PROBABILISTIC PLANNING WITH RISK-SENSITIVE CRITERION

BY

PING HOU

A dissertation submitted to the Graduate School

in partial fulfillment of the requirements

for the degree

Doctor of Philosophy

Major Subject: Computer Science

New Mexico State University

Las Cruces New Mexico

May 2017

"Probabilistic Planning with Risk-Sensitive Criterion," a dissertation prepared by Ping Hou in partial fulfillment of the requirements for the degree, Doctor of Philosophy, has been approved and accepted by the following:

Lou-Vicente Reyes

Dean of the Graduate School

William Yeoh

Chair of the Examining Committee

Date

Committee in charge:

Dr. William Yeoh, Chair

Dr. Son Cao Tran

Dr. Enrico Pontelli

Dr. Wei Tang

Dr. Shlomo Zilberstein

ACKNOWLEDGMENTS

I am indebted to William Yeoh, my advisor, for his help and guidance in my research and beyond, and for making this dissertation possible. I would also like to thank a handful of researchers that I cooperated with: Tran Cao Son, Pradeep Varakantham, Christabel Wayllace, Khoi Hoang and Athena Tabakhi.

ABSTRACT

PROBABILISTIC PLANNING WITH RISK-SENSITIVE CRITERION

BY

PING HOU

Doctor of Philosophy

New Mexico State University

Las Cruces, New Mexico, 2017

Dr. William Yeoh, Chair

Probabilistic planning models – *Markov Decision Processes* (MDPs) and *Partially Observable Markov Decision Processes* (POMDPs) – are models where autonomous agents operate and make sequential decisions under uncertainty environment in successive episodes. To adapt to different scenario and objectives, many criteria and variants for MDPs and POMDPs have been proposed. The *Risk-Sensitive criterion* (RS-criterion) is one criterion that maximize the probability that the accumulated cost of the agent execution is less than a predefined threshold, so that the agent can avoid the situation that an exorbitantly high accumulated cost is encountered as much as possible.

Risk-Sensitive MDPs (RS-MDPs) and *Risk-Sensitive POMDPs* (RS-POMDPs) are risk-sensitive models that combine the RS-criterion with MDPs and POMDPs, respectively. I hypothesize that one can design novel algorithms which are specific for RS-MDPs and RS-POMDPs by applying insights gained

from analyzing properties and structures of risk-sensitive models. The key observation about risk-sensitive models is that the original formalizations can be transformed to new formalizations by maintaining the consumed cost so far and considering it for decisions as well.

To validate my hypothesis, (1) I formally define the RS-MDP model and distinguish the models under different assumptions about cost. For each assumption and model, I introduce new algorithms and discuss related properties. (2) I formally propose the RS-POMDP model and discuss some deficiencies of several existing regular POMDP models. I introduce procedures to complement several existing algorithms and also provide new algorithms for RS-POMDPs. (3) I also provide theoretical properties of the risk-sensitive models as well as empirical evaluations of the new algorithms on randomly generated problems, the NAVIGATION domain from the ICAPS International Probabilistic Planning Competition, and a taxi domain generated with real-world data. For both RS-MDPs and RS-POMDPs, the experiments show my algorithms are more efficient than existing algorithms, which validate my hypothesis.

Contents

List of Algorithms

List of Tables

List of Figures

Chapter 1

Introduction

In artificial intelligence, one core research area is on enabling autonomous agents to make good decisions. If achieving goals require successive decision episodes, the research problems are sequential decision making problems, which is a central research area called planning in artificial intelligence. In order to handle the uncertainty that arise in a system, different planning models are proposed. Probabilistic planning is a main research branch for planning under uncertainty.

Research about probabilistic planning continue over several decades and researchers have proposed many variants and criteria for probabilistic planning models. For example, the *Risk-Sensitive criterion* (RS-criterion) [Yu *et al.*, 1998; Hou *et al.*, 2014b] is one of such criterion. By adopting the RS-criterion for probabilistic planning models, the corresponding risk-sensitive probabilistic planning models can be obtained. In this work, I hypothesize that one can design novel algorithms which are specific for risk-sensitive models by applying insights gained from analyzing properties and structures of risk-sensitive models. To validate my hypotheses, I introduce several new algorithms for various risk-sensitive models with speedups in a variety of problems.

1.1 Probabilistic Planning Problems

Probabilistic planning [Geffner and Bonet, 2013] is a research branch of planning under uncertainty. In probabilistic planning, uncertainty is represented by probabilities, and two important models are *Markov Decision Processes* (MDPs) [Mausam and Kolobov, 2012; Geffner and Bonet, 2013] and *Partially Observable Markov Decision Processes* (POMDPs) [Kaelbling *et al.*, 1998; Geffner and Bonet, 2013]. MDPs and POMDPs model the scene that an autonomous agent operates in a system and executes actions in successive episodes. MDPs and POMDPs use states to represent the environment of the system and assume the states transition to each other under uncertainty, which is caused either by environment dynamics or a result of performing actions. MDPs and POMDPs normally assume that the agent needs to achieve some tasks, namely goals, through actions that incur costs. MDPs assume that the agent knows the exact information of the environment, and POMDPs assume that the agent can only obtain some observations that partially reflect the environment.

The RS-criterion is one probabilistic planning criterion whose objective is to maximize the probability that the agent achieves goals with an accumulated cost that is equal to or less than a predefined threshold. *Risk-Sensitive MDPs* (RS-MDPs) [Yu *et al.*, 1998; Hou *et al.*, 2014b] and *Risk-Sensitive POMDPs* (RS-POMDPs) [Hou *et al.*, 2016] are probabilistic models that combine the RS-criterion wieh MDPs and POMDPs.

1.2 Hypothesis and Contributions

Risk-sensitive probabilistic planning models are based on the RS-criterion, which fall under the category of probabilistic planning models. My hypothesis is as

follows:

> *One can design novel algorithms for risk-sensitive models – RS-MDPs and RS-POMDPs – by applying insights gained from analyzing specific properties and structures of risk-sensitive models.*

Specifically, the key observation about risk-sensitive models is that the original formalization can be transformed to a new formalization by maintaining the consumed cost so far and considering it for decisions as well. For different formalizations, I will explore and exploit their structures and properties and adopt ideas of existing probabilistic planning technique, such as *Value Iteration* (VI) [Bellman, 1957], *Topological Value Iteration* (TVI) [Dai *et al.*, 2011] and *Functional Value Iteration* (FVI) [Liu and Koenig, 2005b; 2006], to accelerate computation process of solving risk-sensitive models – RS-MDPs and RS-POMDPs. I have two contributions thus far:

1. I formally defined the RS-MDP model [Hou *et al.*, 2014b] and distinguish the model under different assumptions about cost. For each model, new algorithms are introduced and related properties are discussed thoroughly.

2. I formally proposed the RS-POMDP model [Hou *et al.*, 2016] and discuss some deficiency of several existing regular POMDP models. I introduce procedures to complement several existing algorithms, provides new algorithms for RS-POMDPs and discuss related properties.

1.3 Dissertation Structure

Having outlined the theme of my dissertation, this proposal is structured as follows: Chapter 2 gives an overview of MDPs and POMDPs. Chapter 3 describes

the different risk attitudes including our RS-criterion. Then, I introduce algorithms for RS-MDPs and RS-POMDPs in Chapter 4 and Chapter 5, respectively. These sections also include comprehensive experimental results for these two models. In additional, a approximate algorithm – Local Search – is introduced in Chapter 6. I discuss related work in Chapter 7 before presenting our conclusions and future work in Chapter 8.

Chapter 2

Background: MDPs and POMDPs

This section begins by providing an overview of the *Markov Decision Process* (MDP) and the *Partially Observable Markov Decision Process* (POMDP) models in Section 2.1 and Section 2.2, respectively.

2.1 Markov Decision Process

In this subsection, we formally define MDPs, describe some of its solution approaches, and also discuss some additional issues related to MDPs.

2.1.1 MDP Definition

There is a variety of MDP models, and some are more expressive than others [Mausam and Kolobov, 2012; Kolobov, 2013; Geffner and Bonet, 2013]. In this dissertation, we focus on *Goal-Directed MDPs* (GD-MDPs) [Geffner and Bonet, 2013] and we will briefly discuss the relationship between different MDP models

as well.

A GD-MDP is defined by a tuple $\langle \mathbf{S}, \mathbf{A}, \mathbf{T}, \mathbf{C}, \mathbf{G}, s_0 \rangle$ where:

- $\mathbf{S}$ is a finite set of all states.
- $\mathbf{A}$ is a finite set of all actions.
- $\mathbf{T} : \mathbf{S} \times \mathbf{A} \times \mathbf{S} \to [0, 1]$ is a transition function that gives the probability $T(s, a, s')$ of transitioning from state s to s' when action a is executed.
- $\mathbf{C} : \mathbf{S} \times \mathbf{A} \times \mathbf{S} \to [0, +\infty)$ is a cost function that gives the cost $C(s, a, s')$ of transitioning from state s to s' when action a is executed. The cost function gives a strictly positive cost $C(s, a, s') > 0$ with the exception of transitions from a goal.
- $\mathbf{G} \subseteq \mathbf{S}$ is the set of goal states. Goal states are *absorbing* and *cost-free*, i.e., $T(s_g, a, s_g) = 1$, $T(s_g, a, s_{\neg g}) = 0$ and $C(s_g, a, s_g) = 0$ for all goal states $s_g \in \mathbf{G}$, actions $a \in \mathbf{A}$ and non-goal states $s_{\neg g} \notin \mathbf{G}$.
- $s_0 \in \mathbf{S}$ is the initial state.

For the cost function in above definition, there exist more general forms. The first generalization is to allow the cost to be zero everywhere rather than only for transitions from goals, or even to be a negative value for transitions from non-goals, i.e., $\mathbf{C} : \mathbf{S} \times \mathbf{A} \times \mathbf{S} \to \mathbb{R}$. This dissertation will have some discussion related to the first generalization in subsequent sections. The second generalization is to include uncertainty in the cost function. Then, the cost function would map transitions, namely state-action-state tuples, into discrete probability distributions over cost values. For the second generalization, it will not be explicitly discussed in this dissertation, but all the work described here can be easily extended to adopt it. In this dissertation, we will focus on GD-MDPs and will thus use the term MDPs to refer to GD-MDPs unless other models are explicitly mentioned.

The state space of an MDP can be visualized as a *directed hyper-graph* called

a *connectivity graph*. A directed hyper-graph generalizes the notion of a regular directed graph by allowing each *hyper-edge* to have one source but several destinations. In a corresponding connectivity graph of an MDP, each vertex corresponds to a state and each pair of state s and action a corresponds to hyper-edge whose source is s and destinations are all states s' where $T(s, a, s') > 0$. The subgraph rooted at the initial state is called a *transition graph*, which only includes vertices corresponding to states reachable from the initial state.

2.1.2 MDP Solution

The solution of an MDP is called a *policy*, denoted as π, which is a strategy that chooses an action based on the current information of the system. For any state s, if we assume it to be the starting state s^0, a policy π defines a *probability* for every *execution trajectory* $\tau : \langle s = s^0, a^0, s^1, a^1, s^2, \ldots \rangle$ starting from s, which is given by the product $P^\pi(\tau) = \prod_{i=0}^{\infty} T(s^i, a^i, s^{i+1})$, and each execution trajectory τ defines an *accumulated cost* as $C(\tau) = \sum_{i=0}^{\infty} C(s^i, a^i, s^{i+1})$, where a^i is the action chosen by π to be executed at the time step i. The *expected accumulated cost*, or simply *expected cost*, of a policy π from state s, denoted as $V^\pi(s)$, stands for the sum of the accumulated costs of the different trajectories, weighted by their probabilities with respect to π.

MDPs assume that the state transitions and costs are Markovian, which means that the future state and cost depend only on the current state and not on the history. Thus, the necessary information for the action choice is just the current state. Since it also assumes that the observation is over full states, the policies are in the form of functions $\pi : \mathbf{S} \to \mathbf{A}$, which map the states into actions, would be optimal. Policies of this type are *stationary* [Puterman, 1994; Mausam and Kolobov, 2012]. On the other hand, a *non-stationary* policy is a function of both state and time.

For *Finite-Horizon* MDPs, which is another MDP model, the optimal policies could be non-stationary, but finite-horizon MDPs can be solved as GD-MDPs by augmenting the state to be the pairs of state and time, i.e., finite-horizon MDPs is a subset of GD-MDPs [Mausam and Kolobov, 2012]. Non-stationary policies is a superset of stationary policies, and they are not strictly needed for GD-MDPs since there exist optimal policies that are stationary already [Puterman, 1994; Mausam and Kolobov, 2012]. In this dissertation, we will use the term policies to refer to stationary policy unless policies with other types are explicitly mentioned. Additionally, the above policies are also called *deterministic*. On the other hand, a *stochastic* policy is a function that maps states into probability distributions over actions. As an example, for *Constrained MDPs* [Altman, 1999], which is variant MDP model that assumes multiple objectives as constraints, the optimal policies could be stochastic. This is due to the presence of multiple objectives, where it may be necessary to randomize over multiple actions in order to trade off between multiple objectives. For the work in this dissertation, stochastic policies are unnecessary.

Policies with the form $\pi : \mathbf{S} \to \mathbf{A}$, $V^{\pi}(s)$ can be defined by the expression

$$V^{\pi}(s) = \sum_{s' \in \mathbf{S}} T(s, \pi(s), s')(C(s, \pi(s), s') + V^{\pi}(s')) \tag{2.1}$$

which defines the *expected cost function* V^{π} for policy π.

Based on the regular *optimality criterion*, or simply *criterion*, a policy π is *optimal* for state s if the expected cost $V^{\pi}(s)$ is minimum among all policies. The optimal policies for MDPs are the policies π^* that are optimal for all states, i.e., $V^{\pi^*}(s) = \min_{\pi} V^{\pi}(s)$ for all states $s \in \mathbf{S}$. We call this criterion as the *Minimizing Expected Cost* (MEC) criterion. The expected cost function V^{π} corresponding to an optimal policy π^* is the optimal expected cost function $V^* = V^{\pi^*}$, which is the

unique solution of the *Bellman equation* [Bellman, 1957]:

$$V(s) = \min_{a \in \mathbf{A}} \sum_{s' \in \mathbf{S}} T(s, a, s')(C(s, a, s') + V(s')) \tag{2.2}$$

for all states.

An optimal policy π^* can be obtained from the optimal expected cost function V^* as:

$$\pi^*(s) = \operatorname*{argmin}_{a \in \mathbf{A}} \sum_{s' \in \mathbf{S}} T(s, a, s')(C(s, a, s') + V^*(s')) \tag{2.3}$$

If there exists a policy π and a execution trajectory $\langle s, a, \ldots, s', \ldots \rangle$ so that the probability π defined for the execution trajectory is larger than 0, then we say that state s can *reach* state s', or state s' is *reachable* from state s. Clearly, states that are not reachable from the initial state s_0 are irrelevant to the optimal cost function value $V^*(s_0)$. For an MDP, if the objective is to find a policy that is optimal for the initial state s_0, we can ignore any states that is not reachable from s_0. In this dissertation, we will not explicitly distinguish between the set of all states and the set of all states that is reachable from s_0, and $\mathbf{S}$ will be misused to denote both.

If an execution trajectory leads to a goal state, then the accumulated cost would be finite since the goal state is absorbing and cost-free. Otherwise, the accumulated cost would be infinite since our MDP definition only allows positive costs for transitions from non-goal states. Thus, a policy π has a finite expected cost $V^\pi(s)$ if and only if all possible trajectories by applying this policy from state s lead to a goal state. A policy is *proper at state s* if applying this policy from state s lead to goals with probability 1. Otherwise, it is *improper at state s*. A policy is *proper* if it is proper at all states. Otherwise, it is *improper*. A proper policy does not always exists for an MDP, and it only exists for MDPs without

dead-ends, which are states from which the goals cannot be reached. Clearly, if s is a dead-end, $V^{\pi}(s)$ is infinite for any policy π. On the other hand, there must exist a proper policy for MDPs without dead-ends. In this dissertation, we will refer to MDPs without dead-ends as *Stochastic Shortest-Path MDPs* (SSP-MDPs) [Mausam and Kolobov, 2012; Geffner and Bonet, 2013], which is another MDP model.[1] In the rest of this subsection, the MDP algorithms we will review are specific for SSP-MDPs and we will also refer SSP-MDPs as MDPs.

2.1.3 Value Iteration (VI)

Value Iteration (VI) is an algorithm for computing the optimal expected cost function V^* for SSP-MDPs. VI first sets a *cost value function* $V_0(s)$ as 0 for goals and an arbitrary value for non-goal states. Then, VI uses Equation 2.2 to perform updates in parallel over all non-goal states to get a new cost value function repeatedly. The update is

$$V_k(s) \leftarrow \min_{a \in \mathbf{A}} \sum_{s' \in \mathbf{S}} T(s, a, s')(C(s, a, s') + V_{k-1}(s')) \tag{2.4}$$

where V_k is the cost value function after performing the update iteration k times. This update of calculating a new cost value function is called *Bellman update* or a *full DP update*. The cost value function approach V^* as more and more Bellman updates are performed. The algorithm terminates when the values *converge* at the kth iteration, that is, the maximum difference between V_{k-1} and V_k of any state is less than a user-defined threshold ϵ, i.e., $\forall s$, $V_k(s) - V_{k-1}(s) < \epsilon$. The above difference is called the *residual*.

[1] This model is actually the weak definition of SSP-MDPs. The strong definition of SSP-MDPs allow non-positive costs and it guarantee any policies π that is improper at s would result in an infinity expected cost from s, i.e., $V^{\pi}(s) = +\infty$.

2.2 Partial Observation Markov Decision Process

In this subsection, we formally define POMDPs and describe solution approaches for POMDPs.

2.2.1 POMDP Definition

POMDPs generalize MDPs by allowing states to be partially observable through sensors that map the actual state of the world into observations according to known probabilities. There are also different POMDP models, and we focus on one formalization – *Goal-Directed POMDPs* (GD-POMDPs) [Bonet and Geffner, 2009; Geffner and Bonet, 2013] – in this dissertation.

A GD-POMDP is defined by a tuple $\langle \mathbf{S}, \mathbf{A}, \mathbf{T}, \mathbf{C}, \mathbf{G}, \mathbf{\Omega}, \mathbf{O}, b_0 \rangle$, where:

- $\mathbf{S}$, $\mathbf{A}$, $\mathbf{T}$, $\mathbf{C}$ and $\mathbf{G}$ are the same as the corresponding components in the MDP definition.
- $\mathbf{\Omega}$ is the finite set of all observations.
- $\mathbf{O} : \mathbf{S} \times \mathbf{A} \times \mathbf{S} \times \mathbf{\Omega} \rightarrow [0, 1]$ is an observation function that gives the probability $O(s, a, s', o)$ of receiving observation o when the action a is executed in state s and state s' is reached.
- b_0 is the initial belief state, that is $b_0(s)$ stands for the probability of s being the true initial state.

For GD-POMDPs, the goals are observable, i.e., $O(s, a, s_{\neg g}, o_{s_g}) = 0$, $O(s, a, s_g, o_{s_g}) = 1$ and $O(s, a, s_g, o) = 0$ for all states $s \in \mathbf{S}$, goal states $s_g \in \mathbf{G}$, non-goal states $s_{\neg g} \notin \mathbf{G}$, actions $a \in \mathbf{A}$, an observation $o_{s_g} \in \mathbf{\Omega}$ that corresponds to s_g and all other observations $o \neq o_{s_g}$. In this dissertation, we will focus on GD-POMDPs and will thus use the term POMDPs to refer to GD-POMDPs unless other models

are explicitly mentioned.

In POMDPs, the observation may not accurately indicate which state the system is in. So, the selection of the best action depends on the *belief state* rather than the state. A belief state b is a probability distribution over the states such that $b(s)$ is the probability of s being the actual state. We use $\mathbf{B}$ to denote the set of belief states. A POMDP can be viewed as an MDP over belief states, but the space of belief states is infinite now for POMDPs. Actually, most of the discussion for MDPs apply also to POMDPs and similar conclusions apply as well. We will not thus repeat related discussion here.

For a POMDP, the initial belief state is given by the prior probability b_0 in the model, and the belief states following an execution are defined recursively. The belief state $b_a^o(s)$, which denotes the belief state after performing action a in belief state b and observing o, is

$$b_a^o(s) = \frac{\sum_{s' \in \mathbf{S}} b(s') T(s', a, s) O(s', a, s, o)}{P(b, a, o)} \tag{2.5}$$

where $P(b, a, o) = \sum_{s \in \mathbf{S}} \sum_{s' \in \mathbf{S}} b(s) T(s, a, s') O(s, a, s', o)$ is the probability of observing o after performing action a in belief state b'. We also define $C(b, a, b_a^o)$ as

$$C(b, a, b_a^o) = \frac{\sum_{s \in \mathbf{S}} \sum_{s' \in \mathbf{S}} b(s) T(s, a, s') O(s, a, s', o) C(s, a, s')}{P(b, a, o)} \tag{2.6}$$

which denotes the cost expectation of observing o after executing action a in belief state b.

A *POMDP policy* $\pi : \mathbf{B} \to \mathbf{A}$ is a mapping from belief states to actions, which is also stationary with respect to belief states. Similar to MDPs, the objective is to find a policy π^* with the minimum expected cost $V^*(b_0)$, defined by

$$V^*(b) = \min_{a \in \mathbf{A}} \sum_{o \in \mathbf{\Omega}} P(b, a, o)(C(b, a, b_a^o) + V^*(b_a^o)) \tag{2.7}$$

for all belief states $b \in \mathbf{B}$.

An optimal policy π^* can be obtained from the optimal expected cost function V^*, as:

$$\pi^*(b) = \underset{a \in \mathbf{A}}{\operatorname{argmin}} \sum_{o \in \mathbf{\Omega}} P(b, a, o)(C(b, a, b_a^o) + V^*(b_a^o)) \tag{2.8}$$

As for MDPs, we will refer to POMDPs without dead-ends as Stochastic Shortest-Path POMDPs (SSP-POMDPs), which is another POMDP model.[2] In the rest of this subsection, the POMDP algorithms we will review are specific for SSP-POMDPs and we will also refer SSP-POMDPs as POMDPs for simplicity.

2.2.2 Value Iteration

Similar to VI for MDPs, a VI-like algorithm solves POMDPs exactly and we also call it VI. Since the number of belief states in POMDPs is infinite, policies and cost value functions cannot be stored explicitly as MDP in algorithms. As in VI for MDPs, the full DP update should map a value function V_{k-1} over all belief states b into the value function V_k. So, exact POMDP algorithms [Kaelbling *et al.*, 1998] use a finite set Γ of $|\mathbf{S}|$-dimensional real vectors, in which each vector can be seen as a cost value function and each element in a vector corresponds to the cost value $\alpha(s)$ of starting at a particular state s. If we pick the minimum vector for each belief state, the set forms a *piecewise linear and concave function* over the belief state space, and this function represents the cost value function. Then, the expected cost of a belief state b is:

$$V(b) = \min_{\alpha \in \Gamma} \sum_{s} b(s)\alpha(s) \tag{2.9}$$

[2]This model is actually the weak definition of SSP-POMDPs. The strong definition of SSP-POMDPs allow non-positive costs and it guarantee any improper policies would result in an infinity expected cost [Geffner and Bonet, 2013]

VI would set the initial vector set Γ_0 to only contain one vector, in which cost values of all goal states are zero, i.e., $\alpha(s_g) = 0$ for all $s_g \in \mathbf{G}$. If we define a function $v : \mathbf{\Omega} \to \Gamma_{k-1}$ to map each observation to the $|\mathbf{S}|$-dimensional real vector in the previous $(k-1)$th iteration, and $\mathcal{V}_{k-1}$ as the set of all such functions, then the full set of possible vectors after the update in iteration k is:

$$\Gamma_k = \{\alpha_{a,v} \mid a \in \mathbf{A}, v \in \mathcal{V}_{k-1}\} \tag{2.10}$$

where $\alpha_{a,v}(s) = \sum_{s',o} T(s,a,s')(C(s,a,s') + O(s,a,s',o)v(o)(s'))$. Some of these vectors are dominated by others in the sense that they do not yield the minimum at any belief state b. Such vectors can be identified by solving linear programs and removed [Kaelbling *et al.*, 1998]. Similar to VI for MDPs, algorithms iteratively perform full DP updates to update the vector set Γ until it converges.

Chapter 3

Risk Attitudes

In this section, we will first describe how utility functions can be used to represent different risk attitudes before describing the risk-sensitive criterion, a specific risk attitude that we will adopt and optimize for in this dissertation.

3.1 Utility Functions

Recall that the *Minimizing Expected Cost* (MEC) criterion, defined in Section 2.1.2, is the typical objective for MDPs and POMDPs. According to this criterion, the optimal policies of MDPs and POMDPs are those with the minimum expected cost, which is equivalent to assuming a risk-neutral attitude. Utility theory assumes that there exist many different risk attitudes aside from the risk-neutral attitude, and it uses different *utility functions* to describe the diverse risk attitudes [Liu and Koenig, 2005a; 2005b]. Formally, a *utility function* $U : \mathbf{W} \to \mathbb{R}$ is a mapping from the *wealth level* to a *utility*, where $\mathbf{W}$ is the set of all possible wealth levels. Also, utility functions are assumed to be strictly monotonically non-decreasing.

Utility functions are an expressive representation for risk attitude, and the

MEC criterion, namely risk-neutral attitude, is equivalent to having a linear utility function that is monotonically increasing. Aside from linear functions, many utility functions with nonlinear forms are studied for different MDP settings, such as exponential functions [Liu and Koenig, 2005a], one-switch functions [Liu and Koenig, 2005b], and skew symmetric bilinear functions [Gilbert *et al.*, 2015]. Additionally, researchers have also solved MDPs with utility functions and worst-case guarantees [Ermon *et al.*, 2012], and it is equivalent to having a utility function that maps to negative infinity when the wealth level is smaller than a threshold $w_\perp$, i.e., $U(w) = -\infty$ if $w < w_\perp$.

In general, utility functions in any form can be approximated by *Piecewise Linear* (PWL) functions, and PWL utility functions are studied for both MDPs [Liu and Koenig, 2006] and POMDPs [Marecki and Varakantham, 2010]. For MDPs and POMDPs, if an initial wealth level w_0 is assumed, a PWL function can be represented by an ordered list of tuples (w^i, k^i, b^i) for $i = 1, 2, \ldots, n$, where $-\infty = w^0 < w^1 < \ldots < w^n = w_0$. A PWL function is formally defined as $U(w) = k^i w + b^i$ where $w \in (w^{i-1}, w^i]$. If an initial wealth level w_0 is assumed, each execution trajectory would correspond to a wealth level, which equals to the initial wealth level minus the accumulated cost. Based on a specific utility function, we can get the utility of an execution trajectory. We can also compute the *expected utility* of a policy π, which is the sum of the utilities of the different trajectories that are possible given π weighted by their probabilities.

3.2 Risk-Sensitive Criterion

For MDPs and POMDPs, if the system only has a limited cost budget at the start and the objective is to accomplish some tasks before the cost budget is

used up, then maximizing the probability of achieving the goals without using up the cost budget may be a more reasonable criterion than the MEC criterion. We call this probability the *reachable probability*.[1] With this motivation in mind, Yu *et al.* [1998] proposed the *Risk-Sensitive criterion* (RS-criterion) for MDPs. The objective of this criterion is to find a policy that maximizes the reachable probability, which is equivalent to finding a policy that maximizes the probability of reaching a goal state with an accumulated cost that is no larger than an initial cost threshold. But, actually, the RS-criterion also represents a risk attitude that can be applied on more general situations. For example, the objective of the regular MEC criterion is to find optimal policies with the minimum expected cost. While such a policy is good in the expected case, it cannot provide any guarantee that it would not result in an exorbitantly high accumulated cost. If the system only executes the optimal policy of the MEC criterion once, the system may encounter an exorbitantly high accumulated cost, which should be avoided. If the system can identify an accumulated cost level and considers accumulated cost greater than this level as risk, then this is also a scenario that the RS-criterion can describe and, hence, the RS-criterion is a risk-sensitive criterion. From the viewpoint of MDPs with utility functions, the RS-criterion is equivalent to having a initial wealth level $w_0 = \theta_0$ and a step utility function:

$$U(\theta) = \begin{cases} 0 & \text{if } \theta < 0 \\ 1 & \text{if } \theta \geq 0 \end{cases} \tag{3.1}$$

[1]The notion of reachable probabilities is also used in regular goal-directed MDPs and POMDPs to refer to the probability of reaching a goal state *without* considering cost budgets. As this definition is equivalent to ours when the cost budget is infinite, we will misuse this terminology to refer to both concepts.

where θ is the current cost threshold (equivalent to wealth level), and the utility is 1 when the cost threshold is non-negative, and 0 otherwise.

In this dissertation, we will discuss *Risk-Sensitive MDPs* (RS-MDPs) and *Risk-Sensitive POMDPs* (RS-POMDPs), which combine the RS-criterion with MDPs and POMDPs, respectively.

Chapter 4

Risk-Sensitive MDPs (RS-MDPs)

Risk-Sensitive MDPs (RS-MDPs) is the probabilistic planning model that combine *Markov Decision Processes* (MDPs) and the *Risk-Sensitive criterion* (RS-criterion). This section begins by describing the RS-MDP (Risk-Sensitive MDP) model in Section 4.1. Then, four algorithms – *Functional Value Iteration* (FVI), *Augmented MDPs*, *Depth First Search* (DFS) and *Dynamic Programming* (DP) – are introduced for RS-MDPs in Section 4.2. Sections 4.3 show the theoretical result for RS-MDPs that allow costs to be negative, and 4.4 discusses RS-MDPs that allow costs to be zero and introduce two new algorithms for it. Section 4.5 gives the complexity of RS-MDPs and its corresponding proof. Finally, we show the experimental result of different RS-MDP algorithms in Section 4.6.

4.1 RS-MDP Model

Risk-Sensitive MDPs (RS-MDPs) are MDPs with the objective of optimizing for the RS-criterion. Specifically, given a cost budget θ_0, the goal is to find a policy that maximizes the reachable probability of the start state $P(s_0, \theta_0)$, that is, a policy that maximizes the probability of reaching a goal state. Specifically, the

reachable probability $P(s, \theta)$ can be formally defined under the situation that the system execution start from state s and the available cost budget is θ. Recall that a general policy π, which may not be stationary, would define a probability $P^{\pi}(\tau)$ for each execution trajectory τ starting from a state s and each execution trajectory τ defines a accumulated cost $C(\tau)$; the reachable probability is defined as $P(s, \theta) = \sum_{\tau \in \mathcal{T}} P^{\pi}(\tau)$, where $\mathcal{T} = \{\tau \mid C(\tau) \leq \theta\}$.

