A \, . \end{aligned} \tag{3.5}$$

For $\beta = 0$ this problem reduces to (3.1) returning $E^*$, for $\beta = 1$ it returns a maximal expected reward solution with the same policy as $\bar{\pi}$. This means that for $\beta$ at values 0 and 1, problems (3.4) and (3.5) 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 $\beta$, the policy obtained by problems (3.4) and (3.5) are different even if $\bar{\pi}$ is obtained from $\bar{x}$. Thus, theorem 1 holds for problem (3.4) but not for (3.5). Table 3.1, obtained experimentally, validates our claim by showing the maximum expected rewards and entropies obtained (entropies in parentheses) from problems (3.4) and (3.5) for various settings of $\beta$, e.g. for $\beta$ = 0.4, problem (3.4) provides a maximum expected reward of 26.29 and entropy of 5.44, while problem (3.5) provides a maximum expected reward of 25.57 and entropy of 6.82.

Table 3.1 shows that for the same value of $\beta$ in Problems (3.4) and (3.5) we get different maximum expected rewards and entropies implying that the optimal policies for both problems are different, hence Theorem 1 does not hold for (3.5). Indeed, while the expected reward of

Problem (3.4) is higher for this example, its entropy is lower than Problem (3.5). Hence to investigate another randomization-reward tradeoff point, we introduce our third algorithm BRLP, which uses problem (3.5) to perform a binary search to attain a policy with expected reward $E_{\min} \in [\overline{E}, E^*]$, adjusting the parameter $\beta$.

| **Beta** | .2 | .4 | .6 | .8 |
|---|---|---|---|---|
| $Problem(3.4)$ | 29.14 (3.10) | 26.29 (5.44) | 23.43 (7.48) | 20.25 (9.87) |
| $Problem(3.5)$ | 28.57 (4.24) | 25.57 (6.82) | 22.84 (8.69) | 20.57 (9.88) |

Table 3.1: Maximum expected rewards(entropies) for various $\beta$

**Algorithm 3** $\mathrm{BRLP}(E_{\min}, \bar{x})$

Set $\beta_l = 0$, $\beta_u = 1$, and $\beta = 1/2$.
Obtain $\bar{\pi}$ from $\bar{x}$
Solve Problem (3.5), let $x_\beta$ and $E(\beta)$ be the optimal solution and expected reward value returned
**while** $|E(\beta) - E_{\min}| > \epsilon$ **do**
    **if** $E(\beta) > E_{\min}$ **then**
        Set $\beta_l = \beta$
    **else**
        Set $\beta_u = \beta$
    **end if**
    $\beta = \frac{\beta_u + \beta_l}{2}$
    Solve Problem (3.5), let $x_\beta$ and $E(\beta)$ be the optimal solution and expected reward value returned
**end while**
**return** $x_\beta$ (expected reward $= E_{\min} \pm \epsilon$, entropy related to $\beta\bar{x}$)

Given input $\bar{x}$, algorithm BRLP runs in polynomial time, since at each iteration it solves an LP and for tolerance of $\epsilon$, 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 $\beta$).

## 3.4 Incorporating models of the adversary

Throughout this thesis, 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 1 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.

## 3.5 Applying to POMDPs

Before turning to agent teams next, we quickly discuss applying these algorithms in single agent POMDPs [Cassandra et al., 1994; Kaelbling et al., 1998]. A POMDP can be represented as a tuple $< S, A, T, \omega, O, R >$, where S is a finite set of states; A is a finite set of actions; T(s, a, s') provides the probability of transitioning from state s to s' when taking action a; $\omega$ is a finite set of observations; O(s', a, o) is probability of observing o after taking an action a and reaching s'; R(s, a) is the reward function. Figure 3.2 shows a POMDP whole control is decomposed into two parts. The agent in this model makes observations and generates actions. It keeps an internal belief state b, that summarizes its previous experience. The component labeled SE is the state estimator: it is responsible for updating the belief state based on the last action, the current observation, and the previous belief state. The component labeled $\pi$ is the policy: as before in

the MDP, it is responsible for generating actions, but this time as a function of the agent's belief state rather than the state of the world.

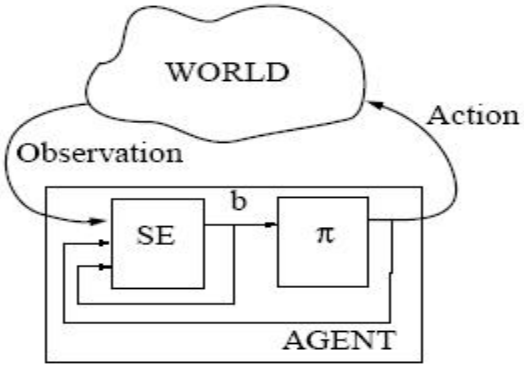



Figure 3.2: Partially Observable Markov Decision Process

For a single-agent finite-horizon POMDPs with known starting belief states [Paquet et al., 2005], we convert the POMDP to (finite horizon) *belief MDP*, allowing BRLP/CRLP to be applied; returning a randomized policy. However, addressing unknown starting belief states is an issue for future work.

# Multi Agent Security Problem

In previous chapter I developed efficient security algorithms for the single agent case. In this chapter I develop an efficient security algorithm for the multiagent case using the single agent security algorithms developed earlier. I assume that the agent team acts in a partially observable environment and hence I first introduce the decentralized POMDP framework to model the agent team. I then describe the procedure to obtain randomized policies in agent teams.

## 4.1 From Single Agent to Agent Teams

We use notation from MTDP (Multiagent Team Decision Problem) [Pynadath and Tambe, 2002] for our decentralized POMDP model; other models are equivalent [Bernstein et al., 2000]. Given a team of $n$ agents, an MTDP is defined as a tuple: $\langle S, A, P, \Omega, O, R \rangle$, where,

- $S$ is a finite set of world states $\{s_1, \ldots, s_m\}$.
- $A = \times_{1 \leq i \leq n} A_i$, where $A_1, \ldots, A_n$, are the sets of action for agents 1 to $n$. A joint action is represented as $\langle a_1, \ldots, a_n \rangle$.
- $P(s_i, \langle a_1, \ldots, a_n \rangle, s_f)$, the transition function, represents the probability that the current state is $s_f$, if the previous state is $s_i$ and the previous joint action is $\langle a_1, \ldots, a_n \rangle$.

- $\Omega = \times_{1 \leq i \leq n} \Omega_i$ is the set of joint observations where $\Omega_i$ is the set of observations for agents $i$.

- $O(s, \langle a_1, \ldots, a_n \rangle, \omega)$, the observation function, represents the probability of joint observation $\omega \in \Omega$, if the current state is $s$ and the previous joint action is $\langle a_1, \ldots, a_n \rangle$. We assume that observations of each agent are independent of each other's observations, i.e. $O(s, \langle a_1, \ldots, a_n \rangle, \omega) = O_1(s, \langle a_1, \ldots, a_n \rangle, \omega_1) \cdot \ldots \cdot O_n(s, \langle a_1, \ldots, a_n \rangle, \omega_n)$.

- The agents receive a single, immediate joint reward $R(s, \langle a_1, \ldots, a_n \rangle)$.

For deterministic policies, each agent $i$ chooses its actions based on its policy, $\Pi_i$, which maps its observation history to actions. Thus, at time $t$, agent $i$ will perform action $\Pi_i(\vec{\omega}_i^t)$ where $\vec{\omega}_i^t = \omega_i^1, ....., \omega_i^t$. $\Pi = \langle \Pi_1, ....., \Pi_n \rangle$ refers to the joint policy of the team of agents. In this model, execution is distributed but planning is centralized; and agents don't know each other's observations and actions at run time.

Unlike previous work, in our work, policies are randomized and hence agents obtain a probability distribution over a set of actions rather than a single action. Furthermore, this probability distribution is indexed by a sequence of action-observation tuples rather than just observations, since observations do not map to unique actions. Thus in MTDP, a randomized policy maps $\mathbf{\Psi_i^t}$ to a probability distribution over actions, where $\mathbf{\Psi_i^t} = \langle \psi_i^1, \ldots, \psi_i^t \rangle$ and $\psi_i^t = \langle a_i^{t-1}, \omega_i^t \rangle$. Thus, at time $t$, agent $i$ will perform an action selected randomly based on the probability distribution returned by $\Pi_i(\mathbf{\Psi_i^t})$. Furthermore we denote the probability of an individual action under policy $\Pi_i$ given $\mathbf{\Psi_i^t}$ as $P^{\Pi_i}(a_i^t | \mathbf{\Psi_i^t})$. However, there are many problems in randomizing an MTDP policy.

1. Existing algorithms for MTDPs produce deterministic policies. New algorithms need to be designed to specifically produce randomized policies.

2. Randomized policies in team settings may lead to miscoordination, unless policies are generated carefully, as explained in the section below.

3. Efficiently generating randomized policies is a key challenge as search space for random policies is larger than for deterministic policies.

## 4.2 Multiagent Randomization

Let $p_i$ be the probability of adversary targeting agent $i$, and $H_W(i)$ be the weighted entropy for agent $i$'s policy. We design an algorithm that maximizes the *multiagent weighted entropy*, given by $\sum_{i=1}^{n} p_i * H_W(i)$, in MTDPs while maintaining the team's expected joint reward above a threshold. Unfortunately, generating optimal policies for decentralized POMDPs is of higher complexity (NEXP-complete) than single agent MDPs and POMDPs [Bernstein et al., 2000], i.e., MTDP presents a fundamentally different class where we cannot directly use the single agent randomization techniques.

Hence, to exploit efficiency of algorithms like BRLP or CRLP, we convert the MTDP into a single agent POMDP, but with a method that changes the state space considered. To this end, our new iterative algorithm called RDR (Rolling Down Randomization) iterates through finding the best randomized policy for one agent while fixing the policies for all other agents — we show that such iteration of fixing the randomized policies of all but one agent in the MTDP leads to a single agent problem being solved at each step. Thus, each iteration can be solved via BRLP or

CRLP. For a two agent case, we fix the policy of agent $i$ and generate best randomized policy for agent $j$ and then iterate with agent $j$'s policy fixed.

Overall RDR starts with an initial joint deterministic policy calculated in the algorithm as a starting point. Assuming this fixed initial policy as providing peak expected reward, the algorithm then rolls down the reward, randomizing policies turn-by-turn for each agent. Rolling down from such an initial policy allows control of the amount of expected reward loss from the given peak, in service of gaining entropy. *The key contribution of the algorithm is in the rolling down procedure that gains entropy (randomization),* and this procedure is independent of how the initial policy for peak reward is determined. The initial policy may be computed via algorithms such as [Hansen et al., 2004] that determine a global optimal joint policy (but at a high cost) or from random restarts of algorithms that compute a locally optimal policy [Nair et al., 2003; Emery-Montemerlo et al., 2004], that may provide high quality policies at lower cost. The amount of expected reward to be rolled down is input to RDR. RDR then achieves the rolldown in $1/d$ steps where $d$ is an input parameter.

The turn-by-turn nature of RDR suggests some similarities to JESP [Nair et al., 2003], which also works by fixing the policy of one agent and computing the best-response policy of the second and iterating. However, there are significant differences between RDR and JESP, as outlined below:

- JESP uses conventional value iteration based techniques whereas RDR creates randomized policies via LP formulations.

- RDR defines a new extended state and hence the belief-update, transition and reward functions undergo a major transformation.

- The d parameter is newly introduced in RDR to control number of rolldown steps.
- RDR climbs down from a given optimal solution rather than JESP's hill-climbing up solution.

## 4.3 RDR Details

For expository purposes, we use a two agent domain, but we can easily generalize to $n$ agents. We fix the policy of one agent (say agent 2), which enables us to create a single agent POMDP if agent 1 uses an extended state, i.e. at each time $t$, agent 1 uses an extended state $e_1^t = \langle s^t, \mathbf{\Psi_2^t} \rangle$. Here, $\mathbf{\Psi_2^t}$ is as introduced in the previous section. By using $e_1^t$ as agent 1's state at time $t$, given the fixed policy of agent 2, we can define a single-agent POMDP for agent 1 with transition and observation function as follows.

$$\begin{aligned} P'(e_1^t, a_1^t, e_1^{t+1}) &= P(\langle s^{t+1}, \mathbf{\Psi_2^{t+1}} \rangle | \langle s^t, \mathbf{\Psi_2^t} \rangle, a_1^t) \\ &= P(\omega_2^{t+1} | s^{t+1}, a_2^t, \mathbf{\Psi_2^t}, a_1^t) \\ &\quad \cdot P(s^{t+1} | s^t, a_2^t, \mathbf{\Psi_2^t}, a_1^t) \cdot P(a_2^t | s^t, \mathbf{\Psi_2^t}, a_1^t) \\ &= P(s^t, \langle a_2^t, a_1^t \rangle, s^{t+1}) \cdot O_2(s^{t+1}, \langle a_2^t, a_1^t \rangle, \omega_2^{t+1}) \\ &\quad \cdot P^{\Pi_2}(a_2^t | \mathbf{\Psi_2^t}) \end{aligned} \tag{4.1}$$

$$\begin{aligned} O'(e_1^{t+1}, a_1^t, \omega_1^{t+1}) &= \Pr(\omega_1^{t+1} | e_1^{t+1}, a_1^t) \\ &= O_1(s^{t+1}, \langle a_2^t, a_1^t \rangle, \omega_1^{t+1}) \end{aligned} \tag{4.2}$$

Thus, we can create a belief state for agent $i$ in the context of $j$'s fixed policy by maintaining a distribution over $e_i^t = \langle s^t, \mathbf{\Psi_j^t} \rangle$. Figure 4.1 shows three belief states for agent 1 in the UAV

domain. For instance $B^2$ shows probability distributions over $e_1^2$. In $e_1^2 = (\text{Left}\langle Sense, OL\rangle)$, Left implies landmine to the left is the current state, Sense is the agent 2's action at time 1, OL (Observe Left) is agent 2's observation at time 2. The belief update rule derived from the transition and observation functions is given in 4.3, where denominator is the transition probability when action $a_1$ from belief state $B_1^t$ results in $\omega_1^{t+1}$ being observed. Immediate rewards for the belief states are assigned using 4.4.

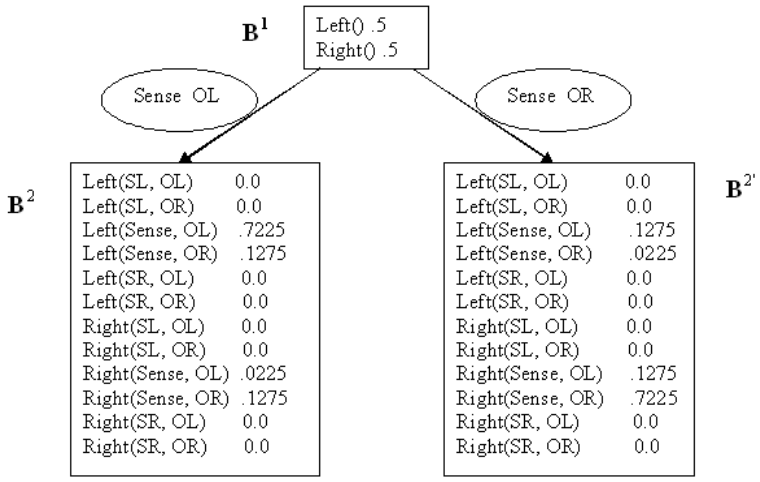



Figure 4.1: Belief Tree for UAV team domain generated by RDR

$$B_1^{t+1}(e_1^{t+1}) = \sum_{s^t} B_1^t(e_1^t) \cdot P(s^t, (a_1^t, a_2^t), s^{t+1}) \cdot P^{\Pi_2}(a_2^t | \mathbf{\Psi_2^t}) \cdot O_2(s^{t+1}, (a_1^t, a_2^t), \omega_2^{t+1}) \cdot O_1(s^{t+1}, (a_1^t, a_2^t), \omega_1^{t+1}) / P(\omega_1^{t+1} | B_1^t, a_1) \quad (4.3)$$

$$\Re(a_1^t, B_1^t) = \sum_{e_1^t} B_1^t(e_1^t) \cdot \sum_{a_2^t} R(s^t, (a_1^t, a_2^t)) \cdot P^{\Pi_2}(a_2^t | \mathbf{\Psi_2^t}) \tag{4.4}$$

Thus, RDR's policy generation implicitly coordinates the two agents, without communication or a correlation device. Randomized actions of one agent are planned taking into account the impact of randomized actions of its teammate on the joint reward. Algorithm 4 presents the pseudo-code for RDR for two agents, and shows how we can use beliefs over extended states $e_i^t$ to construct LPs that maximize entropy while maintaining a certain expected reward threshold. The inputs to this algorithm are the parameters d, *percentdec* and $\bar{x}$. The parameter d specifies the number of iterations taken to solve the problem. It also decides the amount of reward that can be sacrificed at each step of the algorithm for improving entropy. Parameter *percentdec* specifies the percentage of expected reward the agent team forgoes for improving entropy. As with Algorithm 2, parameter $\bar{x}$ provides an input solution with high randomness; and it is obtained using a uniform policy as discussed in Section 3.3. Step 1 of the algorithm uses the Compute joint policy() function which returns a optimal joint deterministic policy from which the individual policies and the expected joint reward are extracted. Obtaining this initial policy is not RDR's main thrust, and could be a global optimal policy obtained via [Hansen et al., 2004] or a very high quality policy from random restarts of local algorithms such as [Nair et al., 2003; Emery-Montemerlo et al., 2004]. The input variable *percentdec* varies between 0 to 1 denoting the percentage of the expected reward that can be sacrificed by RDR. The step size as calculated in the algorithm, denotes the amount of reward sacrificed during each iteration. RDR then iterates 1/d times (step 3).

