

RICE UNIVERSITY

**Bounded Policy Synthesis for POMDPs with  
Safe-Reachability and Quantitative Objectives**

by

**Yue Wang**

A THESIS SUBMITTED  
IN PARTIAL FULFILLMENT OF THE  
REQUIREMENTS FOR THE DEGREE

**Doctor of Philosophy**

APPROVED, THESIS COMMITTEE:



---

Dr. Swarat Chaudhuri, Chair  
Associate Professor of Computer Science



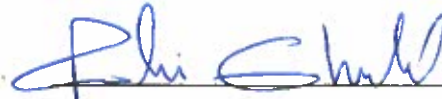
---

Dr. Lydia E. Kavradi  
Noah Harding Professor  
of Computer Science



---

Dr. Moshe Y. Vardi  
Karen Ostrum George Distinguished Service  
Professor in Computational Engineering



---

Dr. Fathi H. Ghorbel  
Professor of Mechanical Engineering

HOUSTON, TEXAS

September 2018

## ABSTRACT

### Bounded Policy Synthesis for POMDPs with Safe-Reachability and Quantitative Objectives

by

Yue Wang

Robots are being deployed for many real-world applications like autonomous driving, disaster rescue, and personal assistance. Effectively planning robust executions under uncertainty is critical for building these autonomous robots. Partially Observable Markov Decision Processes (POMDPs) provide a standard approach to model many robot applications under uncertainty. A key algorithmic problem for POMDPs is the synthesis of policies that specify the actions to take contingent on all possible events. Policy synthesis for POMDPs with two kinds of objectives is considered in this thesis: (1) boolean objectives for a correctness guarantee of accomplishing tasks and (2) quantitative objectives for optimal behaviors. For boolean objectives, this thesis focuses on a common *safe-reachability* objective: with a probability above a threshold, a goal state is eventually reached while keeping the probability of visiting unsafe states below a different threshold.

Previous results have shown that policy synthesis for POMDPs over infinite horizon is generally undecidable. For decidability, this thesis focuses on POMDPs over a *bounded* horizon. Solving POMDPs requires reasoning over a vast space of beliefs (probability distributions). To address this, this thesis introduces the notion of a goal-constrained belief space that only contains beliefs reachable under desired executions that can achieve the safe-reachability objectives. Based on this notion, this thesis presents an offline approach



that constructs policies over the goal-constrained belief space instead of the entire belief space. Simulation experiments show that this offline approach can scale to large belief spaces by focusing on the goal-constrained belief space. A full policy is generally costly to compute. To improve efficiency, this thesis presents an online approach that interleaves the computation of partial policies and execution. A partial policy is parameterized by a replanning probability and only contain a sampled subset of all possible events. This online approach allows users to specify an appropriate bound on the replanning probability to balance efficiency and correctness. Finally, this thesis presents an approximate policy synthesis approach that combines the safe-reachability objectives with the quantitative objectives. The results demonstrate that the constructed policies not only achieve the safe-reachability objective but also are of high quality concerning the quantitative objective.

## Acknowledgments

I would like to express my deepest gratitude to my advisors, Dr. Swarat Chaudhuri and Dr. Lydia Kavraki for their guidance and dedication during my graduate study at Rice University. I benefited a lot from their constructive comments and broad perspective. Without their valuable feedback and encouragement, none of the works presented in this thesis would have reached the modest level of maturity.

I would like to thank my committee members, Dr. Moshe Vardi and Dr. Fathi Ghorbel for their insightful comments during my proposal. They deserve my profound appreciation for spending their valuable time on helping me improve the thesis.

I have been supremely fortunate to collaborate academically with wonderful members from the Kavraki Lab. I learned a lot from Dr. Mark Moll, Dr. Srinivas Nedunuri, Dr. Ryan Luna and Dr. Neil Dantam on many aspects of research. Many thanks to Keliang He for his willingness to listen to me during our countless brainstorming sessions. I must thank Dr. Juan David Hernández, Bryce Willey, Constantinos Chamzas, Zachary Kingston for their kind assistance in preparing the physical experiments on Fetch.

Additionally, the administrative staff in the Department of Computer Science deserve my sincere appreciation for their help with all kinds of things, especially Sherry Nassar, Melissa Cisneros, Beth Rivera, and Lena Sifuentes.

Finally, I would like to thank my wife Yahong, our combined family, and my friends for always being supportive.

The work in this thesis was funded in part by grants from NSF CCF 1139011, NSF CCF 1514372, NSF CCF 1162076 and NSF IIS 1317849.

# Contents

Abstract	ii
Acknowledgments	iv
List of Illustrations	viii
List of Tables	xii
<b>1 Introduction</b>	<b>1</b>
1.1 Overview . . . . .	1
1.2 Related Work . . . . .	4
1.2.1 Partially Observable Markov Decision Processes . . . . .	4
1.2.2 Markov Decision Processes . . . . .	7
1.2.3 Task and Motion Planning Under Uncertainty . . . . .	8
1.2.4 Program Verification and Synthesis . . . . .	9
1.3 Contributions . . . . .	10
1.3.1 POMDPs with Safe-Reachability Objectives . . . . .	11
1.3.2 POMDPs with Both Safe-Reachability and Quantitative Objectives	13
1.4 Thesis Structure . . . . .	13
<b>2 POMDPs with Safe-Reachability Objectives</b>	<b>15</b>
2.1 Definitions . . . . .	16
2.1.1 POMDPs . . . . .	16
2.1.2 Safe-Reachability Objective . . . . .	17
2.1.3 Policy and Conditional Plan . . . . .	18
2.2 Relation to POMDPs with Quantitative Objectives . . . . .	20

### **3 Offline Synthesis for POMDPs with Safe-Reachability Objectives 24**

3.1	Problem Formulation . . . . .	26
3.1.1	Goal-Constrained Belief Space . . . . .	26
3.1.2	Problem Statement . . . . .	27
3.2	Bounded Policy Synthesis . . . . .	27
3.2.1	Constraint Generation . . . . .	30
3.2.2	Plan Generation . . . . .	31
3.2.3	Policy Generation . . . . .	31
3.3	Algorithm Analysis . . . . .	33
3.3.1	Algorithm Complexity . . . . .	33
3.3.2	Observability . . . . .	34
3.4	Experiments . . . . .	35
3.4.1	Performance . . . . .	37
3.4.2	Horizon Bound . . . . .	39
3.4.3	Physical Validation . . . . .	40
3.5	Discussion . . . . .	42

### **4 Online Planning for POMDPs with Safe-Reachability Objectives 44**

4.1	Problem Formulation . . . . .	46
4.1.1	Partial Policy . . . . .	46
4.1.2	Replanning Probability . . . . .	48
4.1.3	Problem Statement . . . . .	51
4.2	Online Partial Policy Synthesis . . . . .	52
4.2.1	Partial Policy Synthesis . . . . .	53
4.2.2	Partial Policy Generation . . . . .	55
4.3	Experiments . . . . .	58
4.3.1	Performance . . . . .	59
4.3.2	Success Rate . . . . .	61

4.3.3	Gains from Updating Replanning Probability Bound . . . . .	62
4.3.4	Physical Validation . . . . .	64
4.3.5	Tag Domain . . . . .	65
4.4	Discussion . . . . .	66
<b>5</b>	<b>Combining Safe-Reachability and Quantitative Objectives</b>	<b>68</b>
5.1	Problem Formulation . . . . .	71
5.1.1	Quantitative Objectives . . . . .	71
5.1.2	Problem Statement . . . . .	72
5.2	Point-Based Policy Synthesis . . . . .	74
5.2.1	Bounded Policy Synthesis . . . . .	75
5.2.2	Policy Iteration . . . . .	75
5.2.3	Algorithm Analysis . . . . .	77
5.3	Experiments . . . . .	82
5.3.1	Results . . . . .	83
5.3.2	Physical Validation . . . . .	85
5.4	Discussion . . . . .	86
<b>6</b>	<b>Conclusions and Future Work</b>	<b>88</b>
6.1	Open Questions . . . . .	90
	<b>Bibliography</b>	<b>92</b>