Formally, an RS-MDP is defined by the tuple $\langle \mathcal{M}, \mathbf{\Theta}, \theta_0 \rangle$, where:

- $\mathcal{M}$ is an MDP
- $\mathbf{\Theta}$ is the set of all possible cost thresholds
- $\theta_0 \in \mathbf{\Theta}$ is a predefined initial cost threshold.

For RS-MDPs, the optimal action choice and the optimal reachable probability depend not only on the current state s, but also on the current cost threshold θ. It means that the optimal policies for RS-MDPs may be non-stationary with respect to the state. Since the cost threshold θ summarizes all necessary information of initial cost threshold θ_0 and the whole execution history, the optimal policies for RS-MDPs are stationary with respect to both state and cost thresholds. Thus, rather than the expected cost function and optimal policy form under the MEC criterion, the reachable probability function and optimal policies have forms as functions $P : \mathbf{S} \times \mathbf{\Theta} \to [0, 1]$ and $\pi : \mathbf{S} \times \mathbf{\Theta} \to \mathbf{A}$, where $\mathbf{\Theta}$ is the set of all possible cost thresholds.

4.2 RS-MDP Algorithms

We describe two general classes of algorithms to solve RS-MDPs. The first is done by representing the RS-criterion using continuous value functions, and then using *Functional Value Iteration* (FVI) [Liu and Koenig, 2005b; 2006; Marecki

and Varakantham, 2010] to solve the problem. The second is done by implicitly representing RS-MDPs as augmented MDPs, and then using classical methods like depth-first search and dynamic programming to solve the problem.

4.2.1 Functional Value Iteration

Functional Value Iteration (FVI) [Liu and Koenig, 2005b; 2006] is an algorithm that solves MDPs with arbitrary utility functions. As such, we first describe its general approach for arbitrary utility functions before describing how we optimize it for the utility functions corresponding to the RS-criterion.

Similar to RS-MDPs, the optimal action choice for MDPs with utility functions depend on both states and wealth levels, which are the cost thresholds when the initial cost threshold equals the initial wealth level, i.e., $\theta_0 = w_0$, namely the amount of unused cost. The expected utility function $V : \mathbf{S} \times \mathbf{W} \to \mathbb{R}$ and optimal policies $\pi : \mathbf{S} \times \mathbf{W} \to \mathbf{A}$ have forms as functions of the pairs of state and wealth level. By considering the above form of expected utility function, it forms an equation similar to the Bellman equation:

$$V(s, w) = \max_{a \in \mathbf{A}} \sum_{s' \in \mathbf{S}} T(s, a, s') V(s', w - C(s, a, s')) \tag{4.1}$$

The set of possible wealth level $\mathbf{W}$ is actually infinitely countable, so the expected utility function cannot be represented as a finite table. FVI considers $\mathbf{W}$ as a continues space and represents the expected utility functions $V(s, w)$ as a function that maps states to utility functions with form $\mathbf{W} \to \mathbb{R}$, namely functions from wealth levels to utilities. Formally, the expected utility functions are represented as the form $V : \mathbf{S} \to (\mathbf{W} \to \mathbb{R})$. Then, the above equation can be rewritten as:

$$V(s)(w) = \max_{a \in \mathbf{A}} \sum_{s' \in \mathbf{S}} T(s, a, s') V(s')(w - C(s, a, s')) \tag{4.2}$$

which operates on utility functions rather than numbers. FVI first sets the function of goal states as the utility function, and uses the above equation to perform updates in parallel over all non-goal states to get a new utility functions repeatedly. FVI looks like original VI except that it uses functions for the original states instead of single numbers.

Recall that any arbitrary risk attitudes can be represented as utility functions, which can then be approximated by *Piecewise Linear* (PWL) utility functions (see Section 3.1). Since the utility function corresponding to the RS-criterion is a step function (see top left subfigure in Figure 4.1), the FVI update iteration would produce a specific type of PWL functions – *Piecewise Constant* (PWC) utility functions. While FVI can be used with any PWL functions, we now describe how it can be optimized when it is used with PWC functions (specific for the RS-criterion).

For PWC utility functions generated by FVI under the RS-criterion, it is clear that they would have zero utility when the cost level is negative. Thus, if an initial cost threshold θ_0 is given, then utility functions are enough to only record values over the interval $[0, \theta_0]$. Each utility function can be represented as an ordered list of pairs (θ^i, p^i) for $i = 0, 1, 2, \ldots, n$, where $\theta^0 = 0$ and $\theta^0 < \theta^1 < \ldots < \theta^n \leq \theta_0$ and a PWC function can be formally defined as $U(\theta) = p^i$ while $\theta \in [\theta^i, \theta^{i+1})$ and $U(\theta_0) = p^n$.

For example, Figure 4.1 shows examples of PWC utility functions. Consider a simple MDP with $\mathbf{S} = \{s_0, s_1, s_2, s_3\}$; $\mathbf{G} = \{s_1, s_2, s_3\}$; and an action a. Let us assume $T(s_0, a, s_i) = P^i$, $C(s_0, a, s_i) = \theta^i$, $\theta^1 < \theta^2 < \theta_0$ and cost $\theta^3 > \theta_0$ exceeds the initial cost threshold. The upper left subfigure of Figure 4.1 is the utility

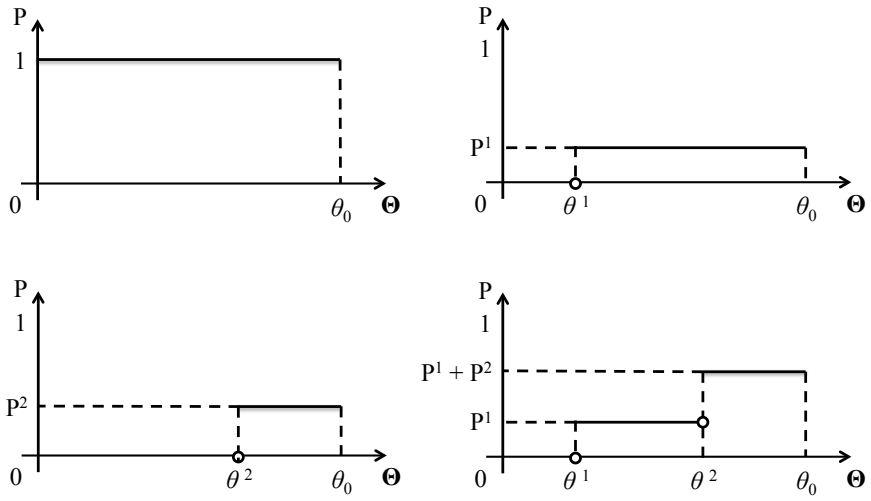

Figure 4.1: PWC Utility Functions

function of the RS-criterion, and it is the utility function for all goal states s_1, s_2, and s_3. The upper right and lower left subfigures of Figure 4.1 represent the utility function components corresponding to transition (s_0, a, s_1) and (s_0, a, s_2), respectively. Taking the upper right subfigure of Figure 4.1 as example, if the agent is at state s_0 and it has a current cost threshold $\theta \in [\theta^1, \theta_0]$, then it receives a utility equal to the transition probability P^1 by applying action a since the goal state s_1 can be reached safely with the transition probability P^1. The utility function of the initial state s_0 is thus the sum of these two utility function components and is shown in the lower right of Figure 4.1, which is a PWC function.

4.2.2 Representing RS-MDPs as Augmented MDPs

We now describe how RS-MDPs can be represented as augmented MDPs, which will allow classical MDP algorithms to be adapted to solve them. Each RS-MDP $\langle \mathcal{M}, \mathbf{\Theta}, \theta_0 \rangle$, where $\mathcal{M} = \langle \mathbf{S}, \mathbf{A}, \mathbf{T}, \mathbf{C}, \mathbf{G}, s_0 \rangle$ is an MDP, can be modeled as an augmented MDP $\langle \widehat{\mathbf{S}}, \mathbf{A}, \widehat{\mathbf{T}}, \mathbf{R}, \widehat{\mathbf{G}}, \hat{s}'_0 \rangle$, where:

- $\widehat{\mathbf{S}} : \mathbf{S} \times \mathbf{\Theta}$ is the set of augmented states (s, θ);
- the transition function $\widehat{\mathbf{T}} : \widehat{\mathbf{S}} \times \mathbf{A} \times \widehat{\mathbf{S}} \to [0, 1]$ is $\widehat{T}((s, \theta), a, (s', \theta')) = T(s, a, s')$ if $\theta' = \theta - C(s, a, s')$, otherwise it equals 0;
- the reward function $\mathbf{R} : \widehat{\mathbf{S}} \times \mathbf{A} \times \widehat{\mathbf{S}} \to [0, 1]$ is $R((s, \theta), a, (s', \theta')) = 1$ if $s \notin G$, $s' \in G$, $\theta' = \theta - C(s, a, s')$ and $\theta' \geq 0$, otherwise it equals 0;
- the set of goal states $\widehat{\mathbf{G}} = \{(s_g, \theta) \mid s_g \in \mathbf{G}, \theta \geq 0\}$; and
- the initial augmented state $\hat{s}_0 = (s_0, \theta_0)$.

Based on this line of thought, one can also generate augmented MDPs for MDPs with utility functions, but the augmented state space $\mathbf{S} \times \mathbf{W}$ would be infinite because the set of possible wealth levels $\mathbf{W}$ is countable infinite. But for RS-MDPs, it is possible to merge all augmented states with negative cost thresholds to a state $\hat{s}_\perp$ and ignore any transitions for $\hat{s}_\perp$ because the reachable probabilities of all augmented states with negative cost are actually 0. Unlike the set of wealth levels $\mathbf{W}$, it is enough to only consider $\mathbf{\Theta}$ as the set of non-negative possible cost thresholds, and then the set of possible cost threshold $\mathbf{\Theta}$ would be finite. If we define $\mathbf{\Theta}_0 = \{\theta_0\}$ and $\mathbf{\Theta}_i = \{\theta \mid \theta = \theta' - c, \theta' \in \mathbf{\Theta}_{i-1}, c \in \mathbf{C}, \theta \geq 0\}$, then $\mathbf{\Theta}$ can be obtained by inductively computing $\mathbf{\Theta}_i$ until it does not change anymore, i.e., $\mathbf{\Theta} = \mathbf{\Theta}_k$ when $\mathbf{\Theta}_k = \mathbf{\Theta}_{k+1}$. Unlike regular MDPs, the goal here would be to find a policy that maximizes the expected reward (equivalent to maximizing the reachable probability in RS-MDPs). Such an MDP is also called a MAXPROB MDP [Kolobov *et al.*, 2011; Kolobov, 2013], which is further discussed in Chapter 7.

For example, Figure 4.2 shows the transition graphs of an example MDP and its corresponding augmented MDP. The left subfigure of Figure 4.2 shows the original MDP and the cost functions. We assume s_0 is the initial state and s_3 is the only goal state. If the initial cost threshold is 6, the right subfigure of

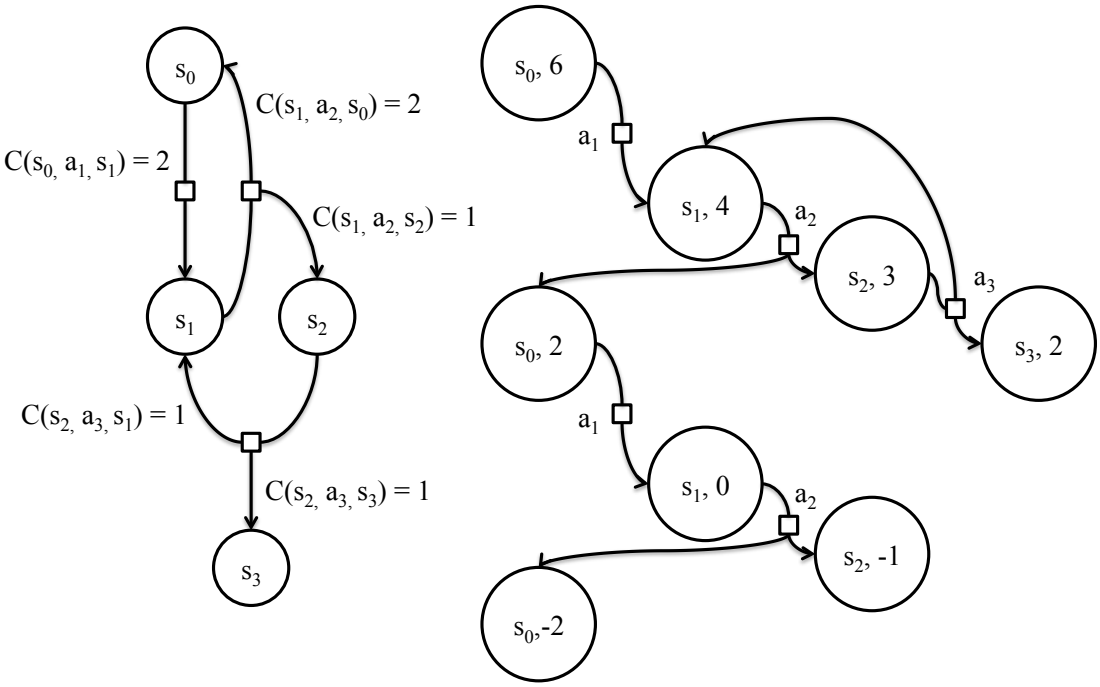

Figure 4.2: Augmented MDP

Figure 4.2 shows its corresponding augmented MDP and the transition graph only shows all reachable augmented states. From the figure, if there exists a transition (s_0, a_1, s_1) and its corresponding cost is 2, then the corresponding transitions in the augmented state space will be $((s_0, 6), a_1, (s_1, 4))$ and $((s_0, 2), a_1, (s_1, 0))$. We omit the transitions in the figure. We also omit the rewards of augmented MDP in the figure. For the augmented MDP, all rewards equal 0 except for the transition $((s_2, 3), a_3, (s_3, 2))$, where $R((s_2, 3), a_3, (s_3, 2)) = 1$ since it is the transition that go into a goal state from a non-goal state with positive cost threshold. In addition, the augmented state $(s_0, -2)$ and $(s_2, -1)$ can be combined to one augmented state since they are with negative cost.

4.2.3 Depth First Search (DFS)

Once an RS-MDP is represented as an augmented MDP, we can use classical methods like depth-first search to solve it. Clearly, the reachable probabilities of any augmented goal states $\hat{s}_g \in \widehat{\mathbf{G}}$ and augmented states with negative cost thresholds are 1 and 0, respectively. By assigning the reachable probabilities of augmented goal states and augmented states with negative cost thresholds in advanced, the relationship between the reachable probability of augmented state (s, θ) and its successor $(s', \theta - C(s, a, s'))$ can be summarize as follows:

$$P(s, \theta) = \max_{a \in \mathbf{A}} \sum_{s' \in \mathbf{S}} T(s, a, s') P(s', \theta - C(s, a, s')) \tag{4.3}$$

Generating the augmented MDP explicitly is very inefficient with respect to both time and memory. If we do not generate the augmented MDP, then augmented goal states and augmented states with negative cost thresholds can be ignored and one can also characterize the reachable probabilities with the following system of equations:

$$P(s, \theta) = \max_{a \in \mathbf{A}} \sum_{s' \in \mathbf{S}} \begin{cases} 0 & \text{if } \theta < C(s, a, s') \\ T(s, a, s') & \text{if } s' \in \mathbf{G},\ \theta \geq C(s, a, s') \\ T(s, a, s') P(s', \theta - C(s, a, s')) & \text{if } s' \notin \mathbf{G},\ \theta \geq C(s, a, s') \end{cases} \tag{4.4}$$

Equation 4.4 shows the components of reachable probability from augmented state (s, θ) assuming that one takes action a from state s and transitions to successor state s', which have three cases, as below:

- If the cost threshold θ is smaller than the action cost $C(s, a, s')$, then the successor can only be reached by exceeding the cost threshold. Thus, the reachable probability is 0.

- If the successor is a goal state and the cost threshold is larger than or equal to the action cost $C(s, a, s')$, then the successor can be reached without exceeding the cost threshold. Thus, the reachable probability is the transition probability $T(s, a, s')$.
- If the successor is not a goal state and the cost threshold is larger than or equal to the action cost $C(s, a, s')$, then the successor can be reached without exceeding the cost threshold. Thus, the reachable probability can be recursively determined as the transition probability $T(s, a, s')$ multiplied by the reachable probability of a new augmented state $P(s', \theta - C(s, a, s'))$.

One can extract the optimal policy by taking the action that is returned by the maximization operator in Equation 4.4 for each augmented state (s, θ).

Because all costs are positive, any augmented successor states have a smaller cost threshold than its predecessor, and the reachable probability of augmented states only depend on the reachable probabilities of augmented states with smaller cost thresholds. The augmented MDP would form a transition graph, in which each node is an augmented state (s, θ). Any transition edges in the transition graph go from augmented states with larger cost thresholds to augmented states with smaller cost thresholds, which indicates that there are no cycles in the transition graph. Therefore, the transition graph forms a *Directed Acyclic Graph* (DAG) of augmented states. Since the transition graph is a DAG of augmented states and augmented states form no cycles, a *Depth First Search* (DFS) style algorithm can be proposed to traverse the transition graph from the initial augmented state (s_0, θ_0) and calculate reachable probability $P(s, \theta)$ for each reachable augmented state (s, θ) based on the *reverse topological sort*. Algorithm 1 shows the pseudocode of the algorithm.

In Algorithm 1, it first applies a DFS style procedure REVERSE-TOPOLOGY-

Algorithm 1: DFS$(\theta_0, \mathcal{M})$

```
1 ⟨(s^1, θ^1), (s^2, θ^2), ..., (s^n, θ^n)⟩ = REVERSE-TOPOLOGY-ORDER(s_0, θ_0)
2 for augmented state (s^i, θ^i) with indices i = 1 to n do
3 |   UPDATE(s^i, θ^i)
4 end
```

ORDER, which traverses all augmented states that is reachable from the initial augmented state (s_0, θ_0), decides their reverse topological sort and record its corresponding indices (line 1). Then the algorithm invokes procedure UPDATE for each augmented states based on their reverse topological sort (lines 2-4). Procedure UPDATE computes the reachable probability $P(s, \theta)$ based exactly on Equation 4.4 (lines 5-22). Every time the reachable probability is computed for an augmented state (s, θ), the procedure records the reachable probability $P(s, \theta)$ and the optimal action $\pi(s, \theta)$ (lines 23-24).

In practice, DFS can be implemented recursively, which will improve its efficiency. The recursive version of DFS executes UPDATE(s_0, θ_0) directly and the Procedure UPDATE has one more line UPDATE(s', θ') between lines 13 and 14.

An important design decision when implementing some of the RS-MDP algorithms, including DFS, is the choice of the global data structure to store the reachable probabilities $P(s, \theta)$. We can either store them as utility functions that dynamically grow (as is used by FVI) or in a fixed-size latticed table (as is used by VI on augmented MDP). Choosing the latter method requires determining the size of the set $\boldsymbol{\Theta}$ in advanced, as the size of the latticed table is proportional to the size of that set. The former method alleviates the need for such restrictions as the size of the utility function can grow as required. However, this comes at the cost of longer runtimes to search and insert elements into the data structure, which take $O(\log(|\boldsymbol{\Theta}|))$ and $O(|\boldsymbol{\Theta}|)$, respectively. The runtime for these two operations

Procedure Update(s, θ)

```
5  P* = 0
6  for a ∈ A do
7      P_a = 0
8      for s' ∈ S | T(s, a, s') > 0 do
9          if θ ≥ C(s, a, s') then
10             if s' ∈ G then
11                 P_a = P_a + T(s, a, s')
12             else
13                 θ' = θ − C(s, a, s')
14                 P_a = P_a + T(s, a, s') · P(s', θ')
15             end
16         end
17     end
18     if P_a > P* then
19         P* = P_a
20         a* = a
21     end
22 end
23 P(s, θ) = P*
24 π(s, θ) = a*
```

is only $O(1)$ when using a latticed table.

If the RS-MDP has integer costs only, i.e., the cost function is $\mathbf{C} : \mathbf{S} \times \mathbf{A} \times \mathbf{S} \to \mathbb{Z}^+$, the set of possible cost threshold $\mathbf{\Theta}$ can be assumed to be $\{0, 1, 2, \ldots, \theta_0\}$. Thus, a latticed table, where each element $[s, i]$ of the table is used to store the reachable probability $P(s, \theta = i)$, can be used. This allows one to use i as the index to reference $P(s, \theta)$, which makes the search and insert operations to be very fast in practice.

In fact, even when the costs are not integers, any real number must have limited precision in a real machine and one can convert problems with higher precision costs and cost thresholds to problems with integer costs and cost thresholds by multiplying them with a sufficiently large constant. If the memory needed for the latticed table with exact precision is too big, the cost threshold can be equally discretized in order to bound the size of the latticed table. Formally, interval $[0, \theta_0]$ can be equally discretized as n interval, and the costs c would be approximated as $\frac{i \cdot \theta_0}{n}$ if $c \in (\frac{(i-1) \cdot \theta_0}{n}, \frac{i \cdot \theta_0}{n}]$, and the set of possible cost threshold can be approximated as $\mathbf{\Theta} = \{\frac{i \cdot \theta_0}{n}\}$ where $i = 0, 1, \ldots, n$. Thus, all possible cost threshold can be indexed directly by an integer i. DFS using this form of latticed table to represent $P(s, \theta)$ is equivalent to having a perfect hash scheme for augmented states.

4.2.4 Dynamic Programming (DP)

Dynamic Programming (DP) is another algorithm to solve RS-MDPs, which can be very effective when it uses a latticed table to store the reachable probabilities. For simplicity, we assume that the costs are all integers in our pseudocode and our description below. It is fairly straightforward to generalize this algorithm for non-integer costs or when utility functions are used to store the reachable probabilities.

For RS-MDPs, the reachable probabilities of augmented states only depend on augmented states with smaller cost thresholds, which is showed by Equation 4.4 because all costs are positive. Thus, the key idea of DP is that reachable probabilities in table $P[s, \theta]$ can be updated from index 0 to θ_0. Algorithm 2 shows the pseudocode of the DP when a latticed table $P[s, \theta]$ is used to represent the reachable probability $P(s, \theta)$.

Unlike DFS, which traverses the transition graph from initial augmented state

Algorithm 2: $\mathrm{DP}(\theta_0, \mathcal{M})$

```
for s_g ∈ G do
    Mark-Predecessor(s_g, 0)
end
for s ∈ S do
    if s ∈ G then
        P[s, 0] = 1
    else
        P[s, 0] = 0
    end
end
for θ = 1 to θ_0 do
    for s ∈ S do
        if (s, θ) is marked then
            Update(s, θ)
            if P[s, θ] > P[s, θ − 1] then
                Mark-Predecessor(s, θ)
            end
        else
            P[s, θ] = P[s, θ − 1]
            π[s, θ] = π[s, θ − 1]
        end
    end
end
```

(s_0, θ_0), DP traverse the *transposed graph* of the connectivity graph of augmented states, where the direction of all transition edges are reversed, from all augmented state $(s_g, 0)$ with a originial goal state and cost threshold 0. The traversal would stop when it encountered augmented states (s, θ) with cost threshold θ greater than θ_0. Firstly, Algorithm 2 mark all predecessors of augmented goal states by a procedure MARK-PREDECESSOR (lines 25-27). Then, DP would update the reachable probabilities in table $P[s, \theta]$ from index 0 to θ_0. When the index is 0, set $P[s_g, 0]$ to 1 while s_g is a goal state, 0 otherwise (lines 28-34). From index 1 to θ_0, if the corresponding augmented states are marked during traversal, DP update it using Procedure UPDATE and continue the traversal, otherwise set it as $P[s, \theta] = P[s, \theta - 1]$ (lines 35-47). An augmented state is marked only if it needs to be updated, and an augmented state needs to be updated only if one of its successor state is updated with a larger reachable probability (lines 39-41).

While DFS is efficient in that it only updates reachable augmented states and ignores the unreachable ones, the policy that DFS finds is not available for all possible augmented states. If the value of the initial threshold θ_0 changes, sometimes one has to run DFS one more time to compute policy for the new initial augmented state. The new round of DFS can be accelerated by reusing previous solution information of reachable probabilities. On the other hand, like FVI, DP finds optimal policies for all thresholds.

4.3 RS-MDPs with Negative Costs

Up to here, almost all the discussion is based on the assumption that the costs are strictly positive except for transitions from goal states. For RS-MDPs, if the costs are allowed to be negative, Theorem 1 shows that solving RS-MDPs with

negative costs is undecidable.

Theorem 1 *Solving RS-MDPs with negative costs is undecidable.*

PROOF: For RS-MDPs with negative costs, if an execution trajectory does not arrive at any goal state, its accumulated cost may not be positive infinite, and sometimes it could even be negative infinity. Stationary policies with respect to the augmented state space may form *full negative cost cycles*, which is similar to negative-weight cycles in *shortest path problems*. If a stationary policy with a full negative cost cycle exists, then its expected cost would be negative infinity. The agent can gain enough cost budget along the full negative cost cycle before then turn to goal states. So, stationary policies with respect to augmented state may not be optimal for RS-MDPs anymore. It has been shown that checking for the existence of a policy, which may not be stationary, that guarantees reaching a goal with an arbitrarily small expected cost is undecidable [Chatterjee *et al.*, 2016]. The objective of RS-MDPs subsumes the above objective. As such, it is undecidable as well. ■

4.4 RS-MDPs with Zero Costs

If RS-MDPs can have zero costs, then FVI and VI on augmented MDPs can still be used to solve them. However, the DFS and DP algorithms will be erroneous due to cycles with zero costs in the augmented MDP. To overcome this limitation, we combine them with the *Topological Value Iteration* (TVI) [Dai *et al.*, 2011] algorithm, which results in the TVI-DFS and TVI-DP algorithms. Below, we first describe the TVI algorithm before describing how it can be used to form the two algorithms.

4.4.1 Topological Value Iteration (TVI)

It is possible to partition the connectivity graph or the transition graph of the MDP into *Strongly Connected Components* (SCCs) in such a way that they form a *Directed Acyclic Graph* (DAG) [Tarjan, 1972; Dai *et al.*, 2011]. More formally, an SCC of a directed graph $G = (V, E)$ is a maximal set of vertices $Y \subseteq V$ such that every pair of vertices u and v in Y are reachable from each other. Since the SCCs form a DAG, it is impossible to transition from a state in a downstream SCC to a state in an upstream SCC.

The upper left subfigure of Figure 4.3 shows one example of an SCC and the right subfigure of Figure 4.3 shows one example of DAG of SCCs, which corresponds to a connectivity graph. In Figure 4.3, we use circles to represent states, arrows to represent action and rectangles to represent SCC. From the example SCC, we could see that each state in this SCC have transition paths to reach every other state, and if a transition go out this SCC, the transition will never return to the SCC. For the DAG example, we omit the states and transitions inside each SCC and only show the transitions between SCCs. The leaves of the DAG are SCCs formed by a single goal state, namely goal SCCs, and we mark the indices of the non-goal SCCs in reverse topological sort order on the upper left corner of rectangles. For MDPs with dead-ends, if a leaf SCC is not a goal SCC, then all states in this SCC are dead-ends

For SSP-MDPs, *Topological Value Iteration* (TVI) [Dai *et al.*, 2011] is an algorithm that exploits an MDP's graphical structure property revealed by the DAG of SCCs. Since the SCCs form a DAG, states in an SCC only affect the states in upstream SCCs. Thus, by choosing the SCCs one by one in reverse topological sort order, TVI performs Bellman update for all states in one SCC until they converge, and then it no longer needs to consider SCCs whose states are already

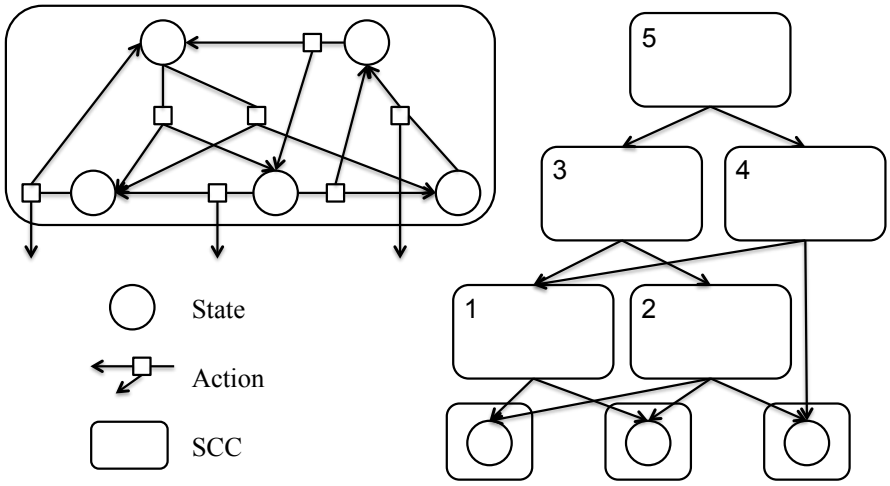

Figure 4.3: SCC and DAG of SCCs in Connectivity Graph

converged anymore.

If all states in transition graph except goal states form only one SCC, then the update iteration process is equivalent to VI, and it would be even slower. TVI would be much more efficient than VI if the transition graph have many SCCs.

4.4.2 TVI-DFS

When the RS-MDP costs can be zero, the connectivity graph of augmented MDPs would have cycles of augmented states, which must be formed by transition edges with cost zero. Thus, the augmented states could form SCCs, but all augmented states in one SCCs must have exact same cost threshold because transition edges that form cycles must be with cost zero. Notice that both the original MDP and the augmented MDP form connectivity graphs, which can be identified as the DAG of SCCs, but the DAG and SCCs are different between the original MDP and the augmented MDP.