**Algorithm 4** RDR$(d, percentdec, \bar{x})$

$\pi_1, \pi_2, Optimalreward \leftarrow Compute\_joint\_policy()$
$stepsize \leftarrow percentdec \cdot Optimalreward \cdot d$
**for** $i \leftarrow 1$ **to** $1/d$ **do**
  $MDP \leftarrow GenerateMDP(b, \Pi_{(i+1)\text{Mod}2}, T)$
  $Entropy, \Pi_{i\text{Mod}2} \leftarrow BRLP(Optimalrew - stepsize * i, \bar{x})$
**end for**

**GenerateMDP**$(b, \pi_2, T)$ :
*reachable*$(0) \leftarrow \{b\}$
**for** $t \leftarrow 1$ **to** $T$ **do**
  **for all** $B^{t-1} \in$ *reachable*$(t-1)$ **do**
    **for all** $a_1 \in A_1, \omega_1 \in \Omega_1$ **do**
      $trans, reachable(t) \overset{\cup}{\leftarrow}$ UPDATE$(B^{t-1}, a_1, \omega_1)$
    **end for**
  **end for**
**end for**
**for** $t \leftarrow T$ **downto** $1$ **do**
  **for all** $B^t \in$ *reachable*$(t)$ **do**
    **for all** $a_1 \in A_1$ **do**
      **for all** $s^t \in S, \psi_2^t \leftarrow \langle a_2^{t-1}, \omega_2^t \rangle$ **do**
        **for all** $a_2^t$ given $\mathbf{\Psi_2^t}$ **do** {Equation 4.4}
          $\Re_t^{a_1}(B^t) \overset{+}{\leftarrow} B^t(s^t, \mathbf{\Psi_2^t}) \cdot R\left(s^t, \langle a_1^t, a_2^t \rangle\right) . P(a_2^t | \mathbf{\Psi_2^t})$
        **end for**
      **end for**
    **end for**
  **end for**
**end for**
**return** $\langle B, A, trans, \Re \rangle$

**UPDATE**$(B^t, a_1, \omega_1)$ :
**for all** $s^{t+1} \in S, \psi_2^t \leftarrow \langle a_2^{t-1}, \omega_2^t \rangle$ **do**
  **for all** $s^t \in S$ **do** {Equation 4.3}
    $B^{t+1}(s^{t+1}, \mathbf{\Psi_2^{t+1}}) \overset{+}{\leftarrow} B^t(s^t, \mathbf{\Psi_2^t}) \cdot P(s^t, (a_1^t, a_2^t), s^{t+1}) \cdot O_1(s^{t+1}, (a_1^t, a_2^t), \omega_1^{t+1}) \cdot O_2(s^{t+1}, (a_1^t, a_2^t), \omega_2^{t+1}) \cdot P(a_2^t | \mathbf{\Psi_2^t})$
  **end for**
  $trans \leftarrow normalize(B^{t+1}(s^{t+1}, \mathbf{\Psi_2^{t+1}}))$
**end for**
**return** $trans, B^{t+1}$

The function GenerateMDP generates all reachable belief states (lines 2 through 6) from a given starting belief state *B* and hence a belief MDP is generated. The number of such belief states is $O(|A_1||\Omega_1|)^{T-1}$ where $T$ is the time horizon. The extended states in each B increases by a factor of $|A_2||\Omega_2|$ with increasing T. Thus the time to calculate B(e) for all extended states e, for all belief states B in agent 1's belief MDP is $O(|S|^2(|A_1||A_2||\Omega_1||\Omega_2|)^{T-1})$. Lines 7 through 12 compute the reward for each belief state. The total computations to calculate the reward is $O(|S||A_1||A_2|(|A_1||A_2||\Omega_1||\Omega_2|)^{T-1})$. The belief MDP generated is denoted by the tuple $\langle B, A, trans, R\rangle$. We reformulate the MDP obtained to problem 3.5 and use our polynomial BRLP procedure to solve it, using $\bar{x}$ as input. Thus, algorithm RDR is exponentially faster than an exhaustive search of a policy space, and comparable to algorithms that generate locally optimal policies [Nair et al., 2003].

# Experimental Results

In this chapter, we present three sets of experimental results. The first set of experiments evaluate the nonlinear algorithm and the two linear approximation algorithms we developed in chapter 3 for the single agent case. The second set of experiments evaluate the RDR algorithm developed for the multiagent case over the various possible parameters of the algorithm. The third set of experiments examine the tradeoffs in entropy of the agent/agent-team and the total number of observations (probes) the enemy needs to determine the agent/agent-team actions at each state. These experiments are performed to show that increasing entropy indeed increases security.

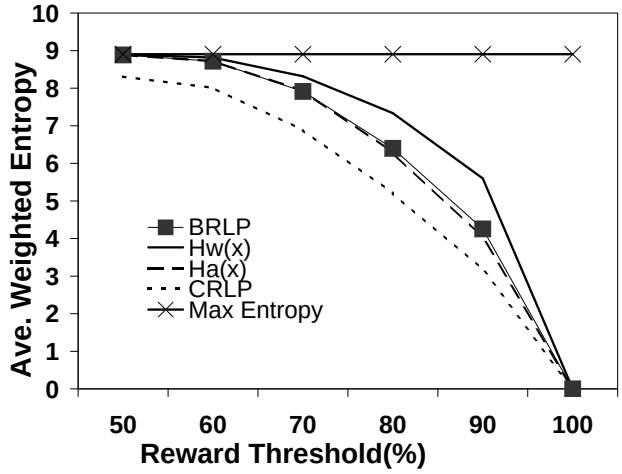
## 5.1 Single Agent Experiments

Our first set of experiments examine the tradeoffs in run-time, expected reward and entropy for single-agent problems. Figures 5.1-a and 5.1-b show the results for these experiments based on generation of MDP policies. The results show averages over 10 MDPs where each MDP represents a flight of a UAV, with state space of 28-40 states. The states represent the regions monitored by the UAVs. The transition function assumes that a UAV action can make a transition from a region to one of four other regions, where the transition probabilities were selected at random. The rewards for each MDP were also generated using random number generators. These experiments compare the performance of our four methods of randomization for single-agent

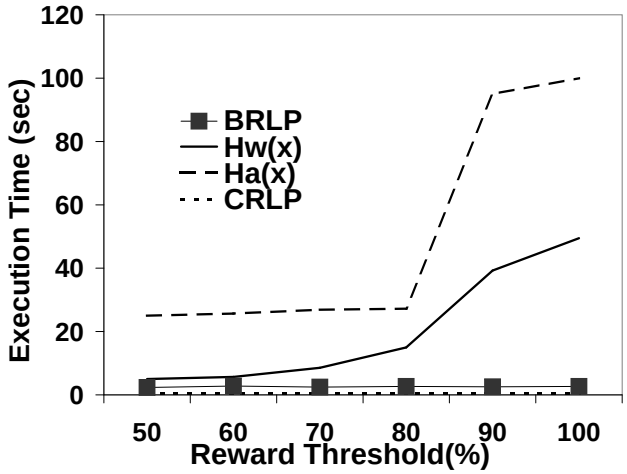
policies. In the figures, *CRLP* refers to algorithm 2; *BRLP* refers to algorithm 3; whereas $H_W(x)$ and $H_A(x)$ refer to Algorithm 1 with these objective functions. Figure 5.1 examines the tradeoff between entropy and expected reward thresholds. It shows the average weighted entropy on the y-axis 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.

Figure 5.1-b 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 as in Figure 5.1-a. 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. Algorithm 1, for both $H_A(x)$ and $H_W(x)$ objectives, exhibits an increase in the runtime as the expected reward threshold increases. This trend that 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 Figure 5.1: (i) CRLP is the fastest but provides the lowest entropy. (ii) BRLP is significantly faster than Algorithm 1, providing 7-fold speedup on average over the 10 MDPs over the entire range of thresholds. (iii) Algorithm 1 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




(a)




(b)

Figure 5.1: Comparison of Single Agent Algorithms

about 18%. In fact, CRLP is unable to reach the maximal possible entropy for the threshold range considered in the plot. Thus, BRLP appears to provide the most favorable tradeoff of run-time to entropy for the domain considered, and we use this method for the multiagent case. However, for time critical domains CRLP might be the algorithm of choice and therefore both BRLP and CRLP provide useful tradeoff points.

## 5.2 Multi Agent Experiments

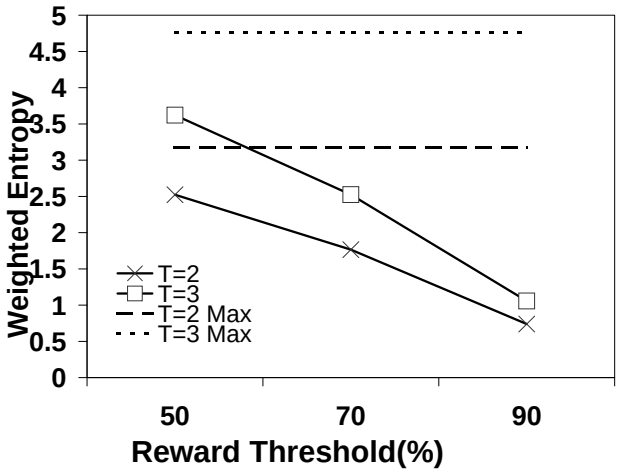
Our second set of experiments examine the tradeoffs in run-time, expected joint reward and entropy for the multiagent case. Table 5.1 shows the runtime results and entropy (in parenthesis) averaged over 10 instances of the UAV team problem based on the original transition and observation functions from [Nair et al., 2003] and its variations. $d$, the input parameter controlling the number of rolldown steps of algorithm 4, varies from 1 to 0.125 for two values of percent reward threshold (90% and 50%) and time horizon $T$=2. We conclude that as $d$ decreases, the run-time increases, but the entropy remains fairly constant for $d \leq .5$. For example, for reward threshold of 50%, for d = 0.5, the runtime is 1.47 secs, but the run-time increases more than 5-fold to 7.47 when d = 0.125; however, entropy only changes from 2.52 to 2.66 with this change in d.

| **Reward Threshold** | **1** | **.5** | **.25** | **.125** |
|---|---|---|---|---|
| 90% | .67(.59) | 1.73(.74) | 3.47(.75) | 7.07(.75) |
| 50% | .67(1.53) | 1.47(2.52) | 3.4(2.62) | 7.47(2.66) |

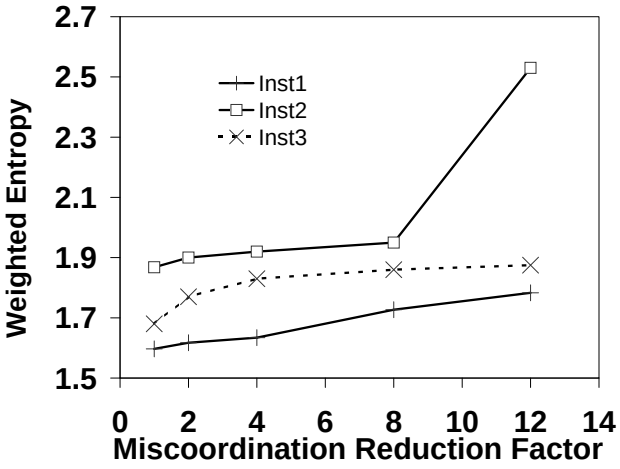
Table 5.1: RDR: Avg. run-time in sec and (Entropy), $T = 2$

Thus in our next set of graphs, we present results for $d$ = .5, as it provides the most favorable tradeoff, if other parameters remain fixed. Figure 5.2-a plots RDR expected reward threshold percent on the $x$-axis and weighted entropy on the y-axis averaged over the same 10 UAV-team instances. Thus, if the team needs to obtain 90% of maximum expected joint reward with a time horizon $T = 3$, it gets a weighted entropy of 1.06 only as opposed to 3.62 if it obtains 50% of the expected reward for the same $d$ and $T$. Similar to the single-agent case, the maximum possible entropy for the multiagent case is also shown by a horizontal line at the top of the graph. Figure 5.2-b studies the effect of changing miscoordination cost on RDR's ability to improve entropy. As explained in Section 2.2, the UAV team incurs a high cost of miscoordination, e.g. if one UAV shoots left and the other shoots right. We now define miscoordination reduction factor (MRF) as the ratio between the original miscoordination cost and a new miscoordination cost. Thus, high MRF implies a new low miscoordination cost, e.g. an MRF of 4 means that the miscoordination cost is cut 4-fold. We plot this MRF on x-axis and entropy on y-axis, with expected joint reward threshold fixed at 70% and the time horizon T at 2. We created 5 reward variations for each of our 10 UAV team instances we used for 5.2-a; only 3 instances are shown, to reduce graph clutter (others are similar). For *instance 2*, the original miscoordination cost provided an entropy of 1.87, but as this cost is scaled down by a factor of 12, the entropy increases to 2.53.

Based on these experiments, we conclude that: (i) Greater tolerance of expected reward loss allows higher entropy; but reaching the maximum entropy is more difficult in multiagent teams — for the reward loss of 50%, in the single agent case, we are able to reach maximum entropy, but we are unable to reach maximum entropy in the multiagent case. (ii) Lower miscoordination costs allow higher entropy for the same expected joint reward thresholds. (iii) Varying $d$ produces




(a)




(b)

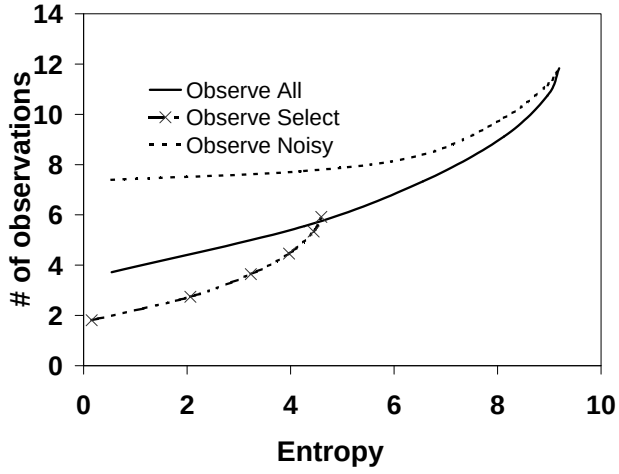
Figure 5.2: Results for RDR

only a slight change in entropy; thus we can use $d$ as high as 0.5 to cut down runtimes. (iv) RDR is time efficient because of the underlying polynomial time BRLP algorithm.

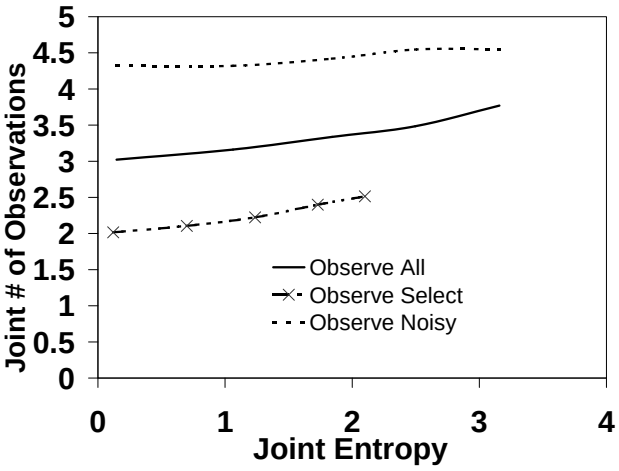
## 5.3 Entropy Increases Security: An Experimental Evaluation

Our fourth set of experiments examine the tradeoffs in entropy of the agent/agent-team and the total number of observations (probes) the enemy needs to determine the agent/agent-team actions at each state. The primary aim of this experiment is to show that maximizing policy entropy indeed makes it more difficult for the adversary to determine/predict our agents' actions, and thus more difficult for the adversary to cause harm, which was our main goal at the beginning of this thesis. Figures 5.3-a and 5.3-b plot the number of observations of enemy as function of entropy of the agent/agent-team. In particular for the experiment we performed, the adversary runs yes-no probes to determine the agent's action at each state, i.e. probes that return an answer yes if the agent is taking the particular action at that state in which case the probing is stopped, and a no otherwise. The average number of yes-no probes at a state is the total number of observations needed by the adversary to determine the correct action taken by the agent in that state. The more deterministic the policy is, the fewer the probes the adversary needs to run; if the policy is completely deterministic, the adversary need not run any probes as it knows the action. Therefore, the aim of the agent/agent-team is to maximize the policy entropy, so that the expected number of probes asked by the adversary is maximized.