## Illustrations

- 1.1 An example uncertain domain. A robot with imperfect actuation and perception needs to navigate through an office to pick up the blue can on the table, while avoiding collisions with uncertain obstacles such as floor signs and file cabinets (boolean objective). There are two regions marked with red tape, and the robot should avoid visiting these red regions as much as possible (quantitative objective). . . . . 2
- 2.1 A conditional plan  $\gamma$  for an uncertain domain with 2 observations ( $o_1$  and  $o_2$ ), represented as a tree rooted at the belief  $b$ . Circle nodes represent beliefs, the edges (e.g.,  $a_1, a_2^0, a_2^1, \dots$ ) from circle nodes to rectangle nodes represent actions and the edges ( $o_1$  and  $o_2$ ) from rectangle nodes to circle nodes represent observations. . . . . 20
- 2.2 An example to show the difference between POMDPs with safe-reachability objectives and unconstrained/C/RS/CC-POMDPs. There are 3 states: start state  $s_{\text{ready}}$ , unsafe state  $s_{\text{unsafe}}$  and goal state  $s_{\text{goal}}$ . Dashed green edges represent transitions of executing left-hand pick-up action  $a_L$  in state  $s_{\text{ready}}$  and solid red edges represent transitions of executing right-hand pick-up action  $a_R$  in state  $s_{\text{ready}}$ . For each edge, the first line is the transition probability and the second line is the tuple of observation probabilities  $(p_{o_{\text{pos}}}, p_{o_{\text{neg}}})$ . . . . . 21

2.3	The belief space transition for the POMDP in Figure 2.2. Blue nodes $(p_{s_{\text{ready}}}, p_{s_{\text{unsafe}}}, p_{s_{\text{goal}}})$ represent beliefs (probability distributions over states), and red nodes represent observation. The edges from blue nodes to red nodes represent actions and the edges from red nodes to blue nodes represent observations and the corresponding probabilities. . . . .	22
3.1	An example of a safe-reachability objective: a robot with uncertain actuation and perception needs to navigate through the kitchen and pick up a green cup from the black storage area (reachability), while avoiding collisions with uncertain obstacles (e.g., chairs) modeled as cylinders in the yellow “shadow” region (safety). . . . .	25
3.2	The core steps of the BPS algorithm. . . . .	27
3.3	An example run of BPS. The black box represents the goal-constrained belief space $\mathcal{B}_{\mathcal{G}}$ over the bounded horizon $k$ . Circle nodes represent beliefs, the edges (e.g., $a_{s+1}^{\sigma_k}, a_i^{\sigma_k}$ ) from circle nodes to rectangle nodes represent actions and the edges (e.g., $o_{s+1}^{\sigma_k}, o'_{s+1}$ ) from rectangles nodes to circle nodes represent observations. The dashed green path represents one candidate plan $\sigma_k$ found by the incremental SMT solver. BPS constructs a policy tree from this candidate plan by considering other branches following the rectangle node for each step. . . . .	29
3.4	Performance of BPS as the number of obstacles $M$ varies. The plot of circles shows the performance of BPS with incremental solving and the plot of squares shows the performance of BPS without incremental solving.	38
3.5	The number of plans checked (i.e, the number of SMT calls) by BPS during policy synthesis as the number of obstacles $M$ varies. . . . .	38
3.6	Performance of BPS for the kitchen domain with $M = 2$ obstacles as the horizon bound $h$ increases. The blue dashed line is the plot $h_{\min} = 9$ . . . . .	40

3.7	An example uncertain domain with safe-reachability objective: a robot with imperfect actuation and perception needs to navigate through an office to pick up the blue can from the table, while avoiding collisions with uncertain obstacles such as floor signs and file cabinets. . . . .	41
3.8	Physical validation of BPS for the domain shown in Figure 3.7. . . . .	42
4.1	A $k$ -step partial policy $\pi^p$ for an uncertain domain with 2 observations ( $o_1$ and $o_2$ ), represented as a partial tree rooted at the initial belief $b_0$ (including only solid branches). Circle nodes represent beliefs, the edges (e.g., $a_1, a_2^0, a_2^1, \dots$ ) from circle nodes to rectangle nodes represent actions, and the edges ( $o_1$ and $o_2$ ) from rectangle nodes to circle nodes represent observations. . . . .	45
4.2	OPPS . . . . .	51
4.3	Partial policy synthesis . . . . .	51
4.4	Performance results for the kitchen domain as the bound $\delta_{p_{\text{replan}}}$ increases. Different plots correspond to tests with different numbers $M$ of obstacles. Missing data points in a plot indicate the timeout. The red dashed line is the timeout (time = 1800 seconds). The blue dashed line passes through the data points generated by BPS. All the results are averaged over 50 independent runs. . . . .	60
4.5	Success rate as $\delta_{p_{\text{replan}}}$ increases. The green dotted line shows the plot of success rate = $1.0 - \delta_{p_{\text{replan}}}$ . The red dashed line is the plot of success rate = 1.0. The blue dashed line passes through the data points generated by BPS. . . . .	62
4.6	Replanning probability and total computation time as the bound $\delta_{p_{\text{replan}}}$ increases ( $M = 4$ ). The green dotted line shows the plot of replanning probability = $\delta_{p_{\text{replan}}}$ . The blue dashed line passes through the data points generated by BPS. . . . .	63



4.7	Physical validation of OPPS for the domain shown in Figure 3.7. . . . .	65
4.8	Performance results for the Tag domain as the replanning probability bound $\delta_{p_{\text{replan}}}$ increases. All the results are averaged over 50 independent runs.	66
5.1	Overview of the PBPS algorithm. PBPS interleaves computation of valid policies and policy iteration. . . . .	72
5.2	Executions of policies constructed by BPS (figures in the first row) and PBPS (figures in the second row) for the domain shown in Figure 1.1. . . .	86

## Tables

5.1	Performance of PBPS with and without the exploration step for different problems. . . . .	84
-----	---	----

# Chapter 1

## Introduction

### 1.1 Overview

Deploying robots in the physical world presents a fundamental challenge with planning robust executions under uncertainty. For example, a personal robot assistant performing daily household tasks needs to consider uncertainty from uncontrollable human activities, a self-driving car on the road needs to consider uncertainty from imperfect controllers and sensors, and an autonomous underwater vehicle exploring oceans needs to consider uncertainty from the unexpected disturbance in the environment. Failing to account uncertainty in these scenarios may cause tasks failing and severe consequences such as human injury and robot damage.

The framework of Partially Observable Markov Decision Processes (POMDPs) [67, 70] offers a standard approach to model a variety of problems under uncertainty [8, 22, 37, 52]. As an example, in robotics, many applications in uncertain domains can be modeled using the POMDP formulation [7, 8, 29, 37].

Perhaps the central algorithmic problem for POMDPs is the synthesis of *policies* [67, 70]: recipes that specify the actions to take contingent on *all possible* events in the environment. While this policy synthesis problem has traditionally been posed with respect to optimality objectives, many applications in robotics are better modeled by POMDPs where the objective is a boolean requirement. For example, in an office environment, a robot needs to navigate through the office to pick up a target object while avoiding collisions

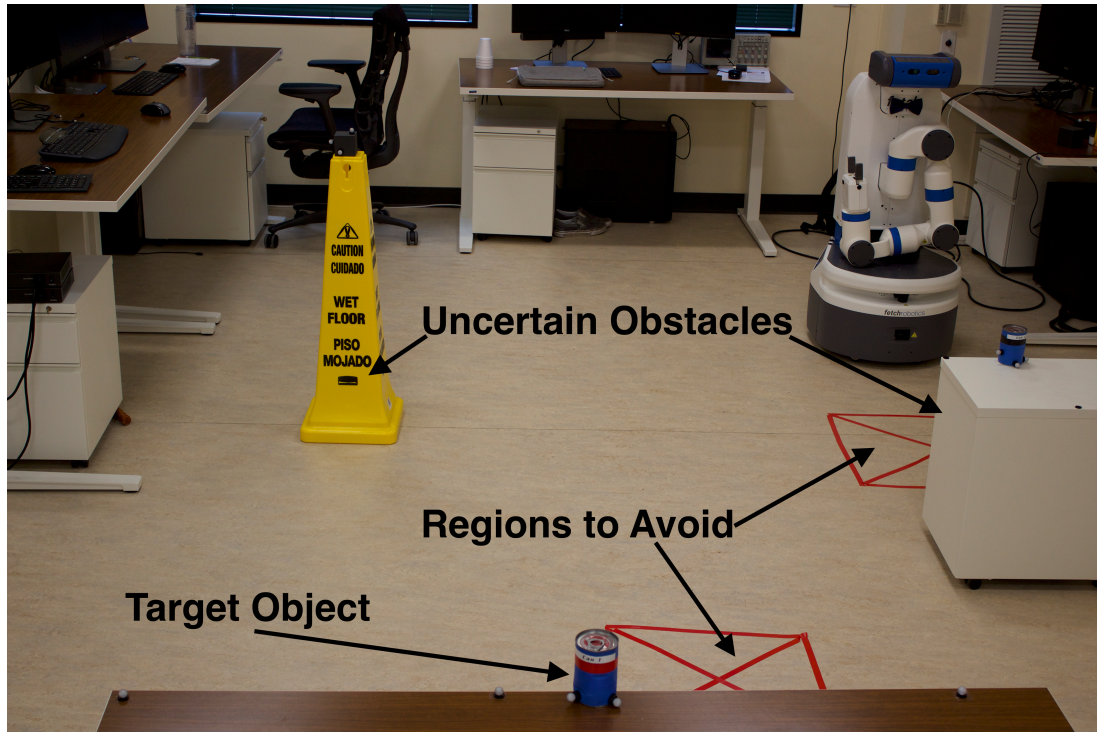


Figure 1.1 : An example uncertain domain. A robot with imperfect actuation and perception needs to navigate through an office to pick up the blue can on the table, while avoiding collisions with uncertain obstacles such as floor signs and file cabinets (boolean objective). There are two regions marked with red tape, and the robot should avoid visiting these red regions as much as possible (quantitative objective).