For example, Figure 4.4 shows the transition graph of one MDP with zero costs and its corresponding augmented MDP. The left of Figure 4.4 shows the

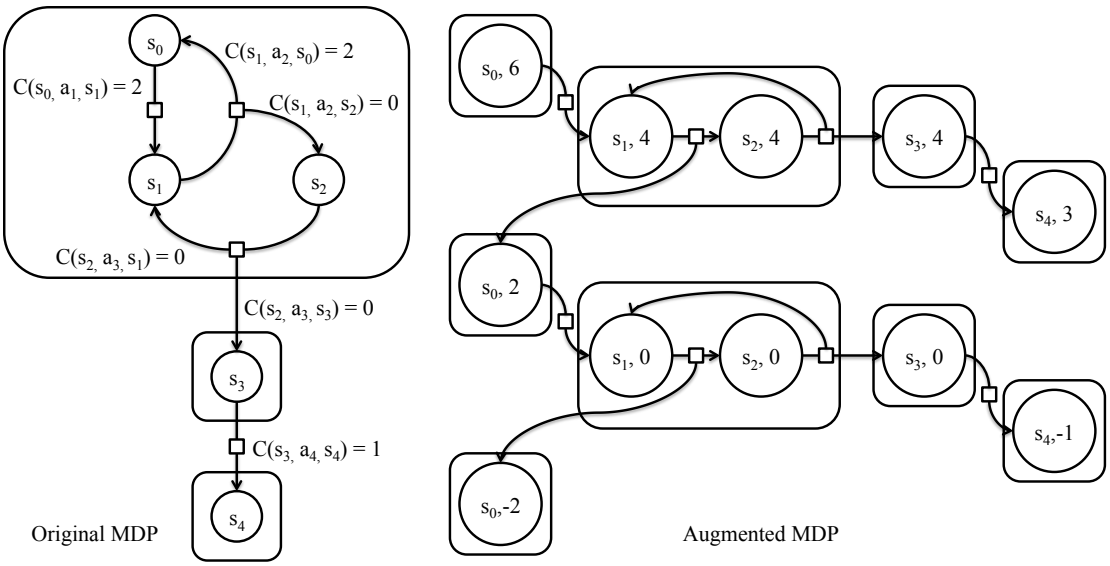

Figure 4.4: Augmented MDP of RS-MDP with Zero Costs

original MDP and cost functions are listed in the subfigure. We assume s_0 is the initial state and the s_4 is the only goal state. If the initial cost threshold is 6, the right subfigure of Figure 4.4 shows its corresponding augmented MDP and the transition graph only shows all reachable augmented states. In the figure, we also use rectangles to denote each SCC for both the original MDP and the augmented MDP. It shows the SCCs from the original MDP and augmented MDP are different and all augmented states in one SCC have exactly the same threshold.

For the DFS algorithm, we modified it to solve RS-MDPs with zero costs, and we call the new algorithm *TVI-DFS*, which adopts ideas from *Topological Value Iteration* (TVI) [Dai *et al.*, 2011]. Similar to DFS, TVI-DFS also traverses the augmented state space from initial augmented state (s_0, θ_0). After the DAG of SCCs are identified, TVI-DFS updates augmented states of SCCs in the reverse topological sort. When augmented states in an SCC are updated in parallel, it performs the update iteration repeatedly until convergence, which is equivalent to performing VI on the set of augmented states in this SCC. At a high level, TVI-

Algorithm 3: TVI-DFS($\theta_0, \mathcal{M}$)

```
Y = FIND-SCCS(s_0, θ_0)
for SCCs y_i ∈ Y with indices i = 1 to n do
    UPDATE-SCC(y_i)
end
```

Procedure Update-SCC(y_i)

```
for (s, θ) ∈ y_i do
    P(s, θ) = 0
end
repeat
    residual = 0
    for (s, θ) ∈ y_i do
        P'(s, θ) = P(s, θ)
        UPDATE(s, θ)
        if residual < |P(s, θ) − P'(s, θ)| then
            residual = |P(s, θ) − P'(s, θ)|
        end
    end
until residual < ε;
```

DFS is identical to TVI, except that it operates on an augmented MDP instead of a original MDP and it uses Equation 4.4 to update the reachable probabilities of each augmented state instead of using the Bellman equation to update the cost value of each state. Recall that TVI is efficient when there are many SCCs in the transition graph, augmented MDPs have this property since each SCC can only contain augmented states with same cost threshold. Algorithm 3 shows the pseudocode of the algorithm.

TVI-DFS first partitions the augmented MDP state space into SCCs with

Tarjan's algorithm [Tarjan, 1972], which traverses the transition graph in a depth-first manner and marks the SCC membership of each state (line 48). Tarjan's algorithm returns an SCC transition tree Y, where the SCC indices are in reverse topological sort order. Then, TVI-DFS updates the augmented states in the SCCs in reverse topological sort order (lines 49-51). Notice that the SCCs TVI-DFS identified in Y do not include any augmented goal states and augmented states with negative cost thresholds. This process is similar to popping elements of a stack that are pushed in depth-first order. For each SCC, the algorithm performs a VI-like update using Equation 4.4 until the *residual* of all augmented states, defined as the difference in the reachable probability between subsequent iterations, are within ϵ (lines 55-64). Additionally, in the case that one SCC is formed by augmented states with dead-ends in original state space, TVI-DFS initializes the reachable probabilities as 0 before the update (lines 52-54) and this can be omitted if dead-ends can be identified in the original state space and all reachable probabilities of augmented states with dead-ends are predefined as 0.

4.4.3 TVI-DP

Corresponding to TVI-DFS, another algorithm – TVI-DP – is available for RS-MDPs with zero costs. Like DP, TVI-DP would perform updates on augmented states from smaller cost thresholds to higher cost thresholds. TVI-DP traverses the transposed graph rooted at all augmented states $(s_g, 0)$ where $s_g \in \mathbf{G}$. Figure 4.5 show the transposed graph corresponding to the RS-MDP in Figure 4.4 and the transposed graph is rooted at $(s_4, 0)$.

TVI-DP only use Equation 4.4 to explicitly update the reachable probability of augmented state that is traversed in the transposed graph rooted at all augmented states $(s_g, 0)$ where $s_g \in \mathbf{G}$. This is because any augmented states that is not in

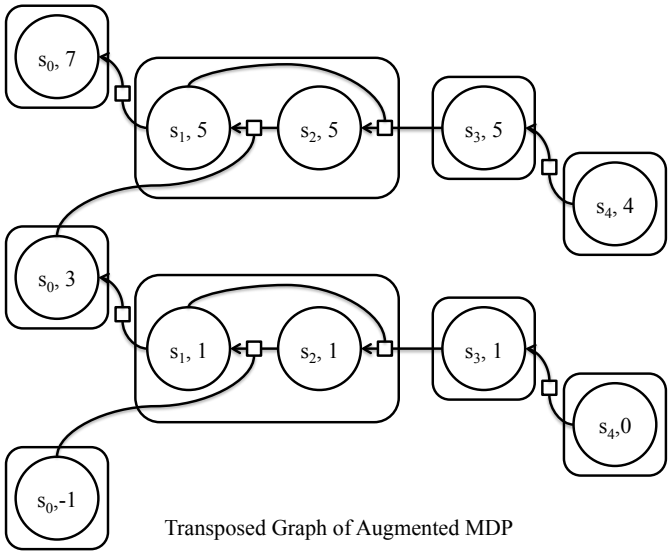

Figure 4.5: Transposed Graph of of RS-MDP with Zero Costs

the transposed graph have the exact same reachable probability as augmented states with the same original state and the adjacent cost threshold that is less. For these augmented state with adjacent cost threshold, the key observation is that there exists exactly the same possible trajectories that reach goal state with non-negative cost threshold. Thus, TVI-DP will assign the same reachable probability for those augmented state. For example, in the RS-MDP showed by Figure 4.4, the reachable probability of augmented state $(s_0, 4)$ is the same as that of augmented state $(s_0, 3)$. So, only $P(s_0, 3)$ is updated explicitly and $P(s_0, 4)$ is assigned as $P(s_0, 3)$, i.e., $P(s_0, 4) = P(s_0, 3)$. Like TVI-DFS, TVI-DP performs a VI-like update on one SCC by another SCC of augmented states. And similar to DP, an SCC of augmented states needs to be updated only if one of its successor state is goal or updated with a larger reachable probability. Additionally, when the augmented states with same cost threshold are update, TVI-DP perform updates based on the reverse topology order of SCCs. TVI-DP will only update augmented states with cost threshold $\theta \in [0, \theta_0]$, so the augmented states $(s_0, -1)$ and $(s_0, 7)$

Algorithm 4: TVI-DP(θ_0)

```
65 for s_g ∈ G do
66 |   P(s_g, 0) = 1
67 end
68 Y = Find-Transposed-SCCs(G, 0)
69 for θ = 0 to θ_0 do
70 |   for SCCs y_j^θ ∈ Y with indices j = 1 to m do
71 |   |   Update-SCC(y_j^θ)
72 |   end
73 |   for (s, θ) ∉ Y do
74 |   |   if θ = 0 then
75 |   |   |   P[s, 0] = 0
76 |   |   else
77 |   |   |   P[s, θ] = P[s, θ − 1]
78 |   |   |   π[s, θ] = π[s, θ − 1]
79 |   |   end
80 |   end
81 end
```

in Figure 4.4 will be ignored by TVI-DP. Algorithm 4 shows the pseudocode of TVI-DP. Like for DP, we assume here that the costs are all integers in the pseudocode and use a latticed table to store the reachable probabilities.

In Algorithm 4, TVI-DP first sets $P[s_g, 0]$ to 1 while s_g is a goal state (lines 65-67). Then, TVI-DP runs Tarjan's algorithm on the transposed graph from all augmented states $(s_g, 0)$ while $s_g \in \mathbf{G}$ (line 68). Once the DAG of SCCs is found, TVI-DP updates the augmented states starting from the augmented states with thresholds $\theta = 0$ to the states with thresholds $\theta = \theta_0$ (lines 69-81). For each group of augmented states with the same threshold, TVI-DP updates the augmented states in the SCCs in reverse topological sort order (lines 70-72). An

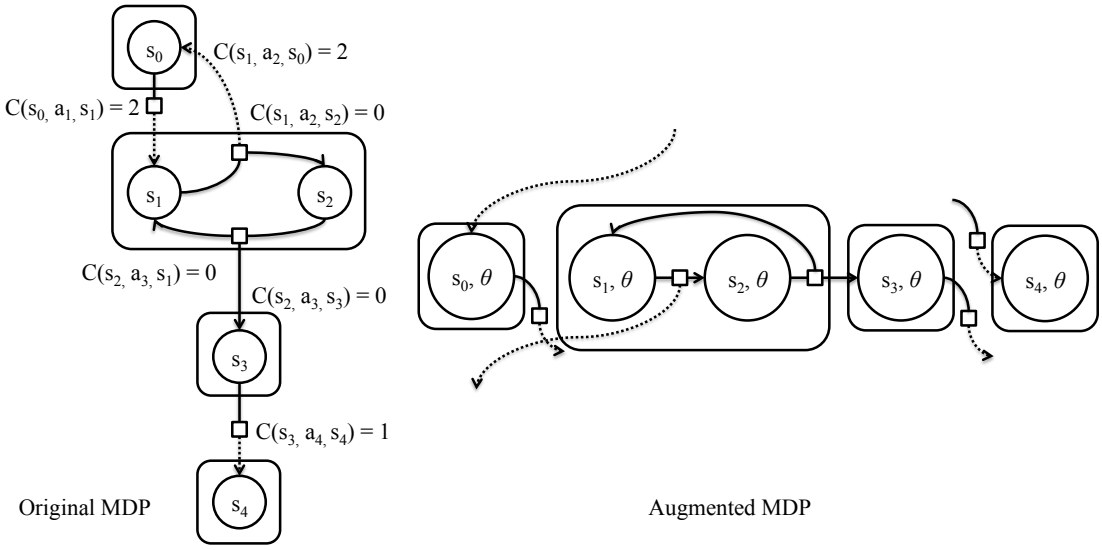

Figure 4.6: SCC of augmented states with a particular cost threshold

important property here is that all augmented states in an SCC must have the same threshold. For each augmented state $(s, 0)$ that is not traversed by FIND-TRANSPOSED-SCCS, TVI-DP set the reachable probability to 0. For each augmented state (s, θ) that is not not traversed by FIND-TRANSPOSED-SCCS and $\theta \neq 0$, TVI-DP updates its reachable probability and record the optimal action to that in the augmented state $(s, \theta - 1)$ (lines 73-80).

Instead of running Tarjan's algorithm to find SCCs in the augmented state space, one can optimize this process by running Tarjan's algorithm on the original state space, and map those SCCs to the augmented state space. Taking the original MDP in Figure 4.4 as an example, the left subfigure of Figure 4.6 shows the corresponding DAG of SCCs when we only consider transitions with zero costs, and the right subfigure of Figure 4.6 shows the DAG of SCCs of augmented states with a particular cost threshold. The figure illustrates that the DAG of SCCs of augmented states with a particular cost threshold and corresponding original states is exactly the same as the DAG of SCCs in the original state space when

we only consider the transitions with zero costs.

4.5 RS-MDP Complexity

The complexity of MDPs are proved in [Papadimitriou and Tsitsiklis, 1987]. By following similar technical steps, we show the complexity of RS-MDPs in this subsection.

Theorem 2 *Solving RS-MDPs optimally is P-hard in the original state space.*

PROOF: To show P-hardness, we shall reduce *Circuit Value Problems* (CVPs) to RS-MDPs. A CVP can be defined as a finite sequence of triples $C = ((o_i, p_i, q_i), i = 1, 2, \ldots, n)$, where o_i is one of the "operations" – TRUE, FALSE, AND, and OR, and p_i and q_i are non-negative integers smaller than i. If o_i is either FALSE or TRUE, then the triple is called an input, and $p_i = q_i = 0$. If o_i is either AND or OR, then the triple is called a gate and $p_i, q_i > 0$. The value of a triple is defined recursively as follows: (1) The value of an input (TRUE, 0, 0) is true, and that of (FALSE, 0, 0) is false; (2) The value of a gate (o_i, p_i, q_i) is the Boolean operation denoted by o_i applied to the values of the p_i-th and q_i-th triples. The value of CVP is the value of the last gate. A CVP needs to decide if its value is true.

A CVP can be reduced to an RS-MDP here. Given a circuit $C = ((o_i, p_i, q_i), i = 1, 2, \ldots, n)$, we construct an MDP $\langle \mathbf{S}, s_0, \mathbf{A}, \mathbf{T}, \mathbf{C}, \mathbf{G} \rangle$ as follows: $\mathbf{S}$ has one state s_i for each triple (o_i, p_i, q_i). The set of goal states $\mathbf{G}$ is the set of states corresponding to all input triples. If (o_i, p_i, q_i) is an AND gate, then there is only 1 available action a_0 for state s_i, with $T(s_i, a_0, s_{p_i}) = T(s_i, a_0, s_{q_i}) = 0.5$, which means that the next state can be either s_{p_i} or s_{q_i}. If (o_i, p_i, q_i) is an OR gate, then there are 2 available actions a_1, a_2 for state s_i and the transition function is $T(s_i, a_1, s_{p_i}) = 1$ and $T(s_i, a_2, s_{q_i}) = 1$, which means that we decide whether the

next state is going to be s_{p_i} or s_{q_i}. The initial state s_0 is s_n, corresponding to the last gate in CVP. If o_i is FALSE, then we set the cost function $C(s, a, s_i) = \theta_\perp$, where s and a could be any state and action with $T(s, a, s_i) > 0$, $\theta_\perp$ is a constant larger than n, and all other costs are 1. Finally, we set the initial cost threshold as a positive constant $n < \theta_0 < \theta_\perp$. Combining the MDP and θ_0, we get desirable RS-MDP.

We claim that the optimal reachable probability is 1 if and only if the value of CVP was TRUE. Let us first assume that the reachable probability is indeed 1. Then, by following the optimal RS-MDP policy, all potential execution trajectories will finally reach a goal state corresponding to a TRUE input. Otherwise, the goal state corresponding to a FALSE input would result in a cost θ' that is larger than θ_0. Following the potential execution trajectories starting from these goal states, it easy to deduce that the value of CVP is TRUE. If the value of CVP was TRUE, then it is easy to construct the optimal RS-MDP policy by picking actions corresponding to TRUE triples among s_{p_i}, s_{q_i} for each OR gate. Then, the potential execution trajectories would always reach goal states corresponding to TRUE input triples, and the reachable probability is 1. ∎

Notice that RS-MDPs are not in P in the original state space, because solving them requires specifying an action for each augmented state in the set $\widehat{S} : \mathbf{S} \times \mathbf{\Theta}$, which, in turn, could be exponential in the size of $\mathbf{S}$ if $|\mathbf{\Theta}| = 2^{|\mathbf{S}|}$.

4.6 RS-MDP Experimental Results

We ran experiments on three domains: (1) randomly generated MDPs, (2) the *Navigation* domain from the ICAPS 2011 *International Probabilistic Planning Competition* (IPPC), and (3) a taxi domain [Ziebart *et al.*, 2008; Varakantham *et*

al., 2012] generated from real-world data. While there are other IPPC domains, we chose the Navigation domain because it is the only one with dead-ends in the original MDP state space. Therefore, this domain highlights the feasibility of our approaches beyond SSP-MDPs, which do not have dead-ends.

In all our three domains, we generated two types of MDPs – ones without zero costs and ones with zero costs. For the former, we compared the following algorithms: FVI, VI on the augmented MDP (which we label as AUG-VI), DFS, and DP. For the latter, we replaced DFS and DP with their TVI-DFS and TVI-DP extensions that work with zero costs. As the performance of DFS, DP, TVI-DFS, and TVI-DP may differ based on the data structure that they use to store the reachable probabilities, we evaluated them for both data structures, where we use "F" or "T" in parentheses to indicate that they are using utility functions or latticed tables to represent the probabilities, respectively.

We report scalability in terms of the percentage of instances solved (%); average runtime in milliseconds of instances solved (t); and average reachable probability of all instances (P).[1] We impose a timeout of 10 minutes. All experiments were conducted on a 3.40 GHz machine with 16GB of RAM.

4.6.1 Randomly Generated MDPs

We ran two types of experiments here – one where we vary the number of states $|\mathbf{S}|$ from 2500 to 40000, and another where we vary the cost threshold θ_0 from 1.25 to 5 times of $\mathcal{C}_d^*$, where $\mathcal{C}_d^*$ is the accumulated cost of the shortest deterministic path from the start state s_0. Each randomly generated MDP has 2 actions per

[1]When an instance is not solved, some algorithms may have found suboptimal solutions for this instance. We include the reachable probabilities of these suboptimal solutions when computing the average.

state and 2 successors per action. We randomly selected a state as the start state and a different state as the goal state, and we chose the costs from the range [0, 1000]. Tables 4.1 and 4.2 show the results for the MDPs without zero costs and with zero costs, respectively. Results are averaged over 50 randomly generated instances.

We make the following observations for MDPs without zero costs:

- In general, either DFS or DP is faster than AUG-VI and FVI. The reason is that they are designed to solve RS-MDPs specifically and, thus, exploits the specific transition properties present in RS-MDPs. In contrast, AUG-VI and FVI do not exploit those properties.
- The runtime of DFS(T) mostly depends on two factors: (1) The initialization time of the latticed table, which increases with increasing state size but is independent of the cost threshold; and (2) The runtime of each of its operations in accessing and manipulating the reachable probabilities in that table is in constant time, independent of the state size and the cost threshold. On the other hand, the runtime of DFS(F) mostly depends on a single factor: Maintaining and updating its utility function, which depends on the complexity of the function (i.e., the number of "pieces" in piecewise constant functions that are needed to represent it). The complexity of such piecewise functions increases with increasing cost thresholds. The worst-case runtime complexity of the search and insert operations are $O(\log(|\mathbf{\Theta}|))$ and $O(|\mathbf{\Theta}|)$, respectively, where $\mathbf{\Theta}$ is the set of cost thresholds.

  In Table 4.1, when the cost threshold is kept at a constant ratio of $\theta_0 = 2.00 \cdot \mathcal{C}_d^*$, DFS(F) is faster than DFS(T) because the overhead of initializing the latticed table is larger than the savings DFS(T) gained from its runtime operations, which is in constant time. As such, the difference in runtime also

increases with the state size. In Table 4.1, when the state size is kept constant, DFS(F) is faster than DFS(T) when cost thresholds are small and vice versa when cost thresholds are large. The reason is that when the cost thresholds are small, the overhead is still larger than the savings in runtime operations. But when the cost thresholds are large, the reverse holds.

- For DP, unlike DFS, DP(T) is usually faster than DP(F) in Tables 4.1. The reason is the following: In both versions, DP has to update the reachable probabilities for the augmented states sequentially from those with cost threshold $\theta = 0$ up to those with the initial cost threshold $\theta = \theta_0$. As such, it needs to order the augmented states that need to be updated according to their cost threshold. When DP uses a latticed table, it can directly mark the cells in the table to indicate whether it needs to be updated or not. However, when DP uses utility functions, it needs to order them using a heap, which will be more expensive, especially as the size of the heap grows with the state size and initial cost threshold.

  In addition, for the same reason, DP(F) may use more memory than DP(T) as it will need to store this heap. As a result, when $\theta_0 = 5.00 \cdot \mathcal{C}_d^*$, DP(F) only solved 76% of instances. For about half of the instances that it failed to solve, they were due to timeouts and the rest were due to DP(F) running out of memory.

- Between the two algorithms, DFS has a slightly larger runtime when updating the reachable probability of a single augmented state. The reason is because it needs to check if its successor augmented states have already been updated while DP does not. This check is in $O(1)$ if it uses a latticed table and is in $O(\log(|\mathbf{\Theta}|))$ if it uses utility functions. However, DFS updates fewer augmented states as it only needs to update those states that are reachable from

the starting augmented state. In contrast, DP needs to update the reachable probabilities for *all* augmented states whose cost threshold $\theta \leq \theta_0$ is no larger than the initial cost threshold. Therefore, when the initial cost threshold is small, the latter effect dominates the former and DFS is faster.

When the initial cost threshold is large, after a certain intermediate cost threshold $\theta' < \theta_0$, almost all augmented states with cost threshold $\theta \leq \theta'$ are also reachable augmented states and are thus updated by both DFS and DP. As a result, the overall overhead of DFS is larger than its savings, and DFS is slower than DP.

- Finally, as expected, the reachable probability P of the augmented start state increases with increasing threshold θ_0. Additionally, since all the algorithms are optimal algorithms, they find solutions with the same reachable probability.

For MDPs with zero costs, most of the trends from above also apply, except for the following:

- TVI-DFS(T) is faster than TVI-DFS(F). The reason is the following: With zero costs, there may be loops in the transition graph (within an SCC), which will cause augmented states within the SCC to be repeatedly updated by VI across multiple iterations before convergence. As the operations on the latticed table, which are in $O(1)$, are faster than the operations with utility functions, which is in $O(\log(|\mathbf{\Theta}|))$, this difference is exaggerated here when there are zero costs. For this reason, the speedup of TVI-DFS(T) over TVI-DFS(F) also increases with increasing state size and cost threshold.
- TVI-DP(T) is faster than TVI-DFS(F) for the same reason as above.

$\|\mathbf{S}\|$	FVI			DFS(F)			DP(F)		
	%	t	P	%	t	P	%	t	P
2500	100	896	9.60e-2	100	7	9.60e-2	100	705	9.60e-2
5000	100	2199	7.17e-2	100	14	7.17e-2	100	1436	7.17e-2
10000	100	7441	5.13e-2	100	45	5.13e-2	100	7677	5.13e-2
20000	100	15243	4.03e-2	100	118	4.03e-2	100	18013	4.03e-2
40000	100	57448	2.68e-2	100	361	2.68e-2	100	53225	2.68e-2
$\|\mathbf{S}\|$	AUG-VI			DFS(T)			DP(T)		
	%	t	P	%	t	P	%	t	P
2500	100	2900	9.60e-2	100	508	9.60e-2	100	148	9.60e-2
5000	100	6846	7.17e-2	100	1270	7.17e-2	100	311	7.17e-2
10000	100	17199	5.13e-2	100	4186	5.13e-2	100	1078	5.13e-2
20000	100	39269	4.03e-2	100	9686	4.03e-2	100	2870	4.03e-2
40000	100	96839	2.68e-2	100	27768	2.68e-2	100	8147	2.68e-2

$\|\mathbf{S}\|$	FVI			DFS(F)			DP(F)		
	%	t	P	%	t	P	%	t	P
$1.25{\cdot}\mathcal{C}_d^*$	100	261	1.24e-2	100	7	1.24e-2	100	88	1.24e-2
$1.50{\cdot}\mathcal{C}_d^*$	100	1133	2.51e-2	100	14	2.51e-2	100	550	2.51e-2
$2.00{\cdot}\mathcal{C}_d^*$	100	14121	5.16e-2	100	52	5.16e-2	100	8782	5.16e-2
$3.00{\cdot}\mathcal{C}_d^*$	96	129913	1.07e-1	100	4861	1.07e-1	100	91823	1.07e-1
$5.00{\cdot}\mathcal{C}_d^*$	24	407102	1.89e-1	100	128576	2.02e-1	76	316178	1.96e-1
$\|\mathbf{S}\|$	AUG-VI			DFS(T)			DP(T)		
	%	t	P	%	t	P	%	t	P
$1.25{\cdot}\mathcal{C}_d^*$	100	10639	1.24e-2	100	195	1.24e-2	100	212	1.24e-2
$1.50{\cdot}\mathcal{C}_d^*$	100	13061	2.51e-2	100	808	2.51e-2	100	347	2.51e-2
$2.00{\cdot}\mathcal{C}_d^*$	100	17802	5.16e-2	100	4378	5.16e-2	100	1191	5.16e-2
$3.00{\cdot}\mathcal{C}_d^*$	100	27600	1.07e-1	100	15613	1.07e-1	100	6321	1.07e-1
$5.00{\cdot}\mathcal{C}_d^*$	100	46960	2.02e-1	100	39087	2.02e-1	100	19255	2.02e-1

Table 4.1: RS-MDP Results of Randomly Generated MDPs without Zero Costs

4.6.2 Navigation Domain

For the navigation domain, we use all 10 IPPC instances. However, we changed the costs to randomly vary from $[1, 1000]$ for RS-MDPs without zero costs and from $[0, 1000]$ for RS-MDPs with zero costs. This domain is specified by a gridworld, where each cell in the gridworld is a state, and a robot needs to navigate from its start state to its goal state, both of which are specified by the instance. It can move in the four cardinal directions and it will move to the neighboring state with

$\|\mathbf{S}\|$	FVI			TVI-DFS(F)			TVI-DP(F)		
	%	t	P	%	t	P	%	t	P
2500	84	10801	7.06e-3	100	508	7.06e-3	100	1367	7.06e-3
5000	86	17215	1.44e-3	100	1878	1.44e-3	100	4237	1.44e-3
10000	96	32623	2.05e-4	100	4013	2.05e-4	100	7847	2.05e-4
20000	88	51013	4.38e-4	100	12806	4.38e-4	100	24201	4.38e-4
40000	86	77977	1.95e-4	100	23980	1.95e-4	100	54071	1.95e-4
$\|\mathbf{S}\|$	AUG-VI			TVI-DFS(T)			TVI-DP(T)		
	%	t	P	%	t	P	%	t	P
2500	100	10934	7.06e-3	100	234	7.06e-3	100	369	7.06e-3
5000	98	23569	1.44e-3	100	866	1.44e-3	100	1020	1.44e-3
10000	100	65304	2.05e-4	100	1946	2.05e-4	100	2151	2.05e-4
20000	94	120126	4.38e-4	100	6424	4.38e-4	100	6778	4.38e-4
40000	76	123923	1.95e-4	100	12984	1.95e-4	100	14122	1.95e-4

$\|\mathbf{S}\|$	FVI			TVI-DFS(F)			TVI-DP(F)		
	%	t	P	%	t	P	%	t	P
$1.25{\cdot}\mathcal{C}_d^*$	90	8067	2.74e-6	100	1273	2.74e-6	100	2830	2.74e-6
$1.50{\cdot}\mathcal{C}_d^*$	90	14300	2.63e-5	100	2316	2.63e-5	100	4804	2.63e-5
$2.00{\cdot}\mathcal{C}_d^*$	90	30526	1.82e-3	100	5192	1.82e-3	100	9885	1.82e-3
$3.00{\cdot}\mathcal{C}_d^*$	88	61007	1.62e-2	100	12741	1.62e-2	100	22757	1.62e-2
$5.00{\cdot}\mathcal{C}_d^*$	86	136390	2.69e-2	98	26235	2.70e-2	98	44333	2.69e-2
$\|\mathbf{S}\|$	AUG-VI			TVI-DFS(T)			TVI-DP(T)		
	%	t	P	%	t	P	%	t	P
$1.25{\cdot}\mathcal{C}_d^*$	100	32032	2.74e-6	100	967	2.74e-6	100	1157	2.74e-6
$1.50{\cdot}\mathcal{C}_d^*$	100	40562	2.63e-5	100	1396	2.63e-5	100	1598	2.63e-5
$2.00{\cdot}\mathcal{C}_d^*$	100	59143	1.82e-3	100	2352	1.82e-3	100	2507	1.82e-3
$3.00{\cdot}\mathcal{C}_d^*$	98	83579	1.62e-2	100	4366	1.62e-2	100	4332	1.62e-2
$5.00{\cdot}\mathcal{C}_d^*$	96	145365	2.71e-2	100	8756	2.71e-2	100	8156	2.71e-2

Table 4.2: RS-MDP Results of Randomly Generated MDPs with Zero Costs

100% probability. However, when it enters a state, it has a probability to fail. These probabilities can differ between states and are specified by the instance; they follow some logical structure and are specified by human IPPC designers.