In contrast, the adversary minimizes the expected number of probes required to determine the agents' actions. Hence, for any given state $s$, the adversary uses the Huffman procedure to optimize the number of probes [Huffman, 1952], and hence the total number of probes over the




(a)




(b)

Figure 5.3: Improved security via randomization

entire MDP state space can be expressed as follows. Let $S = \{s_1, s_2, ...., s_m\}$ be the set of MDP states and $A = \{a_1, a_2, ...., a_n\}$ be the action set at each state. Let $p(s, a) = \{p_1, ....., p_n\}$ be the probabilities of taking the action set $\{a'_1, ....., a'_n\}$, $a'_i \in A$ at state $s$ sorted in decreasing order of probability. The number of yes-no probes at state $s$ is denoted by $\zeta_s = p_1 * 1 + ..... + p_{n-1} * (n - 1) + p_n * (n-1)$. If the weight of the states (see notion of weight introduced in section 3.2) is $W = \{w_1, w_2, ....., w_m\}$, then the number of observations over the set of states is denoted *Observe-all* $= \sum_{s=1...m} \{w_s * \zeta_s\}$. Setting some weights to zero implies that the adversary was not concerned with those states, and the number of observations in this situation is denoted *Observe-select*. While the number of observations in both Observe-all and Observe-select are obtained assuming the adversary obtains an accurate policy of the agent or agent team, in real situations, an adversary may obtain a noisy policy, and the adversary's number of observations in such a case is denoted Observe-noisy.

Figure 5.3-a demonstrates the effectiveness of entropy maximization using the BRLP method against an adversary using yes-no probes procedure as his probing method for the single agent case. The plot shows the number of observations on y-axis and entropy on the x-axis averaged over the 10 MDPs we used for our single-agent experiment. The plot has 3 lines corresponding to the three adversary procedures namely *Observe-all*, *Observe-select* and *Observe-noisy*. *Observe-all* and *Observe-select* have been plotted to study the effect of entropy on the number of probes the adversary needs. For example, for *Observe-all*, when entropy is 8, the average number of probes needed by the adversary is 9. The purpose of the *Observe-noisy* plot is to show that the number of probes that the adversary requires can only remain same or increase when using a noisy policy, as opposed to using the correct agent policy. The noise in our experiments is that

two actions at each state of the MDP have incorrect probabilities. Each data point in the *Observe-noisy* case represents an average of 50 noisy policies, averaging over 5 noisy policies for each reward threshold over each of the 10 MDPs.

Figure 5.3-b plots a similar graph as 5.3-a for the multiagent case, averaged over the 10 UAV-team instances with two UAVs. The plot has three lines namely *Observe-all*, *Observe-select* and *Observe-noisy* with the same definitions as in the single agent case but in a distributed POMDP setting. However, in plot 5.3-b the y-axis represents joint number of yes-no probes and the x-axis represents joint entropy. Both these parameters are calculated as weighted sums of the individual parameters for each UAV, assuming that the adversary assigns equal weight to both the UAVs.

We conclude the following from plots 5.3-a and 5.3-b: (i) The number of observations (yes-no probes) increases as policy entropy increases, whether the adversary monitors the entire state space (*observe-all*) or just a part of it (*observe-select*). (ii) If the adversary obtains a noisy policy (*observe-noisy*), it needs a larger number of observations when compared to the adversary obtaining an accurate policy. (iii) As entropy increases, the agents' policy tends to become more uniform and hence the effect of noise on the number of yes-no probes reduces. In the extreme case where the policy is totally uniform the *Observe-all* and *Observe-noisy* both have same number of probes. This can be observed at the maximal entropy point in plot 5.3-a. The maximal entropy point is not reached in plot 5.3-b as shown in the results for RDR. From the above we conclude that maximizing entropy has indeed made it more difficult for the adversary to determine our agents' actions and cause harm. Hence randomization with quality constraints is an effective procedure to generate secure policies when the agent has no model of the adversary. In the next chapter I introduce the scenario when the agent has a partial model of the adversary. It is shown in that chapter, that randomized policies are still an effective way for avoiding adversarial

actions when there is a partial model of the adversary. However, the amount of randomization needed is dependent on the type of adversaries present in the domain.

# Partial Adversary Model: Limited Randomization Procedure

In the previous chapters we dealt with the problem of developing secure policies for agents when there is no model of the adversary. In the rest of this thesis I develop a limited randomization approach and an exact procedure for generating secure policies for agents assuming that there is a partial model of the adversary available. In particular I first introduce an efficient limited randomization approach for generating secure policies for agents in this chapter. The main advantage of this approach is that it is efficient compared to previously existing approaches and also the policies generated are simple and easy to implement in real applications. In the next chapter, I use the techniques developed here to develop an efficient exact solution.

In some settings, one player must commit to a strategy before the other players choose their strategies. These scenarios are known as Stackelberg games [Fudenberg and Tirole, 1991]. 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.* Stackelberg games are commonly used to model attacker-defender scenarios in security domains. [Brown et al., 2006]. For example consider a domain in which a single security agent is responsible for patrolling a region, searching for robbers. Since the security agent (the leader) cannot be in all areas of the region at once, it must instead choose some strategy of patrolling various areas within the region, one at a time. This strategy could be a mixed strategy in order to be

unpredictable to the robbers (followers). The robbers, after observing the pattern of patrols over time, can then choose their own strategy of choosing a location to rob. Similarly in terms of the domain described earlier, a team of unmanned aerial vehicles (UAVs) [Beard and McLain, 2003] monitoring a region undergoing a humanitarian crisis may also need to choose a patrolling policy. They must make this decision without knowing in advance whether terrorists or other adversaries may be waiting to disrupt the mission at a given location. It may indeed be possible to model the motivations of types of adversaries the agent or agent team is likely to face in order to target these adversaries more closely. However, in both cases, the security robot or UAV team will not know exactly which kinds of adversaries may be active on any given day.

Although the follower in a Stackelberg game is allowed to observe the leader's strategy before choosing its own strategy, there is often an advantage for the leader over the case where both players must choose their moves simultaneously. To see the advantage of being the leader in a Stackelberg game, consider a simple game with the payoff table as shown in Table 6.1, adapted from [Conitzer and Sandholm, 2006]. 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 |

Table 6.1: Payoff table for example normal form game.

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.

The problem of choosing an optimal strategy for the leader to commit to in a Stackelberg game is analyzed in [Conitzer and Sandholm, 2006] 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 [Conitzer and Sandholm, 2006] can be applied to this problem by converting the Bayesian game into a normal-form game by the Harsanyi transformation [Harsanyi and Selten, 1972]. However, by transforming the game, the compact structure of the Bayesian game is lost. In addition, this method requires running a set of multiple 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) [Sandholm et al., 2005] 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. Therefore, finding more efficient and compact techniques for choosing optimal strategies in instances of these games is an important open issue.

To address this open issue, I first introduce an efficient randomization method for generating optimal leader strategy for security domains that have limited randomization, known as ASAP (Agent Security via Approximate Policies). This method has three 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 like the one in the above example. Second, it generates policies with a support which can be expressed as a uniform distribution over a multiset of fixed size as proposed in [Lipton et al., 2003]. This allows for policies that are

simple to understand and represent, as well as a parameter (the size of the multiset) that controls the simplicity of the policy and can be tuned. Third, the method allows for a Bayes-Nash game to be expressed compactly without requiring conversion to a normal-form game, allowing for large speedups over existing Nash methods such as [Sandholm et al., 2005] and [Lemke and Hawson, 1964].

Using the techniques used in deriving the ASAP procedure, I then introduced an efficient exact method for finding the optimal leader strategy for security domains, known as DOBSS (Decomposed Optimal Bayesian Stackelberg Solver). This method has two key advantages. First, the method allows for a Bayesian game to be expressed compactly without requiring conversion to a normal-form game via the Harsanyi transformation. Second, the method requires only one mixed-integer linear program to be solved, rather than a set of such programs.

In most security patrolling domains, the security agents (which could be UAVs [Beard and McLain, 2003] or police [Ruan et al., 2005]) cannot feasibly patrol all areas all the time. Instead, they must choose a policy by which they patrol various routes at different times, taking into account factors such as the likelihood of crime in different areas, possible targets for crime, and the security agents' own resources (number of security agents, amount of available time, fuel, etc.). It is usually beneficial for this policy to be nondeterministic so that robbers cannot safely rob certain locations, knowing that they will be safe from the security agents as shown in the earlier chapters. To demonstrate the utility of our algorithm, we use a simplified version of the police patrolling domain, expressed as a bayesian game as described in chapter 2.

## 6.1 Bayesian Games

A Bayesian game contains a set of $N$ agents, and each agent $n$ must be one of a given set of types $\theta_n$. For our patrolling domain, we have two agents, the security agent and the robber. $\theta_1$ is the set of security agent types and $\theta_2$ is the set of robber types. Since there is only one type of security agent, $\theta_1$ contains only one element. During the game, the robber knows its type but the security agent does not know the robber's type. For each agent (the security agent or the robber) $n$, there is a set of strategies $\sigma_n$ and a utility function $u_n : \theta_1 \times \theta_2 \times \sigma_1 \times \sigma_2 \rightarrow \Re$.

A Bayesian game can be transformed into a normal-form game using the Harsanyi transformation [Harsanyi and Selten, 1972]. Once this is done, new, linear-program (LP)-based methods for finding high-reward strategies for normal-form games [Conitzer and Sandholm, 2006] can be used to find a strategy in the transformed game; this strategy can then be used for the Bayesian game. While methods exist for finding Bayes-Nash equilibria directly, without the Harsanyi transformation [Koller and Pfeffer, 1997], they find only a single equilibrium in the general case, which may not be of high reward. Recent work [Sandholm et al., 2005] has led to efficient mixed-integer linear program techniques to find the best Nash equilibrium for a given agent. However, these techniques do require a normal-form game, and so to compare the policies given by ASAP and DOBSS against the optimal policy, as well as against the highest-reward Nash equilibrium, we must apply these techniques to the Harsanyi-transformed matrix. The next two subsections elaborate on how this is done.

### 6.1.1 Harsanyi Transformation

The first step in solving Bayesian games is to apply the Harsanyi transformation [Harsanyi and Selten, 1972] that converts the incomplete information game into a normal form game. Given that the Harsanyi transformation is a standard concept in game theory, we explain it briefly through a simple example without introducing the mathematical formulations. Let us assume there are two robber types $a$ and $b$ in the Bayesian game. Robber $a$ will be active with probability $\alpha$, and robber $b$ will be active with probability $1 - \alpha$. The rules described in Section 2.3 allow us to construct simple payoff tables.

Assume that there are two houses in the world (1 and 2) and hence there are two patrol routes (pure strategies) for the agent: $\{1,2\}$ and $\{2,1\}$. The robber can rob either house 1 or house 2 and hence he has two strategies (denoted as $1_l$, $2_l$ for robber type l). Since there are two types assumed (denoted as $a$ and $b$), we construct two payoff tables (shown in Table 6.2) corresponding to the security agent playing a separate game with each of the two robber types with probabilities $\alpha$ and $1 - \alpha$. First, consider robber type $a$. Borrowing the notation from the domain section, we assign the following values to the variables: $v_{1,x} = v_{1,q} = 3/4, v_{2,x} = v_{2,q} = 1/4, c_x = 1/2, c_q = 1, p_1 = 1, p_2 = 1/2$. Using these values we construct a base payoff table as the payoff for the game against robber type $a$. For example, if the security agent chooses route $\{1,2\}$ when robber $a$ is active, and robber $a$ chooses house 1, the robber receives a reward of -1 (for being caught) and the agent receives a reward of 0.5 for catching the robber. The payoffs for the game against robber type $b$ are constructed using different values.

Using the Harsanyi technique involves introducing a chance node, that determines the robber's type, thus transforming the security agent's incomplete information regarding the robber

| Security agent: | {1,2} | {2,1} |
|---|---|---|
| **Robber** $a$ | | |
| $1_a$ | -1, .5 | -.375, .125 |
| $2_a$ | -.125, -.125 | -1, .5 |
| **Robber** $b$ | | |
| $1_b$ | -.9, .6 | -.275, .225 |
| $2_b$ | -.025, -.025 | -.9, .6 |

Table 6.2: Payoff tables: Security Agent vs Robbers $a$ and $b$

into imperfect information [Brynielsson and Arnborg, 2004]. The Bayesian equilibrium of the game is then precisely the Nash equilibrium of the imperfect information game. The transformed, normal-form game is shown in Table 6.3. In the transformed game, the security agent is the column player, and the set of all robber types together is the row player. Suppose that robber type

| | {1,2} | {2,1 } |
|---|---|---|
| $\{1_a, 1_b\}$ | $-1\alpha - .9(1-\alpha), .5\alpha + .6(1-\alpha)$ | $-.375\alpha - .275(1-\alpha), .125\alpha + .225(1-\alpha)$ |
| $\{1_a, 2_b\}$ | $-1\alpha - .025(1-\alpha), .5\alpha - .025(1-\alpha)$ | $-.375\alpha - .9(1-\alpha), .125\alpha + .6(1-\alpha)$ |
| $\{2_a, 1_b\}$ | $-.125\alpha - .9(1-\alpha), -.125\alpha + .6(1-\alpha)$ | $-1\alpha - .275(1-\alpha), .5\alpha + .225(1-\alpha)$ |
| $\{2_a, 2_b\}$ | $-.125\alpha - .025(1-\alpha), -.125\alpha - .025(1-\alpha)$ | $-1\alpha - .9(1-\alpha), .5\alpha + .6(1-\alpha)$ |

Table 6.3: Harsanyi Transformed Payoff Table

$a$ robs house 1 and robber type $b$ robs house 2, while the security agent chooses patrol $\{1,2\}$. Then, the security agent and the robber receive an expected payoff corresponding to their payoffs from the agent encountering robber $a$ at house 1 with probability $\alpha$ and robber $b$ at house 2 with probability $1-\alpha$.

### 6.1.2 Existing Procedure: Finding an Optimal Strategy

Although a Nash equilibrium is the standard solution concept for games in which agents choose strategies simultaneously, in our security domain, the security agent (the leader) can gain an advantage by committing to a mixed strategy in advance. Since the followers (the robbers) will know the leader's strategy, the optimal response for the followers will be a pure strategy. Given

the common assumption, taken in [Conitzer and Sandholm, 2006], in the case where followers are indifferent, they will choose the strategy that benefits the leader, there must exist a guaranteed optimal strategy for the leader [Conitzer and Sandholm, 2006].

From the Bayesian game in Table 6.2, we constructed the Harsanyi transformed bimatrix in Table 6.3. We denote $X = \sigma_1^{\theta_2} = \sigma_1$ and $Q = \sigma_2^{\theta_2}$ as the index sets of the security agent and robbers' pure strategies, respectively, with $R$ and $C$ as the corresponding payoff matrices. $R_{ij}$ is the reward of the security agent and $C_{ij}$ is the reward of the robbers when the security agent takes pure strategy $i$ and the robbers take pure strategy $j$. A mixed strategy for the security agent is a probability distribution over its set of pure strategies and will be represented by a vector $x = (p_{x1}, p_{x2}, \ldots, p_{x|X|})$, where $p_{xi} \geq 0$ and $\sum p_{xi} = 1$. Here, $p_{xi}$ is the probability that the security agent will choose its $i$th pure strategy.

The optimal mixed strategy for the security agent can be found in time polynomial in the number of rows in the normal form game using the following linear program formulation from [Conitzer and Sandholm, 2006].

For every possible pure strategy $j$ by the follower (the set of all robber types),

$$\begin{aligned} \max \quad & \textstyle\sum_{i \in X} p_{xi} R_{ij} \\ \text{s.t.} \quad & \forall j' \in Q, \textstyle\sum_{i \in \sigma_1} p_{xi} C_{ij} \geq \sum_{i \in \sigma_1} p_{xi} C_{ij'} \\ & \textstyle\sum_{i \in X} p_{xi} = 1 \\ & \forall_{i \in X}, p_{xi} = 0 \end{aligned} \tag{6.1}$$

Then, for all feasible follower strategies $j$, choose the one that maximizes $\sum_{i \in X} p_{xi} R_{ij}$, the reward for the security agent (leader). The $p_{xi}$ variables give the optimal strategy for the security agent.

Note that while this method is polynomial in the number of rows in the transformed, normal-form game, the number of rows increases exponentially with the number of robber types. Using this method for a Bayesian game thus requires running $|\sigma_2|^{|\theta_2|}$ separate linear programs. This is not a surprise, since finding the optimal strategy to commit to for the leader in a Bayesian game is NP-hard [Conitzer and Sandholm, 2006].

## 6.2 Limited Randomization Approach

In the limited randomization approach 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 $k$. In the previous chapters of this thesis, I have shown that strategies with high entropy are beneficial for security applications when opponents' utilities are completely unknown. In our domain, if utilities are not considered, this method will result in uniform-distribution strategies. One advantage of such strategies is that they are compact to represent (as fractions) and simple to understand; therefore they can be efficiently implemented by real organizations. We aim to maintain the advantage provided by simple strategies for our security application problem, incorporating the effect of the robbers' rewards on the security agent's rewards. Thus, the ASAP will produce strategies which are *k-uniform*. A mixed strategy is denoted $k$-uniform if it is a uniform distribution on a multiset $S$ of pure strategies with $|S| = k$. A multiset is a set whose elements may be repeated multiple times; thus, for example, the mixed strategy corresponding to the multiset $\{1,$

1, 2$\}$ would take strategy 1 with probability 2/3 and strategy 2 with probability 1/3. ASAP allows the size of the multiset to be chosen in order to balance the complexity of the strategy reached with the goal that the identified strategy will yield a high reward.