with uncertain obstacles such as wet-floor signs and file cabinets, as shown in Figure 1.1. This task requirement is naturally formulated as policy synthesis from a high-level boolean objective written in a temporal logic. What's more, in some robotic domains where a strong correctness guarantee of completing tasks is required, formulating boolean requirements as quantitative objectives by assigning penalties for unsafe states and rewards for goal states, may lead to policies that are overly conservative or overly risky [75], depending on the particular reward function chosen. Moreover, designing an appropriate reward function that encodes the boolean requirement exactly is also a non-trivial task for users. Thus, new formulations and algorithms are required to deal with boolean requirements in POMDPs ex-

plicitly. Section 2.2 discusses an example POMDP problem to demonstrate that POMDPs with safe-reachability objectives can provide a better guarantee of both safety and reachability than the existing quantitative POMDP models in some robotic domains.

Policy Synthesis for POMDPs with boolean requirements has been considered in previous work [7, 8, 74, 78, 79]. In [7], the authors introduced two kinds of analysis problems for POMDPs with boolean requirements: (1) the *qualitative* analysis problem that checks whether the boolean requirement can be ensured with *probability 1* (almost-sure satisfaction); and (2) the *quantitative* analysis problem that checks whether the boolean requirement can be ensured with *a probability above a threshold*. It has been shown that both analysis problems are undecidable in general for POMDPs with boolean requirements [10, 50, 57]. These previous works [7, 8, 74] focus on the qualitative analysis problem of POMDPs with boolean requirement restricted to finite-state and memoryless controllers to make the problem tractable.

This thesis studies the quantitative analysis problem of POMDPs with a common boolean requirement: safe-reachability, which requires that with a probability above a certain threshold, a goal state is eventually reached while keeping the probability of visiting unsafe states below a different threshold. Many task requirements of robot applications such as the one shown in Figure 1.1 can be formulated as a safe-reachability objective. As mentioned before, this quantitative analysis problem is undecidable in general [10, 50, 57]. However, when restricted to the bounded horizon, this problem is PSPACE-complete [53, 56]. Therefore, to make the problem tractable, this thesis assumes there is a horizon bound  $h$ , and users are not interested in policies beyond this horizon bound  $h$ . This assumption is particularly reasonable for robot applications since robots often need to complete tasks in bounded steps because of some resource constraints, e.g., time and energy constraints. Based on this assumption, this thesis presents a series of practical policy synthesis approaches for

POMDPs with safe-reachability objectives over a bounded horizon.

## 1.2 Related Work

### 1.2.1 Partially Observable Markov Decision Processes

Partially Observable Markov Decision Processes (POMDPs) [67, 70] provide a principled mathematical framework for modeling a variety of applications in the face of uncertainty [8, 22, 37, 52]. The main algorithmic problem in POMDPs is the synthesis of policies [67, 70] that specify the action choice for every possible event during execution. Policy synthesis for POMDPs can be divided into three different categories, based on the objectives associated with POMDPs that the synthesized policy must achieve.

#### POMDPs with Quantitative Objectives

In the first category, the objectives of POMDPs are quantitative objectives such as (discounted) rewards. Policy synthesis for quantitative POMDPs is to find optimal policies that maximize cumulative expected rewards. Many existing POMDP algorithms for robot applications [1, 6, 29, 33, 42, 49, 58, 69, 71] deal with this quantitative POMDP formulation.

Due to uncertainty, policy synthesis for POMDPs usually reasons over the space of *beliefs*: probability distributions over all possible states. A big challenge in policy synthesis for POMDPs is that belief space is a high-dimensional, continuous space with an infinite number of beliefs, which makes POMDPs extremely difficult to solve [50, 56]. In recent years, there have been significant efforts in developing algorithms that produce approximate policies in order to solve large POMDPs. Many of these approximation algorithms are point-based POMDP solvers based on the notion of (optimally) reachable belief space [1, 6, 42, 49, 58, 60, 65, 66, 68, 69, 71, 72].

The policy synthesis approaches presented in this thesis take inspiration from these efficient point-based POMDP solvers. The offline policy synthesis approach presented in this thesis resembles these efficient point-based POMDP solvers by focusing on the goal-constrained belief space instead of the original belief space. The idea of partial policies behind the online planning approach presented in this thesis is inspired by the state-of-the-art online POMDP solvers based on Determinized Sparse Partially Observable Tree (DESPOT) [6, 69]. Both DESPOT and partial policies only include a subset of all possible executions to approximate a full policy and improve scalability. There are two major differences between DESPOT and partial policies: first, DESPOT deals with POMDPs with quantitative objectives while partial policies are introduced to handle POMDPs with safe-reachability objectives. Second, DESPOT consists of all action branches while partial policies only include one action branch per step, which is part of the desired execution achieving the given safe-reachability objective.

### **Constrained/Risk-Sensitive/Chance-Constrained POMDPs**

In the second category, the objectives of POMDPs extend the quantitative objectives of the traditional POMDPs with notions of risk and cost. Recently, there has been a large body of work in constrained POMDPs (C-POMDPs) [35, 40, 61, 75], chance-constrained POMDP (CC-POMDPs) [63], and risk-sensitive POMDPs (RS-POMDPs) [34, 51] that deal with expected cost or risk constraints explicitly.

There are two significant differences between their models and the formulation of POMDPs with safe-reachability objectives studied in this thesis. First, the goal of these models is to maximize the cumulative expected reward while keeping the expected cost or risk below a threshold, while in POMDPs with safe-reachability objectives, the goal is to satisfy a safe-reachability objective in all possible executions including the worst case.

Therefore, POMDPs with safe-reachability objectives provide a better safety guarantee than the C/RS/CC-POMDPs with expected cost or risk threshold constraints. Second, to encode reachability objectives, C/RS/CC-POMDPs typically assigns a positive reward for goal states. This encoding does not provide any guarantee on the probability of reaching goal states. POMDPs with safe-reachability objectives can directly encode the constraint of reach a goal state with a probability above a threshold as a boolean requirement. Therefore, POMDPs with safe-reachability objectives provide a better reachability guarantee than the C/RS/CC-POMDPs.

While C/RS/CC-POMDPs are suitable for many applications, there are settings in robotics such as autonomous driving and disaster rescue that demand synthesis of policies that can provide such a strong guarantee of reaching goal states safely.

### **POMDPs with Boolean Objectives**

In the third category, the objectives of POMDPs are high-level boolean requirements specified in the form of a temporal logic. There are two kinds of analysis problems for POMDPs with boolean requirements [7]: (1) the *qualitative* analysis problem that asks whether the boolean requirement can be ensured with probability 1 (almost-sure satisfaction); and (2) the *quantitative* analysis problem that asks whether the boolean requirement can be ensured with probability above a threshold. Previous works [10, 50, 57] have shown that both analysis problems are undecidable in general for POMDPs with boolean requirements. Recent work [7, 8, 74] has investigated the *almost-sure satisfaction* problem restricted to finite-state and memoryless controllers to make the problem tractable.

Although the policy with almost-sure satisfaction provides a strong guarantee of completing tasks, this almost-sure satisfaction may not be achievable in general. Therefore, this thesis focuses on the more general quantitative analysis problem of POMDPs with boolean



requirements. Specifically, this thesis studies POMDPs with a common boolean requirement: safe-reachability, which requires that with a probability above a certain threshold, a goal state is eventually reached while keeping the probability of visiting unsafe states below a different threshold. The main challenge is that the quantitative analysis problem is undecidable in general [10, 50, 57]. However, when restricted to the bounded horizon, this problem is PSPACE-complete [53, 56]. Therefore, to make the problem tractable, this thesis assumes there is a horizon bound  $h$ , and users are not interested in policies beyond this horizon bound  $h$ . This assumption is particularly reasonable for robot applications since robots often need to complete tasks in bounded steps because of some resource constraints, e.g., time and energy constraints.