We vary the cost threshold θ_0 from 1.25 to 5 times of $\mathcal{C}_d^*$, and Tables 4.3 and 4.4 show the results for the MDPs without zero costs and with zero costs, respectively. We make the following observations for the case without zero costs, where we focus only on the trends that differ between those in the randomly generated MDPs and

$\|\mathbf{S}\|$	FVI			DFS(F)			DP(F)		
	%	t	P	%	t	P	%	t	P
$1.25{\cdot}\mathcal{C}_d^*$	100	40	2.00e-4	100	42	2.00e-4	100	19	2.00e-4
$1.50{\cdot}\mathcal{C}_d^*$	100	50	1.71e-2	100	66	1.71e-2	100	26	1.71e-2
$2.00{\cdot}\mathcal{C}_d^*$	100	72	1.90e-1	100	133	1.90e-1	100	37	1.90e-1
$3.00{\cdot}\mathcal{C}_d^*$	100	67	4.15e-1	100	226	4.15e-1	100	43	4.15e-1
$5.00{\cdot}\mathcal{C}_d^*$	100	69	5.12e-1	100	388	5.12e-1	100	44	5.12e-1
$\|\mathbf{S}\|$	AUG-VI			DFS(T)			DP(T)		
	%	t	P	%	t	P	%	t	P
$1.25{\cdot}\mathcal{C}_d^*$	100	661	2.00e-4	100	639	2.00e-4	100	21	2.00e-4
$1.50{\cdot}\mathcal{C}_d^*$	100	801	1.71e-2	100	948	1.71e-2	100	26	1.71e-2
$2.00{\cdot}\mathcal{C}_d^*$	100	1078	1.90e-1	100	1495	1.90e-1	100	47	1.90e-1
$3.00{\cdot}\mathcal{C}_d^*$	100	1604	4.15e-1	100	2632	4.15e-1	100	53	4.15e-1
$5.00{\cdot}\mathcal{C}_d^*$	100	2714	5.12e-1	100	4939	5.12e-1	100	81	5.12e-1

Table 4.3: RS-MDP Results of Navigation Domain without Zero Costs

those in this Navigation domain:

- Unlike in randomly generated MDPs, here, DP is fastest, followed by FVI, DFS, and AUG-VI. The reason why DP and FVI are faster than DFS and AUG-VI in this domain is the following: The utility functions in this domain are relatively simple and can be represented by only few piecewise constant functions. As a result, FVI is able to update these functions very efficiently. Additionally, because these utility functions are simple, they also often do not vary across two augmented states (s, θ) and $(s, \theta - 1)$ with the same state s but different adjacent thresholds. Therefore, DP can simply copy the utility function of $(s, \theta - 1)$ for (s, θ) (lines 43-44 of Algorithm 2). DFS and AUG-VI do not exploit this characteristic and are, thus, slower.
- For the same reason as above, both DFS and DP are faster when they use utility functions to represent the reachable probabilities than when they use latticed tables.

For the case with zero costs, most of the trends from above also apply, except for the following:

$\|\mathbf{S}\|$	FVI			TVI-DFS(F)			TVI-DP(F)		
	%	t	P	%	t	P	%	t	P
$1.25{\cdot}\mathcal{C}_d^*$	100	50	2.80e-5	100	118	2.80e-5	100	123	2.80e-5
$1.50{\cdot}\mathcal{C}_d^*$	100	61	1.36e-4	100	194	1.36e-4	100	166	1.36e-4
$2.00{\cdot}\mathcal{C}_d^*$	100	84	1.78e-2	100	348	1.78e-2	100	242	1.78e-2
$3.00{\cdot}\mathcal{C}_d^*$	100	108	2.47e-1	100	641	2.47e-1	100	342	2.47e-1
$5.00{\cdot}\mathcal{C}_d^*$	100	115	3.67e-1	100	1150	3.67e-1	100	384	3.67e-1
$\|\mathbf{S}\|$	AUG-VI			TVI-DFS(T)			TVI-DP(T)		
	%	t	P	%	t	P	%	t	P
$1.25{\cdot}\mathcal{C}_d^*$	100	4469	2.80e-5	100	237	2.80e-5	100	222	2.80e-5
$1.50{\cdot}\mathcal{C}_d^*$	100	5551	1.36e-4	100	344	1.36e-4	100	268	1.36e-4
$2.00{\cdot}\mathcal{C}_d^*$	100	7440	1.78e-2	100	558	1.78e-2	100	360	1.78e-2
$3.00{\cdot}\mathcal{C}_d^*$	100	11255	2.47e-1	100	990	2.47e-1	100	536	2.47e-1
$5.00{\cdot}\mathcal{C}_d^*$	100	18730	3.67e-1	100	1848	3.67e-1	100	884	3.67e-1

Table 4.4: RS-MDP Results of Navigation Domain with Zero Costs

- FVI is faster than TVI-DP. The reason is because there is a large number of SCCs with only a few augmented states in them, and these instances are very structurally similar to those without zero costs. As a result, the runtime of FVI in these two cases are similar. However, TVI-DP is slower than DP because it has the initial overhead of first running Tarjan's algorithm to construct the SCCs. Further, VI within TVI-DP will need to run for several iterations in each SCC, thereby performing updates multiple times per augmented state. Even in the special case where an SCC has only a single augmented state, VI will need to run at least two iterations, once to update the reachable probability of that state and another to check for convergence. As a result, FVI is now faster than TVI-DP.

  Note that this trend differs from the one for randomly generated MDPs with zero costs, where TVI-DP is faster than FVI. The reason is because the runtime of FVI is highly dependent on the complexity of the utility functions. Therefore, in summary, FVI should be preferred when utility functions are simple and TVI-DP should be preferred when utility functions are complex.

$\|\mathbf{S}\|$	FVI			DFS(F)			DP(F)		
	%	t	P	%	t	P	%	t	P
$1.25{\cdot}\mathcal{C}_d^*$	100	416	2.16e-9	100	4	2.16e-9	100	637	2.16e-9
$1.50{\cdot}\mathcal{C}_d^*$	100	670	1.28e-5	100	19	1.28e-5	100	1329	1.28e-5
$2.00{\cdot}\mathcal{C}_d^*$	100	1399	2.29e-2	100	138	2.29e-2	100	3254	2.29e-2
$3.00{\cdot}\mathcal{C}_d^*$	100	2735	1.74e-1	100	442	1.74e-1	100	6723	1.74e-1
$5.00{\cdot}\mathcal{C}_d^*$	100	3965	6.29e-1	100	280	6.29e-1	100	10162	6.29e-1
$\|\mathbf{S}\|$	AUG-VI			DFS(T)			DP(T)		
	%	t	P	%	t	P	%	t	P
$1.25{\cdot}\mathcal{C}_d^*$	100	926	2.16e-9	100	31	2.16e-9	100	126	2.16e-9
$1.50{\cdot}\mathcal{C}_d^*$	100	1140	1.28e-5	100	84	1.28e-5	100	202	1.28e-5
$2.00{\cdot}\mathcal{C}_d^*$	100	1579	2.29e-2	100	218	2.29e-2	100	353	2.29e-2
$3.00{\cdot}\mathcal{C}_d^*$	100	2476	1.74e-1	100	473	1.74e-1	100	575	1.74e-1
$5.00{\cdot}\mathcal{C}_d^*$	100	4192	6.29e-1	100	704	6.29e-1	100	752	6.29e-1

Table 4.5: RS-MDP Results of Taxi Domain without Zero Costs

4.6.3 Taxi Domain

For the taxi domain, states are composed of the tuple $\langle$zone z, time interval $t\rangle$, where there are 100 zones and each time interval is 10 minutes long. Each taxi has two actions: (a_1) move to a zone and (a_2) look for passengers in its zone. Taxis executing a_1 will move to their desired zone with probability 1 and cost c. Taxis executing a_2 have probability $p_{z,t}$ of successfully picking up a passenger and they can accurately observe $p_{z,t}$. If it fails to pick up a passenger, it ends up in the same zone with cost c. The probability $p_{z,t}$; the transition function, which determines which zone a hired taxi moves to; and the cost function, which determines the cost of the hired taxi, is generated with real-world data. As we have data for 24 hours, we partitioned it into 8 instances, where each instance uses data for 3 hours. In each of these instances, we randomly choose a start state and goal states are those states with $t = 18$, which means that the optimization lookahead is for the whole 3 hours. Results are averaged over all 8 instances.

Like in the Navigation domain, we vary the cost threshold θ_0 from 1.25 to 5 times of $\mathcal{C}_d^*$, and Tables 4.5 and 4.6 show the results for the MDPs without

$\|\mathbf{S}\|$	FVI			TVI-DFS(F)			TVI-DP(F)		
	%	t	P	%	t	P	%	t	P
$1.25{\cdot}\mathcal{C}_d^*$	100	554	8.84e-9	100	8	8.84e-9	100	1202	8.84e-9
$1.50{\cdot}\mathcal{C}_d^*$	100	851	2.90e-6	100	39	2.90e-6	100	2135	2.90e-6
$2.00{\cdot}\mathcal{C}_d^*$	100	1615	1.16e-2	100	191	1.16e-2	100	4351	1.16e-2
$3.00{\cdot}\mathcal{C}_d^*$	100	2883	3.21e-1	100	336	3.21e-1	100	7978	3.21e-1
$5.00{\cdot}\mathcal{C}_d^*$	100	4012	6.90e-1	100	242	6.90e-1	100	11489	6.90e-1
$\|\mathbf{S}\|$	AUG-VI			TVI-DFS(T)			TVI-DP(T)		
	%	t	P	%	t	P	%	t	P
$1.25{\cdot}\mathcal{C}_d^*$	100	999	8.84e-9	100	41	8.84e-9	100	1192	8.84e-9
$1.50{\cdot}\mathcal{C}_d^*$	100	1219	2.90e-6	100	94	2.90e-6	100	1486	2.90e-6
$2.00{\cdot}\mathcal{C}_d^*$	100	1688	1.16e-2	100	249	1.16e-2	100	2089	1.16e-2
$3.00{\cdot}\mathcal{C}_d^*$	100	2635	3.21e-1	100	495	3.21e-1	100	3157	3.21e-1
$5.00{\cdot}\mathcal{C}_d^*$	100	4563	6.90e-1	100	742	6.90e-1	100	4930	6.90e-1

Table 4.6: RS-MDP Results of Taxi Domain with Zero Costs

zero costs and with zero costs, respectively. As the actual cost function from the real-world data includes zero costs, we incremented all the costs by 1 for the case without zero costs. We make the following observations for the case without zero costs, where we focus only on the trends that differ between those in the randomly generated MDPs and those in this taxi domain

- Like randomly generated MDPs, DFS is better than DP, FVI, and AUG-VI when the cost thresholds are small. However, when cost thresholds are large, unlike randomly generated MDPs, where DP is better, DFS is still better here. This may be due to the fact that our experiments did not increase the cost threshold to a sufficiently larger value to observe the change in dominance between DFS and DP.
- Interestingly, the runtime of DFS(F) decreased when the cost threshold increased from $\theta_0 = 3.00 \cdot \mathcal{C}_d^*$ to $5.00 \cdot \mathcal{C}_d^*$. The reason is that when the cost threshold is sufficiently large, the reachable probability of a large number of states is 1.0, indicating that it has 100% probability of reaching a goal state. As a result, the utility function representing this reachable probability is made

up of only a *single* "piece", and the operations of DFS(F) is significantly faster.

While DP(F) and FVI also use utility functions, they are not able to exploit this property as they propagate information from the augmented goal states, which are those states with $\theta = 0$, up to the augmented start state with $\theta = \theta_0$. For both algorithms, some of the utility functions will eventually merge into one with a single "piece" at some iteration, but they still have to consider the more complicated functions in the previous iterations.

For the case with zero costs, the corresponding trends for randomly generated MDPs also apply here, except for the following:

- TVI-DFS is faster than TVI-DP. The reason is that TVI-DFS only explores a significantly smaller number of reachable augmented states than TVI-DP in this domain compared to in randomly generated MDPs.
- TVI-DFS is faster when it uses utility functions to represent reachable probabilities results than when it uses latticed tables. The reason is the same as above: As it only explores a small number of reachable augmented states, the overhead of initializing the latticed table is larger than the savings in the operations when it uses a latticed table.

Chapter 5

Risk-Sensitive POMDPs (RS-POMDPs)

Similar to *Risk-Sensitive MDPs* (RS-MDPs), the *Risk-Sensitive POMDP* (RS-POMDP) model can be obtained by combining *Partially Observable Markov Decision Processes* (POMDPs) and the *Risk-Sensitive criterion* (RS-criterion). This section begins by introducing RS-POMDPs and explaining related cost observation issues in Section 5.1. Then, *Functional Value Iteration* (FVI) [Marecki and Varakantham, 2010] is adopted to solve RS-POMDPs in Section 5.2.1. Section 5.2.2 shows how to solve RS-POMDPs by generating and solving their corresponding augmented POMDPs. We show how to adapt *Depth First Search* (DFS) and *Dynamic Programming* (DP) to solve RS-POMDPs in Sections 5.2.3 and 5.2.4, respectively. Section 5.3 give the complexity of RS-POMDPs and corresponding proof. Finally, we show the experimental results of different RS-POMDP algorithms in Section 5.4

5.1 RS-POMDP Model

Formally, like RS-MDPs, an RS-POMDP is defined by the tuple $\langle \mathcal{P}, \mathbf{\Theta}, \theta_0 \rangle$, where:

- $\mathcal{P}$ is a POMDP
- $\mathbf{\Theta}$ is the set of all possible cost thresholds
- $\theta_0 \in \mathbf{\Theta}$ is a predefined initial cost threshold.

For POMDPs, the *execution trajectory* based on states can not be observed, but the *execution history* based on observations can be observed. Let us assume a general policy π, which may not be stationary, is executed, and the system produce an execution history $h : \langle a^0, o^0, a^1, o^1, \ldots \rangle$, Then, a *probability* can be defined, according to π and h, for a execution trajectory $\tau : \langle s^0, a^0, s^1, a^1, s^2, \ldots \rangle$, which is $P^\pi(\tau) = b_0(s^0) \cdot \prod_{i=0}^{\infty} P(s^i, a^i, s^{i+1}, o^i)$, where a^i is the action chosen by π to be executed at the time step i and $P(s^i, a^i, s^{i+1}, o^i) = \frac{T(s^i,a^i,s^{i+1})O(s^i,a^i,s^{i+1},o^i)}{\sum_{s \in \mathbf{S}} T(s^i,a^i,s)O(s^i,a^i,s,o^i)}$, which is the probability that state s^{i+1} is arrived where action a^i is applied in state s^i and observation o^i is received. Thus, we can define the reachable probability with respect to the initial belief state b_0 and initial cost threshold θ_0 as $P_{b_0,\theta_0} = \sum_{\tau \in \mathcal{T}} P^\pi(\tau)$, where $\mathcal{T} = \{\tau \mid C(\tau) \leq \theta_0\}$ and $C(\tau)$ is the accumulated cost of execution trajectory τ. Similar to RS-MDPs, the objective of RS-POMDPs is to find a policy that maximizes P_{b_0,θ_0}.

For original POMDPs, the optimal actions only depend on the belief states, i.e., a stationary policy with respect to belief states could be optimal for original POMDPs. But for RS-POMDPs, the situation is different. From the discussion about RS-criterion for RS-MDPs, we know that the optimal decision is also based on the cost threshold. If two execution trajectory prefixes end at a same state s, their corresponding belief states would both contain s, but their corresponding cost thresholds could be different. So even if the belief states is the same for

an RS-POMDP, the optimal decision choice would be different. Instead of the original belief states, the optimal action choices of RS-POMDPs depend on the augmented belief state, which is probability distributions $\hat{b} : \mathbf{S} \times \mathbf{\Theta} \rightarrow [0, 1]$ on the pair of state and cost threshold. Similar to RS-MDPs, we call pairs (s, θ) augmented states and the augmented state space is $\widehat{\mathbf{S}} : \mathbf{S} \times \mathbf{\Theta}$. Thus, $\hat{b}(s, \theta)$ denote the probability that the system is in state s and with cost threshold θ, and namely $\hat{b}(s, \theta)$ is the probability of the augmented state (s, θ). Let us denote all possible augmented belief state as $\widehat{\mathbf{B}}$. Then optimal policies of RS-POMDP have the form $\widehat{\mathbf{B}} \rightarrow \mathbf{A}$, and the reachable probability of augmented belief states can be defined as $P : \widehat{\mathbf{B}} \rightarrow [0, 1]$. The above policy form is stationary with respect to augmented belief state. Corresponding to the initial belief state b_0 and initial cost threshold θ_0, the initial augmented state can be defined as $\hat{b}_0(s, \theta) = b_0(s)$ if $\theta = \theta_0$, $\hat{b}_0(s, \theta) = 0$ otherwise. Thus, the reachable probability P_{b_0,θ_0} can be denoted as $P(\hat{b}_0)$ as well, i.e., $P_{b_0,\theta_0} = P(\hat{b}_0)$.

5.1.1 Cost Observation

We know that the optimal action choice of RS-POMDPs depend on the cost threshold, but deciding the cost threshold is highly depend on whether the cost can be observed during execution. In most existing POMDP literature, it is often not explicitly stated if agents can or cannot observe the actual cost incurred during execution [Kaelbling *et al.*, 1998; Geffner and Bonet, 2013; Marecki and Varakantham, 2010]. In some real-world applications, the actual costs can indeed be observed. For example, for a robot, costs can correspond to the amount of battery power used, and the robot can observe the drop in its internal battery power. Similarly, if costs correspond to the duration of time taken by an action, time can also be observed. Therefore, in this subsection, we distinguish between

the two cases of whether costs can be observed or not, and we also describe the deficiency of popular POMDP models in existing literature.

In existing POMDP literature, rather than the observation function form we defined in this dissertation, the observation function normally have the simpler form $\mathbf{O}_{\neg c} : \mathbf{A} \times \mathbf{S} \times \mathbf{\Omega}_{\neg c} \rightarrow [0, 1]$, where $\mathbf{\Omega}_{\neg c}$ is the set of corresponding observation. This simpler form assumes that the probability of observing an observation depends on only the action executed and the state reached. When the cost can be observed during execution, the cost observation can not be included in the observations based on this simpler form. For example, let us assume that the observation $o_{\neg c}$ contain information of cost observations and the observed cost corresponding to $o_{\neg c}$ is $c_{o_{\neg c}}$. Then, $O_{\neg c}(a, s, o_{\neg c})$ is positive if the cost of corresponding transition is $c_{o_{\neg c}}$, i.e., $C(s, a, s') = c_{o_{\neg c}}$, otherwise it should equal to 0. But $O_{\neg c}(a, s, o_{\neg c})$ is just one fixed number and it can not reflect the above situations based on different conditions.

When we assume that the cost can be observed, the POMDP model is consistent if and only if all the parameters of the cost function are also parameters of the observation function. Thus, based on the example above, one can make the POMDP model consistent if we include the predecessor state as a parameter of the observation function (i.e., $\mathbf{O} : \mathbf{S} \times \mathbf{A} \times \mathbf{S} \times \mathbf{\Omega} \rightarrow [0, 1]$) because the cost function depends on the predecessor state, action, and successor state (i.e., $\mathbf{C} : \mathbf{S} \times \mathbf{A} \times \mathbf{S} \rightarrow [0, +\infty)$).

Whether the cost can be observed is not only important for RS-POMDPs, and it is also important for original POMDPs. If the cost can be observed, more accurate belief states can be obtained by considering the information of cost observation, and the simpler form of the observation function can not model it. Furthermore, in cases where actual costs can be observed, most POMDP algorithms

do not explicitly use them to update the belief state. Thus, these algorithm may get a less accurate result. In the special case where the cost function depends only on the action (i.e., $\mathbf{C} : \mathbf{A} \to [0, +\infty)$), the actual cost can always be accurately inferred as it does not depend on the predecessor or successor states, and this information can thus be used to refine the belief state. Therefore, in this special case, it does not matter if one assumes whether costs can be observed or not.

In this dissertation, we assume that the observation function of POMDPs have a more general form $\mathbf{O} : \mathbf{S} \times \mathbf{A} \times \mathbf{S} \times \mathbf{\Omega} \to [0, 1]$, which assumes the probability of observing a observation depend on the whole transition (s, a, s'). If the cost can be observed, given a POMDP with the simpler form of observation functions, a corresponding POMDP with the general observation function can be generated by augmenting the observation. Let us denote the simpler form observation function as $O_{\neg c}$ and its corresponding observation as $o_{\neg c}$, which do not contain information of cost observations. Then, the observation of POMDPs with a general observation form would be a pair $(c, o_{\neg c})$, in which c is the observed cost. Then, the general observation function in the new POMDP would be as below:

$$O(s, a, s', (c, o_{\neg c})) = \begin{cases} O_{\neg c}(a, s', o_{\neg c}) & \text{if } c = C(s, a, s') \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

If we consider the above equation as a constraint, the observation function must equal to 0 if the observed cost is not the same as the cost of the corresponding transition, i.e., $c \neq C(s, a, s')$. Any POMDP models that can not satisfy this constraint are not consistent with the assumption that the cost can be observed, even if they use the general form for observation functions.

Actually, when the cost can be observed, RS-POMDPs can be discussed with the simpler observation function, and they are relatively compact [Hou *et al.*, 2016]. Nonetheless, the discussion in the rest of the subsections would be based

on the more general form of observation functions because we want to give a uniform description for both situations whether cost can or cannot be observed.

Additionally, we assume that in both cases, the agent can accurately detect if the actual accumulated cost of its execution trajectory is greater than its initial cost threshold, i.e., when its resource is depleted or its deadline has passed, and it will then stop executing actions. This is an important assumption for RS-POMDPs. Otherwise, the agent would keep executing actions even if goal states can not be reached with accumulated cost less than initial cost thresholds.

5.2 RS-POMDP Algorithms

Similar to RS-MDPs, there are the same two general classes of algorithms to solve RS-POMDPs: *Functional Value Iteration* (FVI) [Marecki and Varakantham, 2010] and classical methods like depth-first search and dynamic programming. However, in addition to these algorithms, we also introduce their point-based variants that are more scalable.

5.2.1 Functional Value Iteration

Functional Value Iteration (FVI) has been proposed for POMDPs with *Piecewise Linear* (PWL) utility functions [Marecki and Varakantham, 2010], but the existing version is designed for *Finite-Horizon* POMDPs and the simpler observation function. In this subsection, we describe how to generalize FVI to solve goal-directed RS-POMDPs with or without cost observations. Additionally, we describe some possible optimizations of FVI for this formalization.

For POMDPs, recall that when the objective is to minimize the expected cost. Cost value functions are represented as a set Γ of $|\mathbf{S}|$-dimensional real vectors. For

POMDPs with utility functions, FVI represents the utility value function as a set Γ of $|\mathbf{S}|$-dimensional vectors, and each element of these vectors is a function that map states to value functions with form $\mathbf{W} \to \mathbb{R}$. In other words, each vector in Γ is a mapping $\mathbf{S} \to (\mathbf{W} \to \mathbb{R})$. Corresponding to Equation 2.10, which describes the update of the vector set, vectors in the new set after the update in iteration k is:

$$\alpha_{a,v}(s)(w) = \sum_{s',o} T(s,a,s')O(s,a,s',o)v(o)(s')(w - C(s,a,s')) \tag{5.2}$$

which also operates on value functions.

FVI Update: Similar to RS-MDPs, the value functions for RS-POMDPs are *Piecewise Constant* (PWC) functions that describe the reachable probability as functions of cost thresholds. In other words, each vector in Γ is a mapping $\mathbf{S} \to (\mathbf{\Theta} \to [0,1])$, that is, a particular state $s \in \mathbf{S}$ maps to a PWC value function, and a particular cost threshold $\theta \in \mathbf{\Theta}$ maps to a reachable probability in that value function. Then, the reachable probability of an augmented belief state $\hat{b}$ is:

$$P(\hat{b}) = \max_{\alpha \in \Gamma} \sum_{s,\theta} \hat{b}(s,\theta)\alpha(s)(\theta) \tag{5.3}$$

For RS-POMDPs, since the intial cost threshold θ_0 is given, value functions need to only record values over the interval $[0, \theta_0]$. Each value function can be represented as an ordered list of pairs (θ^i, p^i) for $i = 0, 1, 2, \ldots, n$, where $\theta^0 = 0$ and $\theta^0 < \theta^1 < \ldots < \theta^n \leq \theta_0$ and a PWC function can be formally defined as $U(\theta) = \alpha(s)(\theta) = p^i$ while $\theta \in [\theta^i, \theta^{i+1})$ and $U(\theta_0) = \alpha(s)(\theta_0) = p^n$.

Similar to exact algorithms of original POMDPs, FVI iteratively updates the vector set Γ until convergence. The full set of possible vectors after the update in

iteration $k+1$ is still:

$$\Gamma_{k+1} = \{\alpha_{a,v} \mid a \in \mathbf{A}, v \in \mathcal{V}_k\} \tag{5.4}$$

where $\mathcal{V}_k$ is the set of functions v, and the definition of $v : \mathbf{\Omega} \to \Gamma_k$ is similar to the one defined for original POMDPs. Also, the update of the vectors $\alpha_{a,v}$ are applied on value functions and Equation 5.2 can be rewritten here:

$$\alpha_{a,v}(s)(\theta) = \sum_{s',o} T(s,a,s')O(s,a,s',o)v(o)(s')(\theta - C(s,a,s')) \tag{5.5}$$

When costs can be observed, the observation function $O(s,a,s',o)$ would be 0 if the observed cost is not equal to the cost function $C(s,a,s')$. Thus, when the above equation is calculated, transitions can be ignored if the observed cost is different from the cost of the transition.

For FVI's update process, we need to initialize the vector set Γ_0 to represent the initial utility value function. Γ_0 would contain only one vector, in which the value functions equal to 1 if corresponding states are goal states, and equal to 0 otherwise, i.e., $\alpha(s_g)(\theta) = 1$ if $s_g \in \mathbf{G}$ and $\alpha(s_{\neg g})(\theta) = 0$ if $s_{\neg g} \notin \mathbf{G}$. Thus, when the update produce new vectors, all value functions associated with goal states would always be $\alpha(s_g)(\theta) = 1$ because goal states are absorbing and cost-free.

Notice that if the value function of dead-ends is initialized with any other value rather than 0, FVI may get incorrect reachable probabilities. Since RS-POMDPs are based on GD-POMDPs, FVI can be optimized by ignoring dead-ends in original state space. If we divide the transition graph into a *Directed Acyclic Graph* (DAG) of *Strongly Connected Components* (SCCs), dead-ends are states in SCCs whose downstream SCCs do not contain any goals. Thus, we can traverse the original state space and indicate all dead-ends. When an FVI update is performed, dead-ends are ignored and their value functions can be set

as $\alpha(s)(\theta) = 0$.

If states in the original state space that are neither dead-ends or goal state, form more than one SCC, FVI can be optimized by adopting the ideas of TVI. Like TVI, by choosing the SCCs one by one in reverse topological sort order, FVI perform update by only considering value functions of all states in one SCC until they converge, and it no longer needs to update value functions of states in SCCs that are already converged when upstream SCCs are updated.

Dominated Vectors Pruning: We now describe how to prune dominated vectors from Γ to scale up FVI. Note that vector α_i is not dominated by other vectors if the following holds for all vectors $\alpha_j \in \Gamma$:

$$\exists \hat{b} : \sum_{s,\theta} \hat{b}(s,\theta) \left[\alpha_i(s)(\theta) - \alpha_j(s)(\theta)\right] \geq 0 \tag{5.6}$$

For one PWC value function, we observe that the reachable probabilities are same for all cost thresholds in each "piece". If we take the union of all cost thresholds in the ordered list of pairs over all value functions $\alpha_i(s)$ in all vectors $\alpha_i \in \Gamma$, an ordered threshold list $\langle \theta^0_{all}, \theta^1_{all}, \ldots, \theta^n_{all} \rangle$ can be obtained, and segments of cost threshold $[\theta^0_{all}, \theta^1_{all}), [\theta^1_{all}, \theta^2_{all}), \ldots, [\theta^{n-1}_{all}, \theta^n_{all})$ and $[\theta^n_{all}, \theta^{n+1}_{all}]$ are formed, where $\theta^{n+1}_{all} > \theta_0$ is a constant. In order to compute $\alpha_i(s)(\theta) - \alpha_j(s)(\theta)$ efficiently for our PWC value functions, observe that the reachable probability for all cost thresholds $\theta \in [\theta^k_{all}, \theta^{k+1}_{all})$ are identical for all value function of all states in all vectors. Therefore, instead of considering all cost thresholds θ in Equation 5.6, one can divide the utility functions into segments of cost thresholds $[\theta^0_{all}, \theta^1_{all}), [\theta^1_{all}, \theta^2_{all}), \ldots, [\theta^{n-1}_{all}, \theta^n_{all})$ and $[\theta^n_{all}, \theta^{n+1}_{all})$, where, for each value function $\alpha_i(s)$, the reachable probabilities $\alpha_i(s)(\theta) = \alpha_i(s)(\theta')$ are identical for all cost thresholds $\theta, \theta' \in [\theta^k_{all}, \theta^{k+1}_{all})$ within a segment.

In the case where actual costs cannot be observed, since the computation of

the difference $\alpha_i(s)(\theta) - \alpha_j(s)(\theta)$ is for the same state s, one can optimize the process above by computing ordered threshold list $\langle \theta_s^0, \theta_s^1, \ldots, \theta_s^n \rangle$ and segments $[\theta_s^0, \theta_s^1), [\theta_s^1, \theta_s^2), \ldots, [\theta_s^{n-1}, \theta_s^n)$ and $[\theta_s^n, \theta_s^{n+1})$ for each state s in order to minimize the number of segments. Similar to above, $\theta_s^{n+1} > \theta_0$ is also a constant. Then, one can use the following condition to check for dominance:

$$\exists \hat{b} : \sum_{s,k} \sum_{\theta \in [\theta_s^k, \theta_s^{k+1})} \hat{b}(s, \theta) \left[\alpha_i(s)(\theta_s^k) - \alpha_j(s)(\theta_s^k) \right] \geq 0 \tag{5.7}$$

where θ_s^k is the start of the k-th cost threshold segment for state s. This dominance check can be implemented with a single linear program.

In the case where actual costs can be observed, recall that for a particular belief state $\hat{b}$, all pairs (s, θ) with non-zero probability $\hat{b}(s, \theta) > 0$ have exactly the same cost threshold θ. Therefore, one needs to only check the following condition for that particular cost threshold θ:

$$\exists \hat{b} : \sum_{s} \hat{b}(s, \theta) \left[\alpha_i(s)(\theta_{all}^k) - \alpha_j(s)(\theta_{all}^k) \right] \geq 0 \tag{5.8}$$

where $\theta \in [\theta_{all}^k, \theta_{all}^{k+1})$ and θ_{all}^k is the start of the k-th cost threshold segment in the union of all cost threshold segments over all states and all vectors. This dominance check can be implemented with n, which is the number of cost threshold segments, linear programs, where $\theta = \theta_{all}^k$ in the k-th linear program.