Another advantage of the ASAP procedure is that it operates directly on the compact Bayesian representation, without requiring the Harsanyi transformation. This is 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 ASAP to yield a decomposition scheme. Note that the LP method introduced by [Conitzer and Sandholm, 2006] to compute optimal Stackelberg policies is unlikely to be decomposable into a small number of games as it was shown to be NP-hard for Bayes-Nash problems. Finally, note that ASAP requires the solution of only one optimization problem, rather than solving a series of problems as in the LP method of [Conitzer and Sandholm, 2006].

For a single follower type, the algorithm works the following way. Given a particular $k$, for each possible mixed strategy $x$ for the leader that corresponds to a multiset of size $k$, evaluate the leader's payoff from $x$ when the follower plays a reward-maximizing pure strategy. We then take the mixed strategy with the highest payoff.

We need only to consider the reward-maximizing pure strategies of the followers (robbers), since for a given fixed strategy $x$ of the security agent, each robber type faces a problem with fixed linear rewards. If a mixed strategy is optimal for the robber, then so are all the pure strategies in the support of that mixed strategy. Note also that because we limit the leader's strategies to take on discrete values, the assumption from Section 6.1.2 that the followers will break ties in the leader's favor is not significant, since ties will be unlikely to arise. This is because, in domains

where rewards are drawn from any random distribution, the probability of a follower having more than one pure optimal response to a given leader strategy approaches zero, and the leader will have only a finite number of possible mixed strategies.

Our approach to characterize the optimal strategy for the security agent makes use of properties of linear programming. We briefly outline these results here for completeness, for detailed discussion and proofs see one of many references on the topic, such as [Bertsimas and Tsitsiklis, 1997]. Every linear programming problem, such as:

$$\max c^T x \mid Ax = b, x \geq 0,$$

has an associated dual linear program, in this case:

$$\min b^T y \mid A^T y \geq c.$$

These primal/dual pairs of problems satisfy weak duality: For any $x$ and $y$ primal and dual feasible solutions respectively, $c^T x \leq b^T y$. Thus a pair of feasible solutions is optimal if $c^T x = b^T y$, and the problems are said to satisfy strong duality. In fact if a linear program is feasible and has a bounded optimal solution, then the dual is also feasible and there is a pair $x^*, y^*$ that satisfies $c^T x^* = b^T y^*$. These optimal solutions are characterized with the following optimality conditions:

- primal feasibility: $Ax = b,\ x \geq 0$
- dual feasibility: $A^T y \geq c$
- complementary slackness: $x_i(A^T y - c)_i = 0$ for all $i$.

Note that this last condition implies that

$$c^T x = x^T A^T y = b^T y,$$

which proves optimality for primal dual feasible solutions $x$ and $y$.

In the following subsections, we first define the problem in its most intuitive form as a mixed-integer quadratic program, and then show how this problem can be converted into a mixed-integer linear program.

### 6.2.1 Mixed-Integer Quadratic Program

We begin with the case of a single type of follower. Let the leader be the row player and the follower the column player. We denote by $x$ the vector of strategies of the leader and $q$ 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 $C$ correspond to: $R_{ij}$ 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 $j$. Let $k$ be the size of the multiset.

We first fix the policy of the leader to some $k$-uniform policy $x$. The value $x_i$ is the number of times pure strategy $i$ is used in the $k$-uniform policy, which is selected with probability $x_i/k$.

We formulate the optimization problem the follower solves to find its optimal response to $x$ as the following linear program:

$$\begin{aligned} \max \quad & \sum_{j \in Q} \sum_{i \in X} \frac{1}{k} C_{ij} x_i \, q_j \\ \text{s.t.} \quad & \textstyle\sum_{j \in Q} q_j = 1 \\ & q \geq 0. \end{aligned} \tag{6.2}$$

The objective function maximizes the follower's expected reward given $x$, while the constraints make feasible any mixed strategy $q$ for the follower. The dual to this linear programming problem is the following:

$$\begin{aligned} \min \quad & a \\ \text{s.t.} \quad & a \geq \sum_{i \in X} \frac{1}{k} C_{ij} x_i \quad j \in Q. \end{aligned} \tag{6.3}$$

From strong duality and complementary slackness we obtain that the maximum reward value for the follower $a$ is the value of every pure strategy with $q_j > 0$, that is in the support of the optimal mixed strategy. Therefore each of these pure strategies is optimal. Optimal solutions to the follower's problem are characterized by linear programming optimality conditions: primal feasibility constraints in (6.2), dual feasibility constraints in (6.3), and complementary slackness

$$q_j \left( a - \sum_{i \in X} \frac{1}{k} C_{ij} x_i \right) = 0 \qquad j \in Q.$$

These conditions must be included in the problem solved by the leader in order to consider only best responses by the follower to the $k$-uniform policy $x$.

The leader seeks the $k$-uniform solution $x$ that maximizes its own payoff, given that the follower uses an optimal response $q(x)$. Therefore the leader solves the following integer problem:

$$\begin{aligned} \max \quad & \sum_{i \in X} \sum_{j \in Q} \frac{1}{k} R_{ij} q(x)_j \; x_i \\ \text{s.t.} \quad & \textstyle\sum_{i \in X} x_i = k \\ & x_i \in \{0, 1, \ldots, k\}. \end{aligned} \tag{6.4}$$

Problem (6.4) maximizes the leader's reward with the follower's best response ($q_j$ for fixed leader's policy $x$ and hence denoted $q(x)_j$) by selecting a uniform policy from a multiset of constant size $k$. We complete this problem by including the characterization of $q(x)$ through linear programming optimality conditions. To simplify writing the complementary slackness conditions, we will constrain $q(x)$ to be only optimal pure strategies by just considering integer solutions of $q(x)$. The leader's problem becomes:

$$\begin{aligned} \max_{x,q} \quad & \sum_{i \in X} \sum_{j \in Q} \frac{1}{k} R_{ij} x_i q_j \\ \text{s.t.} \quad & \textstyle\sum_i x_i = k \\ & \textstyle\sum_{j \in Q} q_j = 1 \\ & 0 \leq (a - \textstyle\sum_{i \in X} \frac{1}{k} C_{ij} x_i) \leq (1 - q_j) M \\ & x_i \in \{0, 1, ...., k\} \\ & q_j \in \{0, 1\}. \end{aligned} \tag{6.5}$$

Here, the constant $M$ is some large number. The first and fourth constraints enforce a $k$-uniform 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). In fact, since only one pure strategy can be selected by the follower, say $q_h = 1$, this last constraint enforces that $a = \sum_{i \in X} \frac{1}{k} C_{ih} x_i$ imposing no additional constraint for all other pure strategies which have $q_j = 0$.

We conclude this subsection noting that Problem (6.5) is an integer program with a non-convex quadratic objective in general, as the matrix $R$ need not be positive-semi-definite. Efficient solution methods for non-linear, non-convex integer problems remains a challenging research question. In the next section we show a reformulation of this problem as a linear integer programming problem, for which a number of efficient commercial solvers exist.

### 6.2.2 Mixed-Integer Linear Program

We can linearize the quadratic program of Problem 6.5 through the change of variables $z_{ij} = x_i q_j$, obtaining the following problem

$$
\begin{aligned}
\max_{q,z} \quad & \textstyle\sum_{i \in X} \sum_{j \in Q} \frac{1}{k} R_{ij} z_{ij} \\
\text{s.t.} \quad & \textstyle\sum_{i \in X} \sum_{j \in Q} z_{ij} = k \\
& \textstyle\sum_{j \in Q} z_{ij} \leq k \\
& k q_j \leq \textstyle\sum_{i \in X} z_{ij} \leq k \\
& \textstyle\sum_{j \in Q} q_j = 1 \\
& 0 \leq (a - \textstyle\sum_{i \in X} \frac{1}{k} C_{ij} (\sum_{h \in Q} z_{ih})) \leq (1 - q_j) M \\
& z_{ij} \in \{0, 1, ...., k\} \\
& q_j \in \{0, 1\}
\end{aligned}
\tag{6.6}
$$

**Theorem 2** *Problems (6.5) and (6.6) are equivalent.*

**Proof:** Consider $x$, $q$ a feasible solution of (6.5). We will show that $q$, $z_{ij} = x_i q_j$ is a feasible solution of (6.6) of same objective function value. The equivalence of the objective functions, and constraints 4, 6 and 7 of (6.6) are satisfied by construction. The fact that $\sum_{j \in Q} z_{ij} = x_i$ as $\sum_{j \in Q} q_j = 1$ explains constraints 1, 2, and 5 of (6.6). Constraint 3 of (6.6) is satisfied because $\sum_{i \in X} z_{ij} = k q_j$.

Let us now consider $q, z$ feasible for (6.6). We will show that $q$ and $x_i = \sum_{j \in Q} z_{ij}$ are feasible for (6.5) with the same objective value. In fact all constraints of (6.5) are readily satisfied by construction. To see that the objectives match, notice that if $q_h = 1$ then the third constraint in (6.6) implies that $\sum_{i \in X} z_{ih} = k$, which means that $z_{ij} = 0$ for all $i \in X$ and all $j \neq h$. Therefore,

$$x_i q_j = \sum_{l \in Q} z_{il} q_j = z_{ih} q_j = z_{ij}.$$

This last equality is because both are 0 when $j \neq h$. This shows that the transformation preserves the objective function value, completing the proof.

Given this transformation to a mixed-integer linear program (MILP), we now show how we can apply our decomposition technique on the MILP to obtain significant speedups for Bayesian games with multiple follower types.

## 6.3 Decomposition for Multiple Adversaries

The MILP developed in the previous section handles only one follower. Since our security scenario contains multiple follower (robber) types, we change the response function for the follower from a pure strategy into a weighted combination over various pure follower strategies where the weights are probabilities of occurrence of each of the follower types.

### 6.3.1 Decomposed MIQP

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 $C^l$.

Using this modified notation, we characterize the optimal solution of follower $l$'s problem given the leaders k-uniform policy $x$, with the following optimality conditions:

$$\begin{aligned}
\sum_{j \in Q} q_j^l &= 1 \\
a^l - \sum_{i \in X} \frac{1}{k} C_{ij}^l x_i &\geq 0 \\
q_j^l (a^l - \sum_{i \in X} \frac{1}{k} C_{ij}^l x_i) &= 0 \\
q_j^l &\geq 0
\end{aligned}$$

Again, considering only optimal pure strategies for follower $l$'s problem we can linearize the complementarity constraint above. We incorporate these constraints on the leader's problem that

selects the optimal $k$-uniform policy. Therefore, given *a priori* probabilities $p^l$, with $l \in L$ of facing each follower, the leader solves the following problem:

$$\begin{aligned} \max_{x,q} \quad & \sum_{i\in X}\sum_{l\in L}\sum_{j\in Q}\frac{p^l}{k}R^l_{ij}x_i q^l_j \\ \text{s.t.} \quad & \textstyle\sum_i x_i = k \\ & \textstyle\sum_{j\in Q} q^l_j = 1 \\ & 0 \le (a^l - \textstyle\sum_{i\in X}\frac{1}{k}C^l_{ij}x_i) \le (1-q^l_j)M \\ & x_i \in \{0,1,....,k\} \\ & q^l_j \in \{0,1\}. \end{aligned} \tag{6.7}$$

Problem (6.7) for a Bayesian game with multiple follower types is indeed equivalent to Problem (6.5) on the payoff matrix obtained from the Harsanyi transformation of the game. In fact, every pure strategy $j$ in Problem (6.5) corresponds to a sequence of pure strategies $j_l$, one for each follower $l \in L$. This means that $q_j = 1$ if and only if $q^l_{j_l} = 1$ for all $l \in L$. In addition, given the *a priori* probabilities $p^l$ of facing player $l$, the reward in the Harsanyi transformation payoff table is $R_{ij} = \sum_{l\in L} p^l R^l_{ij_l}$. The same relation holds between $C$ and $C^l$. These relations between a pure strategy in the equivalent normal form game and pure strategies in the individual games with each followers are key in showing these problems are equivalent.

### 6.3.2 Decomposed MILP

We can linearize the quadratic programming problem 6.7 through the change of variables $z_{ij}^l = x_i q_j^l$, obtaining the following problem

$$
\begin{aligned}
\max_{q,z} \quad & \textstyle\sum_{i\in X}\sum_{l\in L}\sum_{j\in Q}\frac{p^l}{k}R_{ij}^l z_{ij}^l \\
\text{s.t.} \quad & \textstyle\sum_{i\in X}\sum_{j\in Q} z_{ij}^l = k \\
& \textstyle\sum_{j\in Q} z_{ij}^l \leq k \\
& k q_j^l \leq \textstyle\sum_{i\in X} z_{ij}^l \leq k \\
& \textstyle\sum_{j\in Q} q_j^l = 1 \\
& 0 \leq (a^l - \textstyle\sum_{i\in X}\frac{1}{k}C_{ij}^l(\sum_{h\in Q} z_{ih}^l)) \leq (1-q_j^l)M \\
& \textstyle\sum_{j\in Q} z_{ij}^l = \sum_{j\in Q} z_{ij}^1 \\
& z_{ij}^l \in \{0,1,....,k\} \\
& q_j^l \in \{0,1\}
\end{aligned}
\tag{6.8}
$$

**Theorem 3** *Problems (6.7) and (6.8) are equivalent.*

**Proof:** Consider $x$, $q^l$, $a^l$ with $l \in L$ a feasible solution of (6.7). We will show that $q^l$, $a^l$, $z_{ij}^l = x_i q_j^l$ is a feasible solution of (6.8) of same objective function value. The equivalence of the objective functions, and constraints 4, 7 and 8 of (6.8) are satisfied by construction. The fact that $\sum_{j\in Q} z_{ij}^l = x_i$ as $\sum_{j\in Q} q_j^l = 1$ explains constraints 1, 2, 5 and 6 of (6.8). Constraint 3 of (6.8) is satisfied because $\sum_{i\in X} z_{ij}^l = k q_j^l$.

Lets now consider $q^l, z^l, a^l$ feasible for (6.8). We will show that $q^l$, $a^l$ and $x_i = \sum_{j\in Q} z_{ij}^1$ are feasible for (6.7) with the same objective value. In fact all constraints of (6.7) are readily satisfied

by construction. To see that the objectives match, notice for each $l$ one $q_j^l$ must equal 1 and the rest equal 0. Let us say that $q_{j_l}^l = 1$, then the third constraint in (6.8) implies that $\sum_{i \in X} z_{ij_l}^l = k$, which means that $z_{ij}^l = 0$ for all $i \in X$ and all $j \neq j_l$. In particular this implies that

$$x_i = \sum_{j \in Q} z_{ij}^1 = z_{ij_1}^1 = z_{ij_l}^l,$$

this last equality from constraint 6 of (6.8). Therefore $x_i q_j^l = z_{ij_l}^l q_j^l = z_{ij}^l$. This last equality is because both are 0 when $j \neq j_l$. 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.

## 6.4 Experimental results

The patrolling domain and the payoffs for the associated game are detailed in Sections 2.3 and 6.1. We performed experiments for this game in worlds of three and four houses with patrols consisting of two houses. The description given in Section 2.3 is used to generate a base case for both the security agent and robber payoff functions. The payoff tables for additional robber types are constructed and added to the game by adding a random distribution of varying size to the payoffs in the base case. All games are normalized so that, for each robber type, the minimum and maximum payoffs to the security agent and robber are 0 and 1, respectively.

Using the data generated, we performed the experiments using four methods for generating the security agent's strategy:

- uniform randomization
- ASAP
- the multiple linear programs method from [Conitzer and Sandholm, 2006] (to find the true optimal strategy)
- the highest reward Bayes-Nash equilibrium, found using the MIP-Nash algorithm [Sandholm et al., 2005]

The last three methods were applied using CPLEX 8.1. Because the last two methods are designed for normal-form games rather than Bayesian games, the games were first converted using the Harsanyi transformation [Harsanyi and Selten, 1972]. The uniform randomization method is simply choosing a uniform random policy over all possible patrol routes. We use this method as a simple baseline to measure the performance of other procedures. We anticipated that the uniform policy would perform reasonably well since maximum-entropy policies have been shown to be effective in multiagent security domains. The highest-reward Bayes-Nash equilibria were used in order to demonstrate the higher reward gained by looking for an optimal policy rather than an equilibria in Stackelberg games such as our security domain.

Based on our experiments we present three sets of graphs to demonstrate (1) the runtime of ASAP compared to other common methods for finding a strategy, (2) the reward guaranteed by ASAP compared to other methods, and (3) the effect of varying the parameter $k$, the size of the multiset, on the performance of ASAP. In the first two sets of graphs, ASAP is run using a multiset of 80 elements; in the third set this number is varied.