### 1.2.2 Markov Decision Processes

Markov Decision Processes (MDPs) are a standard model for stochastic systems featuring non-determinism. The fundamental problem in MDPs is to design a strategy that optimizes the value of a given objective function. For MDPs, many different objectives, such as  $\omega$ -regular or LTL properties [14, 15, 77], conditional value-at-risk [43], and multiple objectives [11, 12, 24, 28, 81] have been studied with a variety of applications.

In MDPs, states are fully observable while in POMDPs, states are only partially observable. However, beliefs (probability distributions over all possible states) are always fully observable. A POMDP can be reduced to an MDP with a continuous state space, i.e., the belief space. When restricted to a bounded horizon, POMDPs can be reduced to an MDP with a finite state space, i.e., the reachable belief space  $\mathcal{B}_{b_0}$ . Many existing approaches can deal with MDPs with boolean objectives, e.g., [14, 15, 20, 24, 44, 46–48, 77, 82].

This thesis focuses on POMDP formulations rather than reducing POMDPs to the corresponding MDPs and applying existing approaches for MDPs with boolean objectives,

based on two reasons. First, in order to employ existing MDP-based approaches, it is typically required to represent the state space explicitly. As mentioned above, POMDPs with a bounded horizon can be reduced to MDPs with a finite state space, i.e., the reachable belief space. It is clear that the size of the reachable belief space grows exponentially as the horizon increases. Therefore, explicitly representing the reachable belief space may not be tractable in practice. Second, value functions of POMDPs over finite-horizon has nice properties, which are piece-wise linear and convex. Moreover, they can be represented as a finite set of vectors called  $\alpha$ -vectors [37, 67, 70]. This vector representation contains not only the value information but also the gradient of the value functions, which can be used as a global approximation over the entire belief space. This is one of the key ideas behind many efficient point-based POMDP solvers [1, 6, 33, 42, 49, 58, 69, 71]. The thesis also exploits this idea to efficiently generate good approximated policies for POMDPs with both boolean and quantitative objectives.

### 1.2.3 Task and Motion Planning Under Uncertainty

Task and Motion Planning (TMP) [2, 16–18, 21, 23, 25–27, 31, 32, 39, 41, 45, 73, 80] describes a class of challenging problems that combine low-level motion planning and high-level task reasoning. Most of these TMP approaches focus on deterministic domains, while several of them apply to uncertain domains with uncertainty in perception [31, 39]. These TMP under uncertainty works perform *online* planning with a determinized approximation of belief space dynamics [59] assuming the most likely observation will be obtained. In some cases, due to the limited time budget for replanning, the online approach constructs a plan over a small subset of possible events which may miss some rare events that are critical to safety [49]. Therefore it is worth investigating both online and offline approaches to gain a better understanding of planning under uncertainty and to develop methodologies that can

combine both approaches to achieve a good balance between efficiency and safety.

#### 1.2.4 Program Verification and Synthesis

Policy synthesis for robotics is closely related to program synthesis and program verification for general software; both require determining the correct response to a variety of possible events or inputs. The classical program synthesis solves the problem of discovering a program, which implements the required system, from user intent expressed in the form of some logical formulas [30]. Recent work [4, 5, 13] in program synthesis and program verification has investigated combining boolean and quantitative objectives.

Synthesizing policies and programs is computationally hard due to the large and possibly infinite search space. In the context of synthesizing policies for POMDPs, the search space is a high-dimensional, continuous space of probability distributions (beliefs) called belief space. To address this challenge, this thesis introduces the notion of goal-constrained belief space that only contains beliefs reachable from a given initial belief under desired executions that can achieve the safe-reachability objectives. Since not every execution can satisfy the safe-reachability objective, the goal-constrained belief space is much smaller than the original belief space in general. This thesis applies techniques from Bounded Model Checking (BMC) [3] to compactly represent the goal-constrained belief space over a bounded horizon as a set of symbolic constraints. BMC verifies whether a finite state system satisfies a given temporal logic specification. Thanks to the tremendous increase in the reasoning power of practical SMT (SAT) solvers, BMC can scale up to large systems with hundreds of thousands of states. The goal-constrained belief space is efficiently explored through a modern, incremental SMT solver [19]. It has been shown that the incremental capability of the SMT solver leads to an efficient planning algorithm for TMP [17, 54, 55, 80]. Inspired by this result, this thesis now leverages incremental SMT solvers for belief space

policy synthesis.

### 1.3 Contributions

Traditionally, POMDPs are posed with quantitative objectives such as (discounted) rewards [1, 6, 33, 42, 49, 58, 63, 69]. Recently, there has been a growing interest in POMDPs with *boolean* objectives [7, 8, 74, 78]. Many robot tasks in uncertain domains, such as the one shown in Figure 1.1, is naturally formulated as POMDPs with a high-level boolean objective written in a temporal logic. POMDPs with quantitative objectives provide an optimality guarantee but may lead to overly conservative or overly risky behaviors [75], depending on the particular reward function chosen. On the other hand, POMDPs with boolean objectives provides a strong correctness guarantee of completing tasks safely [78], but the constructed policy may not be optimal. For the example domain shown in Figure 1.1, there are many valid policies that can achieve the task objective, i.e., satisfy the boolean objective. Among these valid policies, the preferable policy is the one that passes the smallest number of red regions that the robot should avoid. Therefore, for domains that desire both correctness and optimality, POMDPs with both boolean and quantitative objectives are natural formulations.

Policy synthesis for POMDPs with both boolean and quantitative objectives has been studied before [9]. In their work, the goal is to find an optimal policy that also ensures a goal state is reached with probability 1 (almost-sure satisfaction). A more general policy synthesis problem of POMDPs with both boolean and quantitative objectives is to synthesize an optimal policy that satisfies the boolean objective with a probability above a threshold. This thesis studies this problem for the particular case of *safe-reachability* objectives, which require that with a probability above a threshold, a goal state is eventually reached while keeping the probability of visiting unsafe states below a different threshold.

Many robot tasks such as the one shown in Figure 1.1 can be formulated as POMDPs with safe-reachability and quantitative objectives.

This thesis introduces the formulation of POMDPs with safe-reachability objectives and demonstrates that in specific domains that require a strong correctness guarantee of accomplishing tasks, POMDPs with safe-reachability objectives can provide a better guarantee of both safety and reachability than the existing POMDP models. Based on this formulation, this thesis first presents two policy synthesis approaches for POMDPs with safe-reachability objectives: (1) an offline policy synthesis approach for a strong correctness guarantee; and (2) an online planning approach for a balance between efficiency and correctness. Based on these approaches, this thesis then presents a policy synthesis approach for POMDPs with both safe-reachability and quantitative objectives.

### **1.3.1 POMDPs with Safe-Reachability Objectives**

This thesis presents both offline synthesis and online planning approaches for POMDPs with safe-reachability objectives.

#### **Offline Synthesis**

The major challenge in policy synthesis for POMDPs is reasoning over a high-dimensional, continuous space of probability distributions (beliefs) called belief space. It is intractable to compute a full policy that specifies an action choice for every belief of the entire belief space. To address this challenge, this thesis introduces the notion of goal-constrained belief space that only contains beliefs reachable from a given initial belief under desired executions that can achieve the safe-reachability objectives. Since not every execution can satisfy the safe-reachability objective, the goal-constrained belief space is much smaller than the original belief space in general. Based on this notion, this thesis presents an approach

called Bounded Policy Synthesis (BPS) that synthesize policies with a bounded horizon offline over the goal-constrained belief space. The performance of BPS is evaluated in both simulated experiments on PR2 and physical experiments on Fetch. The results show that BPS can solve problems with large belief space by focusing on the goal-constrained belief space.