5.2.2 Representing RS-POMDPs as Augmented POMDPs

Like RS-MDPs, solving an RS-POMDP is equivalent to solving an augmented POMDP, which can be constructed by considering pairs of state and cost threshold as augmented states $\hat{s} = (s, \theta)$. Based on augmented states (s, θ), an augmented

POMDP $\langle \widehat{\mathbf{S}}, \mathbf{A}, \widehat{\mathbf{T}}, \mathbf{R}, \widehat{\mathbf{G}}, \mathbf{\Omega}, \widehat{\mathbf{O}}, \hat{b}_0 \rangle$ can be generated. In the augmented POMDP, $\widehat{\mathbf{S}}$, $\widehat{\mathbf{T}}$, $\mathbf{R}$ and $\widehat{\mathbf{G}}$ are the same as the corresponding components in augmented MDPs for RS-MDPs. The observation function $\widehat{\mathbf{O}} : \widehat{\mathbf{S}} \times \mathbf{A} \times \widehat{\mathbf{S}} \times \mathbf{\Omega} \to [0, 1]$ is $\widehat{O}((s, \theta), a, (s', \theta'), o) = O(s, a, s', o)$ if $\theta' = \theta - C(s, a, s')$, otherwise it equal to 0. Like augmented MDPs for RS-MDPs, augmented POMDPs can also merge all states with negative cost threshold to one state $\hat{s}_\perp$. Unlike regular POMDPs, the goal here would be to find a policy that maximizes the expected reward (equivalent to maximizing the reachable probability in RS-POMDPs).

5.2.3 Depth First Search (DFS)

Similar to the *Depth First Search* (DFS) algorithm for RS-MDPs with positive costs, it can also be adapted to solve RS-POMDPs with positive costs. Instead of *augmented states* that are reachable from initial augmented state (s_0, θ_0), DFS for RS-POMDPs traverses *augmented belief states* that are reachable from initial augmented belief states $\hat{b}_0$. For an augmented belief state $\hat{b}$ and any augmented state (s, θ) that $\hat{b}(s, \theta) > 0$, its successors have smaller cost thresholds since all costs are positive. Along the transition, the augmented belief state would finally have a negative cost threshold or it reached goal states. We know any augmented states with negative cost thresholds would have reachable probability 0, so they can be combined and ignored. Thus, all augmented belief states that are reachable from initial augmented state $\hat{b}_0$ would also form a graph without cycles. Since augmented belief states do not form cycles, DFS traverses all augmented belief states and updates their reachable probabilities based on the reverse topological sort order.

In RS-POMDPs, the update is on augmented belief states rather than the original belief states in original POMDPs. Since we assume that goal states and

negative cost thresholds are observable, the agent would certainly know if the cost threshold become negative or goal states are reached. Thus, the belief update is only necessary for receiving an observation o that indicate neither goal states or negative cost threshold. Let us use $\hat{b}_a^o$ to denote successor augmented belief state after performing action a in belief state b and observing o, which indicate neither goal states or negative cost threshold. $\hat{b}_a^o$ can be calculated as below:

$$\hat{b}_a^o(s,\theta) = \frac{1}{Z}\sum_{s',\theta'} \hat{b}(s',\theta')T(s',a,s)O(s',a,s,o) \tag{5.9}$$

where $\theta = \theta' - C(s,a,s')$ and Z is the normalizing factor $\sum_{s',\theta'}\sum_{s,\theta}\hat{b}(s,\theta)$ $T(s,a,s')O(s,a,s',o)$. Notice that for every augmented state (s,θ) where $\hat{b}_a^o(s,\theta) > 0$, s is not a goal state and θ is non-negative, i.e., $s \notin \mathbf{G}$ and $\theta \geq 0$.

For an augmented belief state $\hat{b}$, one can compute its reachable probability $P(\hat{b})$ using the system of linear equations below:

$$P(\hat{b}) = \max_a \sum_{s',\theta'} \begin{cases} 0 & \text{if } \theta' < 0 \\ \sum_{s,\theta}\hat{b}(s,\theta)T(s,a,s') & \text{if } s' \in \mathbf{G},\ \theta' \geq 0 \\ \sum_{s,\theta}\hat{b}(s,\theta)T(s,a,s')\sum_o O(s,a,s',o)P(\hat{b}_a^o) & \text{if } s' \notin \mathbf{G},\ \theta' \geq 0 \end{cases} \tag{5.10}$$

where $\theta' = \theta - C(s,a,s')$.

The above equation shows the component of reachable probability from an augmented belief state $\hat{b}$. For each possible augmented transition $((s,\theta),a,(s',\theta'))$ from $\hat{b}$, i.e., $\hat{b}(s,\theta) > 0$, $T(s,a,s') > 0$ and $\theta' = \theta - C(s,a,s')$, the agent takes action a from augmented state (s,θ) and transits to successor augmented state (s',θ'), which have three cases, as below:

- If the resulting cost threshold θ' is negative, then the successor cannot be reached. Thus, the reachable probability is 0. Note that there is no need to

Algorithm 5: DFS()

82 DFS-UPDATE($\hat{b}_0$)

consider the observation here since agents can accurately identify if the cost threshold is negative.

- If the successor s' is a goal state and the resulting cost threshold θ' is non-negative, the agent would observe this situation and the goal state can be reached. Thus, the reachable probability is $\hat{b}(s,\theta)T(s,a,s')$, which is the probability the agent transits from (s,θ) to (s',θ'). Note that there is no need to consider the observation function here since agents can accurately identify if they have reached goal states.
- If the successor is not a goal state and the resulting cost threshold is non-negative, then the successor can be reached. Thus, the reachable probability can be recursively computed as the probability $\hat{b}(s,\theta)T(s,a,s')$ multiplied by the product of the observation probability $O(s,a,s',o)$ and the reachable probability of the resulting belief state $P(\hat{b}_a^o)$ summed over all observations o.

One can extract the optimal policy by taking the action that is returned by the maximization operator in Equations 5.10 for each belief b.

DFS would traverse all augmented belief states that are reachable from the initial augmented belief states. Since it is pretty hard to design a efficient mechanism, such as a hash table, to search and insert related information of an augmented belief, pruning a subtree of augmented belief states by checking if it has been traversed before may not accelerate the algorithm. For RS-POMDPs, one possible way to store all reachable augmented belief states and their related information is to save the policy tree corresponding to possible execution histories. Without pruning, Algorithm 5 shows one possible pseudo-code of DFS for RS-POMDP.

Procedure DFS-Update($\hat{b}$)

$P^*(\hat{b}) = 0$
for actions $a \in \mathbf{A}$ **do**
    $P_a = 0$
    for observations $o \in \mathbf{\Omega}$ **do**
        $P_{a,o} = 0$
    end
    for augmented states $(s, \theta) \mid \hat{b}(s, \theta) > 0$ **do**
        for states $s' \in \mathbf{S} \mid T(s, a, s') > 0$ **do**
            $\theta' = \theta - C(s, a, s')$
            if $s' \in \mathbf{G}$ and $\theta' \geq 0$ **then**
                $P_a = P_a + \hat{b}(s, \theta) \cdot T(s, a, s')$
            else if $s' \notin \mathbf{G}$ and $\theta' \geq 0$ **then**
                for observations $o \in \mathbf{\Omega}$ **do**
                    $P_{a,o} = P_{a,o} + \hat{b}(s, \theta) \cdot T(s, a, s') \cdot O(s, a, s', o)$
                end
            end
        end
        for observations $o \in \mathbf{\Omega} \mid P_{a,o} > 0$ **do**
            $\hat{b}_a^o \leftarrow$ Belief_Update(b, a, o)
            DFS-Update$(\hat{b}_a^o)$
            $P_a = P_a + P_{a,o} \cdot P^*(\hat{b}_a^o)$
        end
    end
    if $P_a > P^*$ **then**
        $P^*(\hat{b}) = P_a$
        $\pi^*(\hat{b}) = a$
    end
end

Algorithm 5 is implemented recursively and it calls Procedure DFS-UPDATE starting from the initial augmented belief state $\hat{b}_0$. Procedure DFS-UPDATE corresponds to Equation 5.10, where it computes the reachable probability for each belief state. For each belief state $\hat{b}$, Procedure DFS-UPDATE uses variable $P^*(\hat{b})$ to store the reachable probability and it is initialized as 0 at the beginning (line 83). Additionally, temporary variables P_a (line 85) and $P_{a,o}$ (lines 86-88) are used to record the reachable probability for all possible actions a and the transition probability of transiting from belief state $\hat{b}$ to $\hat{b}_a^o$ by taking action a. For each possible augmented transition $((s,\theta),a,(s',\theta'))$, lines 89-105 compute the components of the reachable probability with respect to a and add them up to get P_a. Corresponding to Equation 5.10, if an augmented transition reach an augmented state with negative cost, it is ignored; otherwise the algorithm add up the corresponding transition probability to P_a if the successor state is a goal state (lines 92-93) or to $P_{a,o}$ if the successor state is a non-goal state (lines 94-98). Lines 100-104 add the reachable probability component of successor augmented belief state to P_a. The function BELIEF_UPDATE implements Equations 5.9 (line 101). For each reachable augmented belief state, DFS update its reachable probability based on reverse topological sort order (line 102). Finally, it stores the largest probability $P^*(\hat{b})$ and the optimal action $\pi^*(\hat{b})$ (lines 106-109).

If costs can be observed, then the augmented belief state $\hat{b}$ would have a specific form, in that all pairs (s,θ) with non-zero probability $\hat{b}(s,\theta) > 0$ have exactly the same cost threshold θ since costs can be observed. In the implementation, the augmented belief states can be simplified as (b,θ), where b is the belief state on original state space, because the cost threshold for all augmented states are same. Finally, all dead-ends in original state space can be identified and ignored by assuming their value functions always equal to 0.

5.2.4 Dynamic Programming (DP)

Similar to the *Dynamic Programming* (DP) algorithm for RS-MDPs, it can also be adapted to solve RS-POMDPs when costs can be observed. However, instead of *augmented states*, DP for RS-POMDPs operate on *augmented belief states*. Additionally, like in RS-MDPs, the reachable probabilities of augmented belief states only depend on augmented belief states with smaller cost thresholds. Thus, the high-level idea here, too, is that the reachable probabilities of each augmented belief state $\hat{b}(s, \theta)$ can be updated from $\theta = 0$ to $\theta = \theta_0$.

DP partitions the augmented POMDP into multiple partitions corresponding to the available cost threshold θ. Thus, each partition contains all the augmented belief states $\hat{b}(s, \theta)$ with the same cost threshold θ. Further, transitions between partitions must be from an upstream augmented belief state $\hat{b}(s, \theta)$ to a downstream augmented belief state $\hat{b}(s', \theta')$, where $\theta > \theta'$. Therefore, DP sequentially solves the partitions, where each partition corresponds to a regular POMDP, starting from the partition for $\theta = 0$ up to the partition for $\theta = \theta_0$.

Note that this approach applies only when costs can be observed. The reason is that through this assumption, all pairs (s, θ) with non-zero probability $\hat{b}(s, \theta) > 0$ have exactly the same cost threshold θ. Thus, all these augmented belief states are in the same partition, and DP can solve the different partitions sequentially. If costs cannot be observed, then the set of augmented belief states with non-zero probabilities may be distributed across multiple partitions, thereby requiring all partitions to be solved concurrently.

5.2.5 Point-Based Algorithms

Since the number of vectors in Γ grows exponentially in each update iteration, researchers have introduced point-based algorithms [Pineau *et al.*, 2003; Shani

et al., 2013] that update the vectors by considering only a restricted subset of belief state points. In each iteration, these algorithms keep only a vector with the smallest cost for each belief point, thereby restricting the number of vectors in Γ to the number of belief state points. Naturally, this comes at the cost of optimality and most point-based algorithms do not have quality guarantees on the solutions found.

Similar to point-based POMDP algorithms, one can extend FVI, VI for augmented POMDPs, and DP to their point-based counterparts *Point-Based FVI* (PB-FVI), *Point-Based VI* (PB-VI), and *Point-Based DP* (PB-DP). For all these point-based algorithms, they update their vectors in Γ by considering only a restricted subset of augmented belief state points. In each of their iterations, they keep only a vector with the largest reachable probability for each augmented belief state point, thereby restricting the number of vectors in Γ to the number of augmented belief state points. A difference between the three algorithms is that FVI and augmented VI maintain only a single subset of augmented belief state points to represent the whole augmented state space, but DP maintains a single subset for each partition of the augmented POMDP.

5.3 RS-POMDP Complexity

The complexity of POMDPs is proved in [Papadimitriou and Tsitsiklis, 1987]. By following similar technical steps, we show the complexity of RS-POMDPs in this subsection.

Theorem 3 *Solving RS-POMDPs optimally is PSPACE-hard in the original state space.*

Proof: To show PSPACE-hardness, we shall reduce *Quantified SAT* (QSAT)

to RS-POMDP. A QSAT can be defined as a logic formula $F(x_1, \ldots, x_n)$, which has n variables $x_1, \ldots, x_n$. Each variable x_i is either an existential or universal variable. By a standard transformation, formula $F(x_1, \ldots, x_n)$ can be transformed to *Conjunctive Normal Form* (CNF) $C_1 \wedge C_2 \wedge \ldots \wedge C_m$ with m clauses $C_1, \ldots, C_m$. A QSAT needs to decide if its value is true.

A QSAT can be reduced to an RS-POMDP here. Starting from any quantified formula $\exists x_1 \ldots \exists x_k \forall x_{k+1} \ldots \forall x_n F(x_1, \ldots, x_n)$, with n variables (existential or universal) and m clauses $C_1, \ldots, C_m$, we construct an RS-POMDP such that its optimal policy has reachable probability 1 if and only if the formula is true.

We first describe $\mathbf{S}$, b_0, and $\mathbf{G}$. The initial belief state b_0 includes only one state s_0, i.e. $b_0(s_0) = 1$. Besides s_0, $\mathbf{S}$ contains six states $s^T_{i,j}$, $s^F_{i,j}$, $t^T_{i,j}$, $t^F_{i,j}$, $f^T_{i,j}$, $f^F_{i,j}$ that each corresponds to clause C_i and variable x_j. There are also $2m$ states $s^T_{i,n+1}$, $s^F_{i,n+1}$. Finally, there is a goal state s_g that forms the set of goal states $\mathbf{G}$. Next, we describe $\mathbf{A}$, $\mathbf{T}$, and $\mathbf{C}$. There are, in total, three actions a, a_t, and a_f, i.e. $\mathbf{A} = \{a, a_t, a_f\}$. At s_0, there is only one applicable action a, leading to the states $s^F_{i,1}$, $i = 1, \ldots, m$ with equal probability. If x_j is an existential variable, then there are two applicable actions a_t and a_f for states $s^T_{i,j}$, leading with certainty from $s^T_{i,j}$ to $t^T_{i,j}$ and $f^T_{i,j}$, respectively. Similarly, there are two applicable actions a_t and a_f for states $s^F_{i,j}$, leading with certainty from $s^F_{i,j}$ to $t^F_{i,j}$ and $f^F_{i,j}$, respectively. If x_j is a universal variable, then there is only one applicable action a for state $s^T_{i,j}$, leading with equal probability from $s^T_{i,j}$ to $t^T_{i,j}$ and $f^T_{i,j}$. Similarly, there is only one applicable action a for states $s^F_{i,j}$, leading with equal probability from $s^F_{i,j}$ to $t^F_{i,j}$ and $f^F_{i,j}$. From the states $t^T_{i,j}$, $f^T_{i,j}$, $t^F_{i,j}$, and $f^F_{i,j}$, there is only one applicable action a that leads with certainty from $t^T_{i,j}$, $f^T_{i,j}$, $t^F_{i,j}$, and $f^F_{i,j}$ to $s^T_{i,j+1}$, $s^T_{i,j+1}$, $s^F_{i,j+1}$, and $s^F_{i,j+1}$, respectively, with two exceptions: If x_j appears positively in C_i, then the transition from $t^F_{i,j}$ is to $s^T_{i,j+1}$ instead of $s^F_{i,j+1}$; and if x_j appears negatively

in C_i, then the transition from $f^F_{i,j}$ is to $s^T_{i,j+1}$ instead of $s^F_{i,j+1}$. Finally, out of $s^T_{i,n+1}$ and $s^F_{i,n+1}$, there is only one applicable action a, leading to the goal state s_g with certainty. In the entire process, all actions have cost 1, except for the action out of $s^F_{i,n+1}$, which incurs a cost of 2. Finally, we set the initial cost threshold $\theta_0 = 2n + 2$. Then, the threshold set $\mathbf{\Theta}$ is naturally $\{0, 1, 2, \ldots, 2n + 2\}$. This completes the construction of the process.

We claim that the optimal reachable probability is 1 if and only if the value of QSAT was TRUE. Let us first assume that the reachable probability is indeed 1. Then by following the optimal RS-POMDP policy, all potential execution trajectories will finally reach states $s^T_{i,n+1}$ and their accumulated costs will be all $2n + 2$. Otherwise, a transition from $s^F_{i,n+1}$ will result in a cost of 2 and its accumulated costs will be $2n + 3$, which is larger than θ_0. Following the potential execution trajectories, it easy to deduce that the formula value is TRUE under all possible combination of variables' assignments. If the value of QSAT was TRUE, it is easy to construct the optimal RS-POMDP policy by picking corresponding actions that transit to states that will transit to $s^T_{i,j+1}$ for each existential variable x_j, then the potential execution trajectories would always reach states $s^T_{i,n+1}$, and the reachable probability is 1. ■

Notice that RS-POMDPs are not in PSPACE in the original state space, because solving them requires specifying an action for each augmented belief state, and the augmented state space could be exponential in the size of the original state space $\mathbf{S}$ if $|\mathbf{\Theta}| = 2^{|\mathbf{S}|}$.

	Observable Actual Costs	Unobservable Actual Costs
With Zero Costs	• FVI • AUG-VI • DP	• FVI • AUG-VI
Without Zero Costs	• FVI • AUG-VI • DP • DFS	• FVI • AUG-VI • DFS

Table 5.1: RS-POMDP Configurations and their Applicable Algorithms

5.4 RS-POMDP Experimental Results

We ran experiments on the same three domains as in RS-MDPs, except that they now have partial observability: (1) randomly generated POMDPs, (2) the *Navigation* domain from the ICAPS 2011 *International Probabilistic Planning Competition* (IPPC), and (3) a taxi domain [Ziebart *et al.*, 2008; Varakantham *et al.*, 2012] generated from real-world data. Also, just like in RS-MDPs, we generated two types of POMDPs here as well – ones without zero costs and ones with zero costs. Along a different dimension, we also differentiated the two cases of whether the actual costs can be observed or cannot be observed. Therefore, for each domain, we will have four different configurations.

In general, we compared the following algorithms: FVI, VI on the augmented POMDP (which we label as AUG-VI), DP, and DFS. For the first three algorithms, we experimented with their different variants: with pruning, without pruning, and their point-based counterparts, where we varied the number of points between 10 and 1000. However, some of the algorithms are only applicable in certain con-

figurations, and we only applied them there. Specifically, DP is only applicable for configurations where actual costs can be observed and DFS is only applicable in configurations without zero costs. FVI and AUG-VI are applicable in all configurations. Table 5.1 tabulates the applicable algorithms for each configuration. While we have results for all four configurations in all three domains described above, to improve readability, we only show the results for RS-POMDPs without zero costs but observable actual costs in the subsections below. The results for the other configurations are provided in the appendix.

Similar to the results for RS-MDPs, we report scalability in terms of the percentage of instances solved (%); average runtime in milliseconds of instances solved (t); and average reachable probability of all instances (P). We impose a timeout of 10 minutes. All experiments were conducted on a 3.40 GHz machine with 16GB of RAM.

5.4.1 Randomly Generated POMDPs

Just like in the RS-MDP experiment, we ran two types of experiments here – one where we vary the number of states $|\mathbf{S}|$ from 25 to 200, and another where we vary the cost threshold θ_0 from 1.25 to 3 times of $\mathcal{C}_d^*$, where $\mathcal{C}_d^*$ is the accumulated cost of the shortest deterministic path from the start state s_0. We fix the cost threshold $\theta = 1.5 \cdot \mathcal{C}_d^*$ when we vary the number of states, and we fix the number of states $|\mathbf{S}| = 50$ when we vary the cost thresholds.

Each randomly generated POMDP has 2 actions per state, 2 successors per action, and 2 possible observations per transition. We randomly selected a state as the start state and another state as the goal state, and we chose the costs from the range [0, 100]. Tables 5.2 and 9.1 to 9.3 show the results for the four different POMDP test configurations. Results are averaged over 50 randomly generated

(a) FVI

$\|\mathbf{S}\|$	w/o Pruning			w/ Pruning			PB(10)			PB(100)			PB(1000)		
	%	t	P	%	t	P	%	t	P	%	t	P	%	t	P
25	8	6084	1.61e-1	18	9639	2.03e-1	100	4	2.67e-1	100	28	2.89e-1	100	347	2.90e-1
50	6	7	5.31e-2	8	7490	6.01e-2	100	7	1.61e-1	100	65	2.19e-1	100	1067	2.23e-1
100	2	33	4.89e-2	6	125913	5.00e-2	100	23	1.21e-1	100	409	1.57e-1	100	7846	1.65e-1
200	0	–	1.21e-2	0	–	1.21e-2	100	54	6.15e-2	100	749	1.06e-1	100	41453	1.22e-1

θ_0	w/o Pruning			w/ Pruning			PB(10)			PB(100)			PB(1000)		
	%	t	P	%	t	P	%	t	P	%	t	P	%	t	P
$1.25{\cdot}\mathcal{C}_d^*$	12	1004	6.68e-2	18	52917	7.96e-2	100	1	1.66e-1	100	25	1.91e-1	100	455	1.92e-1
$1.50{\cdot}\mathcal{C}_d^*$	12	17506	6.19e-2	12	3099	7.30e-2	100	4	2.13e-1	100	102	2.43e-1	100	1481	2.46e-1
$2.00{\cdot}\mathcal{C}_d^*$	6	12	6.19e-2	8	1690	7.06e-2	100	24	2.57e-1	100	932	3.08e-1	100	32117	3.14e-1
$3.00{\cdot}\mathcal{C}_d^*$	2	62	4.78e-2	4	270480	5.05e-2	100	223	3.07e-1	100	34486	4.06e-1	62	119265	4.18e-1

(b) AUG-VI

$\|\mathbf{S}\|$	w/o Pruning			w/ Pruning			PB(10)			PB(100)			PB(1000)		
	%	t	P	%	t	P	%	t	P	%	t	P	%	t	P
25	8	10071	1.54e-1	18	53691	2.09e-1	100	63	2.67e-1	100	184	2.89e-1	100	438	2.90e-1
50	6	41	5.31e-2	8	71881	6.58e-2	100	105	1.61e-1	100	416	2.19e-1	100	1344	2.23e-1
100	2	1029	4.89e-2	6	194281	5.04e-2	100	541	1.21e-1	100	2555	1.57e-1	100	9382	1.65e-1
200	0	–	8.90e-3	0	–	1.21e-2	100	1152	6.15e-2	100	6177	1.06e-1	98	49222	1.22e-1

θ_0	w/o Pruning			w/ Pruning			PB(10)			PB(100)			PB(1000)		
	%	t	P	%	t	P	%	t	P	%	t	P	%	t	P
$1.25{\cdot}\mathcal{C}_d^*$	12	1947	6.68e-2	16	88291	8.18e-2	100	62	1.66e-1	100	169	1.91e-1	100	557	1.92e-1
$1.50{\cdot}\mathcal{C}_d^*$	12	30942	6.19e-2	12	37720	8.42e-2	100	130	2.13e-1	100	641	2.43e-1	100	1993	2.46e-1
$2.00{\cdot}\mathcal{C}_d^*$	6	213	6.19e-2	8	46738	7.30e-2	100	452	2.57e-1	100	5435	3.08e-1	98	25275	3.14e-1
$3.00{\cdot}\mathcal{C}_d^*$	2	126	3.98e-2	2	9626	7.33e-2	100	1644	3.07e-1	100	29195	4.06e-1	68	95499	4.25e-1

(c) DP

$\|\mathbf{S}\|$	w/o Pruning			w/ Pruning			PB(10)			PB(100)			PB(1000)		
	%	t	P	%	t	P	%	t	P	%	t	P	%	t	P
25	8	1814	9.02e-2	54	88521	2.33e-1	100	11	2.86e-1	100	714	2.90e-1	100	16518	2.90e-1
50	6	1	4.21e-2	16	108098	1.22e-1	100	37	1.82e-1	100	4221	2.23e-1	98	198203	2.21e-1
100	2	125	4.01e-2	8	23757	5.05e-2	100	128	1.25e-1	100	23447	1.65e-1	18	195205	1.06e-1
200	0	–	0.00e+0	0	–	1.37e-2	100	291	6.71e-2	100	69943	1.22e-1	6	247197	2.23e-2

θ_0	w/o Pruning			w/ Pruning			PB(10)			PB(100)			PB(1000)		
	%	t	P	%	t	P	%	t	P	%	t	P	%	t	P
$1.25{\cdot}\mathcal{C}_d^*$	12	812	3.98e-2	32	71190	1.07e-1	100	16	1.74e-1	100	2836	1.92e-1	96	120095	1.92e-1
$1.50{\cdot}\mathcal{C}_d^*$	12	8587	4.60e-2	18	15482	1.13e-1	100	33	2.18e-1	100	4992	2.46e-1	86	181167	2.41e-1
$2.00{\cdot}\mathcal{C}_d^*$	6	19	4.60e-2	12	68226	1.15e-1	100	81	2.76e-1	100	10047	3.14e-1	60	243267	2.78e-1
$3.00{\cdot}\mathcal{C}_d^*$	2	1	4.60e-2	8	26823	1.15e-1	100	196	3.38e-1	100	21021	4.32e-1	22	196940	3.11e-1

(d) DFS

$\|\mathbf{S}\|$	%	t	P
25	88	21236	2.58e-1
50	84	5103	1.89e-1
100	88	26557	1.53e-1
200	76	57157	9.68e-2

θ_0	%	t	P
$1.25{\cdot}\mathcal{C}_d^*$	96	2657	1.85e-1
$1.50{\cdot}\mathcal{C}_d^*$	86	16287	2.11e-1
$2.00{\cdot}\mathcal{C}_d^*$	62	32669	1.93e-1
$3.00{\cdot}\mathcal{C}_d^*$	26	70590	1.04e-1

Table 5.2: RS-POMDP Results of Randomly Generated POMDPs without Zero Costs but Observable Actual Costs

(a) FVI

θ_0	w/o Pruning			w/ Pruning			PB(10)			PB(100)			PB(1000)		
	%	t	P	%	t	P	%	t	P	%	t	P	%	t	P
$1.25 \cdot \mathcal{C}_d^*$	20	8	5.36e-3	50	22906	7.84e-3	100	1	0.00e+0	100	19	4.79e-3	100	490	7.81e-3
$1.50 \cdot \mathcal{C}_d^*$	20	41097	2.10e-2	50	201174	4.73e-2	100	1	0.00e+0	100	11	5.05e-3	100	1007	6.54e-2
$2.00 \cdot \mathcal{C}_d^*$	10	20	2.10e-2	30	129184	4.41e-2	100	1	0.00e+0	100	35	5.23e-3	100	2773	1.08e-1
$3.00 \cdot \mathcal{C}_d^*$	10	239	4.21e-3	10	1009	5.37e-2	100	1	0.00e+0	100	203	5.08e-2	100	29020	2.19e-1

(b) AUG-VI

θ_0	w/o Pruning			w/ Pruning			PB(10)			PB(100)			PB(1000)		
	%	t	P	%	t	P	%	t	P	%	t	P	%	t	P
$1.25 \cdot \mathcal{C}_d^*$	40	83984	7.84e-3	30	140743	7.84e-3	100	38	0.00e+0	100	93	4.79e-3	100	605	7.81e-3
$1.50 \cdot \mathcal{C}_d^*$	20	792	2.49e-2	30	189695	3.10e-2	100	45	0.00e+0	100	177	5.05e-3	100	2403	6.54e-2
$2.00 \cdot \mathcal{C}_d^*$	10	142	3.17e-2	10	2818	4.39e-2	100	67	0.00e+0	100	766	5.23e-3	100	8732	1.08e-1
$3.00 \cdot \mathcal{C}_d^*$	10	849	2.52e-2	10	28141	3.75e-2	100	126	0.00e+0	100	7259	5.08e-2	90	13476	2.19e-1

(c) DP

θ_0	w/o Pruning			w/ Pruning			PB(10)			PB(100)			PB(1000)		
	%	t	P	%	t	P	%	t	P	%	t	P	%	t	P
$1.25 \cdot \mathcal{C}_d^*$	60	1711	1.45e-2	70	115671	1.45e-2	100	48	4.58e-3	100	2993	1.45e-2	100	41284	1.45e-2
$1.50 \cdot \mathcal{C}_d^*$	30	152099	6.11e-2	60	115092	6.27e-2	100	70	6.59e-3	100	5547	6.80e-2	100	78387	6.80e-2
$2.00 \cdot \mathcal{C}_d^*$	20	1685	6.78e-2	40	171183	8.52e-2	100	78	7.55e-3	100	13247	1.15e-1	90	92916	1.22e-1
$3.00 \cdot \mathcal{C}_d^*$	10	15	6.80e-2	30	186990	1.33e-1	100	155	3.58e-2	100	33922	2.92e-1	80	127857	3.11e-1

(d) DFS

θ_0	%	t	P
$1.25 \cdot \mathcal{C}_d^*$	80	37074	5.36e-3
$1.50 \cdot \mathcal{C}_d^*$	70	28569	5.20e-2
$2.00 \cdot \mathcal{C}_d^*$	50	4849	7.49e-2
$3.00 \cdot \mathcal{C}_d^*$	40	58267	1.62e-1

Table 5.3: RS-POMDP Results of Navigation Domain without Zero Costs but Observable Actual Costs

(a) FVI

θ_0	w/o Pruning			w/ Pruning			PB(10)			PB(100)			PB(1000)		
	%	t	P	%	t	P	%	t	P	%	t	P	%	t	P
$1.25 \cdot \mathcal{C}_d^*$	0	–	0.00e+0	0	–	0.00e+0	100	29	0.00e+0	100	498	1.79e-5	100	23374	1.64e-4
$1.50 \cdot \mathcal{C}_d^*$	0	–	0.00e+0	0	–	0.00e+0	100	13	0.00e+0	100	935	9.83e-3	100	130153	2.25e-2
$2.00 \cdot \mathcal{C}_d^*$	0	–	0.00e+0	0	–	0.00e+0	100	74	1.25e-1	100	13949	5.36e-1	62	404460	6.18e-1
$3.00 \cdot \mathcal{C}_d^*$	0	–	0.00e+0	0	–	0.00e+0	100	257	1.00e+0	100	2024	1.00e+0	100	31566	1.00e+0

(b) AUG-VI

θ_0	w/o Pruning			w/ Pruning			PB(10)			PB(100)			PB(1000)		
	%	t	P	%	t	P	%	t	P	%	t	P	%	t	P
$1.25 \cdot \mathcal{C}_d^*$	0	–	0.00e+0	0	–	0.00e+0	100	95	0.00e+0	100	896	1.79e-5	100	17175	1.64e-4
$1.50 \cdot \mathcal{C}_d^*$	0	–	0.00e+0	0	–	0.00e+0	100	133	0.00e+0	100	1583	9.83e-3	100	59432	2.25e-2
$2.00 \cdot \mathcal{C}_d^*$	0	–	0.00e+0	0	–	0.00e+0	100	592	1.25e-1	100	18536	5.36e-1	100	147611	6.18e-1
$3.00 \cdot \mathcal{C}_d^*$	0	–	0.00e+0	0	–	0.00e+0	100	2106	1.00e+0	100	3857	1.00e+0	100	22053	1.00e+0

(c) DP

θ_0	w/o Pruning			w/ Pruning			PB(10)			PB(100)			PB(1000)		
	%	t	P	%	t	P	%	t	P	%	t	P	%	t	P
$1.25 \cdot \mathcal{C}_d^*$	0	–	0.00e+0	37	160259	1.67e-4	100	63	1.19e-4	100	480	1.30e-4	100	21803	1.98e-4
$1.50 \cdot \mathcal{C}_d^*$	0	–	0.00e+0	0	–	2.47e-4	100	134	1.58e-2	100	1117	2.10e-2	100	27934	2.29e-2
$2.00 \cdot \mathcal{C}_d^*$	0	–	0.00e+0	0	–	2.47e-4	100	334	5.24e-1	100	2997	6.20e-1	100	47479	6.23e-1
$3.00 \cdot \mathcal{C}_d^*$	0	–	0.00e+0	0	–	2.47e-4	100	707	1.00e+0	100	6830	1.00e+0	100	90851	1.00e+0

(d) DFS

θ_0	%	t	P
$1.25 \cdot \mathcal{C}_d^*$	100	12283	1.98e-4
$1.50 \cdot \mathcal{C}_d^*$	100	96307	2.29e-2
$2.00 \cdot \mathcal{C}_d^*$	100	386248	6.23e-1
$3.00 \cdot \mathcal{C}_d^*$	100	429773	1.00e+0

Table 5.4: RS-POMDP Results of Taxi Domain without Zero Costs but Observable Actual Costs

instances.