The first set of graphs, shown in Figure 6.1 shows the runtime graphs for three-house (left column) and four-house (right column) domains. Each of the three rows of graphs corresponds

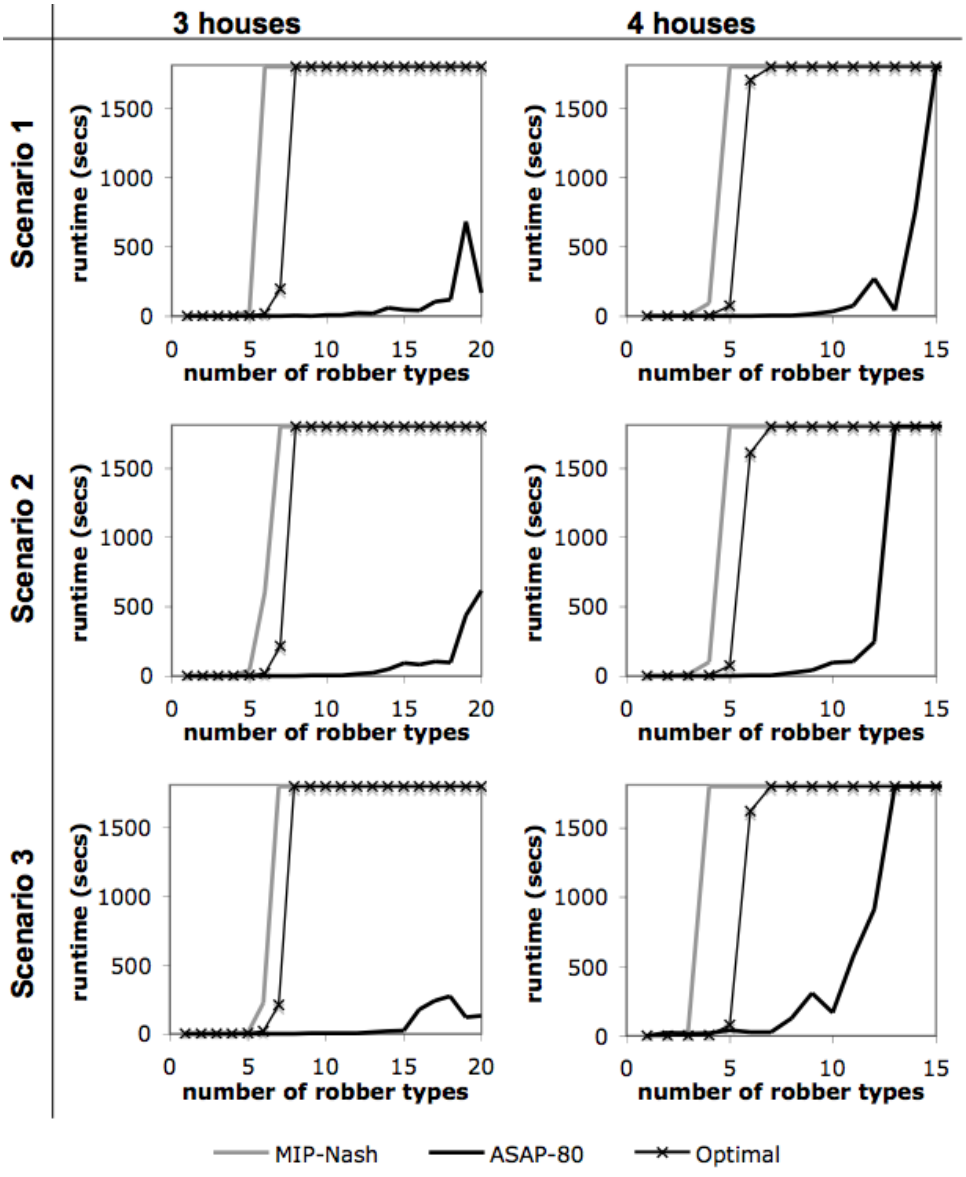



Figure 6.1: Runtimes for various algorithms on problems of 3 and 4 houses.

to a different randomly-generated scenario. The $x$-axis shows the number of robber types the security agent faces and the $y$-axis of the graph shows the runtime in seconds. All experiments that were not concluded in 30 minutes (1800 seconds) were cut off. The runtime for the uniform policy is always negligible irrespective of the number of adversaries and hence is not shown.

The ASAP algorithm clearly outperforms the optimal, multiple-LP method as well as the MIP-Nash algorithm for finding the highest-reward Bayes-Nash equilibrium with respect to runtime. For a domain of three houses, the optimal method cannot reach a solution for more than seven robber types, and for four houses it cannot solve for more than six types within the cutoff time in any of the three scenarios. MIP-Nash solves for even fewer robber types within the cutoff time. On the other hand, ASAP runs much faster, and is able to solve for at least 20 adversaries for the three-house scenarios and for at least 12 adversaries in the four-house scenarios within the cutoff time. The runtime of ASAP does not increase strictly with the number of robber types for each scenario, but in general, the addition of more types increases the runtime required.

The second set of graphs, Figure 6.2, shows the reward to the patrol agent given by each method for three scenarios in the three-house (left column) and four-house (right column) domains. This reward is the utility received by the security agent in the patrolling game, and not as a percentage of the optimal reward, since it was not possible to obtain the optimal reward as the number of robber types increased. The uniform policy consistently provides the lowest reward in both domains; while the optimal method of course produces the optimal reward. The ASAP method remains consistently close to the optimal, even as the number of robber types increases. The highest-reward Bayes-Nash equilibria, provided by the MIP-Nash method, produced rewards higher than the uniform method, but lower than ASAP. This difference clearly illustrates the gains

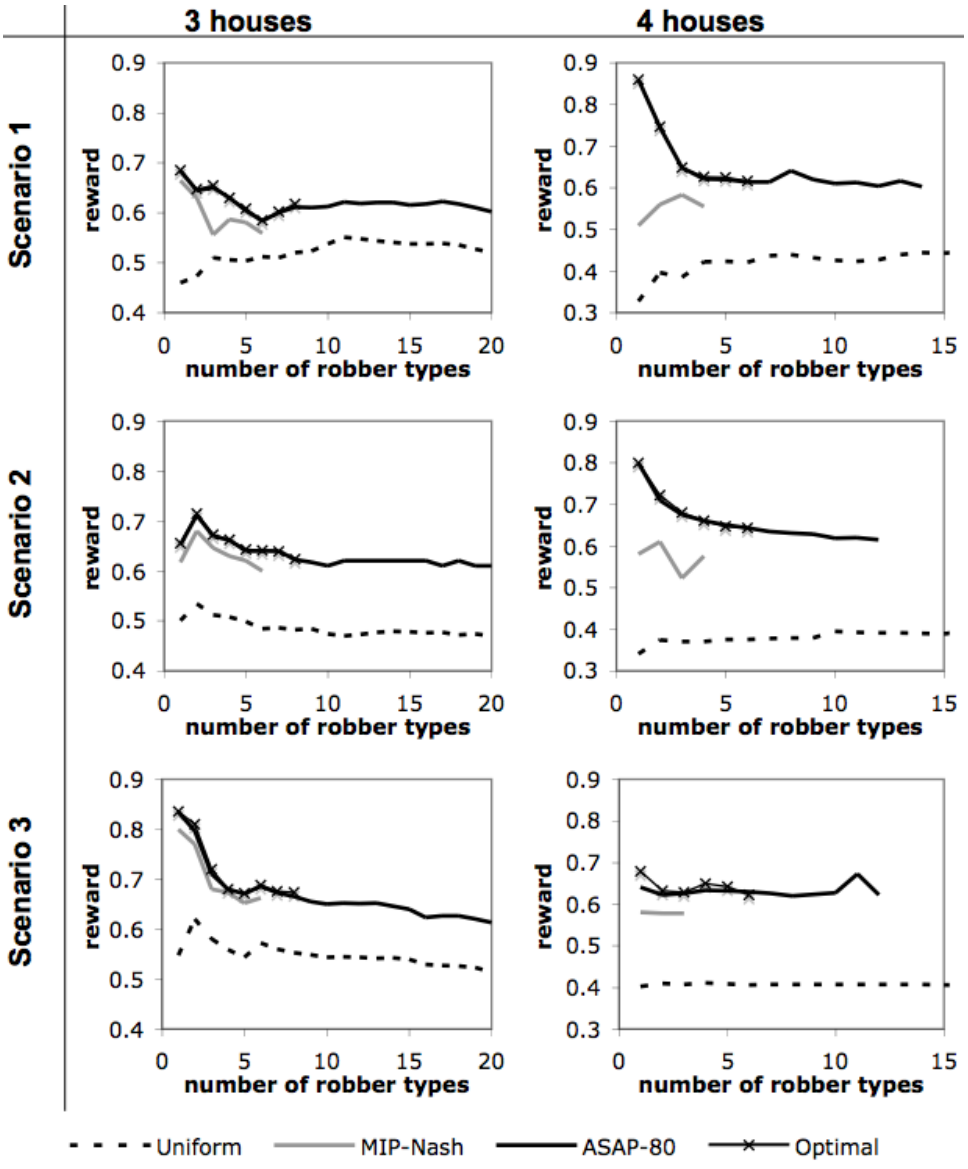



Figure 6.2: Reward for various algorithms on problems of 3 and 4 houses.

in the patrolling domain from committing to a strategy as the leader in a Stackelberg game, rather than playing a standard Bayes-Nash strategy.

The third set of graphs, shown in Figure 6.3 shows the effect of the multiset size on runtime in seconds (left column) and reward (right column), again expressed as the reward received by the security agent in the patrolling game, and not a percentage of the optimal reward. Results here are for the three-house domain. The trend is that as as the multiset size is increased, the runtime and reward level both increase. Not surprisingly, the reward increases monotonically as the multiset size increases, but what is interesting is that there is relatively little benefit to using a large multiset in this domain. In all cases, the reward given by a multiset of 10 elements was within at least 96% of the reward given by an 80-element multiset. The runtime does not always increase strictly with the multiset size; indeed in one example (scenario 2 with 20 robber types), using a multiset of 10 elements took 1228 seconds, while using 80 elements only took 617 seconds. In general, runtime should increase since a larger multiset means a larger domain for the variables in the MILP, and thus a larger search space. However, an increase in the number of variables can sometimes allow for a policy to be constructed more quickly due to more flexibility in the problem.

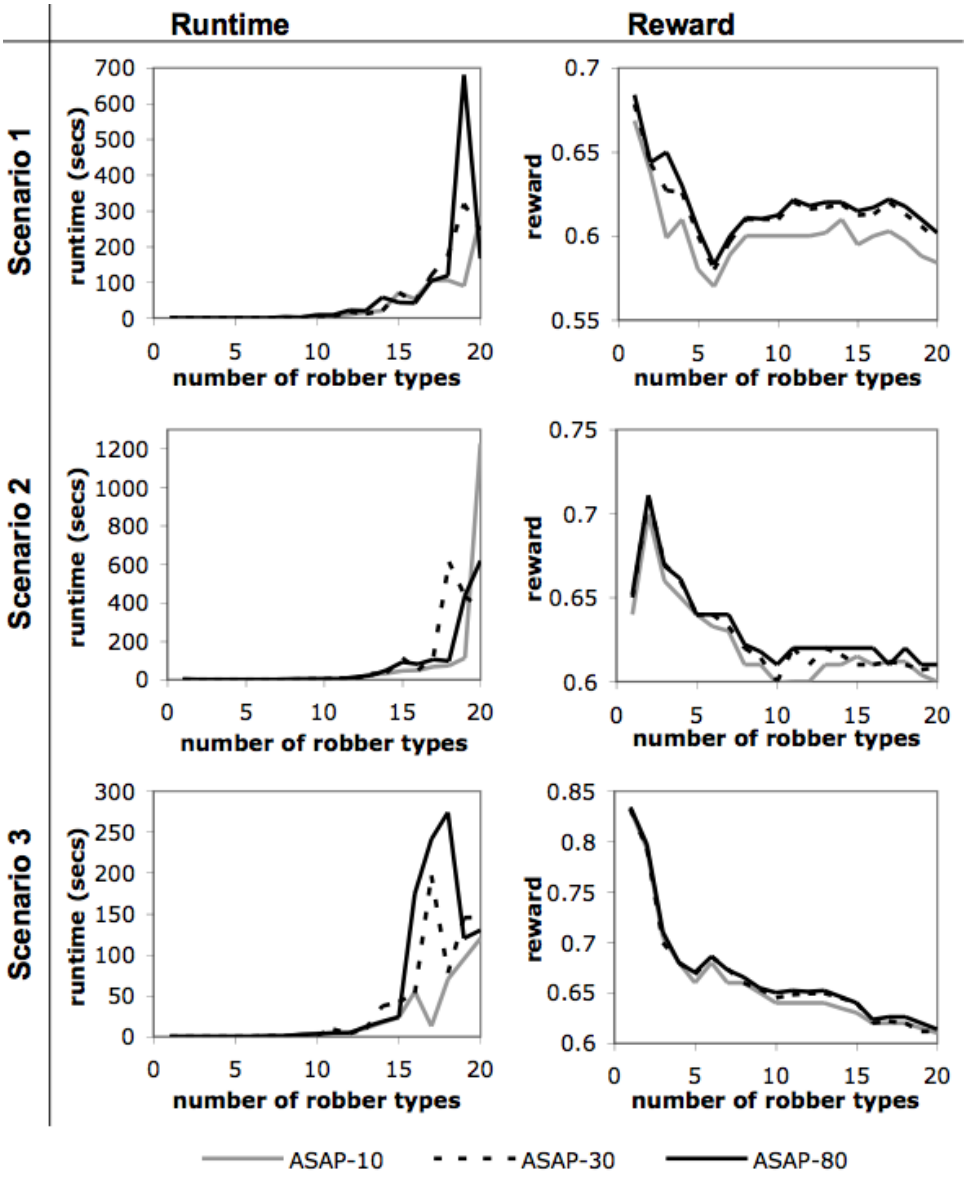



Figure 6.3: Reward for ASAP using multisets of 10, 30, and 80 elements

# Partial Adversary Model: Exact Solution

The ASAP procedure introduced in the previous chapter can operate directly on the untransformed Bayesian game to find a high-quality strategy for the leader; however it does not guarantee an optimal solution. In this chapter we introduce the DBOSS algorithm which guarantees an optimal solution.

## 7.1 Approach

Similar to the ASAP procedure, one advantage of the DOBSS algorithm is that it operates directly on the compact Bayesian representation, without requiring the Harsanyi transformation. This is because the different follower 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. This decomposition is analogous to that of the ASAP method shown in the previous chapter; however, while ASAP provides limited randomization procedure for choosing the leader's strategy, DOBSS provides the optimal solution without any bounds on randomization. In addition, our experiments show that ASAP is significantly slower than DOBSS as the number of follower types increases and may be unable to provide a solution in many cases (where DOBSS is able to). Note that the LP method introduced by [Conitzer and

Sandholm, 2006] to compute optimal Stackelberg policies is unlikely to be decomposable into a small number of games as it was shown to be NP-hard for Bayes-Nash problems; DOBSS has the advantage of decomposition, but must work with mixed-integer linear programs (MILPs) rather than LPs. Finally, note that DOBSS requires the solution of only one optimization problem, rather than solving a series of problems as in the LP method of [Conitzer and Sandholm, 2006].

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.

In the following subsections, we first define the problem in its most intuitive form as a mixed-integer quadratic program, and then show how this problem can be decomposed and converted into an MILP. For a detailed discussion of MILPs, please see one of many references on the topic, such as [Wolsey, 1998].

### 7.1.1 Mixed-Integer Quadratic Program

We begin with the case of a single type of follower. Let the leader be the row player and the follower the column player. We denote by $x$ the vector of strategies of the leader and $q$ 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 $x$. The value $x_i$ is the proportion of times in which pure strategy $i$ is used in the policy. We formulate the optimization problem the follower solves to find its optimal response to $x$ as the following linear program:

$$\begin{aligned} \max \quad & \sum_{j \in Q} \sum_{i \in X} C_{ij} x_i \, q_j \\ \text{s.t.} \quad & \textstyle\sum_{j \in Q} q_j = 1 \\ & q \geq 0. \end{aligned} \tag{7.1}$$

The objective function maximizes the follower's expected reward given $x$, while the constraints make feasible any mixed strategy $q$ for the follower. The dual to this linear programming problem is the following:

$$\begin{aligned} \min \quad & a \\ \text{s.t.} \quad & a \geq \sum_{i \in X} C_{ij} x_i \quad j \in Q. \end{aligned} \tag{7.2}$$

From strong duality and complementary slackness we obtain that the follower's maximum reward value, $a$, is the value of every pure strategy with $q_j > 0$, that is in the support of the optimal mixed strategy. Therefore each of these pure strategies is optimal. Optimal solutions to the follower's problem are characterized by linear programming optimality conditions: primal feasibility constraints in (7.1), dual feasibility constraints in (7.2), and complementary slackness

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

These conditions must be included in the problem solved by the leader in order to consider only best responses by the follower to the policy $x$.