### **Online Planning**

BPS is an offline method that synthesizes a full policy over the goal-constrained belief space before execution. Another category of approaches for solving POMDPs is online planning that interleaves the computation of a single plan and execution [62]. The choice of the approach depends on the problem at hand. A policy provides faster responses during execution, while a single plan is cheaper to compute. Offline policy synthesis provides a strong correctness guarantee, but it is difficult to scale. Online planning is much more scalable but makes it hard to ensure correctness. To balance between efficiency and correctness, this thesis introduces the notion of a *partial policy*, which is parameterized by a replanning probability and only contains a sampled subset of all possible events to approximate a full policy. This thesis proves that the probability of the constructed partial policy failing is bounded by the replanning probability. This theoretical result indicates that the replanning probability is a good measure of the approximation quality of a partial policy. Based on this idea, this thesis presents an online approach called Online Partial Policy Synthesis (OPPS) that interleaves the computation of partial policies and execution. OPPS allows users to specify an appropriate bound on the replanning probability to achieve a good balance between efficiency and correctness. Simulation results demonstrate that OPPS scales better than BPS and can solve problems that are beyond the capabilities of BPS.

### 1.3.2 POMDPs with Both Safe-Reachability and Quantitative Objectives

Both BPS and OPPS focus on boolean reasoning: the goal is to synthesize a policy that satisfies a safe-reachability objective. However, in some robot applications where many resource constraints such as energy and time are required to be considered, boolean reasoning is not enough. A safe-reachability objective is naturally used to set a “lower bound” on the desirability of the synthesized policy: the policy should at least satisfy the given safe-reachability objective. However, there can be many policies that meet the safe-reachability objective, and some of them are more desirable than others. Given this, it is appropriate to consider policy synthesis where the synthesized policy must not only satisfy a safe-reachability objective but also be optimal with respect to a quantitative objective. To complete the picture, this thesis presents a policy synthesis approach, called Point-Based Policy Synthesis (PBPS) that combines the safe-reachability objectives with the traditional quantitative objectives. This thesis provides a theoretical analysis for this PBPS approach, showing that the value difference between the policy constructed by PBPS and the potential solution policy that is both optimal and satisfies the safe-reachability objective is bounded. Both simulation and physical experiment results demonstrate that the policies constructed by PBPS achieve the safe-reachability objective and are of high quality with respect to the quantitative objective.

## 1.4 Thesis Structure

The remainder of this thesis is organized as follows:

In Chapter 2, the formulation of POMDPs with safe-reachability is formally defined. An example POMDP domain is discussed to demonstrate the advantage of this POMDP formulation against the existing POMDP models.

In Chapter 3 and 4, policy synthesis for POMDPs with only safe-reachability objectives is studied. Chapter 3 presents an offline policy synthesis approach for POMDPs with safe-reachability objectives and the corresponding experimental results. Chapter 4 presents an online planning approach for POMDPs with safe-reachability objectives and the corresponding theoretical and empirical analysis.

In Chapter 5, policy synthesis for POMDPs with both safe-reachability and quantitative objectives is investigated.

Conclusions and discussions of future work are addressed in Chapter 6.



## Chapter 2

### POMDPs with Safe-Reachability Objectives

Deploying robots in the physical world presents a fundamental challenge with planning robust executions under uncertainty, e.g., uncertain effects from imperfect controllers and uncertain observations from noisy sensors. The framework of POMDPs [67] is a standard approach for modeling a variety of robot tasks under uncertainty (e.g., [6, 7, 37, 58, 78]).

Perhaps the central algorithmic problem for POMDPs is the synthesis of *policies* [67] or *conditional plans* [33]: recipes that specify the actions to take contingent on *all possible* events in the environment. Traditionally, this policy synthesis problem is posed with quantitative objectives such as (discounted) rewards [1, 6, 33, 42, 49, 58, 63, 69]. Recently, there has been a growing interest in POMDPs with *boolean* objectives [7, 8, 74, 78, 79]. Many robot tasks in uncertain domains, such as the one shown in Figure 1.1, is naturally formulated as POMDPs with a high-level boolean objective written in a temporal logic.

On the one hand, POMDPs with quantitative objectives provide an optimality guarantee but may lead to overly conservative or overly risky behaviors [75], depending on the particular reward function chosen. On the other hand, POMDPs with boolean objectives provides a strong correctness guarantee of completing tasks [7, 8, 74, 78], but the constructed policy may not be optimal. For the example domain shown in Figure 1.1, many valid policies can achieve the task objective, i.e., satisfying the boolean objective. Among these valid policies, the preferable policy is the one that passes the smallest number of red regions that the robot should avoid. Therefore, for domains that require both correctness and optimality, POMDPs with both boolean and quantitative objectives are natural formulations.

This thesis considers policy synthesis for POMDPs with both boolean and quantitative objectives. Specifically, this thesis study a common boolean objective: *safe-reachability* objectives, which require that with a probability above a threshold, a goal state is eventually reached while keeping the probability of visiting unsafe states below a different threshold. Many robot tasks such as the one shown in Figure 1.1 can be formulated as POMDPs with safe-reachability and quantitative objectives.

This chapter provides the formulation of POMDPs with safe-reachability objectives. We follow the notation in [78, 79].

## 2.1 Definitions

### 2.1.1 POMDPs

**Definition 2.1.1** (POMDP).

A POMDP is a tuple  $P = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{O}, \mathcal{Z}, r)$ :

- $\mathcal{S}$  is a *finite* set of states.
- $\mathcal{A}$  is a *finite* set of actions.
- $\mathcal{T}$  is a probabilistic transition function:  $\mathcal{T}(s, a, s') = \Pr(s'|s, a)$  is the probability of moving to state  $s' \in \mathcal{S}$  after taking action  $a \in \mathcal{A}$  in state  $s \in \mathcal{S}$ .
- $\mathcal{O}$  is a *finite* set of observations.
- $\mathcal{Z}$  is a probabilistic observation function:  $\mathcal{Z}(s', a, o) = \Pr(o|s', a)$  is the probability of observing  $o \in \mathcal{O}$  after taking action  $a \in \mathcal{A}$  and reaching  $s' \in \mathcal{S}$ .
- $r$  is a reward function:  $r(s, a)$  defines the reward of executing action  $a \in \mathcal{A}$  in state  $s \in \mathcal{S}$ .

Due to uncertainty in transition and observation, the actual state is partially observable and typically the robot maintains a *belief*, which is a probability distribution over all possible states  $b : \mathcal{S} \rightarrow [0, 1]$  with  $\sum_{s \in \mathcal{S}} b(s) = 1$ . The set of beliefs  $\mathcal{B} = \{b : \mathcal{S} \rightarrow [0, 1] \mid \sum_{s \in \mathcal{S}} b(s) = 1\}$  is known as the *belief space*.

The belief space transition function  $\mathcal{T}_{\mathcal{B}} : \mathcal{B} \times \mathcal{A} \times \mathcal{O} \rightarrow \mathcal{B}$  is *deterministic*.  $b_a^o = \mathcal{T}_{\mathcal{B}}(b, a, o)$  is the successor belief for a belief  $b \in \mathcal{B}$  after taking an action  $a \in \mathcal{A}$  and receiving an observation  $o \in \mathcal{O}$ , defined according to Bayes rule:

$$\forall s' \in \mathcal{S}, b_a^o(s') = \frac{\mathcal{Z}(s', a, o) \sum_{s \in \mathcal{S}} \mathcal{T}(s, a, s')b(s)}{\Pr(o|b, a)} \quad (2.1)$$

where  $\Pr(o|b, a) = \sum_{s' \in \mathcal{S}} \mathcal{Z}(s', a, o) \sum_{s \in \mathcal{S}} \mathcal{T}(s, a, s')b(s)$  is the probability of receiving the observation  $o$  after taking the action  $a$  in the belief  $b$ .

**Definition 2.1.2** (*k*-Step Plan).

A *k*-step *plan* is a sequence  $\sigma = (b_0, a_1, o_1, \dots, a_k, o_k, b_k)$  such that for all  $i \in (0, k]$ , the belief updates satisfy the transition function  $\mathcal{T}_B$ , i.e.,  $b_i = \mathcal{T}_B(b_{i-1}, a_i, o_i)$ , where  $a_i \in \mathcal{A}$  is an action and  $o_i \in \mathcal{O}$  is an observation.  $|\sigma| = k$  is the length of the *k*-step plan  $\sigma$ .

### 2.1.2 Safe-Reachability Objective

This thesis studies a common boolean objective: safe-reachability, defined as follows:

**Definition 2.1.3** (Safe-Reachability Objective).

A *safe-reachability objective* is a tuple  $\mathcal{G} = (Dest, Safe)$ :

- *Safe* is a set of safe beliefs
- *Dest* is a set of goal beliefs. In general, goal beliefs are safe beliefs, i.e.,  $Dest \subseteq Safe$ .