We make the following observations for POMDPs without zero costs but observable actual costs:

- With increasing initial cost threshold or number of states, in general, scalability decreases and runtime increases for all algorithms. The reason is that with a larger cost threshold or a larger number of states, each algorithm has to search over a larger search space.
- DFS is faster and more scalable than DP, FVI, and AUG-VI because DFS ignores non-reachable belief states while DP, FVI, and AUG-VI do not.
- DP is faster and more scalable than FVI and AUG-VI because it partitions the augmented belief states into several partitions, one for each cost threshold, and updates the reachable probabilities sequentially from one partition to the next. As a result, once convergence is reached for a partition, it is no longer updated. In contrast, FVI and AUG-VI update the reachable probabilities for all augmented belief states concurrently. Thus, they may continue to check belief states that have already converged. This reason is also similar to the reason why TVI (for MDPs) is faster and more scalable than VI (for MDPs).
- The point-based versions of DP, FVI, and AUG-VI are faster and more scalable than their respective optimal counterparts. They find better solutions with increasing number of belief points but at the cost of increasing runtime and decreasing scalability. So much so that with 1000 belief points, PB-DP is less scalable than DFS even though DFS is an optimal algorithm and PB-DP is not.
- Finally, pruning improves the scalability of FVI, AUG-VI, and DP.

For the other POMDP configurations, the above observations generally apply as well. However, we can also make the following additional observation: All

the optimal algorithms generally found better solutions (with larger reachable probabilities) when costs can be observed because they can exploit the observed cost to more accurately update the beliefs. This observation may not hold for the point-based algorithms as they are not guaranteed to solve the problem optimally.

5.4.2 Navigation Domain

For the navigation domain, similar to RS-MDPs, we also use all 10 IPPC instances. This domain is as described in Section 4.6.2. However, the range of costs is now only up to 100 instead of 1000. Further, the robot now may not have complete certainty on its actual state and must maintain a belief over its possible states. Tables 5.3 and 9.4 to 9.6 show the results for the four different RS-POMDP configurations described in Table 5.1.

In general, all the observations from the randomly generated POMDP domain apply here as well except for the following: The point-based algorithms successfully solve most of the instances in all configurations. However, they often do not find good quality solutions unlike in randomly generated POMDPs. For example, PB-FVI finds infeasible solutions (the reachable probability is 0) when the number of points is small (= 10) in Table 5.3. This observation highlights the fact that the behavior of point-based algorithms is highly dependent on domain structure.

5.4.3 Taxi Domain

For the taxi domain, it is as described in Section 4.6.3. However, states now include the hired rate level $p_{z,t}$ in its tuple, which is either high (= 0.75) or low (= 0.25). Taxis looking for passengers in its zone (i.e., action a_2) have probability $p_{z,t}$ of successfully picking up a passenger and they can accurately observe $p_{z,t}$ with probability 0.8. This probability is generated with the same real-world dataset.

Finally, we reduce the number of zones from 100 to 10, by clustering some of the original zones into a single zone. Tables 5.4 and 9.7 to 9.9 show the results for the four different RS-POMDP configurations described in Table 5.1.

In general, all the observations from the randomly generated POMDP and Navigation domains apply here as well except for the following: Unlike those two domains, FVI, AUG-VI, and DP often failed to solve all instances, and they must rely on their point-based counterparts to find suboptimal solutions. However, surprisingly, DFS successfully solved all instances (see Tables 5.4 and 9.7). Therefore, this result shows that in this domain, there is a very large number of meaningful augmented belief states, but only a small number of them are reachable.

Chapter 6

Local Search

Instead of solving *Risk-Sensitive MDPs* (RS-MDPs) optimally using the algorithms described in Chapter 4, we now introduce a local search algorithm that solves them approximately.

6.1 Local Search Algorithm

In theory, if an optimal policy of the *Minimizing Expected Cost* (MEC) criterion is executed on an RS-MDP, the quality is unbounded. However, in some problems, optimal policies based on the MEC-criterion can provide pretty good solution qualities.

Figure 6.1 show one such example. In this instance, the initial state is s_0 and all other states are goal states. There are two actions – a_1 and a_2 – from s_0, and the corresponding transition and cost functions are shown in the figure. Based on the MEC-criterion, a_2 is the optimal action since it has an expected cost of 16 ($= 0.8 \cdot 15 + 0.2 \cdot 20$) and a_1 has an expected cost of 17 ($= 0.3 \cdot 10 + 0.7 \cdot 20$).

However, based on the RS-criterion, when the initial cost threshold is between $\theta_0 = [10, 15)$, the optimal action is a_1 as it has a reachable probability of 0.3 and

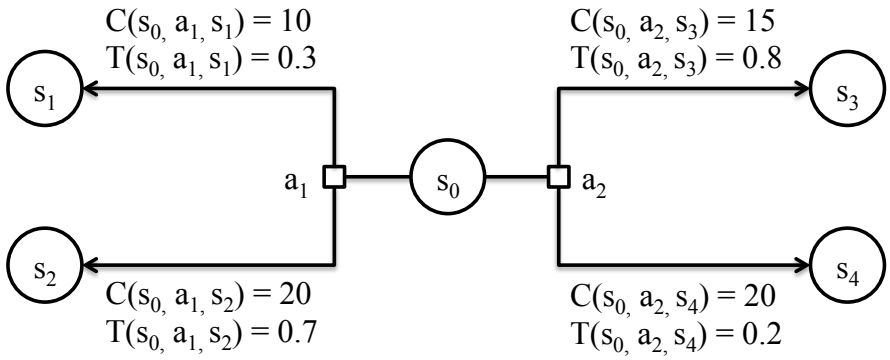

Figure 6.1: MDP Example Illustrating the Relationship between MEC- and RS-optimal Policies

a_2 has a reachable probability of 0. When the initial cost threshold is between $\theta_0 = [15, 20)$, then the optimal action is a_2 with reachable probability 0.8, while action a_1 has a reachable probability of 0.3. Thus, the MEC optimal policy is the same as the RS optimal policy when the initial cost threshold is between $[15, 20)$. This example also highlights a qualitative observation that the the MEC optimal policy and RS optimal policy are close to each other when the initial cost threshold is close to the optimal expected cost.

Since finding an optimal MEC policies is normally easier than finding an optimal RS policy, We introduce an approximate algorithm – Local Search (LS), whose primary idea is to adjust the optimal MEC policy to adapt to the risk-sensitive criterion. The high-level ideas of LS is as follows: It will first solve for and obtain an optimal policy based on the MEC-criterion, and then assume that every augmented state (s, θ) for a state s and all possible cost thresholds θ have the same policy. Let us call this augmented policy an *abstracted policy*. Given this abstracted policy, we can compute the set of reachable augmented states. Then, we iterate through this set of reachable augmented states, and for each augmented state in this set, we evaluate all the actions and choose the action

Algorithm 6: LS$(\theta_0, \mathcal{M})$

```
111 π_c = MEC-SOLVER(M)
112 EVALUATE(s_0, θ_0)
113 while π has not converged do
114 |   UPDATE-ITERATION(s_0, θ_0)
115 end
```

that maximizes the reachable probability under the assumption that every other augmented state uses the abstracted policy. Once a different action is chosen for an augmented state, we now have a new policy for this augmented state. Once we have gone through all the reachable augmented states, we recompute the set of reachable augmented states again with the new updated abstracted policy and the abstracted policy will keep improve along this process. We keep repeating this process until there are no new reachable augmented states that it can find and the selected action for all reachable augmented states are the same as in the previous iteration. Algorithm 6 shows the pseudocode of the LS algorithm.

In Algorithm 6, it first computes the optimal policy π_c with respect to the MEC-criterion (line 110), and evaluates the current abstracted policy on the reachable augmented state space (line 111). Then, LS iterates over the risk-sensitive policy until it converges (lines 112-114).

Procedure UPDATE-ITERATION expands the reachable augmented state space recursively based on the current abstracted policy (lines 118-123), applies the Bellman update on each augmented state to decide which action is currently optimal (lines 124-141), and updates the abstracted policy (lines 142-143). Procedure EVALUATE would evaluate the default abstracted policy on augmented states if they are visited for the first time.

Similar to other local search algorithms, the LS algorithm may get stuck at a

Procedure Update-Iteration(s, θ)

```
116 if P(s, θ) is converge then
117 |   return;
118 end
119 for s' ∈ S | T(s, π(s, θ), s') > 0 do
120 |   if θ ≥ C(s, π(s, θ), s') then
121 |   |   θ' = θ − C(s, π_c(s), s')
122 |   |   UPDATE-ITERATION(s', θ')
123 |   end
124 end
125 P* = 0
126 for a ∈ A do
127 |   P_a = 0
128 |   for s' ∈ S | T(s, π(s, θ), s') > 0 do
129 |   |   if θ ≥ C(s, π(s, θ), s') then
130 |   |   |   if s' ∈ G then
131 |   |   |   |   P_a = P_a + T(s, a, s')
132 |   |   |   else
133 |   |   |   |   θ' = θ − C(s, π(s, θ), s')
134 |   |   |   |   P_a = P_a + T(s, π(s, θ), s') · EVALUATE(s', θ')
135 |   |   |   end
136 |   |   end
137 |   end
138 |   if P_a > P* then
139 |   |   P* = P_a
140 |   |   a* = a
141 |   end
142 end
143 P(s, θ) = P*
144 π(s, θ) = a*
```

Procedure Evaluate(s, θ)

```
145 if visited(s, θ) then
146 |   return P(s, θ)
147 end
148 P(s, θ) = 0
149 π(s, θ) = π_c(s)
150 for s' ∈ S | T(s, π_c(s), s') > 0 do
151 |   if θ ≥ C(s, π_c(s), s') then
152 |   |   if s' ∈ G then
153 |   |   |   P(s, θ) = P(s, θ) + T(s, π_c(s), s')
154 |   |   else
155 |   |   |   θ' = θ − C(s, π_c(s), s')
156 |   |   |   P(s, θ) = P(s, θ) + T(s, π_c(s), s') · EVALUATE(s', θ')
157 |   |   end
158 |   end
159 end
160 return P(s, θ)
```

local optima and, thus, do not guarantee optimality. The reason is that it would terminate when the optimal actions do not change based on the current abstracted policy, but the current abstracted policy may not be optimal.

6.2 Using a Random Policy as the Initial Policy

We proposed using an optimal MEC policy as the initial policy of the local search as we hypothesized that this policy is very similar to the optimal RS policy. However, this may not be true in practice. Further, the computation time to search for the MEC policy can relatively larger than the computation time of the local search improvements. Therefore, we also propose a variant of the local

search algorithm, where we start with a random policy as the initial policy and continuously improve this policy until convergence.

6.3 Local Search Experimental Results

We ran experiments on two domains in the RS-MDPs section: (1) randomly generated POMDPs and (2) a taxi domain [Ziebart *et al.*, 2008; Varakantham *et al.*, 2012] generated from real-world data. The *Navigation* domain from the ICAPS 2011 *International Probabilistic Planning Competition* (IPPC) is not applied here because it has dead-ends and it is impossible to compute optimal policy based on the MEC-criterion. For instances with zero costs, local search needs to expand the reachable state space once in every value iteration even when the reachable states do not change anymore. So, it is rather inefficient. For algorithms with utility functions, it is impossible to maintain consistency (augmented states with higher cost thresholds must have equal or higher reachable probability) by only maintaining one utility function. If we maintain two utility functions, it would involve a large number of copy operations, which would be inefficient. Further, maintaining multiple utility functions would contradict the original motivation of using utility functions, that is, to potentially use less memory. Thus, we only show the results for instances without zero costs and with algorithms that are based on latticed tables. Every experiment setting is exactly the same as in Chapter 4. Finally, we compared both local search variants – LS-MEC refers to the variant that uses the optimal MEC policy as the initial policy and LS-RAND refers to the variant that uses a random policy – against DFS and DP in these experiments. Since the initial policy of LS-RAND is randomized, we average each data point over 5 different runs for each instance.

$\|\mathbf{S}\|$	DFS(T)			DP(T)			LS-MEC(T)			LS-RAND(T)		
	%	t	P	%	t	P	%	t	P	%	t	P
2500	100	508	9.60e-2	100	148	9.60e-2	100	215	9.45e-2	100	32	2.42e-2
5000	100	1270	7.17e-2	100	311	7.17e-2	100	475	6.99e-2	100	81	1.04e-2
10000	100	4186	5.13e-2	100	1078	5.13e-2	100	1595	5.00e-2	100	218	5.92e-3
20000	100	9686	4.03e-2	100	2870	4.03e-2	100	5980	3.93e-2	100	638	3.75e-3
40000	100	27768	2.68e-2	100	8147	2.68e-2	66	22073	1.61e-2	66	1285	8.98e-4

θ_0	DFS(T)			DP(T)			LS-MEC(T)			LS-RAND(T)		
	%	t	P	%	t	P	%	t	P	%	t	P
$1.25 \cdot \mathcal{C}_d^*$	100	195	1.24e-2	100	212	1.24e-2	100	1425	1.14e-2	100	123	6.77e-4
$1.50 \cdot \mathcal{C}_d^*$	100	808	2.51e-2	100	347	2.51e-2	100	1428	2.37e-2	100	121	1.77e-3
$2.00 \cdot \mathcal{C}_d^*$	100	4378	5.16e-2	100	1191	5.16e-2	100	1593	5.01e-2	100	224	4.31e-3
$3.00 \cdot \mathcal{C}_d^*$	100	15613	1.07e-1	100	6321	1.07e-1	100	5759	1.06e-1	100	2993	1.18e-2
$5.00 \cdot \mathcal{C}_d^*$	100	39087	2.02e-1	100	19255	2.02e-1	100	56418	2.01e-1	100	48159	4.66e-2

Table 6.1: RS-MDP Results of Randomly Generated MDPs without Zero Costs

6.3.1 Randomly Generated MDPs

Table 6.3.1 shows the results for randomly generated MDPs without zero costs. We make the following observations:

- Generally, LS-MEC has comparable performance as DFS and DP. Normally, LS-MEC finds a solution very close to the optimal solution.
- For MDPs with different state sizes and the same initial cost threshold level ($\theta_0 = 2.00 \cdot \mathcal{C}_d^*$), DP is faster than DFS because it has a larger overhead of expanding augmented states than DP; the reason is further elaborated in detail in Section 4.6. In these problems, LS-MEC is faster than DFS but slower than DP. The overhead in expanding augmented states for LS-MEC is similar to that for DFS. When combined with the fact that LS-MEC expands fewer augmented states than DFS before it converges, LS-MEC is thus faster than DFS. However, since the overhead of LS-MEC is larger than DP, it is still slower than DP despite expanding fewer augmented states.

  When the instances become lager ($|\mathbf{S}| = 40000$), LS-MEC solves fewer

instances, and it fails because it ran out of memory. The reason is that LS-MEC actually requires more memory since it needs to store more latticed tables than DP and DFS. Specifically, LS-MEC also needs latticed tables to store optimal actions and to mark convergence of augmented states.

- For MDPs with different initial cost threshold levels and the same state size, LS-MEC is slower at the start, then faster, and finally slower again as the cost threshold increases. At the start, RS-MDPs are actually easy to solve compared to corresponding regular MDPs. Thus, the running time of LS-MEC is actually dominated by the running time to solve the regular MDPs with respect to the MEC-criterion ($\theta_0 = 1.25 \cdot \mathcal{C}_d^*$ to $2.00 \cdot \mathcal{C}_d^*$). When $\theta_0 = 3.00 \cdot \mathcal{C}_d^*$, the initial cost threshold becomes close to a certain ratio level of the expected optimal cost, under which the optimal policy with respect to the MEC-criterion and RS-criterion are mostly similar. Thus, LS-MEC can converge very quickly after a few number of iterations. When the initial cost threshold becomes larger ($\theta_0 = 5.00 \cdot \mathcal{C}_d^*$), then the optimal policy with respect to the RS-criterion become more different compared to the optimal MEC policy again. Thus, LS-MEC becomes slower again since it needs relatively much more iterations before converging.
- Comparing LS-MEC and LS-RAND, LS-RAND is much faster but the solutions found have worse qualities since it starts from a random policy.

6.3.2 Taxi Domain

Table 6.3.2 shows the results for taxi domain without zero costs. Recall that DFS is faster than DP in the taxi domain as it traverses fewer augmented states. See description in Section 4.6 for details. As the running time of LS-MEC and DFS are comparable, LS-MEC is thus also faster than DP. Other observations from the

θ_0	DFS(T)			DP(T)			LS-MEC(T)			LS-RAND(T)		
	%	t	P	%	t	P	%	t	P	%	t	P
$1.25 \cdot \mathcal{C}_d^*$	100	31	2.16e-9	100	126	2.16e-9	100	27	1.82e-9	100	3	0.00e+0
$1.50 \cdot \mathcal{C}_d^*$	100	84	1.28e-5	100	202	1.28e-5	100	78	1.28e-5	100	1	0.00e+0
$2.00 \cdot \mathcal{C}_d^*$	100	218	2.29e-2	100	353	2.29e-2	100	221	2.29e-2	100	13	3.10e-3
$3.00 \cdot \mathcal{C}_d^*$	100	473	1.74e-1	100	575	1.74e-1	100	481	1.74e-1	100	89	1.40e-1
$5.00 \cdot \mathcal{C}_d^*$	100	704	6.29e-1	100	752	6.29e-1	100	648	6.27e-1	100	84	6.25e-1

Table 6.2: RS-MDP Results of Taxi Domain without Zero Costs

randomly generated MDP domain apply here as well.

Chapter 7

Related Work

In this section, we will discuss the literature that is related to the work presented in this dissertation. Figure 7.1 illustrates the relationships between the different models, where we broadly classify the related models into two large classes:

- Risk-based MDP and POMDP models, where the notion of risk is explicitly or implicitly considered in the models. Within this large class, we further describe four general MDP and POMDP models: (a) MDPs and POMDPs with utility functions; (b) MDPs and POMDPs with reachable probabilities; (c) Multi-objective MDPs and POMDPs; and (d) Uncertain MDPs and POMDPs. We also describe how RS-MDPs and RS-POMDPs are related to each of these models in detail in the subsections below.
- Continuous MDP and POMDP models, where certain components of the problem can be continuous. RS-MDPs and RS-POMDPs are related to these models in that RS-MDPs and RS-POMDPs can be viewed as Continuous MDPs and POMDPs.

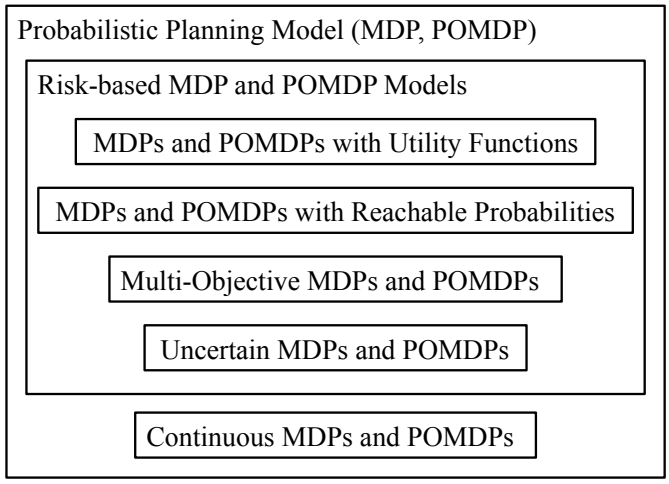

Figure 7.1: Overview of Related Models

7.1 Risk-based MDP and POMDP Models

We now describe the four general MDP and POMDP models that are risk based.

7.1.1 MDPs and POMDPs with Utility Functions

Recall that RS-MDPs and RS-POMDPs can be represented as MDPs and POMDPs with *Piecewise Constant* (PWC) utility functions. Therefore, they are also related to the more general problems of MDPs and POMDPs with arbitrary utility functions. Liu and Koenig; Liu and Koenig; Liu and Koenig [2005a; 2005b; 2006] introduced the original *Functional Value Iteration* (FVI) algorithm to finds optimal policies for MDPs with exponential, one-switch, and piecewise linear utility functions. Finally, Marecki and Varakantham [2010] extended FVI to solve POMDPs with piecewise linear utility functions. We have described FVI in great detail in Section 4.2.1 and 5.2.1, including how they can be used and adapted to solve RS-MDPs and RS-POMDPs. Therefore, we omit them here. In general, compared to FVI, we believe that our solution approaches should scale to much

larger problems better as we exploit some utility-dependent properties.

Ermon *et al.* [2012] extended the work above for MDPs by including a requirement that the returned policy needs to satisfy certain worst-case guarantees in addition to the expected utility maximization criterion. They also introduced an approach based on dynamic programming, but the algorithm recursively solves the problem backwards from the horizon to the starting time step. In contrast, our dynamic programming algorithm solves the problem from a cost threshold of 0 to the user-defined maximum cost threshold.

Another related work is by McMillen and Veloso [2007], where they solve a specific type of RS-MDPs – finite-horizon RS-MDPs with zero-sum utility functions – inspired by robot soccer. They used a dynamic programming based algorithm similar to that introduced by Ermon *et al.* [2012], which performs a one-swap backup from the horizon to the starting time step, to solve their problem.

7.1.2 MDPs and POMDPs with Reachable Probabilities

Another important body of work that is relevant to ours are the MDP and POMDP models that explicitly uses the reachable probabilities in their optimization criteria [Kolobov, 2013; Steinmetz *et al.*, 2016; Chatterjee *et al.*, 2016]. A general issue with goal-directed MDPs is the presence of *dead-ends*. Dead-ends can be classified into two types: avoidable and unavoidable dead-ends. Dead-ends are avoidable if there exist *proper* policies, which are policies that reach goal states with probability 1. However, some MDPs may not have proper policies, in which case the dead-ends are unavoidable. In such MDPs with unavoidable dead-ends, finding policies that minimize the expected cost is futile since all policies have a cost of $-\infty$, as they continuously incur a positive cost at dead-ends.

To remedy this issue, Kolobov *et al.* introduced the MAXPROB MDP

model, which is a goal-directed MDP with the objective of maximizing the probability of getting to a goal independent of cost [Kolobov *et al.*, 2011; Kolobov, 2013]. MAXPROB MDPs treat reachable probabilities as the reward function. In other words, it sets the transition to goal states to have a reward of 1 while all other transitions have a reward of 0. Thus, transitioning to dead-end states will accumulate a reward of 0 instead of incurring a cost of $-\infty$ as in regular MDPs. As such, it is possible to solve MAXPROB MDPs using Value Iteration. In summary, a MAXPROB MDP is equivalent to an RS-MDP where the objective is to find a policy π that maximizes $Pr(c^{\pi}(s_0) < \infty)$.

For MDPs with avoidable dead-ends, Chatterjee *et al.* [2016] introduced a different optimization criterion for POMDPs called as the *Optimal Cost Almost-Sure Reachability* criterion. Here, they are interested in finding a policy that minimizes the expected cost but guarantees that the reachable probability of the starting belief with this policy is 1. The general approach that they take to solve goal-directed POMDPs with this criterion is similar to local search, where they first find a proper policy and iteratively improve this policy while guaranteeing that the new policy found is also a proper policy.

Therefore, in summary, one can classify goal-directed MDPs and POMDPs as belonging to three categories, each with a reasonable criterion that one should optimize for.

- Goal-directed MDPs and POMDPs without dead-ends: In such (classical) problems, one should use the classical criterion of minimizing the expected cost since all policies can reach goals with probability 1.
- Goal-directed MDPs and POMDPs with avoidable dead-ends: In such problems, one should use the *Optimal Cost Almost-Sure Reachability* criterion since we are interested in differentiating between policies that can reach goals with

probability 1.

- Goal-directed MDPs and POMDPs with unavoidable dead-ends: In such problems, since no policy can reach goals with probability 1, we should use the MAXPROB criterion, which is to find a policy with the highest reachable probability, independent of cost.

However, in practice, an issue is that one typically cannot directly identify if the goal-directed MDP or POMDP at hand has dead-ends or not, let alone whether the dead-ends are avoidable or unavoidable. To solve this issue, one can combine the three criteria above in the following way to obtain a general criteria that can be used for all goal-directed MDPs and POMDPs:

(1) Solve the MDP or POMDP using the MAXPROB criteria. If there are unavoidable dead-ends (i.e., the reachable probability of the policy returned with the MAXPROB criteria is less than 1), then the problem is solved as it does not make sense to optimize for the remaining two criteria.

(2) If there are no unavoidable dead-ends (i.e., the reachable probability of the policy returned with the MAXPROB criteria is 1), then use Tarjan's algorithm to construct strongly connected components (SCCs) for the MDP or POMDP. If there exists a leaf SCC without any goal states, then there exists a dead-end and one should find a policy that optimizes the Optimal Cost Almost-Sure Reachability criterion.

(3) If there are no leaf SCCs without goal states, then there are no dead-ends, and one can use the classical criterion of minimizing the expected cost.

One limitation of the above criteria is that they only use reachable probabilities and expected costs of those trajectories as metrics to measure the quality of the trajectories. However, there may be other metrics that one might want to consider, especially domain-dependent metrics that may exist in some applications. We now

describe the body of work that investigates the use of these other metrics.

In general, this body of work categorizes trajectories into different types, based on a particular chosen metric. For example, for the metric of reachable probabilities, we can separate the trajectories into those that reach a goal and those that do not. Similarly, for the metric of expected costs, we can separate the trajectories into those that incur a cost larger than a particular cost threshold and those that incur a cost within the threshold. In RS-MDPs and RS-POMDPs, the objective is to find trajectories that reach a goal and whose cost is within a cost threshold.

If we wish to enforce that the probability of trajectories being in a category (e.g., that its cost is within a cost threshold) is within some probability threshold (e.g., the trajectories are in a category with probability greater than 0.7), then those satisfaction conditions can be enforced by *chance constraints*. Defourny *et al.* [2008] introduced one such criterion, where they seek to find a policy π that minimizes the expected cost *and* satisfies the *chance constraint* $Pr(c^{\pi}(s_0) \leq \theta_0) \geq p$, where p is a user-defined minimum probability threshold.

de Rodrigues Quemel e Assis Santana *et al.* [2016] proposed a different metric to distinguish the different trajectories: Whether a trajectory crosses a particular undesirable state s or not. They then enforce the requirement that the trajectories should cross undesirable states with probability no larger than a user-defined probability threshold using chance constraints. Specifically, they use the chance constraint $Pr(s \in \tau \mid s \in C \wedge \tau \in \mathcal{T}^{\pi}) \leq p$, where C is a set of user-defined undesirable states, $\mathcal{T}^{\pi}$ is the set of all execution trajectories of policy π, and we misuse the notation $s \in \tau$ to indicate that the execution trajectory τ crosses state s. Therefore, they seek to find policies that will avoid undesirable states with probability $1 - p$. Aside from using this criterion for MDPs and POMDPs, researchers have also used it for scheduling problems [Fang *et al.*, 2014; Yu *et al.*,

2015].

The *Optimal Cost Almost-Sure Reachability* criterion described above can also be viewed as an optimization with chance constraints: It is seeking a policy π that minimizes the expected cost and satisfying the chance constraint $Pr(s \in \tau \mid s \in \mathbf{G} \wedge \tau \in \mathcal{T}^{\pi}) \geq p$, except that $p = 1$. Thus, this constraint isn't strictly a "chance" constraint since it does not allow trajectories that violate the constraint.

7.1.3 Multi-Objective MDPs and POMDPs

One can view RS-MDPs and RS-POMDPs from the lens of multi-objective optimization, where two objectives are optimized – expected cost and reachable probability. In Multi-Objective MDPs and POMDPs [Roijers *et al.*, 2013], there are n, potentially competing, objective functions that must be taken into account. Typical methods weigh each objective i by a weight w_i. If the weights are defined a priori, then one can optimize for the weighted sum of all the objectives. If the weights are not known a priori but are known at runtime, then one can find the convex hull for all possible weights. Finally, the last approach finds solutions at the Pareto frontier, which may be fed to users to rank. Roijers *et al.* [2013] provides a detailed overview and characterization of all these methods.

Another recent approach proposed by Wray *et al.*; Wray and Zilberstein [2015; 2015] is for MDPs and POMDPs with multiple cost functions that must be optimized. However, instead of weights for each of these cost functions, they proposed a lexicographical ordering on the functions. Then, the goal is to first optimize for the first cost function, and allowing a certain slack in the optimal cost values after the optimal costs are found. Then, among all policies that are within the slack range, they optimize for the second cost function, and allowing a certain slack again. This process continues until all the objectives are optimized for.