The leader seeks the solution $x$ that maximizes its own payoff, given that the follower uses an optimal response $q(x)$. Therefore the leader solves the following integer problem:

$$\begin{aligned} \max \quad & \sum_{i \in X} \sum_{j \in Q} R_{ij} q(x)_j \; x_i \\ \text{s.t.} \quad & \textstyle\sum_{i \in X} x_i = 1 \\ & x_i \in [0 \ldots 1]. \end{aligned} \tag{7.3}$$

Problem (7.3) maximizes the leader's reward with the follower's best response ($q_j$ for fixed leader's policy $x$ and hence denoted $q(x)_j$). We complete this problem by including the characterization of $q(x)$ through linear programming optimality conditions. To simplify writing the complementary slackness conditions, we will constrain $q(x)$ to be only optimal pure strategies by just considering integer solutions of $q(x)$. The leader's problem becomes:

$$\begin{aligned} \max_{x,q} \quad & \sum_{i \in X} \sum_{j \in Q} R_{ij} x_i q_j \\ \text{s.t.} \quad & \textstyle\sum_i x_i = 1 \\ & \textstyle\sum_{j \in Q} q_j = 1 \\ & 0 \leq (a - \textstyle\sum_{i \in X} C_{ij} x_i) \leq (1 - q_j) M \\ & x_i \in [0 \ldots 1] \\ & q_j \in \{0, 1\}. \end{aligned} \tag{7.4}$$

Here, the constant $M$ is some large number. 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). In fact, since only one pure strategy can be selected by the follower, say $q_h = 1$, this last constraint enforces that $a = \sum_{i \in X} C_{ih} x_i$ imposing no additional constraint for all other pure strategies which have $q_j = 0$.

We conclude this subsection noting that Problem (7.4) is an integer program with a non-convex quadratic objective. Efficient solution methods for non-linear, non-convex integer problems remains a challenging research question. In the next section we show how this formulation can be decomposed in order to handle multiple follower types without requiring the Harsanyi transformation.

### 7.1.2 Decomposed MIQP

The MIQP developed in the previous section handles only one follower. To extend this Stackelberg model to handle multiple follower types we follow a Bayesian approach and assume that there is an a priori probability $p^l$ that a follower of type $l$ will appear, with $L$ denoting the set of follower types. We also adapt the notation defined in the previous section to reason... strategies of follower $l$. 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 $C^l$.

Using this modified notation, we characterize the optimal solution of follower $l$'s problem given the leader's policy $x$, with the following optimality conditions:

$$\begin{aligned} \sum_{j \in Q} q_j^l &= 1 \\ a^l - \sum_{i \in X} C_{ij}^l x_i &\geq 0 \\ q_j^l (a^l - \sum_{i \in X} C_{ij}^l x_i) &= 0 \\ q_j^l &\geq 0 \end{aligned}$$

Again, considering only optimal pure strategies for follower $l$'s problem we can linearize the complementarity constraint above. We incorporate these constraints on the leader's problem that selects the optimal policy. Therefore, given *a priori* probabilities $p^l$, with $l \in L$ of facing each follower, the leader solves the following problem:

$$\begin{aligned} \max_{x,q} \quad & \sum_{i \in X} \sum_{l \in L} \sum_{j \in Q} p^l R_{ij}^l x_i q_j^l \\ \text{s.t.} \quad & \textstyle\sum_i x_i = 1 \\ & \textstyle\sum_{j \in Q} q_j^l = 1 \\ & 0 \leq (a^l - \textstyle\sum_{i \in X} C_{ij}^l x_i) \leq (1 - q_j^l) M \\ & x_i \in [0 \ldots 1] \\ & q_j^l \in \{0, 1\}. \end{aligned} \tag{7.5}$$

For a Bayesian Stackelberg game with multiple follower types we will show that the decomposed MIQP of Problem (7.5) above is equivalent to the MIQP of Problem (7.4) on the payoff matrix obtained from the Harsanyi transformation of the game.

The Harsanyi transformation introduces a chance node that determines the follower's type according to the a priori probabilities $p^l$. For the leader it is as if there is a single follower, whose action set is the cross product of the actions of every follower type. Each pure strategy $q$ of the single follower in the transformed game corresponds to a vector of pure strategies $(q^1_{j_1}, \ldots, q^{|L|}_{j_{|L|}})$ one for each follower type, and the rewards for these actions need to be weighted by the probabilities of occurrence of each follower. Thus, the reward matrices for the Harsanyi transformation are constructed from the individual reward matrices as follows, as in [Harsanyi and Selten, 1972]:

$$R_{ij} = \sum_{l \in L} p^l R^l_{ij} \text{ and } \mathrm{C_{ij}} = \sum_{l \in L} p^l C^l ij. \tag{7.6}$$

**Theorem 4** *Problem (7.5) for a Bayesian game with multiple follower types is equivalent to Problem (7.4) on the payoff matrices given by the Harsanyi transformation (7.6).*

**Proof:** We first notice that if a pure strategy $j$ corresponds to $(j_1, \ldots, j_{|L|})$ and $q_j = 1$ then

$$\sum_{i \in X} \sum_{h \in Q} x_i R_{ih} q_h = \sum_{i \in X} x_i \sum_{l \in L} p^l R^l_{ij_l} = \sum_{i \in X} \sum_{l \in L} p^l x_i \sum_{h \in Q} R^l_{ih} q^l_h. \tag{7.7}$$

Consider $x, q^l, a^l$ with $l \in L$ a feasible solution to Problem (7.5). From its second constraint and integrality of $q$ we have that for every $l$ there is a $j_l$ such that $q^l_{j_l} = 1$. Let $j$ be the Harsanyi pure strategy that corresponds to $(j_1, \ldots, j_{|L|})$. Note that $q_j = 1$ and thus the objectives match. We show that $x$, $q$ such that $q_j = 1$ and $q_h = 0$ for $h \neq j$, and $a = \sum_{l \in L} p^l a^l$ is feasible for Problem (7.4) and of the same objective function value. Constraints 1, 2, 4, 5 in (7.4) are easily

satisfied by the proposed solution. Constraint 3 in (7.5) means that $\sum_{i\in X} x_i C^l_{ij_l} \geq \sum_{i\in X} x_i C^l_{ih}$, for every $h \in J$ and $l \in L$, leading to

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

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

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

Now we start with $(x, q, a)$ feasible for (7.4). This means that $q_j = 1$ for some pure strategy $j$. Let $(j_1, \ldots, j_{|L|})$ be the corresponding strategies 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 (7.5). By construction this solution satisfies constraints 1, 2, 4, 5 and has a matching objective function. We now show that $\sum_{i\in X} x_i C^l_{ij_l} \geq \sum_{i\in X} x_i C^l_{ih}$ 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 (7.4) implies that $\sum_{i\in X} x_i C_{ij} \geq \sum_{i\in X} x_i C_{ih}$ for all $h$.

### 7.1.3 Decomposed MILP

We can linearize the quadratic programming problem 7.5 through the change of variables $z_{ij}^l = x_i q_j^l$, obtaining the following problem

$$
\begin{aligned}
\max_{q,z} \quad & \textstyle\sum_{i\in X}\sum_{l\in L}\sum_{j\in Q} p^l R_{ij}^l z_{ij}^l \\
\text{s.t.} \quad & \textstyle\sum_{i\in X}\sum_{j\in Q} z_{ij}^l = 1 \\
& \textstyle\sum_{j\in Q} z_{ij}^l \leq 1 \\
& q_j^l \leq \textstyle\sum_{i\in X} z_{ij}^l \leq 1 \\
& \textstyle\sum_{j\in Q} q_j^l = 1 \\
& 0 \leq (a^l - \textstyle\sum_{i\in X} C_{ij}^l (\sum_{h\in Q} z_{ih}^l)) \leq (1 - q_j^l)M \\
& \textstyle\sum_{j\in Q} z_{ij}^l = \sum_{j\in Q} z_{ij}^1 \\
& z_{ij}^l \in [0 \ldots 1] \\
& q_j^l \in \{0, 1\}
\end{aligned}
\tag{7.8}
$$

**Theorem 5** *Problems (7.5) and (7.8) are equivalent.*

**Proof:** Consider $x$, $q^l$, $a^l$ with $l \in L$ a feasible solution of (7.5). We will show that $q^l$, $a^l$, $z_{ij}^l = x_i q_j^l$ is a feasible solution of (7.8) of same objective function value. The equivalence of the objective functions, and constraints 4, 7 and 8 of (7.8) are satisfied by construction. The fact that $\sum_{j\in Q} z_{ij}^l = x_i$ as $\sum_{j\in Q} q_j^l = 1$ explains constraints 1, 2, 5 and 6 of (7.8). Constraint 3 of (7.8) is satisfied because $\sum_{i\in X} z_{ij}^l = q_j^l$.

Lets now consider $q^l, z^l, a^l$ feasible for (7.8). We will show that $q^l$, $a^l$ and $x_i = \sum_{j\in Q} z_{ij}^1$ are feasible for (7.5) with the same objective value. In fact all constraints of (7.5) are readily satisfied

by construction. To see that the objectives match, notice for each $l$ one $q_j^l$ must equal 1 and the rest equal 0. Let us say that $q_{j_l}^l = 1$, then the third constraint in (7.8) implies that $\sum_{i \in X} z_{ij_l}^l = 1$, which means that $z_{ij}^l = 0$ for all $i \in X$ and all $j \neq j_l$. In particular this implies that

$$x_i = \sum_{j \in Q} z_{ij}^1 = z_{ij_1}^1 = z_{ij_l}^l,$$

the last equality from constraint 6 of (7.8). Therefore $x_i q_j^l = z_{ij_l}^l q_j^l = z_{ij}^l$. This last equality is because both are 0 when $j \neq j_l$. Effectively, constraint 6 ensures that all the adversaries are calculating their best responses against a particular fixed policy of the agent. 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.

## 7.2 Experimental Results

We performed two sets of experiments. The first set of experiments compares the runtime of DOBSS against ASAP as well as the multiple linear programs method of [Conitzer and Sandholm, 2006]. Note that ASAP provides an approximation, while the other two methods provide the optimal solution. In addition, the method of [Conitzer and Sandholm, 2006] requires a normal-form game, and so the Harsanyi transformation is required as an initial step; this preprocessing time is not recorded here. In this set of experiments, we created games in worlds of three and four houses with patrols consisting of two houses, with payoff tables as described in the previous subsection.

The set of graphs in Figure 7.1 shows the runtime graphs for three-house (left column) and four-house (right column) domains. Each of the graphs corresponds to a differently randomly-generated scenario. The $x$-axis shows the number of follower types the leader faces and the $y$-axis of the graph shows the runtime in seconds. All experiments that were not concluded in 30 minutes (1800 seconds) were cut off.

DOBSS outperforms the multiple LPs method with respect to runtime. For a domain of three houses, the LPs method cannot reach a solution of more than seven follower types, and for four houses it cannot solve for more than six types within the cutoff time in any of the three scenarios. On the other hand, DOBSS method runs much faster, and is able to solve for at least 20 adversaries for the three-house scenarios and for at least 12 adversaries in the four-house scenario within the cutoff time. The reason for the speedup for the DBOSS procedure over the previous procedure i.e., the multiple LPs method is as follows: The previous approach solves an LP over the exponentially blown harsanyi transformed matrix for each joint strategy of the adversaries (also exponential in number). In contrast, the DBOSS procedure introduces one integer per strategy for each new adversary. Branch and bound procedure in the DBOSS formulation would then lead to one exponent blowup thus saving the other exponent.

The runtime of DOBSS does not always increase strictly as the number of follower types increases. However the trend is that the addition of more types increases the runtime. The runtimes of DOBSS and ASAP are comparable in all scenarios; however DOBSS actually finds the optimal solution, while ASAP is a non-optimal procedure that only approximates the solution. For example, when the problem is scaled up in number of leader strategies as discussed below, the limited randomization character of ASAP creates difficulties in obtaining a feasible solution.

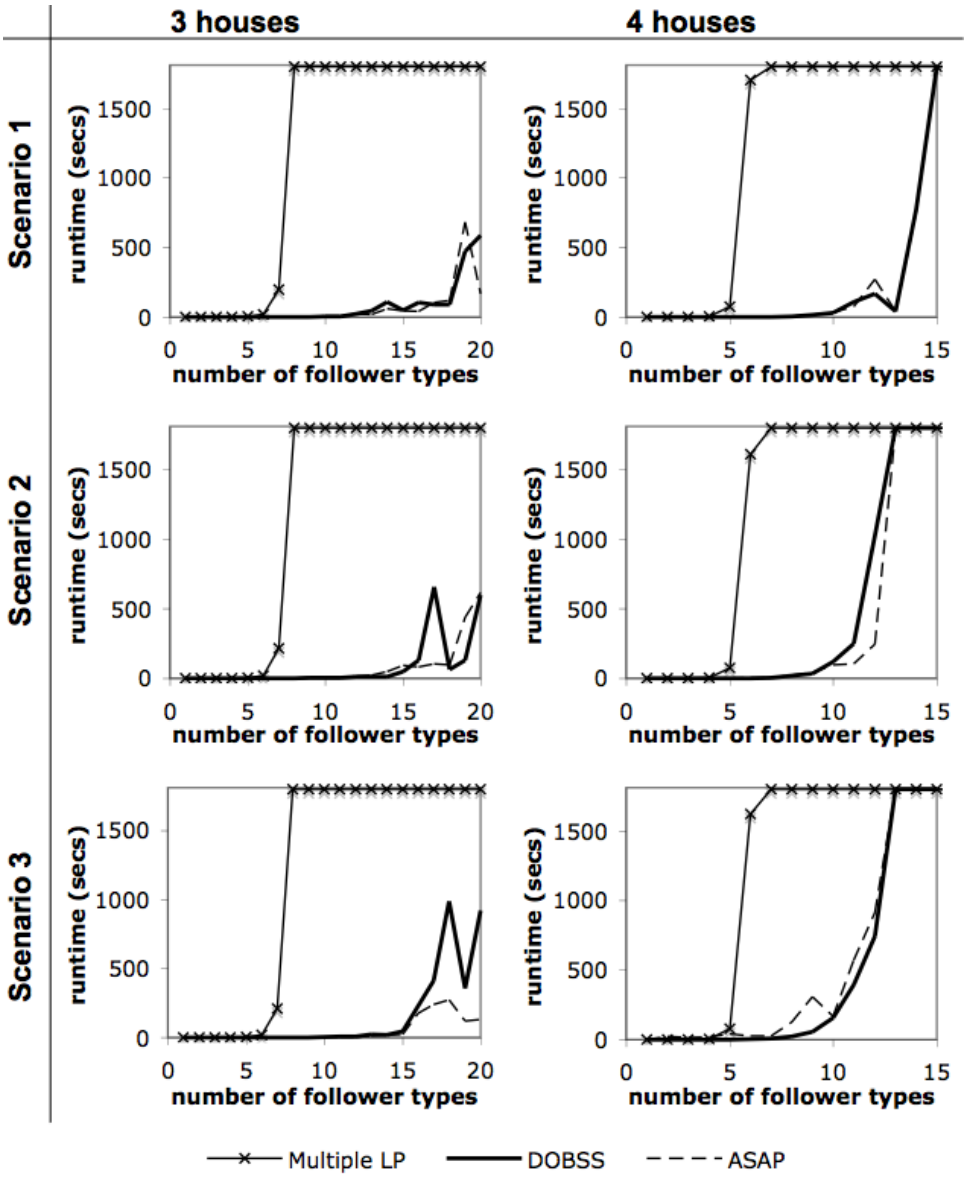



Figure 7.1: DOBSS vs. ASAP and multiple LP methods

This experiment clearly shows the speedups of DOBSS over the multiple LP method, although for these examples its runtime is comparable to that of the ASAP procedure. We now introduce a second of experimental results in Figure 7.2 to demonstrate the speedups of DOBSS over ASAP as the number of leader strategies increases. In this experiment, for a single scenario, the number of houses was varied between three and seven and the number of follower types was varied from one to nine. Patrols of length two were again used; thus the number of pure strategies for the leader ranged from three to 21 (and the number of pure strategies for each follower ranged from three to seven.) In this experiment, DOBSS outperformed ASAP in every instance, except in cases when both methods went past 1800 seconds and were cut off. *In many cases, ASAP was unable to return a feasible solution; DOBSS returned a solution in all these cases.*

Figure 7.3 illustrates the changes in the leader's patrolling strategies in a single scenario (three houses, Scenario 1, from Figure 7.1) as the number of followers increased; we focused here on the cases of two, six, and 10 follower types. This figure shows how, in this experimental domain, the addition of follower types caused a significant difference in the strategies chosen by the leader, and did not simply cause small variations on a single strategy. In other words, the results for this experimental domain are meaningful in general because the addition of more follower types had a clear effect on the strategy chosen by the leader; they were not irrelevant additions that would be ignored by the MILP solver. For example, with two follower types, strategies 1 and 6 were chosen with highest probability, whereas with 10 follower types, strategy 1 was not chosen at all.