A safe-reachability objective  $\mathcal{G}$  compactly represents the set  $\Omega_{\mathcal{G}}$  of valid plans in belief space:

**Definition 2.1.4** (Valid  $k$ -Step Plan). A  $k$ -step plan  $(b_0, a_1, o_1, \dots, a_k, o_k, b_k)$  is *valid* w.r.t. a safe-reachability objective  $\mathcal{G} = (Dest, Safe)$  if there exists a goal belief  $b_j$  ( $j \leq k$ ) at step  $j$  ( $b_j \in Dest$ ) and all beliefs  $b_i$  ( $i < j$ ) visited before step  $j$  are safe beliefs ( $b_i \in Safe$ ).

Note that safe-reachability objectives are defined using sets of beliefs (probability distributions). The quantitative analysis problem of POMDPs with requirements of a goal state is eventually reached with a probability above some threshold while keeping the probability of visiting unsafe states below some threshold can be easily formulated as a safe-reachability objective  $\mathcal{G} = (Dest, Safe)$  defined as follows:

$$Dest = \{b \in \mathcal{B} \mid \left( \sum_{s \text{ is a goal state}} b(s) \right) > 1 - \delta_1\} \quad (2.2)$$

$$Safe = \{b \in \mathcal{B} \mid \left( \sum_{s \text{ violates safety}} b(s) \right) < \delta_2\} \quad (2.3)$$

Where  $\delta_1$  and  $\delta_2$  are small values that represent tolerance.

### 2.1.3 Policy and Conditional Plan

Previous results [10, 50, 57] have shown that policy synthesis of POMDPs is generally undecidable. However, when restricted to a bounded horizon, policy synthesis of POMDPs becomes PSPACE-complete [53, 56]. Therefore, this thesis focuses on computing a bounded policy  $\pi$  and the horizon (number of steps) of the policy is less than a given bound  $h$ . This bounded policy  $\pi$  is essentially a set of conditional plans [33]:

**Definition 2.1.5** (Conditional Plan).

A conditional plan  $\gamma$  is a tuple  $\gamma = (a, \nu)$ , where  $a \in \mathcal{A}$  is an action and  $\nu$  is an *observation*

*strategy* that maps an observation  $o \in \mathcal{O}$  to a conditional plan  $\gamma'$ .

Figure 2.1 shows an example of a conditional plan  $\gamma = (a_1, \nu)$  represented as a tree rooted at the belief  $b$ .  $\gamma$  together with the belief  $b$  defines a set  $\Omega_{\gamma,b}$  of  $k$ -step plans  $\sigma_k = (b, a_1, o_1, \dots, a_k, o_k, b_k)$  in the belief space. For each plan  $\sigma_k \in \Omega_{\gamma,b}$ , the execution is the following process: initially, the execution starts at the belief  $b$  and take the action  $a_1$  specified by  $\gamma$ . Upon receiving an observation  $o$ , the execution moves to the successor belief  $b_a^o = \mathcal{T}_{\mathcal{B}}(b, a_1, o)$  and start executing the conditional plan  $\gamma_o$  for the observation  $o$  specified by the observation strategy  $\nu$  of the conditional plan  $\gamma$ . This process repeats until the execution reaches a terminal belief. The horizon  $h_\gamma = \max_{\sigma \in \Omega_{\gamma,b}} |\sigma|$  of a conditional plan  $\gamma$  is defined as the maximum length of the plans in  $\Omega_{\gamma,b}$ .

**Definition 2.1.6** (Valid Conditional Plan).

A conditional plan  $\gamma$  is *valid* starting from a belief  $b \in \mathcal{B}$  w.r.t. a safe-reachability objective  $\mathcal{G}$  if every plan in  $\Omega_{\gamma,b}$  is valid ( $\Omega_{\gamma,b} \subseteq \Omega_{\mathcal{G}}$ ).

**Definition 2.1.7** ( $k$ -Step Policy).

A  $k$ -step policy  $\pi = \{\gamma_1, \gamma_2, \dots\}$  is a set of conditional plans.  $\pi$  is associated with a set of beliefs  $\mathcal{B}_\pi \subseteq \mathcal{B}$ . Each conditional plan  $\gamma \in \pi$  is associated with a belief  $b \in \mathcal{B}_\pi$  and  $\pi(b) = \gamma$  specifies this conditional plan  $\gamma$  for this belief  $b$ . For a  $k$ -step policy  $\pi$ , the horizon  $h_\pi = \max_{\gamma \in \pi} h_\gamma$  is  $k$ .

Similarly, the  $k$ -step policy  $\pi = \{\gamma_1, \gamma_2, \dots\}$  defines a set  $\Omega_\pi = \bigcup_{b \in \mathcal{B}_\pi} \Omega_{\gamma,b}$  of plans, where  $\gamma$  is the conditional plan specified for the belief  $b$  by the policy  $\pi$ .

**Definition 2.1.8** (Valid  $k$ -Step Policy).

A  $k$ -step policy  $\pi$  is *valid* w.r.t. a safe-reachability objective  $\mathcal{G}$  if every plan in the set  $\Omega_\pi$  is valid ( $\Omega_\pi \subseteq \Omega_{\mathcal{G}}$ ).

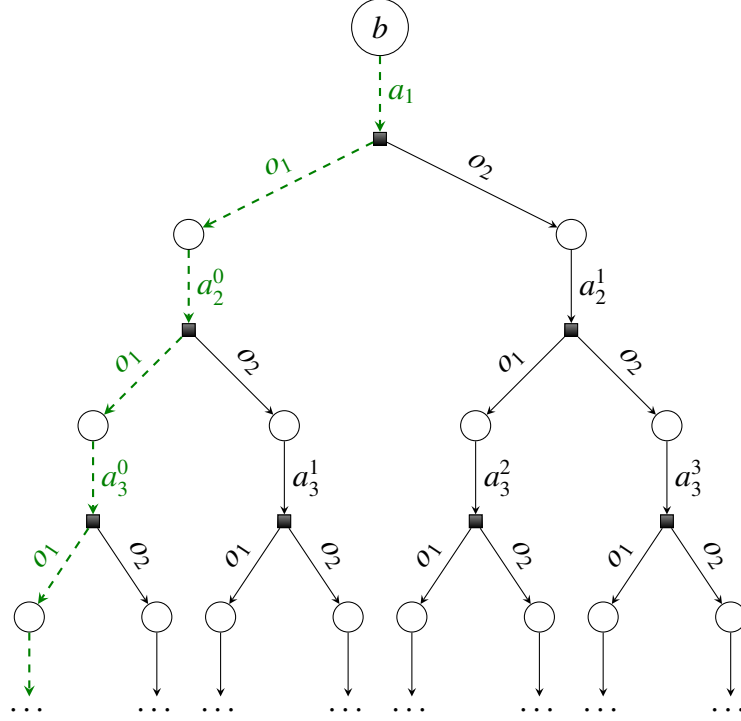


Figure 2.1 : A conditional plan  $\gamma$  for an uncertain domain with 2 observations ( $o_1$  and  $o_2$ ), represented as a tree rooted at the belief  $b$ . Circle nodes represent beliefs, the edges (e.g.,  $a_1, a_2^0, a_2^1, \dots$ ) from circle nodes to rectangle nodes represent actions and the edges ( $o_1$  and  $o_2$ ) from rectangle nodes to circle nodes represent observations.

## 2.2 Relation to POMDPs with Quantitative Objectives

There are two distinct approaches that can model safe-reachability objectives *implicitly* using the existing POMDP models in the literature. The first approach is to incorporate safety and reachability constraints as negative penalties for unsafe states and positive rewards for goal states in *unconstrained* POMDPs with quantitative objectives. However, the authors of [75] have shown a counterexample that demonstrates formulating constraints as unconstrained POMDPs with quantitative objectives does not always yield good policies. The second approach is to encode safe-reachability objectives implicitly as C/RS/CC-POMDPs that extend unconstrained POMDPs with notions of risk and cost [34, 35, 40, 51, 61, 63, 75].