In RS-MDPs and RS-POMDPs, we also consider two objectives: The probability of reaching a goal and the expected cost of doing so. Both of these measurements are encapsulated within the reachable probability definition. One can cast RS-MDPs and RS-POMDPs using a multi-objective optimization framework that is similar to the one above, by first optimizing for trajectories that reach the goal while allowing a slack on the range of cost thresholds $\theta_0 \geq \theta_0^*$, where θ_0^* is the actual user-defined cost threshold. Then, among all trajectories that are within the slack range, we only consider the trajectories whose costs are within θ_0^*. However, it is likely that this method will not be very efficient as there is a lot of search effort that is expanded but not needed (e.g., by searching for trajectories that have costs larger than θ_0^*), and it is more efficient to encapsulate both objectives and optimize for them together as is done by the current RS-MDP and RS-POMDP algorithms.

Finally, another related body of work are Constrained MDPs and POMDPs [Altman, 1999; Dolgov and Durfee, 2005; Poupart *et al.*, 2015], where the goal is to optimize for one of the objectives, while enforcing constraints on the expected value of other objectives. For example, it may seek to find a policy π that maximizes the reachable probability and satisfying the constraint that the expected cost of that policy is no larger than some user-defined threshold. To solve Constrained MDPs and POMDPs, one can no longer use standard MDP or POMDP algorithms as the optimal policy is now a stochastic policy instead of a deterministic one. Therefore, different variants of linear and non-linear programing methods are typically used to solve them [Altman, 1999; Dolgov and Durfee, 2005; Poupart *et al.*, 2015]. They can be used to model a number of applications. For instance, in spoken dialog systems [Williams and Young, 2007] the main objective may be to minimize the number of turns while ensuring that the probability of

completing a booking task is above some threshold. In mobile assistive technologies [Hoey *et al.*, 2012], the goal may be to maximize the probability of completing a task while bounding energy consumption. In wireless communications, the goal of opportunistic spectrum access [Zhao *et al.*, 2007] is to maximize the utilization of wireless spectrum by allowing multiple devices to use the same wireless channel while satisfying regulatory requirements on the maximum collision rate between devices.

While RS-MDPs and RS-POMDPs share similar motivations with Constrained MDPs and POMDPs, their key difference is on the thresholds required by the problems. Constrained MDPs and POMDPs require thresholds on the expected value (e.g., expected costs must be within some threshold) while RS-MDPs and POMDPs require thresholds on the exact value (e.g., actual accumulated costs must be within some threshold). In some applications, setting thresholds on the expected value is reasonable (e.g., the wait time for patients in a hospital should be on average 10 minutes). Therefore, Constrained MDPs and POMDPs can be applied in those applications. However, in some applications, setting thresholds on the actual value is more realistic (e.g., the amount of power used by a robot must be within the available battery charge) than thresholds on the expected value. Therefore, in such applications, RS-MDPs and RS-POMDPs are more applicable. In the event that the problem is deterministic, then, clearly, both models are equivalent.

7.1.4 Uncertain MDPs and POMDPs

The notion of risk in RS-MDPs and RS-POMDPs is due to the stochasticity in the cost, transition, and observation functions. However, RS-MDPs and RS-POMDPs assume that such functions are known accurately. However, in real-

world scenarios, such functions are usually approximated, through advice from domain experts and/or approximated from real-world observations. As such, there is a *different* notion of risk incurred when planning based on a model that is possibly inaccurate.

Uncertain MDPs and POMDPs model the uncertainty in these parameters explicitly and techniques for such models typically take these uncertainty factors into account. For example, researchers have taken *proactive approaches* to solve Uncertain MDPs by explicitly representing the cost and transition functions as fixed but unknown parameters. Some assume that the parameters are elements of a known bounded set, called the uncertainty set, and use robust optimization techniques to solve the Uncertain MDPs [Givan *et al.*, 2000; Iyengar, 2005; Nilim and Ghaoui, 2005; Regan and Boutilier, 2009; 2011; Mannor *et al.*, 2012]. Alternatively, some assume that the parameters are random variables that follow some distribution and use percentile optimization techniques [Delage and Mannor, 2010], distributional robustness [Xu and Mannor, 2012], or sampling-based approaches [Ahmed *et al.*, 2013] to solve the Uncertain MDPs. Many proactive Uncertain MDP and POMDP techniques are also based on minimax-like methods [Iyengar, 2005; Wiesemann *et al.*, 2013], where the idea is to find a policy with the minimum expected cost across all worst-case scenarios. In addition, there also exists some work that keep a baseline policy and directly minimize the regret with respect to the current policy [Ahmed *et al.*, 2013; Ghavamzadeh *et al.*, 2016].

In contrast, researchers have also proposed *reactive approaches* to solve the Uncertain MDPs without explicitly representing the uncertainty in the cost and transition functions [Hou *et al.*, 2014a]. Instead, they find a policy for their current (possibly incorrect) MDP model, execute it, and if inconsistencies between the

environment and the model are observed, then the model is updated and the agent searches for a new policy for the new model. This process repeats until the agent gets to its goal. The goal in such problems is typically to reuse as much information as possible between subsequent searches to speed up the search process. Researchers have also applied this reactive approach to deterministic planning problems, where they are called *incremental search techniques* [Stentz, 1995; Koenig and Likhachev, 2002; Koenig *et al.*, 2004; Likhachev *et al.*, 2003; Sun *et al.*, 2008; 2009; 2010a; 2010b]. Incremental search algorithms typically use A* [Hart *et al.*, 1968] to find a plan for the initial problem, and replans each time the problem changes.

Another category of reactive approaches is based on reinforcement learning [Sutton and Barto, 1998; Thomas, 2015], which is useful in problems where only partial information about the model is known (e.g., only the range of the transition and cost functions are known). By applying reinforcement learning based methods, more information on the model can be learned (e.g., the transition and cost functions can be made more accurate), and this new information can be used to find new and improved policies, while ensuring that the probability of bad policies being proposed is low. Note that the scenario where this approach is useful assumes that very little information is known about the underlying MDP model, even less than that assumed by Uncertain MDPs and the incremental approaches above.

7.1.5 Alternative Risk Criteria for MDPs and POMDPs

Aside from the four general models above, researchers have also correlated the risk of a policy with the variance in the distribution of cumulative cost of that policy, where a policy is riskier if it has a larger variance [Mannor and Tsitsiklis, 2011;

2013]. In such formulations, the goal can be to find a policy with the smallest expected cost and break ties in favor of policies with smaller cost variance, or to find a policy with the smallest expected cost among a set of policies whose cost variance is within some user-defined threshold.

Another criterion is one called *dynamically consistent risk measure* [Ruszczynski, 2010; Petrik and Subramanian, 2012], which uses the regular MDP/POMDP optimization criterion – minimizing expected cost – except that it penalizes some specific trajectories of the policy. Specifically, given a reference probability for each transition function, all adverse realizations (with probability greater than the reference probability) are penalized in the regular optimization criterion.

7.2 Continuous MDPs and POMDPs

When formalizing RS-MDPs and RS-POMDPs as augmented MDPs and POMDPs, the augmented state space is continuous. These MDPs and POMDPs are normally considered as Continuous MDPs and POMDPs [Guestrin *et al.*, 2004; Li and Littman, 2005; Marecki *et al.*, 2007; Sanner *et al.*, 2011; Zamani *et al.*, 2012]. Different from discrete MDPs and POMDPs, the continuous models assume that their components (i.e., states, actions, and both transition and cost/reward functions) may be continuous. There are many different Continuous MDP and POMDP variants, where each variant makes different assumptions on which component is continuous [Guestrin *et al.*, 2004; Li and Littman, 2005; Marecki *et al.*, 2007; Sanner *et al.*, 2011; Zamani *et al.*, 2012]. In general, the algorithm to solve Continuous MDPs depend on which components are assumed to be continuous [Guestrin *et al.*, 2004; Li and Littman, 2005; Marecki *et al.*, 2007; Sanner *et al.*, 2011; Zamani *et al.*, 2012].

It is usually clear from the name of the model which component is assumed to be continuous. For example, the Continuous State MDP model assumes that states are continuous. A simple and straightforward way to solve this problem approximately is to discretize the continuous state space into a discrete state space and use regular techniques for discrete MDPs to solve it. This method is thus similar to how point-based POMDP methods approximate exact POMDP methods. Alternatively, if the costs are continuous values, they too can be discretized into bins (e.g., by rounding them to the nearest integer). By carefully designing the rounding strategies based on the underlying problem structure, some times the complexity of solving some specific probabilistic problems can reduce from exponential to polynomial [Wu *et al.*, 2014].

A different way to solve the problem exactly is to use continuous functions to represent the value function of the continuous state space. Then, one can update these value functions using a modified Bellman equation, similar to how FVI updates its value functions. But since the continuous value functions can be of any arbitrary form (e.g., piecewise constant, piecewise linear, exponential functions, one-switch functions, skew symmetric bilinear functions, etc.), updating them exactly, in general, can be difficult. However, updating the value functions of specific types exactly may be significantly simpler and more efficient. For example, the RS-MDP algorithms we have proposed assume that the value functions are piecewise constant functions and our update procedures are tailored for that type of functions only. As such, Continuous State MDPs typically assume that their value functions have specific forms or can be approximated by specific forms. For example, Discrete and Continuous State MDPs (a model with both continuous and discrete states) assume that their value functions are hyper-rectangle piecewise linear functions [Sanner *et al.*, 2011].

As we mentioned above, as RS-MDPs and RS-POMDPs have continuous augmented state spaces, they can be viewed as one type of Continuous MDPs and POMDPs. Nonetheless, the number of states that need to be considered is finite because the cost function is discrete and the cost thresholds that need to be considered are only positive cost thresholds that are within the initial cost threshold. Therefore, we are able to adopt classical MDP and POMDP methods (e.g., DFS and DP) in addition to Continuous MDP and POMDP methods (e.g., FVI) to solve RS-MDPs and RS-POMDPs.

One category of Continuous MDPs and POMDPs that are closely related to RS-MDPs and POMDPs are MDPs and POMDPs with real-valued resources. Many MDP and POMDP models with limited resources have been proposed [Marecki *et al.*, 2007; Marecki and Tambe, 2009]. One such model includes finite-horizon MDPs, where the resource is the number of time steps. For finite-horizon problems, Li and Littman [2005] assumes that transition and cost functions are continuous and they apply Continuous MDP techniques to solve them. To achieve computing efficiency, they use piecewise constant functions to approximate the value functions after each iteration. Marecki *et al.* [2007] assumes that actions have associated durations and time is a limited resource. Instead of approximating the value functions as piecewise constant functions, they approximate the probability distributions of the action durations using phase-type distributions, which allow them to better approximate the overall value functions. Both of these models and RS-MDPs assume that there is a limited resource budget. But both of those models seek to maximize the accumulated reward under the limited resource budget, while RS-MDPs seek to maximize the probability of reaching a goal under the limited resource budget.

The other category of Continuous MDPs and POMDPs that are closely

related are Temporal MDP and POMDP models that are typically used to model scheduling problems. In these temporal models, actions have associated durations and time is part of the augmented state space. As such, the state space is continuous since time is continuous [Younes and Simmons, 2004; Little *et al.*, 2005; Mausam and Weld, 2008]. If the durations are deterministic, then these problems can be reduced to classical discrete MDPs and POMDPs and be solved using classical techniques [Younes and Simmons, 2004]. If the durations are stochastic, then they must be solved using Continuous MDP and POMDP techniques. Further, some more complex models even allow for concurrent actions to be taken at the same time [Little *et al.*, 2005; Mausam and Weld, 2008]. These temporal models are related to RS-MDPs and RS-POMDPs in that the augmented state space contains both the actual state element as well as a resource element. In the case of temporal models, that resource element is time, while in RS-MDPs and RS-POMDPs, that resource element is the cost budget. In both models, the resource is continuous, and, as a result, the augmented state space is continuous as well.

Chapter 8

Conclusions and Future Work

In this dissertation, we revisit the *Risk-Sensitive MDP* (RS-MDP) model and generalize it to *Risk-Sensitive POMDPs* (RS-POMDPs). Unlike conventional MDPs and POMDPs, the goal in these problems is to find a policy that maximizes the probability that an agent achieves its goal with an accumulated cost that is no larger than a predefined threshold. We showed that this specific risk-sensitive criterion can be viewed as a specific type of risk attitude. Most risk attitudes can be represented by utility functions, and our risk-sensitive criterion corresponds to a step function.

To solve the new models, we adapted and optimized the existing *Functional Value Iteration* (FVI), which can be applied on more general risk attitudes, to solve RS-MDPs and RS-POMDPs. Additionally, we show that one can naively represent RS-(PO)MDPs as augmented (PO)MDPs and solve them using the traditional *Value Iteration* (VI) algorithm on the augmented (belief) state space. Finally, we also introduce a number of new algorithms, based on *Depth First Search* (DFS) and *Dynamic Programming* (DP) techniques, that also operate on this augmented (belief) state space. For RS-MDPs with zero costs, we also combine the DFS- and

DP-based algorithms with *Topological Value Iteration* (TVI).

Additionally, we implemented the DFS- and DP-based algorithms to use two different kinds of data structures – latticed tables and utility functions – to store their reachable probabilities as there are inherent tradeoffs between the two. Latticed tables have a small worst-case runtime for the search and insert operations, which are both $O(1)$. However, its memory requirement to store the table is proportional to $O(|\mathbf{S}| \cdot |\mathbf{\Theta}|)$, where $\mathbf{S}$ is the original state space and $\mathbf{\Theta}$ is the set of all possible cost thresholds. In contrast, utility functions have a larger worst-case runtime for the search and insert operations, which are $O(\log(|\mathbf{\Theta}|))$ and $O(|\mathbf{\Theta}|)$, respectively. However, its memory requirement grows with the complexity of the utility function and, in many cases, will be smaller than the memory requirement to store latticed tables.

We empirically compare these algorithms on three domains: Randomly generated (PO)MDPs, the *Navigation* domain from the ICAPS 2011 *International Probabilistic Planning Competition* (IPPC), and a taxi domain generated from real-world data. Our experimental results for RS-MDPs show that either the new DFS- or DP-based algorithms are faster than VI and FVI. The exception is when utility functions are extremely simple, such as in the Navigation domain, where FVI is the fastest. When utility functions are simple, both DFS- and DP-based algorithms are typically faster when they use utility functions than when they use latticed tables. Finally, DFS is typically faster than DP when most of the state space is not reachable or when utility functions are simple.

In RS-POMDPs, most of the trends from RS-MDPs still apply. However, DFS and DP are not applicable in some RS-POMDP configurations, in which case we must rely on FVI or VI. As expected, FVI, VI, and DP are typically faster and more scalable when they are able to prune the belief state space. Their point-based

counterparts are also more scalable but at the cost of reduced solution quality. Finally, in this dissertation, we distinguish between the two cases of whether costs can or cannot be observed, and the algorithms typically find better solutions when they can observe the costs. This dissertation thus provides a comprehensive evaluation of these algorithms on a large number of RS-MDP and RS-POMDP configurations.

For future work, there are multiple promising directions that one can pursue. The first is the relationship between our *Risk-Sensitive* (RS) criterion with the *Minimizing Expected Cost* (MEC) criterion. For example, an interesting question is whether a theoretical error bound exists between the optimal reachable probability of a policy that optimizes the RS-criterion and the reachable probability of a policy that optimizes the MEC criterion. In general, a tight bound does not exist as it is possible to construct examples where the error is close to 1 (i.e., the reachable probability of the MEC policy is almost 0 while the optimal reachable probability is marginally less than 1). However, one may be able to exploit specific problem structures to obtain a tighter bound. Further, this line of research actually applies to the more general class of all risk attitudes discussed in Chapter 7, and not just restricted to the RS-criterion To the best of our knowledge, the relationships between the MEC criterion and these other risk attitudes are still open questions.

The second possible direction is to generalize our RS-MDP and RS-POMDP models to more general models. For example, a reasonably straightforward extension is to one where the utility function can have constant tails with arbitrary values. More formally,

$$U(w) = \begin{cases} u_\perp & \text{if } w < w_\perp \\ f(w) & \text{if } w \geq w_\perp \end{cases} \tag{8.1}$$

where $u_\perp$ and $w_\perp$ are constants and $f(\cdot)$ is one of any monotonically non-decreasing functions. Ermon *et al.* [2012] discusses one specific example where $u_\perp = \infty$. It is relatively straightforward to extend our algorithms to handle this case as well, for example, by rewriting Equation 4.4 to the following:

$$V(s,\theta) = \max_{a \in \mathbf{A}} \sum_{s' \in \mathbf{S}} \begin{cases} u_\perp & \text{if } \theta' < w_\perp \\ T(s,a,s') \cdot U(\theta') & \text{if } s' \in \mathbf{G},\ \theta' \geq w_\perp \\ T(s,a,s') \cdot V(s',\theta') & \text{if } s' \notin \mathbf{G},\ \theta' \geq w_\perp \end{cases} \tag{8.2}$$

where $\theta' = \theta - C(s,a,s')$. $V(s,\theta)$ now denote the expected utility with respect to the utility function $U(w)$ with a constant tail.

Another interesting extension is one where the model *can* have negative costs but there is a negative bound on the accumulated cost of the agent, similar to the model proposed by Brázdil *et al.* [2016]. This extension may readily apply in many real-world applications. For example, in robotic navigation problems, the initial cost threshold may correspond to the initial battery level of the robot, which is drained when the robot takes actions with positive costs. However, the robot may charge its battery, which corresponds to it taking an action with a negative cost. Further, there is a capacity on the amount of charge the battery can carry; hence the negative bound on the accumulated cost. This constraint is important as we have shown that solving general RS-MDPs with negative costs is undecidable (Theorem 1).

Finally, one may cast the RS-criterion as one of the other risk attitudes discussed in Chapter 7. For example, one may view the RS-criterion as a bi-objective optimization problem with a lexicographic ordering of the objectives [Wray *et al.*, 2015; Wray and Zilberstein, 2015]. Then, the objective of satisfying the cost threshold is more important than the objective of maximizing the reachable probability. Further, it may also be possible to combine the RS-criterion with other

orthogonal risk criteria, such as the mean-variance criterion.

Chapter 9

Appendix

The tables below are for the three RS-POMDP configurations not reported in the RS-POMDP experimental results (see Section 5.4).

(a) FVI

$\|\mathbf{S}\|$	w/o Pruning			w/ Pruning			PB(10)			PB(100)			PB(1000)		
	%	t	P	%	t	P	%	t	P	%	t	P	%	t	P
25	10	8	2.84e-1	30	3292	2.74e-1	100	1	2.87e-1	100	12	2.88e-1	100	398	2.88e-1
50	8	42	1.91e-1	16	15865	1.79e-1	100	1	1.79e-1	100	37	2.13e-1	100	1397	2.13e-1
100	4	19	1.20e-1	10	103965	9.91e-2	100	1	1.32e-1	100	175	1.56e-1	100	3934	1.56e-1
200	0	–	7.42e-2	0	–	4.36e-2	100	10	8.58e-2	100	1694	1.13e-1	100	34996	1.15e-1

θ_0	w/o Pruning			w/ Pruning			PB(10)			PB(100)			PB(1000)		
	%	t	P	%	t	P	%	t	P	%	t	P	%	t	P
$1.25 \cdot \mathcal{C}_d^*$	16	42	1.77e-1	28	10552	1.70e-1	100	1	1.75e-1	100	14	1.89e-1	100	683	1.89e-1
$1.50 \cdot \mathcal{C}_d^*$	14	169	2.09e-1	20	6048	1.87e-1	100	1	2.17e-1	100	55	2.29e-1	100	1226	2.30e-1
$2.00 \cdot \mathcal{C}_d^*$	8	5	2.40e-1	10	315	2.00e-1	100	15	2.69e-1	100	1621	2.85e-1	94	49990	2.85e-1
$3.00 \cdot \mathcal{C}_d^*$	2	1	2.48e-1	8	32364	1.97e-1	100	137	3.32e-1	92	31228	3.60e-1	50	116641	3.35e-1

(b) AUG-VI

$\|\mathbf{S}\|$	w/o Pruning			w/ Pruning			PB(10)			PB(100)			PB(1000)		
	%	t	P	%	t	P	%	t	P	%	t	P	%	t	P
25	10	79	2.84e-1	22	1629	2.71e-1	100	29	2.87e-1	100	347	2.88e-1	100	9639	2.88e-1
50	8	187	1.82e-1	8	52773	1.78e-1	100	56	1.79e-1	100	1414	2.13e-1	98	35561	2.13e-1
100	4	223	9.44e-2	6	1170	9.91e-2	100	328	1.32e-1	100	5687	1.56e-1	98	131322	1.56e-1
200	0	–	3.57e-2	0	–	4.17e-2	100	1148	8.65e-2	100	35217	1.13e-1	76	199009	1.14e-1

θ_0	w/o Pruning			w/ Pruning			PB(10)			PB(100)			PB(1000)		
	%	t	P	%	t	P	%	t	P	%	t	P	%	t	P
$1.25 \cdot \mathcal{C}_d^*$	16	211	1.70e-1	22	2775	1.66e-1	100	28	1.75e-1	100	582	1.89e-1	100	25078	1.89e-1
$1.50 \cdot \mathcal{C}_d^*$	14	611	2.01e-1	14	26163	1.85e-1	100	72	2.17e-1	100	1795	2.29e-1	100	43377	2.30e-1
$2.00 \cdot \mathcal{C}_d^*$	8	117	2.17e-1	10	4115	1.95e-1	100	288	2.69e-1	100	22252	2.85e-1	76	74808	2.81e-1
$3.00 \cdot \mathcal{C}_d^*$	2	1	2.05e-1	2	125	1.97e-1	100	937	3.32e-1	88	106480	3.60e-1	30	130739	3.06e-1

(c) DFS

$\|\mathbf{S}\|$	%	t	P
25	88	18626	2.57e-1
50	84	3945	1.82e-1
100	90	32294	1.48e-1
200	76	52754	9.24e-2

θ_0	%	t	P
$1.25 \cdot \mathcal{C}_d^*$	96	1439	1.83e-1
$1.50 \cdot \mathcal{C}_d^*$	88	19899	2.05e-1
$2.00 \cdot \mathcal{C}_d^*$	64	32184	1.85e-1
$3.00 \cdot \mathcal{C}_d^*$	32	113983	1.20e-1

Table 9.1: RS-POMDP Results of Randomly Generated POMDPs without Zero Costs and Unobservable Actual Costs

(a) FVI

$\|\mathbf{S}\|$	w/o Pruning			w/ Pruning			PB(10)			PB(100)			PB(1000)		
	%	t	P	%	t	P	%	t	P	%	t	P	%	t	P
25	6	120184	1.90e-1	16	46121	2.03e-1	100	47	2.68e-1	100	6524	2.93e-1	96	5132	2.95e-1
50	6	6	1.35e-1	10	4320	1.38e-1	100	62	1.65e-1	100	2746	1.92e-1	92	48092	1.95e-1
100	0	–	5.58e-2	4	90858	5.58e-2	100	75	7.93e-2	100	3098	9.53e-2	90	68605	9.89e-2
200	0	–	4.64e-3	2	3683	5.76e-3	100	131	1.33e-2	100	4728	2.17e-2	80	102246	2.79e-2

θ_0	w/o Pruning			w/ Pruning			PB(10)			PB(100)			PB(1000)		
	%	t	P	%	t	P	%	t	P	%	t	P	%	t	P
$1.25{\cdot}\mathcal{C}_d^*$	8	187	5.51e-2	16	29336	6.22e-2	100	24	6.88e-2	100	1259	8.43e-2	96	17896	8.54e-2
$1.50{\cdot}\mathcal{C}_d^*$	6	244	5.53e-2	16	36710	6.90e-2	100	83	8.53e-2	100	5416	1.11e-1	88	24080	1.13e-1
$2.00{\cdot}\mathcal{C}_d^*$	4	351	5.51e-2	16	36841	6.43e-2	100	3419	1.13e-1	100	16842	1.38e-1	78	40443	1.47e-1
$3.00{\cdot}\mathcal{C}_d^*$	4	367	5.51e-2	14	33672	6.43e-2	100	4995	1.63e-1	92	66782	2.15e-1	60	66226	2.24e-1

(b) AUG-VI

$\|\mathbf{S}\|$	w/o Pruning			w/ Pruning			PB(10)			PB(100)			PB(1000)		
	%	t	P	%	t	P	%	t	P	%	t	P	%	t	P
25	6	70127	1.90e-1	20	65395	2.36e-1	100	198	2.68e-1	100	6923	2.93e-1	96	2793	2.95e-1
50	6	9	1.35e-1	10	4528	1.47e-1	100	235	1.65e-1	100	5460	1.92e-1	100	42916	1.95e-1
100	0	–	5.58e-2	6	120378	5.61e-2	100	575	7.93e-2	100	4022	9.53e-2	96	48246	9.89e-2
200	0	–	4.64e-3	2	3352	8.27e-3	100	334	1.33e-2	100	3953	2.17e-2	86	75031	2.79e-2

θ_0	w/o Pruning			w/ Pruning			PB(10)			PB(100)			PB(1000)		
	%	t	P	%	t	P	%	t	P	%	t	P	%	t	P
$1.25{\cdot}\mathcal{C}_d^*$	8	211	5.51e-2	22	69462	6.45e-2	100	131	6.88e-2	100	2027	8.43e-2	100	15969	8.54e-2
$1.50{\cdot}\mathcal{C}_d^*$	6	280	5.53e-2	20	59783	6.91e-2	100	512	8.53e-2	98	6487	1.11e-1	96	36666	1.13e-1
$2.00{\cdot}\mathcal{C}_d^*$	4	421	5.51e-2	20	60944	6.56e-2	98	615	1.13e-1	98	16221	1.38e-1	84	37581	1.47e-1
$3.00{\cdot}\mathcal{C}_d^*$	4	413	5.35e-2	20	125149	6.44e-2	98	3884	1.63e-1	88	27666	2.15e-1	64	33770	2.27e-1

(c) DP

$\|\mathbf{S}\|$	w/o Pruning			w/ Pruning			PB(10)			PB(100)			PB(1000)		
	%	t	P	%	t	P	%	t	P	%	t	P	%	t	P
25	6	3979	7.08e-2	20	104172	1.01e-1	100	67	1.49e-1	100	1486	1.50e-1	92	26478	1.44e-1
50	6	1	9.61e-2	12	497	1.10e-1	100	2380	1.15e-1	100	6423	1.26e-1	80	60790	1.27e-1
100	0	–	4.08e-2	6	11283	4.16e-2	100	34	5.72e-2	98	16076	5.95e-2	68	140716	5.77e-2
200	0	–	1.01e-3	6	76513	7.45e-3	100	51	6.95e-3	98	22090	7.14e-3	57	35683	7.16e-3

θ_0	w/o Pruning			w/ Pruning			PB(10)			PB(100)			PB(1000)		
	%	t	P	%	t	P	%	t	P	%	t	P	%	t	P
$1.25{\cdot}\mathcal{C}_d^*$	8	4	4.60e-2	24	13516	5.71e-2	100	33	4.03e-2	100	9962	4.94e-2	84	41731	4.91e-2
$1.50{\cdot}\mathcal{C}_d^*$	6	5	4.63e-2	20	15274	5.74e-2	100	47	4.77e-2	100	13351	5.69e-2	82	66476	5.66e-2
$2.00{\cdot}\mathcal{C}_d^*$	4	8	4.63e-2	20	16114	5.74e-2	100	86	4.85e-2	100	28837	5.93e-2	72	62545	5.70e-2
$3.00{\cdot}\mathcal{C}_d^*$	4	8	4.63e-2	20	18313	5.74e-2	100	158	5.46e-2	94	27790	6.71e-2	64	36505	5.70e-2

Table 9.2: RS-POMDP Results of Randomly Generated POMDPs with Zero Costs and Observable Actual Costs

(a) FVI

$\|\mathbf{S}\|$	w/o Pruning			w/ Pruning			PB(10)			PB(100)			PB(1000)		
	%	t	P	%	t	P	%	t	P	%	t	P	%	t	P
25	6	14	2.58e-1	16	2616	2.61e-1	100	3	2.83e-1	100	196	2.87e-1	94	2949	2.87e-1
50	6	1	1.61e-1	14	4517	1.59e-1	100	169	1.67e-1	100	3687	1.91e-1	88	30156	1.92e-1
100	0	–	8.06e-2	4	2278	7.69e-2	100	33	8.12e-2	100	2255	9.54e-2	88	43629	9.76e-2
200	0	–	1.52e-2	2	165	1.32e-2	100	9	1.32e-2	100	1953	2.68e-2	84	45734	2.81e-2

θ_0	w/o Pruning			w/ Pruning			PB(10)			PB(100)			PB(1000)		
	%	t	P	%	t	P	%	t	P	%	t	P	%	t	P
$1.25 \cdot \mathcal{C}_d^*$	8	8	7.01e-2	24	11662	7.25e-2	100	6	6.92e-2	98	1282	8.32e-2	96	16609	8.42e-2
$1.50 \cdot \mathcal{C}_d^*$	6	10	8.78e-2	22	7212	8.96e-2	100	5391	8.67e-2	100	2906	1.08e-1	88	25346	1.09e-1
$2.00 \cdot \mathcal{C}_d^*$	4	8	1.08e-1	20	6972	9.98e-2	100	4851	1.13e-1	100	27766	1.36e-1	76	44858	1.36e-1
$3.00 \cdot \mathcal{C}_d^*$	4	8	1.36e-1	16	6065	1.16e-1	100	3461	1.75e-1	86	23231	2.01e-1	56	92163	1.96e-1

(b) AUG-VI

$\|\mathbf{S}\|$	w/o Pruning			w/ Pruning			PB(10)			PB(100)			PB(1000)		
	%	t	P	%	t	P	%	t	P	%	t	P	%	t	P
25	6	15	2.58e-1	12	5773	2.59e-1	100	40	2.83e-1	100	1515	2.87e-1	94	29371	2.87e-1
50	6	5	1.61e-1	10	439	1.56e-1	100	607	1.67e-1	98	5525	1.91e-1	82	52868	1.91e-1
100	0	–	7.84e-2	2	7663	7.66e-2	100	288	8.12e-2	100	20320	9.54e-2	74	87634	9.75e-2
200	0	–	1.11e-2	2	168	1.23e-2	100	86	1.32e-2	100	15272	2.67e-2	70	119213	2.80e-2