| | | No. of houses 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| **DOBSS** | | | | | | |
| **No. of follower types** | **1** | 0.01 | 0.01 | 0.03 | 0.03 | 0.04 |
| | **2** | 0.01 | 0.01 | 0.07 | 0.12 | 0.25 |
| | **3** | 0.02 | 0.06 | 0.19 | 0.76 | 2.08 |
| | **4** | 0.06 | 0.13 | 0.86 | 5.30 | 12.70 |
| | **5** | 0.11 | 0.27 | 4.55 | 33.60 | 141.70 |
| | **6** | 0.32 | 0.80 | 37.10 | 197.15 | 779.29 |
| | **7** | 0.45 | 1.85 | 88.64 | >1800 | >1800 |
| | **8** | 0.91 | 3.67 | 178.79 | >1800 | >1800 |
| | **9** | 1.13 | 14.45 | >1800 | >1800 | >1800 |
| **ASAP** | | | | | | |
| **No. of follower types** | **1** | 0.02 | 0.03 | 0.12 | infeasible | 0.80 |
| | **2** | 0.03 | 0.06 | 0.17 | infeasible | infeasible |
| | **3** | 0.06 | 0.12 | 0.58 | infeasible | >1800 |
| | **4** | 0.09 | 0.24 | 2.37 | 45.08 | infeasible |
| | **5** | 0.18 | 0.72 | 6.80 | infeasible | infeasible |
| | **6** | 0.45 | 1.46 | infeasible | infeasible | infeasible |
| | **7** | 0.52 | 3.88 | infeasible | >1800 | >1800 |
| | **8** | 2.65 | 4.40 | 270.82 | >1800 | >1800 |
| | **9** | 1.85 | 15.30 | >1800 | >1800 | >1800 |

Figure 7.2: DOBSS vs. ASAP for larger strategy spaces

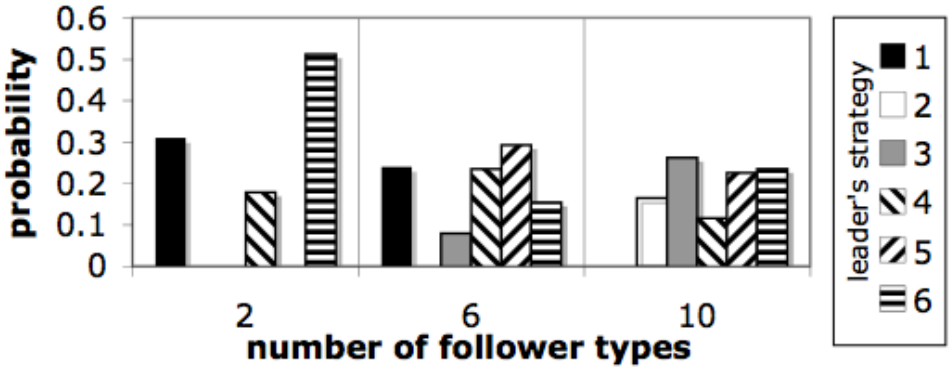



Figure 7.3: Effect of additional followers on leader's strategy

# Related Work

There are five broad areas of related work that I would be presenting below. First, I present literature on randomized policies in decision theoretic literature like MDPs and POMDPs. In the next section I provide related work to my security algorithms developed using game theoretic techniques. In the third section I focus on the application of randomization to preserve privacy of agents. The next section is devoted to related work done in modeling patrol agents. In the final section I describe the usage of randomization in designing algorithms called randomized algorithms.

## 8.1 Randomized policies for MDPs/POMDPs

Decision theoretic frameworks 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 [Puterman, 1994] framework can model stochastic actions and hence can handle transition uncertainty. The POMDP [Cassandra et al., 1994; Kaelbling et al., 1995] and Decentralized POMDP [Pynadath and Tambe,

2002; Becker et al., 2003; Goldman and Zilberstein, 2003] frameworks are more general and can model both action and observation uncertainty.

- Efficient algorithms have been devised for generating optimal plans for agents and agent teams modeled using these frameworks [Puterman, 1994; Cassandra et al., 1997].

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 useful when the agents act in environments where security is not an important issue. 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.

CMDPs (constrained MDPs) are a standard framework for modeling resource constrained agents that act in uncertain environments [Paruchuri et al., 2004; Dolgov and Durfee, 2003a]. Efficient algorithms have been developed involving usage of Linear Programs, to find the optimal policies that ensure that the resource constraints are not violated [Altman, 1999; Dolgov and Durfee, 2003c]. However, the randomized optimal policies occur in CMDPs as a side-effect of resource constraints. Additionally, the amount of randomization is a function of the resource constraints and cannot be varied as needed for security purposes. Randomized policies in team settings, which arise as a side-effect of resource constraints, lead to miscoordination and hence lower the team rewards [Paruchuri et al., 2004]. The effect of this miscoordination can be mitigated through costly communication actions or through techniques that can take the miscoordination costs into consideration while generating optimal policies [Paruchuri et al., 2004]. Due to the

miscoordination costs, deterministic policies are preferred over randomized policies in team settings and research in CMDPs has focused on developing optimal deterministic policies [Dolgov and Durfee, 2003c].

Similarly, randomized policies occur as side-effect in POMDPs where there are memory constraints in representation of policies. The policies in such POMDPs map directly from the most recent observation to an action and are referred to as memoryless policies. When considering such POMDP policies, randomized policies obtain higher expected reward than deterministic policies [Jaakkola et al., 1994]. In addition it has been pointed out [Parr and Russel, 1995; Kaelbling et al., 1995] that memoryless deterministic policies tend to exhibit looping behavior. A desire to escape this undesirable behavior motivated finding methods to obtain randomized memoryless policies [Bartlett and Baxter, 2000; Jaakkola et al., 1994]. Unfortunately, such randomization is unable to achieve the goal of maximizing expected entropy while attaining a certain threshold reward because the focus is on maximizing the expected reward.

## 8.2 Related work in game theory

In the second half of my thesis I dealt with adversarial domains where the adversaries' actions and payoffs are known but the exact adversary type is unknown to the security agent. I described the police patrol domain in detail and modeled it as Bayesian Stackelberg game. Stackelberg games [Stackelberg, 1934; Roughgarden, 2001] are commonly used to model attacker-defender scenarios in security domains [Brown et al., 2006]. In particular [Brown et al., 2006] develop 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 face many such adversaries but have incomplete information about them. Bayesian games have been a popular choice to model such incomplete information games [Conitzer and Sandholm, 2006; Brynielsson and Arnborg, 2004] and the solution concept is called the Bayes-Nash equilibrium [Fudenberg and Tirole, 1991].

Nash equilibrium [Nash, 1950] as a solution concept has received lot of attention and many efficient techniques have been developed to obtain the equilibrium solutions for normal form games [Lemke and Hawson, 1964; McKelvey et al., 2004; Savani and Stengel, 2004; Porter et al., 2004; Sandholm et al., 2005]. The Bayes-Nash equilibrium which is the nash equilibrium equivalent of Bayesian games can be found by simply converting the Bayesian game using the Harsanyi transformation [Harsanyi and Selten, 1972] into a normal form game and then applying the efficient Nash equilibrium techniques [Sandholm et al., 2005]. However the exponential complexity of the Harsanyi transformation itself is a bottleneck to find the Bayes-Nash equilibrium. The Gala toolkit is one method for finding the equilibrium solutions for the Bayesian games [Koller and Pfeffer, 1995] without requiring the game to be represented in normal form via the Harsanyi transformation [Harsanyi and Selten, 1972]. Much work has been done on finding optimal Bayes-Nash equilibria for subclasses of Bayesian games, finding single Bayes-Nash equilibria for general Bayesian games using the sequence form representation [Koller et al., 1996; Koller and Pfeffer, 1997] or approximate Bayes-Nash equilibria [Singh et al., 2004]. Unfortunately, they do not address the issue of finding the optimal strategy to commit to for the agent in a stackelberg kind scenario, which is the focus of the latter half of my thesis.

It has been shown in [Stengel and Zamir, 2004] that for any generic two player normal form games Nash equilibrium will be lower bound for the optimal strategy that the agent commits to in a stackelberg game. Less attention has been paid to finding the optimal strategy to commit to in a Bayesian game (the Stackelberg scenario). However, the complexity of this problem was shown to be NP-hard in the general case [Conitzer and Sandholm, 2006]. My thesis therefore focused on the development of efficient solution techniques for the Bayesian Stackelberg games. While developing the ASAP procedure I used $k$-uniform strategies. These strategies are similar to the $k$-uniform strategies of [Lipton et al., 2003]. While that work provides epsilon error-bounds based on the $k$-uniform strategies, their solution concept is still that of a Nash equilibrium, and they do not provide efficient algorithms for obtaining such $k$-uniform strategies. This contrasts with ASAP, where my emphasis is on a highly efficient approach that is not focused on equilibrium solutions. The main advantage of my ASAP solution is that the policies are simple and easy to implement in practice. I then converted the discrete strategy set of the leader in ASAP into a continuous strategy space to obtain the optimal leader strategies using the DOBSS algorithm.

While bayesian stackelberg games are useful in modeling and development of efficient solutions for security applications, they can model other scenarios as well. Traditionally stackelberg games have been used to model duopoly markets where a leader firm moves first, choosing a quantity and a follower firm moves next and picks a quantity based on the leaders choice [wikipedia, 2007]. These scenarios occur in markets where there is a leader who has a monopoly over the industry and the follower is a new entrant into the market. The market can be modeled as a Bayesian stackelberg game as the number of followers increase.

## 8.3 Randomization for Privacy

The next two paragraphs elaborate on the usage of randomization to increase privacy of agent strategies. Intentional randomization of agent strategies has been used as a technique to increase privacy. [Otterloo, 2005] deals with intentional randomization of agent strategies to increase privacy using strategic game settings. However, the tradeoff structure between randomization and reward is different from our structure, thus needing development of different optimization techniques. Furthermore, the work focuses on equilibrium solutions and proving equilibrium properties whereas my thesis focuses on providing efficient policy generation algorithms for both single and multi-agent teams acting in uncertain environments.

Randomization has also been used to increase privacy of data transmission in sensor networks. For example, [Ozturk et al., 2004] describes a flexible routing strategy called phantom routing. This routing scheme was designed for maintaining source-location privacy in a energy-constrained sensor network. Phantom routing is a two stage routing scheme that first consists of a directed walk along a random direction, followed by routing from phantom source to the sink. The algorithm has been tested on a panda-hunter game and the randomization was shown to preserve source location privacy. Yet another application of randomization has been described in [Borisov and Waddle, 2005]. This paper defines an entropy based anonymity metric and proposes a randomized routing protocol to increase the anonymity in structured peer-to-peer networks for purposes of user privacy.

## 8.4 Randomization and Patrolling

The patrolling problem which motivated my work has recently received growing attention from the multiagent community due to its wide range of applications [Chevaleyre, 2004; Machado et al., 2002; Carroll et al., 2005; Lewis et al., 2005; Billante, 2003; Ruan et al., 2005; Beard and McLain, 2003]. [Gui and Mohapatra, 2005] describes a surveillance application using wireless sensor networks. The SENSTROL protocol described in this work is focused on limiting the energy consumption involved in patrolling due to which deterministic sweeping patterns are generated by the virtual patroller. In [Chevaleyre, 2004; Machado et al., 2002], an informal notion of good patrolling strategy has been defined which states that a good strategy is one that minimizes the time lag between two passages to the same place and for all the places. These papers then calculate the optimal policies using this notion of goodness and find that cyclical strategies which are deterministic are the best. However deterministic strategies maximize the information given to the adversary, thus making them suboptimal if the adversary learns the agent's policy and acts. [Beard and McLain, 2003] describes the scenario where a team of UAVs cooperatively search an area of interest that contains regions of opportunity and regions of potential hazard. The aim of the UAV team is to visit as many opportunities as possible while avoiding as many hazards as possible. The UAV team is further constrained by distance and collision constraints. The static nature of the hazards allow the UAVs to find an optimal deterministic policy in these scenarios as opposed to a dynamic opponent in our patrolling domains. [Ruan et al., 2005] describes an algorithm for generating many sub-optimal patrol routes for agents acting in stochastic environment. The patrol team would then decide on one particular patrol route randomly. However this work

doesn't have an explicitly defined metric for randomness and a procedure to evaluate the tradeoffs between the reward and randomness parameters of the domain.

[Carroll et al., 2005; Lewis et al., 2005; Billante, 2003] describe practical patrol teams where randomized policies have been found to be useful. [Carroll et al., 2005] describes a real patrol unit that automatically moves randomly to and throughout designated patrol areas. While on random patrol, the patrol unit conducts surveillance, checks for intruders, conducts product inventory etc. [Lewis et al., 2005] describes an application for security/sentry vehicles using randomized patrols for avoiding predictive movements. [Billante, 2003] describes how a randomized police patrol has turned to be a key factor in the drop of crime rate in New York city. However, in these papers no specific algorithm/procedure has been provided for the generation of the randomized policies. In addition, no metrics have been proposed to quantify randomization and hence the randomness of the patrol policy cannot be explicitly controlled as a function of other parameters of the domain like reward. Also, neither single nor decentralized MDP/POMDP teams were considered here. In my thesis, I focused on development of efficient security algorithms that reason about the tradeoffs between the randomness and domain constraints. The development of these efficient security algorithms that are polynomial complexity involved extensive usage of mathematical optimization. While a significant amount of research has been done on global optimization algorithms, none of these have polynomial complexity in general [Vavasis, 1995; Floudas, 1999].

## 8.5 Randomized algorithms

Randomization has been used as a technique to devise efficient algorithms that have good expected runtimes in a large class of algorithms appropriately named randomized or probabilistic algorithms [Motwani and Raghavan, 1995]. These algorithms are used in wide variety of applications like cryptography, monte-carlo methods and even appear in simple sorting techniques like the quicksort algorithm. [Ramanathan et al., 2004] describes usage of randomization techniques for generating an efficient algorithm for leader election in large scale distributed systems. In all these algorithms, randomization serves the purpose of speeding up the process of obtaining the optimal solution and has no bearing on the type of the final solution obtained, which can be deterministic. This contrasts with my research where the focus is on obtaining a final randomized policy.

# Conclusion

My thesis focuses on the problem of providing security for agents acting in uncertain adversarial environments having limited information about their 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/regions [Carroll et al., 2005; Billante, 2003; Lewis et al., 2005], agents assisting routine security checks at airports [Poole and Passantino, 2003] or agents providing privacy in sensor network routing [Ozturk et al., 2004]. I addressed this problem of providing security, broadly by considering two realistic situations: First, when agents have no information about their adversaries and second, when the agents have partial information about their adversaries. In both these cases the adversary is assumed to know the agent's policy and hence can exploit it to its advantage. When the agents have no information about their adversaries, I developed policy randomization for single-agent and decentralized (PO)MDPs with guaranteed expected rewards, as a solution technique to minimize the information gained by the adversaries. To this end, my thesis provides two key contributions:

1. Provides novel algorithms, in particular the polynomial-time CRLP and BRLP algorithms, to randomize single-agent MDP policies, while attaining a certain level of expected reward;

2. Provides RDR, a new algorithm to generate randomized policies for decentralized POMDPs. RDR can be built on BRLP or CRLP, and thus is able to efficiently provide randomized policies.

While the techniques developed are applied for analyzing randomization-reward tradeoffs, they could potentially be applied more generally to analyze different tradeoffs between competing objectives in single/decentralized (PO)MDP.

When the agent has partial information about the adversaries, I model the security domains as Bayesian Stackelberg games. The reason is as follows: Bayesian games capture the fact that there are many adversary types while Stackelberg games model an important property which is that the agents act first while the adversaries observe the agent's policy and act i.e., an explicit notion of a leader and follower. My contributions in this case are as follows:

1. Developed an efficient procedure named ASAP for finding the optimal agent policy with limited randomization when faced with multiple opponents. I provided experimental results to show the speedups provided by ASAP over previous approaches. Further, the policies generated by ASAP are simple and easy to implement in practice.

2. Based on the ideas developed for ASAP, I derived an exact procedure named DOBSS for finding the optimal agent policy.

For both of my methods, I developed an efficient Mixed Integer Linear Program (MILP) implementation and also provided experimental results illustrating significant speedups and higher rewards over previously existing approaches.

For multiagent systems to be actually deployed and function effectively in real world they should be able to address the challenge of environmental uncertainty and security. In addition,

as agents become increasingly independent of human intervention they might need to deal with unexpected security challenges. Following are some of the problems that I would like to address in the future towards creation of secure agents:

- Incorporating Machine Learning: The environments in which agents act are many times dynamic. It might not be possible to develop a complete model of the environment beforehand for generation of optimal policies. For example, in a patrolling domain, the model describing the adversaries and their strategies may not capture correctly the situation in the real world. This is because new adversaries can appear who were previously unmodeled or adversaries can devise new strategies for which the agents might not have planned. Learning components can be incorporated into existing algorithms where the current model gets updated whenever new information is learned and optimal policies generated against the updated model.

- Resource constrained agents: Agents acting in the real world will encounter situations where there are not enough resources to execute their plans [Paruchuri et al., 2004]. This may arise because there are not enough resources in the environment or because there are other agents competing for these resources. Furthermore, the resource constraints may change dynamically making it a hard problem to solve.

- Modeling risk: While the standard assumption in game and decision theoretic literature for optimal policy generation algorithms is that the agent or agent team maximizes the expected utility thus making them risk neutral. However, agents can be risk seeking or risk averse, and thus need development of different solution models such as the optimistic or robust solutions [Nilim and Ghaoui, 2003].

I believe that by addressing these uncertainty and security challenges I will bring agents close to the reality of being deployed and function effectively in real world.