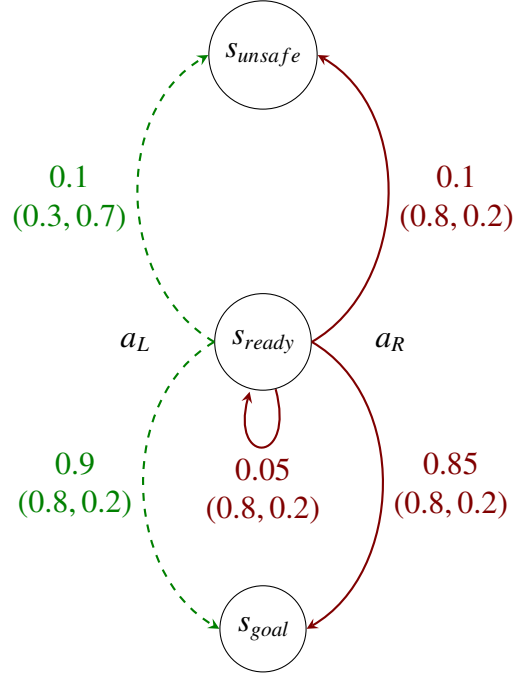


Figure 2.2 : An example to show the difference between POMDPs with safe-reachability objectives and unconstrained/C/RS/CC-POMDPs. There are 3 states: start state  $s_{ready}$ , unsafe state  $s_{unsafe}$  and goal state  $s_{goal}$ . Dashed green edges represent transitions of executing left-hand pick-up action  $a_L$  in state  $s_{ready}$  and solid red edges represent transitions of executing right-hand pick-up action  $a_R$  in state  $s_{ready}$ . For each edge, the first line is the transition probability and the second line is the tuple of observation probabilities  $(p_{o_{pos}}, p_{o_{neg}})$ .

This section shows the differences between POMDPs with safe-reachability objectives and unconstrained/C/RS/CC-POMDPs through an example.

In this simple example, the robot tries to pick up a target cup from the storage area. There are two action choices: pick-up using the left hand (action  $a_L$ ) and pick-up using the right hand (action  $a_R$ ). Both  $a_L$  and  $a_R$  are uncertain, and the robot may hit the storage while executing  $a_L$  or  $a_R$ , which results in an unsafe collision state  $s_{unsafe}$ . There are two possible observations after executing  $a_L$  or  $a_R$ : observation  $o_{pos}$  representing the robot observes a cup in its hand and observation  $o_{neg}$  representing the robot observes no cup in its hand (Note that the actual state may be different from the observation due to uncertainty). The task

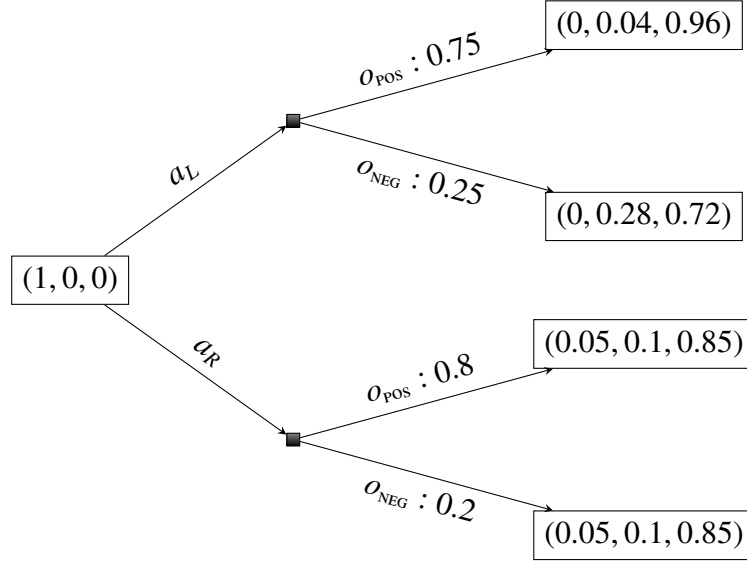


Figure 2.3 : The belief space transition for the POMDP in Figure 2.2. Blue nodes  $(p_{s_{\text{ready}}}, p_{s_{\text{unsafe}}}, p_{s_{\text{goal}}})$  represent beliefs (probability distributions over states), and red nodes represent observation. The edges from blue nodes to red nodes represent actions and the edges from red nodes to blue nodes represent observations and the corresponding probabilities.

objective is to reach a goal state  $s_{\text{goal}}$  where the robot holds a cup in its hand with a probability greater than 0.8 (reachability) while keeping the probability of visiting unsafe state  $s_{\text{unsafe}}$  below the threshold 0.2 (safety). The probability transition and observation functions are shown in Figure 2.2. Based on Formula 2.1, the transition in the corresponding belief space is constructed (see Figure 2.3).

If this problem is modeled as an unconstrained POMDP by assigning a negative penalty  $-P$  ( $P > 0$ ) for unsafe state  $s_{\text{unsafe}}$  and a positive reward  $R$  ( $R > 0$ ) for goal state  $s_{\text{goal}}$ , the optimal action for  $s_{\text{ready}}$  that achieves the maximum reward is always  $a_L$ , no matter what values of  $P$  and  $R$  are. This is because the expected reward of action  $a_L$  ( $0.9R - 0.1P$ ) is greater than the expected reward of  $a_R$  ( $0.85R - 0.1P$ ). However, action  $a_L$  does not satisfy the original safe-reachability objective in the worst case where the robot observing  $o_{\text{neg}}$  after executing action  $a_L$  and the resulting belief state  $(0, 0.28, 0.72)$  violates the original



safety-reachability objective.

If this problem is model this problem as a C/RS/CC-POMDP by assigning a positive reward  $R$  for goal state  $s_{\text{goal}}$  and a cost 1 for visiting unsafe state  $s_{\text{unsafe}}$ , the best action for  $s_{\text{ready}}$  will be  $a_L$  since both  $a_L$  and  $a_R$  satisfies the cost/risk constraint (expected cost/risk  $0.1 < 0.2$ ) and the expected reward of  $a_L$  ( $0.9R$ ) is greater than the expected reward of  $a_R$  ( $0.85R$ ). However, action  $a_L$  violates the original safe-reachability objective for the same reason explained above.

On the contrary, using the formulation of POMDPs with safe-reachability objectives, the best action for  $s_{\text{ready}}$  will be  $a_R$ . This is because, as shown in Definition 2.1.8, a valid policy should satisfy the safe-reachability objective in all possible executions and only  $a_R$  satisfies the safe-reachability objective in every possible execution.

This simple example intends to illustrate that in some domains that require the robot to accomplish the task safely, the formulation of POMDPs with safe-reachability objectives can provide a better guarantee of both safety and reachability than the existing POMDP models. While the formulations of cost/risk as negative penalties in unconstrained POMDPs and expected cost/risk threshold constraints in C/RS/CC-POMDPs are suitable for many applications, there are domains such as autonomous driving and disaster rescue that demand synthesis of policies that can provide such a strong guarantee of reaching goal states safely, especially when violating safety requirements results in irreversible damage to robots.

## Chapter 3

### Offline Synthesis for POMDPs with Safe-Reachability Objectives

This chapter presents an offline policy synthesis algorithm called Bounded Policy Synthesis (BPS) for POMDPs with safe-reachability objectives, first published in [78].

Previous results [10, 50, 57] have shown that the quantitative analysis problem of POMDPs with reachability objectives is undecidable. To make the problem tractable, this thesis assumes there exists a *bounded horizon*  $h$  such that  $h$  is sufficiently large to prove the existence of a valid policy or the user is not interested in plans beyond the bounded horizon  $h$ . This assumption is particularly reasonable for robotic domains because robots are often required to accomplish a task in bounded steps due to some resource constraints such as energy/time constraints. Figure 3.1 shows an example of such a scenario: a robot with uncertain actuation and perception needs to navigate through a kitchen to pick up an object in bounded steps while avoiding collisions with uncertain obstacles.

Like most other algorithms for POMDP policy synthesis, BPS is based on reasoning about the space of *beliefs*, or probability distributions over possible states of the POMDP. The primary algorithmic challenge is that the belief space is a vast, high-dimensional space of probability distributions. To address this challenge, BPS exploits the notion of a *goal-constrained belief space*. This notion takes inspiration from recent advances in point-based algorithms [42, 49, 58, 69] for POMDPs with discounted reward objectives. These POMDP algorithms exploit the notion of the *reachable belief space*  $\mathcal{B}_{b_0}$  from an initial belief  $b_0$  and compute an approximately optimal policy over  $\mathcal{B}_{b_0}$  rather than the entire belief space. Sim-