θ_0	w/o Pruning			w/ Pruning			PB(10)			PB(100)			PB(1000)		
	%	t	P	%	t	P	%	t	P	%	t	P	%	t	P
$1.25 \cdot \mathcal{C}_d^*$	8	4	7.01e-2	16	1486	7.10e-2	100	65	6.92e-2	98	10166	8.32e-2	90	43651	8.42e-2
$1.50 \cdot \mathcal{C}_d^*$	6	10	8.78e-2	16	5337	8.77e-2	98	461	8.67e-2	98	7397	1.08e-1	80	37693	1.09e-1
$2.00 \cdot \mathcal{C}_d^*$	4	7	1.08e-1	16	5421	9.93e-2	98	277	1.13e-1	96	38842	1.36e-1	60	47789	1.36e-1
$3.00 \cdot \mathcal{C}_d^*$	4	8	1.26e-1	14	3301	1.20e-1	98	3377	1.75e-1	84	54605	2.01e-1	42	48517	1.86e-1

Table 9.3: RS-POMDP Results of Randomly Generated POMDPs with Zero Costs and Unobservable Actual Costs

(a) FVI

θ_0	w/o Pruning			w/ Pruning			PB(10)			PB(100)			PB(1000)		
	%	t	P	%	t	P	%	t	P	%	t	P	%	t	P
$1.25 \cdot \mathcal{C}_d^*$	20	1	7.84e-3	70	14154	1.19e-2	100	1	2.16e-4	100	13	4.61e-3	100	653	1.19e-2
$1.50 \cdot \mathcal{C}_d^*$	20	82	2.49e-2	60	9246	6.37e-2	100	1	2.16e-4	100	9	5.06e-3	100	832	6.33e-2
$2.00 \cdot \mathcal{C}_d^*$	10	2	2.51e-2	50	24341	9.77e-2	100	1	3.28e-4	100	16	6.11e-3	100	3044	9.09e-2
$3.00 \cdot \mathcal{C}_d^*$	10	16	2.52e-2	10	102	1.24e-1	100	1	5.44e-4	100	62	1.41e-1	100	14210	2.07e-1

(b) AUG-VI

θ_0	w/o Pruning			w/ Pruning			PB(10)			PB(100)			PB(1000)		
	%	t	P	%	t	P	%	t	P	%	t	P	%	t	P
$1.25 \cdot \mathcal{C}_d^*$	90	79935	1.19e-2	50	1158	1.19e-2	100	14	2.16e-4	100	348	4.61e-3	100	36562	1.19e-2
$1.50 \cdot \mathcal{C}_d^*$	70	145253	6.34e-2	60	8325	6.37e-2	100	17	2.16e-4	100	430	5.06e-3	100	45524	6.33e-2
$2.00 \cdot \mathcal{C}_d^*$	30	28526	8.17e-2	20	34555	9.74e-2	100	25	3.28e-4	100	699	6.17e-3	90	77027	9.09e-2
$3.00 \cdot \mathcal{C}_d^*$	10	60	7.72e-2	20	76501	1.24e-1	100	43	5.44e-4	100	2995	1.41e-1	80	105526	2.07e-1

(c) DFS

θ_0	%	t	P
$1.25 \cdot \mathcal{C}_d^*$	80	69471	5.36e-3
$1.50 \cdot \mathcal{C}_d^*$	70	50456	5.05e-2
$2.00 \cdot \mathcal{C}_d^*$	50	8043	7.48e-2
$3.00 \cdot \mathcal{C}_d^*$	40	87771	1.47e-1

Table 9.4: RS-POMDP Results of Navigation Domain without Zero Costs and Unobservable Actual Costs

(a) FVI

θ_0	w/o Pruning			w/ Pruning			PB(10)			PB(100)			PB(1000)		
	%	t	P	%	t	P	%	t	P	%	t	P	%	t	P
$1.25 \cdot \mathcal{C}_d^*$	10	3	5.57e-3	70	117457	5.58e-3	100	10	0.00e+0	100	15	2.82e-3	100	1294	3.80e-3
$1.50 \cdot \mathcal{C}_d^*$	10	3	5.62e-3	40	23103	6.67e-3	100	1	0.00e+0	100	19	5.65e-3	100	1490	6.64e-3
$2.00 \cdot \mathcal{C}_d^*$	10	11	5.99e-3	30	81527	3.18e-2	100	1	0.00e+0	100	12	6.67e-3	100	2053	1.67e-2
$3.00 \cdot \mathcal{C}_d^*$	10	22597	5.30e-3	10	71137	4.05e-2	100	1	0.00e+0	100	18	1.09e-2	100	4417	9.36e-2

(b) AUG-VI

θ_0	w/o Pruning			w/ Pruning			PB(10)			PB(100)			PB(1000)		
	%	t	P	%	t	P	%	t	P	%	t	P	%	t	P
$1.25 \cdot \mathcal{C}_d^*$	20	709	5.58e-3	50	31566	5.57e-3	100	25	0.00e+0	100	40	2.82e-3	100	1281	3.80e-3
$1.50 \cdot \mathcal{C}_d^*$	10	11	6.67e-3	40	139532	6.64e-3	100	36	0.00e+0	100	64	5.65e-3	100	1470	6.64e-3
$2.00 \cdot \mathcal{C}_d^*$	10	59	3.26e-2	30	148047	3.16e-2	100	22	0.00e+0	100	111	6.67e-3	100	2337	1.67e-2
$3.00 \cdot \mathcal{C}_d^*$	10	469	1.48e-2	10	72848	3.93e-2	100	49	0.00e+0	100	199	1.09e-2	100	5508	9.36e-2

(c) DP

θ_0	w/o Pruning			w/ Pruning			PB(10)			PB(100)			PB(1000)		
	%	t	P	%	t	P	%	t	P	%	t	P	%	t	P
$1.25 \cdot \mathcal{C}_d^*$	20	12	2.15e-3	60	118259	5.10e-3	100	20	1.03e-3	100	302	1.03e-3	100	4321	1.03e-3
$1.50 \cdot \mathcal{C}_d^*$	20	195	3.23e-3	50	132747	5.10e-3	100	13	1.03e-3	100	458	1.03e-3	100	6300	1.03e-3
$2.00 \cdot \mathcal{C}_d^*$	10	3	4.08e-3	30	10640	6.73e-3	100	17	1.03e-3	100	764	1.03e-3	100	10589	1.03e-3
$3.00 \cdot \mathcal{C}_d^*$	10	11	4.08e-3	20	9668	7.25e-3	100	45	1.03e-3	100	1453	1.03e-3	100	19877	1.03e-3

Table 9.5: RS-POMDP Results of Navigation Domain with Zero Costs and Observable Actual Costs

(a) FVI

θ_0	w/o Pruning			w/ Pruning			PB(10)			PB(100)			PB(1000)		
	%	t	P	%	t	P	%	t	P	%	t	P	%	t	P
$1.25 \cdot \mathcal{C}_d^*$	20	238981	5.57e-3	80	2548	7.38e-3	100	1	1.03e-3	100	16	5.57e-3	100	3402	5.57e-3
$1.50 \cdot \mathcal{C}_d^*$	10	1	5.88e-3	60	2539	8.54e-3	100	1	2.78e-3	100	24	5.69e-3	100	13175	6.68e-3
$2.00 \cdot \mathcal{C}_d^*$	10	3	3.07e-2	50	4338	4.37e-2	100	1	1.03e-3	100	22	6.72e-3	100	71777	4.09e-2
$3.00 \cdot \mathcal{C}_d^*$	10	2019	1.40e-2	20	2823	8.97e-2	100	1	0.00e+0	100	1834	8.42e-2	80	93573	1.21e-1

(b) AUG-VI

θ_0	w/o Pruning			w/ Pruning			PB(10)			PB(100)			PB(1000)		
	%	t	P	%	t	P	%	t	P	%	t	P	%	t	P
$1.25 \cdot \mathcal{C}_d^*$	20	74	5.57e-3	60	3354	6.91e-3	100	6	1.03e-3	100	205	5.57e-3	100	54782	5.57e-3
$1.50 \cdot \mathcal{C}_d^*$	20	4534	6.88e-3	60	3145	8.22e-3	100	12	2.78e-3	100	289	6.67e-3	90	30683	6.68e-3
$2.00 \cdot \mathcal{C}_d^*$	10	6	3.52e-2	60	14210	4.29e-2	100	10	1.03e-3	100	443	6.72e-3	90	38749	4.09e-2
$3.00 \cdot \mathcal{C}_d^*$	10	45	7.49e-2	20	3228	8.97e-2	100	16	0.00e+0	100	48418	8.42e-2	50	76410	1.18e-1

Table 9.6: RS-POMDP Results of Navigation Domain with Zero Costs and Unobservable Actual Costs

(a) FVI

θ_0	w/o Pruning			w/ Pruning			PB(10)			PB(100)			PB(1000)		
	%	t	P	%	t	P	%	t	P	%	t	P	%	t	P
$1.25 \cdot \mathcal{C}_d^*$	0	–	0.00e+0	0	–	0.00e+0	100	10	0.00e+0	100	652	2.38e-5	100	18254	1.98e-4
$1.50 \cdot \mathcal{C}_d^*$	0	–	0.00e+0	0	–	0.00e+0	100	10	0.00e+0	100	164	0.00e+0	100	42271	2.26e-2
$2.00 \cdot \mathcal{C}_d^*$	0	–	0.00e+0	0	–	0.00e+0	100	78	1.25e-1	100	7742	6.02e-1	100	60106	6.19e-1
$3.00 \cdot \mathcal{C}_d^*$	0	–	0.00e+0	0	–	0.00e+0	100	183	1.00e+0	100	1375	1.00e+0	100	19474	1.00e+0

(b) AUG-VI

θ_0	w/o Pruning			w/ Pruning			PB(10)			PB(100)			PB(1000)		
	%	t	P	%	t	P	%	t	P	%	t	P	%	t	P
$1.25 \cdot \mathcal{C}_d^*$	0	–	0.00e+0	0	–	0.00e+0	100	109	0.00e+0	100	4930	1.98e-4	100	207715	1.98e-4
$1.50 \cdot \mathcal{C}_d^*$	0	–	0.00e+0	0	–	0.00e+0	100	1406	1.72e-2	100	13190	2.26e-2	100	307616	2.26e-2
$2.00 \cdot \mathcal{C}_d^*$	0	–	0.00e+0	0	–	0.00e+0	100	4612	6.13e-1	100	24511	6.17e-1	62	413842	6.18e-1
$3.00 \cdot \mathcal{C}_d^*$	0	–	0.00e+0	0	–	0.00e+0	100	2467	1.00e+0	100	16316	1.00e+0	87	498494	8.75e-1

(c) DFS

θ_0	%	t	P
$1.25 \cdot \mathcal{C}_d^*$	100	28505	1.98e-4
$1.50 \cdot \mathcal{C}_d^*$	100	128973	2.26e-2
$2.00 \cdot \mathcal{C}_d^*$	100	392193	6.20e-1
$3.00 \cdot \mathcal{C}_d^*$	100	429201	1.00e+0

Table 9.7: RS-POMDP Results of Taxi Domain without Zero Costs and Unobservable Actual Costs

(a) FVI

θ_0	w/o Pruning			w/ Pruning			PB(10)			PB(100)			PB(1000)		
	%	t	P	%	t	P	%	t	P	%	t	P	%	t	P
$1.25 \cdot \mathcal{C}_d^*$	0	–	0.00e+0	0	–	0.00e+0	100	25	0.00e+0	100	396	1.32e-5	100	22001	1.07e-4
$1.50 \cdot \mathcal{C}_d^*$	0	–	0.00e+0	0	–	0.00e+0	100	9	0.00e+0	100	870	4.45e-3	100	137277	1.57e-2
$2.00 \cdot \mathcal{C}_d^*$	0	–	0.00e+0	0	–	0.00e+0	100	97	1.25e-1	100	9828	4.12e-1	75	420595	5.91e-1
$3.00 \cdot \mathcal{C}_d^*$	0	–	0.00e+0	0	–	0.00e+0	100	215	1.00e+0	100	1980	1.00e+0	100	31100	1.00e+0

(b) AUG-VI

θ_0	w/o Pruning			w/ Pruning			PB(10)			PB(100)			PB(1000)		
	%	t	P	%	t	P	%	t	P	%	t	P	%	t	P
$1.25 \cdot \mathcal{C}_d^*$	0	–	0.00e+0	0	–	0.00e+0	100	88	0.00e+0	100	683	1.32e-5	100	16353	1.07e-4
$1.50 \cdot \mathcal{C}_d^*$	0	–	0.00e+0	0	–	0.00e+0	100	147	0.00e+0	100	1502	4.45e-3	100	62261	1.57e-2
$2.00 \cdot \mathcal{C}_d^*$	0	–	0.00e+0	0	–	0.00e+0	100	708	1.25e-1	100	13224	4.12e-1	100	142321	5.91e-1
$3.00 \cdot \mathcal{C}_d^*$	0	–	0.00e+0	0	–	0.00e+0	100	1735	1.00e+0	100	3827	1.00e+0	100	21601	1.00e+0

(c) DP

θ_0	w/o Pruning			w/ Pruning			PB(10)			PB(100)			PB(1000)		
	%	t	P	%	t	P	%	t	P	%	t	P	%	t	P
$1.25 \cdot \mathcal{C}_d^*$	0	–	0.00e+0	50	299073	1.10e-4	100	48	0.00e+0	100	92	0.00e+0	100	18045	0.00e+0
$1.50 \cdot \mathcal{C}_d^*$	0	–	0.00e+0	0	–	2.18e-4	100	77	0.00e+0	100	115	0.00e+0	100	18113	0.00e+0
$2.00 \cdot \mathcal{C}_d^*$	0	–	0.00e+0	0	–	1.51e-4	100	112	0.00e+0	100	154	0.00e+0	100	18039	0.00e+0
$3.00 \cdot \mathcal{C}_d^*$	0	–	0.00e+0	0	–	1.51e-4	100	204	0.00e+0	100	241	0.00e+0	100	18219	0.00e+0

Table 9.8: RS-POMDP Results of Taxi Domain with Zero Costs and Observable Actual Costs

(a) FVI

θ_0	w/o Pruning			w/ Pruning			PB(10)			PB(100)			PB(1000)		
	%	t	P	%	t	P	%	t	P	%	t	P	%	t	P
$1.25 \cdot \mathcal{C}_d^*$	0	–	0.00e+0	0	–	0.00e+0	100	7	0.00e+0	100	545	1.91e-5	100	17917	1.39e-4
$1.50 \cdot \mathcal{C}_d^*$	0	–	0.00e+0	0	–	0.00e+0	100	9	0.00e+0	100	169	0.00e+0	100	37379	1.59e-2
$2.00 \cdot \mathcal{C}_d^*$	0	–	0.00e+0	0	–	0.00e+0	100	93	1.25e-1	100	4626	4.83e-1	100	52932	5.94e-1
$3.00 \cdot \mathcal{C}_d^*$	0	–	0.00e+0	0	–	0.00e+0	100	156	1.00e+0	100	1367	1.00e+0	100	19380	1.00e+0

(b) AUG-VI

θ_0	w/o Pruning			w/ Pruning			PB(10)			PB(100)			PB(1000)		
	%	t	P	%	t	P	%	t	P	%	t	P	%	t	P
$1.25 \cdot \mathcal{C}_d^*$	0	–	0.00e+0	0	–	0.00e+0	100	104	0.00e+0	100	4651	1.39e-4	100	197520	1.39e-4
$1.50 \cdot \mathcal{C}_d^*$	0	–	0.00e+0	0	–	0.00e+0	100	540	5.04e-3	100	13350	1.56e-2	100	293996	1.55e-2
$2.00 \cdot \mathcal{C}_d^*$	0	–	0.00e+0	0	–	0.00e+0	100	4128	5.88e-1	100	22141	5.89e-1	75	412387	5.90e-1
$3.00 \cdot \mathcal{C}_d^*$	0	–	0.00e+0	0	–	0.00e+0	100	1671	1.00e+0	100	16032	1.00e+0	87	496325	1.00e+0

Table 9.9: RS-POMDP Results of Taxi Domain with Zero Costs and Unobservable Actual Costs

Bibliography

[Ahmed *et al.*, 2013] Asrar Ahmed, Pradeep Varakantham, Yossiri Adulyasak, and Patrick Jaillet. Regret based robust solutions for uncertain Markov decision processes. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 881–889, 2013.

[Altman, 1999] Eitan Altman. *Constrained Markov Decision Processes*. Chapman & Hall/CRC, 1999.

[Bellman, 1957] Richard Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[Bonet and Geffner, 2009] Blai Bonet and Hector Geffner. Solving POMDPs: RTDP-Bel vs. point-based algorithms. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1641–1646, 2009.

[Brázdil *et al.*, 2016] Tomás Brázdil, Krishnendu Chatterjee, Martin Chmelik, Anchit Gupta, and Petr Novotný. Stochastic shortest path with energy constraints in POMDPs (extended abstract). In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1465–1466, 2016.

[Chatterjee *et al.*, 2016] Krishnendu Chatterjee, Martin Chmelik, Raghav Gupta, and Ayush Kanodia. Optimal cost almost-sure reachability in POMDPs. *Artificial Intelligence*, 234:26–48, 2016.

[Dai *et al.*, 2011] Peng Dai, Mausam, Daniel Weld, and Judy Goldsmith. Topological value iteration algorithms. *Journal of Artificial Intelligence*, 42(1):181–209, 2011.

[de Rodrigues Quemel e Assis Santana *et al.*, 2016] Pedro Henrique de Rodrigues Quemel e Assis Santana, Sylvie Thiébaux, and Brian Charles Williams. RAO*: An algorithm for chance-constrained POMDP's. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 3308–3314, 2016.

[Defourny *et al.*, 2008] Boris Defourny, Damien Ernst, and Louis Wehenkel. Risk-aware decision making and dynamic programming. In *NIPS 2008 Workshop on Model Uncertainty and Risk in Reinforcement Learning*, 2008.

[Delage and Mannor, 2010] Erick Delage and Shie Mannor. Percentile optimization for Markov decision processes with parameter uncertainty. *Operations Research*, 58(1):203–213, 2010.

[Dolgov and Durfee, 2005] Dmitri A. Dolgov and Edmund H. Durfee. Stationary deterministic policies for constrained MDPs with multiple rewards, costs, and discount factors. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1326–1331, 2005.

[Ermon *et al.*, 2012] Stefano Ermon, Carla P. Gomes, Bart Selman, and Alexander Vladimirsky. Probabilistic planning with non-linear utility functions and worst-case guarantees. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 965–972, 2012.

[Fang *et al.*, 2014] Cheng Fang, Peng Yu, and Brian Charles Williams. Chance-constrained probabilistic simple temporal problems. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 2264–2270, 2014.

[Geffner and Bonet, 2013] Hector Geffner and Blai Bonet. *A Concise Introduction to Models and Methods for Automated Planning*. Morgan & Claypool Publishers, 2013.

[Ghavamzadeh *et al.*, 2016] Mohammad Ghavamzadeh, Marek Petrik, and Yinlam Chow. Safe policy improvement by minimizing robust baseline regret. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 2298–2306, 2016.

[Gilbert *et al.*, 2015] Hugo Gilbert, Olivier Spanjaard, Paolo Viappiani, and Paul Weng. Solving MDPs with skew symmetric bilinear utility functions. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1989–1995, 2015.

[Givan *et al.*, 2000] Robert Givan, Sonia Leach, and Thomas Dean. Bounded-parameter Markov decision processes. *Artificial Intelligence*, 122(1–2):71–109, 2000.

[Guestrin *et al.*, 2004] Carlos Guestrin, Milos Hauskrecht, and Branislav Kveton. Solving factored MDPs with continuous and discrete variables. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 235–242, 2004.

[Hart *et al.*, 1968] Paul Hart, Nils Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, SSC4(2):100–107, 1968.

[Hoey *et al.*, 2012] Jesse Hoey, Xiao Yang, Eduardo Quintana, and Jesús Favela. Lacasa: Location and context-aware safety assistant. In *Proceedings of International Conference on Pervasive Computing Technologies for Healthcare*, pages 171–174, 2012.

[Hou *et al.*, 2014a] Ping Hou, William Yeoh, and Tran Cao Son. Solving uncertain MDPs by reusing state information and plans. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 2285–2292, 2014.

[Hou *et al.*, 2014b] Ping Hou, William Yeoh, and Pradeep Varakantham. Revisiting risk-sensitive MDPs: New algorithms and results. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pages 136–144, 2014.

[Hou *et al.*, 2016] Ping Hou, William Yeoh, and Pradeep Varakantham. Solving risk-sensitive POMDPs with and without cost observations. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 3138–3144, 2016.

[Iyengar, 2005] Garud Iyengar. Robust dynamic programming. *Mathematics of Operations Research*, 30:257–280, 2005.

[Kaelbling *et al.*, 1998] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134, 1998.

[Koenig and Likhachev, 2002] Sven Koenig and Maxim Likhachev. D* Lite. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 476–483, 2002.

[Koenig *et al.*, 2004] Sven Koenig, Maxim Likhachev, Yaxin Liu, and David Furcy. Incremental heuristic search in AI. *AI Magazine*, 25(2):99–112, 2004.

[Kolobov *et al.*, 2011] Andrey Kolobov, Mausam, Daniel S. Weld, and Hector Geffner. Heuristic search for generalized stochastic shortest path MDPs. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pages 130–137, 2011.

[Kolobov, 2013] Andrey Kolobov. *Scalable Methods and Expressive Models for Planning Under Uncertainty*. PhD thesis, University of Washington, 2013.

[Li and Littman, 2005] Lihong Li and Michael L. Littman. Lazy approximation for solving continuous finite-horizon MDPs. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 1175–1180, 2005.

[Likhachev *et al.*, 2003] Maxim Likhachev, Geoff Gordon, and Sebastian Thrun. ARA*: Anytime A* with provable bounds on sub-optimality. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2003.

[Little *et al.*, 2005] Iain Little, Douglas Aberdeen, and Sylvie Thiébaux. Prottle: A probabilistic temporal planner. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 1181–1186, 2005.

[Liu and Koenig, 2005a] Yaxin Liu and Sven Koenig. Existence and finiteness conditions for risk-sensitive planning: Results and conjectures. In *Proceedings of the Conference in Uncertainty in Artificial Intelligence (UAI)*, pages 354–363, 2005.

[Liu and Koenig, 2005b] Yaxin Liu and Sven Koenig. Risk-sensitive planning with one-switch utility functions: Value iteration. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 993–999, 2005.

[Liu and Koenig, 2006] Yaxin Liu and Sven Koenig. Functional value iteration for decision-theoretic planning with general utility functions. In *Proceedings*

of the National Conference on Artificial Intelligence (AAAI), pages 1186–1193, 2006.

[Mannor and Tsitsiklis, 2011] Shie Mannor and John N. Tsitsiklis. Mean-variance optimization in Markov decision processes. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 177–184, 2011.

[Mannor and Tsitsiklis, 2013] Shie Mannor and John N. Tsitsiklis. Algorithmic aspects of mean-variance optimization in Markov decision processes. *European Journal of Operational Research*, 231(3):645–653, 2013.

[Mannor *et al.*, 2012] Shie Mannor, Ofir Mebel, and Huan Xu. Lightning does not strike twice: Robust MDPs with coupled uncertainty. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2012.

[Marecki and Tambe, 2009] Janusz Marecki and Milind Tambe. Planning with continuous resources for agent teams. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1089–1096, 2009.

[Marecki and Varakantham, 2010] Janusz Marecki and Pradeep Varakantham. Risk-sensitive planning in partially observable environments. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1357–1368, 2010.

[Marecki *et al.*, 2007] Janusz Marecki, Sven Koenig, and Milind Tambe. A fast analytical algorithm for solving Markov decision processes with real-valued resources. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2536–2541, 2007.

[Mausam and Kolobov, 2012] Mausam and Andrey Kolobov. *Planning with Markov Decision Processes : An AI Perspective*. Morgan & Claypool Publishers, 2012.

[Mausam and Weld, 2008] Mausam and Daniel S. Weld. Planning with durative actions in stochastic domains. *Journal of Artificial Intelligence Research (JAIR)*, 31:33–82, 2008.

[McMillen and Veloso, 2007] Colin McMillen and Manuela Veloso. Thresholded rewards: Acting optimally in timed, zero-sum games. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 1250–1255, 2007.

[Nilim and Ghaoui, 2005] Arnab Nilim and Laurent El Ghaoui. Robust Markov decision processes with uncertain transition matrices. *Operations Research*, 53(5):780–798, 2005.

[Papadimitriou and Tsitsiklis, 1987] Christos H. Papadimitriou and John N. Tsitsiklis. The complexity of Markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, August 1987.

[Petrik and Subramanian, 2012] Marek Petrik and Dharmashankar Subramanian. An approximate solution method for large risk-averse Markov decision processes. In *Proceedings of the Conference in Uncertainty in Artificial Intelligence (UAI)*, pages 805–814, 2012.

[Pineau *et al.*, 2003] Joelle Pineau, Geoffrey J. Gordon, and Sebastian Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1025–1032, 2003.

[Poupart *et al.*, 2015] Pascal Poupart, Aarti Malhotra, Pei Pei, Kee-Eung Kim, Bongseok Goh, and Michael Bowling. Approximate linear programming for constrained partially observable Markov decision processes. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 3342–3348, 2015.

[Puterman, 1994] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994.

[Regan and Boutilier, 2009] Kevin Regan and Craig Boutilier. Regret-based reward elicitation for Markov decision processes. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 444–451, 2009.

[Regan and Boutilier, 2011] Kevin Regan and Craig Boutilier. Robust online optimization of reward-uncertain MDPs. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2165–2171, 2011.

[Roijers *et al.*, 2013] Diederik M. Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research (JAIR)*, 48:67–113, 2013.

[Ruszczynski, 2010] Andrzej Ruszczynski. Risk-averse dynamic programming for Markov decision processes. *Mathematical Programming*, 125(2):235–261, 2010.

[Sanner *et al.*, 2011] Scott Sanner, Karina Valdivia Delgado, and Leliane Nunes de Barros. Symbolic dynamic programming for discrete and continuous state MDPs. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 643–652, 2011.

[Shani *et al.*, 2013] Guy Shani, Joelle Pineau, and Robert Kaplow. A survey of point-based POMDP solvers. *Autonomous Agents and Multi-Agent Systems*, 27(1):1–51, 2013.

[Steinmetz *et al.*, 2016] Marcel Steinmetz, Jörg Hoffmann, and Olivier Buffet. Goal probability analysis in probabilistic planning: Exploring and enhancing the state of the art. *Journal of Artificial Intelligence Research (JAIR)*, 57:229–271, 2016.

[Stentz, 1995] Anthony Stentz. The focussed D* algorithm for real-time replanning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1652–1659, 1995.

[Sun *et al.*, 2008] Xiaoxun Sun, Sven Koenig, and William Yeoh. Generalized adaptive A*. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 469–476, 2008.

[Sun *et al.*, 2009] Xiaoxun Sun, William Yeoh, and Sven Koenig. Efficient incremental search for moving target search. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 615–620, 2009.

[Sun *et al.*, 2010a] Xiaoxun Sun, William Yeoh, and Sven Koenig. Generalized fringe-retriving A*: Faster moving target search on state lattices. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1081–1087, 2010.

[Sun *et al.*, 2010b] Xiaoxun Sun, William Yeoh, and Sven Koenig. Moving target D* lite. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 67–74, 2010.

[Sutton and Barto, 1998] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, 1998.

[Tarjan, 1972] Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.

[Thomas, 2015] Philip S. Thomas. *Safe Reinforcement Learning.* PhD thesis, University of Massachusetts Amherst, 2015.

[Varakantham *et al.*, 2012] Pradeep Varakantham, Shih-Fen Cheng, Geoffrey J. Gordon, and Asrar Ahmed. Decision support for agent populations in uncertain and congested environments. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2012.

[Wiesemann *et al.*, 2013] Wolfram Wiesemann, Daniel Kuhn, and Berç Rustem. Robust Markov decision processes. *Mathematics of Operations Research*, 38(1):153–183, 2013.

[Williams and Young, 2007] Jason D. Williams and Steve J. Young. Partially observable Markov decision processes for spoken dialog systems. *Computer Speech & Language*, 21(2):393–422, 2007.

[Wray and Zilberstein, 2015] Kyle Hollins Wray and Shlomo Zilberstein. Multi-objective POMDPs with lexicographic reward preferences. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1719–1725, 2015.

[Wray *et al.*, 2015] Kyle Hollins Wray, Shlomo Zilberstein, and Abdel-Illah Mouaddib. Multi-objective MDPs with conditional lexicographic reward preferences. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 3418–3424, 2015.

[Wu *et al.*, 2014] XiaoJian Wu, Daniel Sheldon, and Shlomo Zilberstein. Rounded dynamic programming for tree-structured stochastic network design. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 479–485, 2014.

[Xu and Mannor, 2012] Huan Xu and Shie Mannor. Distributionally robust Markov decision processes. *Mathematics of Operations Research*, 37(2):288–300, 2012.

[Younes and Simmons, 2004] Håkan L. S. Younes and Reid G. Simmons. Solving generalized semi-Markov decision processes using continuous phase-type distributions. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 742–748, 2004.

[Yu *et al.*, 1998] Stella Yu, Yuanlie Lin, and Pingfan Yan. Optimization models for the first arrival target distribution function in discrete time. *Journal of Mathematical Analysis and Applications*, 225:193–223, 1998.

[Yu *et al.*, 2015] Peng Yu, Cheng Fang, and Brian Charles Williams. Resolving over-constrained probabilistic temporal problems through chance constraint relaxation. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 3425–3431, 2015.

[Zamani *et al.*, 2012] Zahra Zamani, Scott Sanner, and Cheng Fang. Symbolic dynamic programming for continuous state and action MDPs. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2012.

[Zhao *et al.*, 2007] Qing Zhao, Lang Tong, Ananthram Swami, and Yunxia Chen. Decentralized cognitive MAC for opportunistic spectrum access in ad hoc networks: A POMDP framework. *IEEE Journal on Selected Areas in Communications*, 25(3):589–600, 2007.

[Ziebart *et al.*, 2008] Brian Ziebart, Andrew Maas, Anind Dey, and J. Andrew Bagnell. Navigate like a cabbie: Probabilistic reasoning from observed context-

aware behavior. In *Proceedings of Ubiquitous Computing (Ubicomp)*, pages 322–331, 2008.