# Bibliography


E. Altman. *Constrained Markov Decision Process*. Chapman and Hall, 1999.

P. Bartlett and J. Baxter. Estimation and approximation bounds for gradient-based reinforcement learning. In *Technical Report, Australian National University*, 2000.

R. Beard and T. McLain. Multiple uav cooperative search under collision avoidance and limited range communication constraints. In *IEEE CDC*, 2003.

R. Becker, V. Lesser, and C.V. Goldman. Transition-independent decentralized markov decision processes. In *Proceedings of AAMAS*, 2003.

D.S. Bernstein, S. Zilberstein, and N. Immerman. The complexity of decentralized control of MDPs. In *UAI*, 2000.

D. Bertsimas and J. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1997.

Nicole Billante. The beat goes on: Policing for crime prevention. *http://www.cis.org.au/IssueAnalysis/ia38/ia38.htm*, 2003.

N. Borisov and J. Waddle. Anonymity in structured peer-to-peer networks. In *University of California, Berkeley, Technical Report No. UCB/CSD-05-1390*, 2005.

G. Brown, M. Carlyle, J. Salmeron, and K. Wood. Defending critical infrastructures. *Interfaces*, 36(6):530–544, 2006.

J. Brynielsson and S. Arnborg. Bayesian games for threat prediction and situation analysis. In *FUSION*, 2004.

Daniel M. Carroll, Chinh Nguyen, H.R. Everett, and Brian Frederick. Development and testing for physical security robots. *http://www.nosc.mil/robots/pubs/spie5804-63.pdf*, 2005.

A. Cassandra, L. Kaelbling, and M. Littman. Acting optimally in partially observable stochastic domains. In *Proceedings of National Conference on Artificial Intelligence*, 1994.

A. Cassandra, M. Littman, and N. Zhang. Incremental pruning: A simple, fast, exact method for partially observable Markov decision processes. In *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-97)*, pages 54–61, 1997.

Y. Chevaleyre. Theoretical analysis of multi-agent patrolling problem. In *Proceedings of AAMAS*, 2004.

V. Conitzer and T. Sandholm. Choosing the optimal strategy to commit to. In *ACM Conference on Electronic Commerce*, 2006.

D. Dolgov and E. Durfee. Approximating optimal policies for agents with limited execution resources. In *Proceedings of IJCAI*, 2003a.

D. Dolgov and E. Durfee. Constructing optimal policies for agents with constrained architectures. Technical report, Univ of Michigan, 2003b.

D. Dolgov and E. Durfee. Constructing optimal policies for agents with constrained architectures, 2003c. URL `citeseer.ist.psu.edu/dolgov03constructing.html`.

R. Emery-Montemerlo, G. Gordon, J. Schneider, and S. Thrun. Approximate solutions for partially observable stochastic games with common payoffs. In *AAMAS*, 2004.

A. Christodoulos Floudas. *Deterministic Global Optimization: Theory, Methods and Applications*, volume 37 of *Nonconvex Optimization And Its Applications*. Kluwer, 1999.

D. Fudenberg and J. Tirole. *Game Theory*. MIT Press, 1991.

Claudia V. Goldman and Shlomo Zilberstein. Optimizing information exchange in cooperative multi-agent systems. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS-03)*, pages 137–144, 2003.

C. Gui and P. Mohapatra. Virtual patrol: A new power conservation design for surveillance using sensor networks. In *IPSN*, 2005.

E.A. Hansen, D.S. Bernstein, and S. Zilberstein. Dynamic programming for partially observable stochastic games. In *AAAI*, 2004.

J. C. Harsanyi and R. Selten. A generalized Nash solution for two-person bargaining games with incomplete information. *Management Science*, 18(5):80–106, 1972.

D. A. Huffman. A method for the construction of minimum redundancy codes. In *Proc. IRE 40*, 1952.

T. Jaakkola, S. Singh, and M. Jordan. Reinforcement learning algorithm for partially observable markov decision problems. *Advances in NIPS*, 7, 1994.

L. Kaelbling, M. Littman, and A. Cassandra. Planning and acting in partially observable stochastic domains. In *Technical Report, Brown University*, 1995.

L. Kaelbling, M. Littman, and A. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(2):99–134, 1998.

D. Koller, N. Meggido, and B. V. Stengel. Efficient computation of equilibria for extensive two-person games. In *Games and Economic Behavior*, 1996.

D. Koller and A. Pfeffer. Generating and solving imperfect information games. In *IJCAI*, 1995.

D. Koller and A. Pfeffer. Representations and solutions for game-theoretic problems. *Artificial Intelligence*, 94(1):167–215, 1997.

C. Lemke and J. Hawson. Equilibrium points of bimatrix games. In *Journal of Society of Industrial and Applied Mathematics*, 1964.

Paul J. Lewis, Mitchel R. Torrie, and Paul M. Omilon. Applications suitable for unmanned and autonomous missions utilizing the tactical amphibious ground support (tags) platform. *http://www.autonomoussolutions.com/Press/SPIE%20TAGS.html*, 2005.

R. J. Lipton, E. Markakis, and A. Mehta. Playing large games using simple strategies. In *ACM Conference on Electronic Commerce*, 2003.

M. Littman. Markov games as a framework for multi-agent reinforcement learning. In *ML*, 1994. URL `citeseer.ist.psu.edu/littman94markov.html`.

A. Machado, G. Ramalho, J. D. Zucker, and A. Drougoul. Multi-agent patrolling: an empirical analysis on alternative architectures. In *MABS*, 2002.

R. D. McKelvey, A. M. McLennan, and T. L. Turocy. Gambit: Software tools for game theory. In *Version 0.97.1.5*, 2004.

R. Motwani and P. Raghavan. Randomized algorithms. In *Cambridge University Press*, 1995.

R. Nair, D. Pynadath, M. Yokoo, M. Tambe, and S. Marsella. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *IJCAI*, 2003.

J. Nash. Equilibrium points in n-person games. In *Proceedings of National Academy of Sciences*, 1950.

A. Nilim and L. E. Ghaoui. Robustness in markov decision problems with uncertain transition matrices. In *NIPS*, 2003.

S. Otterloo. The value of privacy: Optimal strategies for privacy minded agents. In *AAMAS*, 2005.

C. Ozturk, Y. Zhang, and W. Trappe. Source-location privacy in energy-constrained sensor network routing. 2004.

S. Paquet, L. Tobin, and B. Chaib-draa. An online POMDP algorithm for complex multiagent environments. In *AAMAS*, 2005.

R. Parr and S. Russel. Approximating optimal policies for partially observable stochastic domains. In *Proceedings of IJCAI*, 1995.

P. Paruchuri, M. Tambe, F. Ordonez, and S. Kraus. Towards a formalization of teamwork with resource constraints. In *AAMAS*, 2004.

R. Poole and G. Passantino. A risk based airport security policy. 2003.

R. Porter, E. Nudelman, and Y. Shoham. Simple search methods for finding a nash equilibrium. In *AAAI*, 2004.

M.L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley and Sons, 1994.

D. V. Pynadath and M. Tambe. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *JAIR*, 16:389–423, 2002.

M. Ramanathan, R. Ferreira, and A.Grama S. Jagannathan. Randomized leader election. In *Purdue University Technical Report*, 2004.

T. Roughgarden. Stackelberg scheduling strategies. In *ACM Symposium on TOC*, 2001.

S. Ruan, C. Meirina, F. Yu, K. R. Pattipati, and R. L. Popp. Patrolling in a stochastic environment. In *10th Intl. Command and Control Research Symposium*, 2005.

T. Sandholm, A. Gilpin, and V. Conitzer. Mixed-integer programming methods for finding nash equilibria. In *AAAI*, 2005.

Sasemas. http://aose.ift.ulaval.ca/index.php?module=postcalendar=details 2005.

R. Savani and B. V. Stengel. Exponentially many steps for finding a nash equilibrium in a bimatrix game. In *FOCS*, 2004.

C. Shannon. A mathematical theory of communication. *The Bell Labs Technical Journal*, pages 379–457,623,656, 1948.

S. Singh, V. Soni, and M. Wellman. Computing approximate Bayes-Nash equilibria with tree-games of incomplete information. In *ACM Conference on Electronic Commerce*, 2004.

M. Smith, S.L. Urban, and H.M. Avila. Retaliate: Learning winning policies in first-person shooter games. 2007.

H. V. Stackelberg. Marktform and gleichgewicht. In *Springer*, 1934.

B. V. Stengel and S. Zamir. Leadership with commitment to mixed strategies. In *CDAM Research Report LSE-CDAM-2004-01*, 2004.

Jo Twist. Eternal planes to watch over us. *http://news.bbc.co.uk/1/hi/sci/tech/4721091.stm*, 2005.

S. A. Vavasis. Complexity issues in global optimization: a survey. In R. Horst and P.M. Pardalos, editors, *Handbook of Global Optimization*, pages 27–41. Kluwer, 1995.

S.A. Vavasis. Nonlinear optimization: Complexity issues. In *University Press, New York*, 1991.

Y. Wen. Efficient network diagnosis algorithms for all-optical networks with pobabilistic link failures. In *thesis MIT*, 2005.

wikipedia. Stackelberg competition. 2007. URL `http://en.wikipedia.org/wiki/Stackelberg%_competition`.

L. Wolsey. *Integer Programming*. Wiley, 1998.

# Curriculum Vitae

## CURRENT POSITION

**Graduate Research Assistant**
Computer Science Department
Viterbi School of Engineering
University of Southern California
http://teamcore.usc.edu/praveen
paruchur@usc.edu

USC Powell Hall 514
3737 Watt Way
Los Angeles, CA 90089
Phone: (213)740-9560
Fax: (213)740-7877

## RESEARCH INTERESTS

Applied Artificial Intelligence, Single/Multi Agent Systems, Linear/Nonlinear/Mixed Integer Programming based Solution Techniques, Decision and Game Theoretic Reasoning, Decision Making under Uncertainty, Safety and Security Issues for Practical Agent Systems, Bayesian Games, Randomized Policies

## EDUCATION

**Doctor of Philosophy (in progress)**
Computer Science — 09/'02 - Present
University of Southern California
Dissertation Topic: "Reasoning in Uncertain Adversarial Environments in Agent/Multiagent Systems"
Advisor: Prof. Milind Tambe
Committee: Prof. Milind Tambe (chair), Prof. Leana Golubchik, Prof. Sarit Kraus, Prof. Stacy Marsella, Prof. Fernando Ordóñez and Prof. Gaurav S. Sukhatme

**Bachelor of Technology**
Computer Science and Information Technology — 09/'98 - 06/'02
International Institute of Information Technology, India
Thesis: Multiagent Simulation of Unorganized Traffic
Advisor: Prof. Kamalakar Karlapalem

## AWARDS AND HONORS

**Best Paper Award** at the SASEMAS Workshop, AAMAS '05: Paper titled "Safety in multiagent systems by policy randomization" selected as the Best Paper at the International Workshop on Safety and Security in Multiagent Systems.

**Deans Merit Scholarship** at IIIT '01: Awarded the Dean's Merit Scholarship during undergraduate studies at IIIT-Hyderabad for best GPA for the semester.

Represented IIIT at the **ACM Asia Programming Contest '01**: Team of three members selected to represent IIIT at the ACM Asia Programming Contest-01 held at IIT-Kanpur.

**National Talent Search Exam (NTSE) Scholar '96**: Awarded the NTSE scholarship in 1996. The National Council of Education Research and Training, New Delhi, India awards 1000 scholarships, every year under its NTS Schema, at the end of class X picked from all over the country.

## EXPERIENCE

**Graduate Research Assistant**
Teamcore Research Group (Prof. Milind Tambe) 09/'02 - Present
Computer Science Department, USC
Los Angeles, CA
*Extensively employed Linear and Mixed Integer programming techniques to develop efficient Game and Decision Theoretic algorithms for security of multiagent systems acting in uncertain environments.*

**Member of CREATE center** 05/'05 - Present
Center for Risk and Economic Analysis of Terrorism Events, USC
Los Angeles, CA
*Developing a police patrol policy generator for the Department of Public Safety(DPS), University of Southern California(USC).*

**Graduate Teaching Assistant**
USC CS Department (Advanced AI) 08/'03 - 12/'03
Los Angeles, CA

**Summer Intern**
Language Technologies Research Center 04/'00 - 07/'00
IIIT, Hyderabad, India

## Funding Proposal Experience

Worked on generating a proposal titled “Specialized Robot/Human Teams for Limited Tactical Maneuvers”. This proposal was funded for SBIR.

## PUBLICATIONS

### RIGOROUSLY REVIEWED CONFERENCE PAPERS

**Praveen Paruchuri**, Jonathan P. Pearce, Milind Tambe, Fernando Ordóñez and Sarit Kraus. *An Efficient Heuristic Approach for Security Against Multiple Adversaries*. Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multiagent Systems, **AAMAS-2007** (Oral Presentation, Acceptance Rate 23%).

**Praveen Paruchuri**, Milind Tambe, Fernando Ordóñez and Sarit Kraus. *Security in Multiagent Systems by Policy Randomization*. Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems, **AAMAS-2006** (Oral Presentation, Acceptance Rate 11%).

M. Tambe, E. Bowring, H. Jung, G. Kaminka, R. Maheswaran, J. Marecki, P. J. Modi, R. Nair, J. Pearce, **Praveen Paruchuri**, D. Pynadath, P. Scerri, N. Schurr and P. Varakantham. *Conflicts in teamwork: Hybrids to the rescue*. Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems, **AAMAS-2005** (Invited Paper).

**Praveen Paruchuri**, Milind Tambe, Fernando Ordóñez and Sarit Kraus. *Towards a formalization of teamwork with resource constraints*. Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, **AAMAS-2004** (Acceptance Rate 24%).

**Praveen Paruchuri**, Alok Reddy Pullalarevu and Kamalakar Karlapalem. *Multi Agent Simulation of Unorganized Traffic*. Proceedings of the First International Joint Conference on Autonomous Agents and Multi Agent Systems, **AAMAS-2002** (Acceptance Rate 27%).

### MAGAZINE ARTICLES AND BOOK CHAPTERS

**Praveen Paruchuri**, Emma Bowring, Ranjit Nair, Jonathan P. Pearce, Nathan Schurr, Milind Tambe and Pradeep Varakantham. *Teamcore Research Group: Hybrids in Multiagent Teamwork*. Communications of the Computer Society of India, 2006.

**Praveen Paruchuri**, Don Dini, Milind Tambe, Fernando Ordóñez and Sarit Kraus. *Coordinating Randomized Policies for Increasing Security in Multiagent Systems*. To appear, Lecture Notes in Artificial Intelligence Hot Topic Volume, Springer Publications, 2007.

WORKSHOP AND SYMPOSIUM PAPERS

**Praveen Paruchuri**, Jonathan P. Pearce, Milind Tambe, Fernando Ordóñez and Sarit Kraus. *An Efficient Heuristic for Security Against Multiple Adversaries in Stackelberg Games*. AAAI Spring Symposium on Game and Decision-Theoretic Agents, GTDT-2007.

**Praveen Paruchuri**, Milind Tambe, Fernando Ordóñez and Sarit Kraus. *Increasing Security through Communication and Policy Randomization in Multiagent Systems*. Proceedings of the MSDM workshop at AAMAS-2006.

**Praveen Paruchuri**, Milind Tambe, Fernando Ordóñez and Sarit Kraus. *Randomizing Policies for agents and agent-teams*. Proceedings of the AI and Math Symposium, AI/Math-2006.

**Praveen Paruchuri**, Don Dini, Milind Tambe, Fernando Ordóñez and Sarit Kraus. *Safety in Multiagent Systems by Policy Randomization*. Proceedings of the SASEMAS workshop at AAMAS-2005 (**Best Paper Award).**

**Praveen Paruchuri**, Don Dini, Milind Tambe, Fernando Ordóñez and Sarit Kraus. *Intentional Randomization for Single Agents and Agent-Teams*. Proceedings of the GTDT workshop at IJCAI-2005.

**Praveen Paruchuri**, Milind Tambe, Spiros Kapetanakis and Sarit Kraus. *Between collaboration and competition: An initial Formalization using Distributed POMDPs*. Proceedings of the GTDT workshop at AAMAS-2003.

## REFERENCES

**Milind Tambe**
Professor
Computer Science Department
University of Southern California
Powell Hall of Engineering 208
3737 Watt Way
Los Angeles, CA 90089-0781
Phone: 213 740 6447
Fax: 213 740 7285
Email: tambe@usc.edu
Website: http://teamcore.usc.edu/tambe

**Sarit Kraus**
Professor
Department of Computer Science
Bar-Ilan University

Ramat Gan, 52900 Israel
Office: Room 8 Brain Science Building
Phone: (972) 3-531-8762
Fax: (972) 3-535-2184
Email: sarit@cs.biu.ac.il
Website: http://www.cs.biu.ac.il/~sarit

**Fernando Ordóñez**
Assistant Professor
Computer Science, Industrial and Systems Engineering, USC
3715 McClintock Ave., GER-240
Los Angeles, CA 90089-0193
Phone: 213 821 2413
Fax: 213 740 1120
Email: fordon@usc.edu
Website: http://www-rcf.usc.edu/~fordon