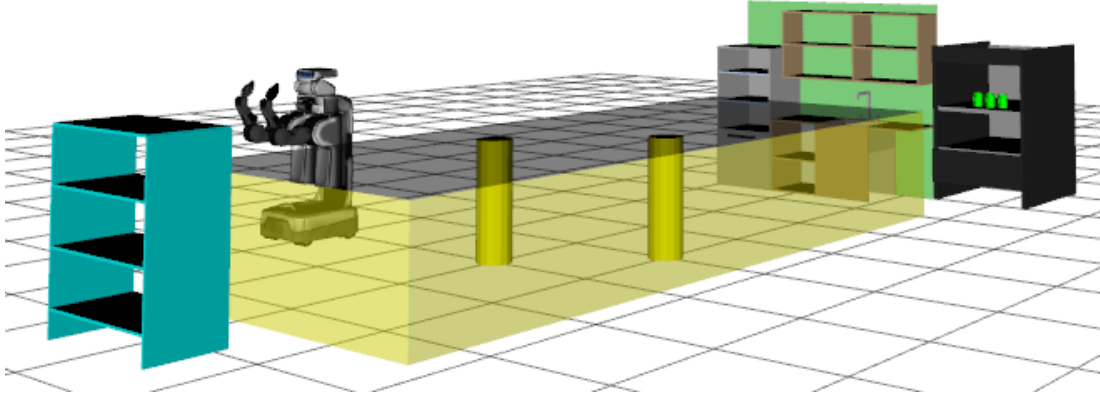


Figure 3.1 : An example of a safe-reachability objective: a robot with uncertain actuation and perception needs to navigate through the kitchen and pick up a green cup from the black storage area (reachability), while avoiding collisions with uncertain obstacles (e.g., chairs) modeled as cylinders in the yellow “shadow” region (safety).

ilarly, BPS computes a valid policy over a *goal-constrained belief space*, which contains beliefs visited by desired executions that can achieve the safe-reachability objective. The goal-constrained belief space is generally much smaller than the original belief space.

BPS computes a valid policy by iteratively searching for a candidate plan in the goal-constrained belief space and constructing a policy from this candidate plan. BPS compactly represents the goal-constrained belief space over a *bounded* horizon using symbolic constraints. The applicability of constraint-based methods has been already advocated in several robotics planning algorithms [17, 39, 54, 80]. Many of these algorithms take advantage of a modern, incremental SMT solver [19] for efficiency. Inspired by this, BPS applies the SMT solver to efficiently explore the symbolic goal-constrained belief space to generate candidate plans. Note that a candidate plan is a single path that only covers a particular observation at each step, while a valid policy is contingent on all possible observations. Therefore, once a candidate plan is found, BPS tries to generate a valid policy from the candidate plan by considering all possible events at each step. If this policy generation

fails, BPS adds additional constraints that *block* invalid plans and force the SMT solver to generate other better plans. The incremental capability of the SMT solver allows BPS to generate alternate candidate plans when the constraints are updated efficiently. If there is no new candidate plan for the current horizon, BPS increases the horizon and repeats the above steps until it finds a valid policy or reaches a given horizon bound.

The scalability of BPS is evaluated using a case study involving a partially observable robotic domain with uncertain obstacles (Figure 3.1). The experimental results demonstrate that BPS can scale up to large belief spaces by focusing on the goal-constrained belief space.

### 3.1 Problem Formulation

#### 3.1.1 Goal-Constrained Belief Space

Computing a full policy that selects an action for every belief is intractable, due to the curse of dimensionality [56]: the belief space  $\mathcal{B}$  is a high-dimensional, continuous space with an infinite number of beliefs. One way to make the problem tractable is to focus on the *reachable belief space*  $\mathcal{B}_{b_0}$  [42, 58], which only contains beliefs reachable from the given initial belief  $b_0$  and is generally much smaller than the original belief space  $\mathcal{B}$ .

The safe-reachability objective  $\mathcal{G}$  defines a set  $\Omega_{\mathcal{G}}$  of plans that satisfy  $\mathcal{G}$ . Combining the reachable belief space  $\mathcal{B}_{b_0}$  and the set  $\Omega_{\mathcal{G}}$  of valid plans defined by the safe-reachability objective  $\mathcal{G}$  defines a *goal-constrained belief space*  $\mathcal{B}_{\mathcal{G}}$  that contains beliefs reachable from the initial belief  $b_0$  under a valid plan  $\sigma \in \Omega_{\mathcal{G}}$ . The *goal-constrained belief space*  $\mathcal{B}_{\mathcal{G}}$  is usually much smaller than the reachable belief space  $\mathcal{B}_{b_0}$ . Thus, computing policies over the goal-constrained belief space  $\mathcal{B}_{\mathcal{G}}$  can lead to a substantial gain in efficiency.

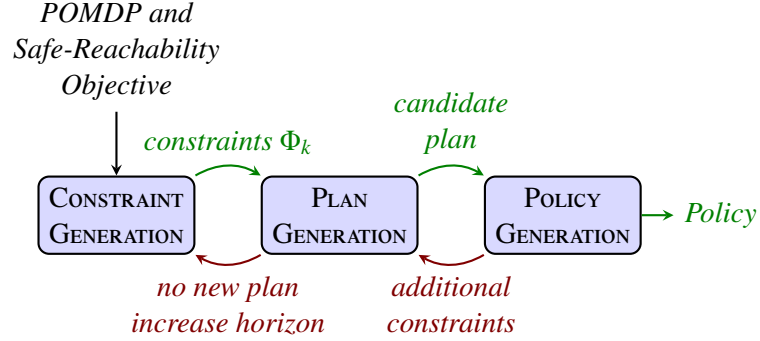


Figure 3.2 : The core steps of the BPS algorithm.

### 3.1.2 Problem Statement

Given a POMDP  $P$ , an initial belief  $b_0$  and a safe-reachability objective  $\mathcal{G}$ , the goal is to synthesize a *valid* policy  $\pi$  over the corresponding goal-constrained belief space  $\mathcal{B}_{\mathcal{G}}$ .

## 3.2 Bounded Policy Synthesis

The core steps of BPS (Algorithm 1) are shown in Figure 3.2. BPS computes a valid policy by iteratively searching for a candidate plan in the goal-constrained belief space  $\mathcal{B}_{\mathcal{G}}$  and constructing a valid policy from this candidate plan. Figure 3.3 graphically depicts one example run of BPS.

First BPS compactly encodes the goal-constrained belief space  $\mathcal{B}_{\mathcal{G}}$  (the black box in Figure 3.3) w.r.t. the given POMDP  $P$ , the initial belief  $b_0$  and the safe-reachability objective  $\mathcal{G}$  over a bounded *horizon*  $k$  as a logical formula  $\Phi_k$  (Algorithm 1, lines 2, 6, 8). More details of the constraints that encode the goal-constrained belief space are discussed in Section 3.2.1.

Then BPS computes a candidate plan by checking the satisfiability of the constraint  $\Phi_k$  (line 10) through a modern, incremental SMT solver [19]. Note that the horizon  $k$  restricts the length of the plan and thus the robot can only execute  $k$  actions.

---

**Algorithm 1: BPS**


---

**Input:**  
 POMDP  $P$   
 Initial Belief  $b_0$   
 Safe-Reachability Objective  $\mathcal{G}$   
 Start Step  $s$   
 Horizon Bound  $h$   
**Output:** A Valid Policy  $\pi$

```

1  $k \leftarrow s;$  /* Initial horizon */
2  $\Phi_k \leftarrow (b_s = b_0);$  /* Initial belief */
3 while  $k \leq h$  do
4    $\sigma_k \leftarrow \emptyset;$  /*  $\sigma_k$ : Candidate plan */
5   /* Add transition at step  $k$  if  $k > s$  */
6   if  $k > s$  then
7      $\Phi_k \leftarrow \Phi_k \wedge (b_k = \mathcal{T}_B(b_{k-1}, a_k, o_k));$ 
8      $\text{push}(\Phi_k);$  /* Push scope */
9     /* Add goal constraints at step  $k$  (Formula 3.1) */
10     $\Phi_k \leftarrow \Phi_k \wedge G(\sigma_k, \mathcal{G}, k);$ 
11    while  $\emptyset = \sigma_k$  do
12      /* Candidate generation */
13       $\sigma_k \leftarrow \text{IncrementalSMT}(\Phi_k);$ 
14      if  $\emptyset = \sigma_k$  then /* No new plan */
15        break;
16      else
17        /*  $\phi$ : constraints for blocking invalid plans */
18         $\pi, \phi = \text{PolicyGeneration}(P, \mathcal{G}, \sigma_k, s + 1, h);$ 
19        if  $\pi = \emptyset$  then /* Generation failed */
20           $\Phi_k \leftarrow \Phi_k \wedge \phi;$ 
21        else
22          return  $\pi;$ 
23         $\sigma_k \leftarrow \emptyset;$ 
24      /* Pop scope: pop goal and  $\phi$  at step  $k$  */
25       $\text{pop}(\Phi_k);$ 
26       $k \leftarrow k + 1;$  /* Increase horizon */
27 return  $\emptyset;$ 

```

---

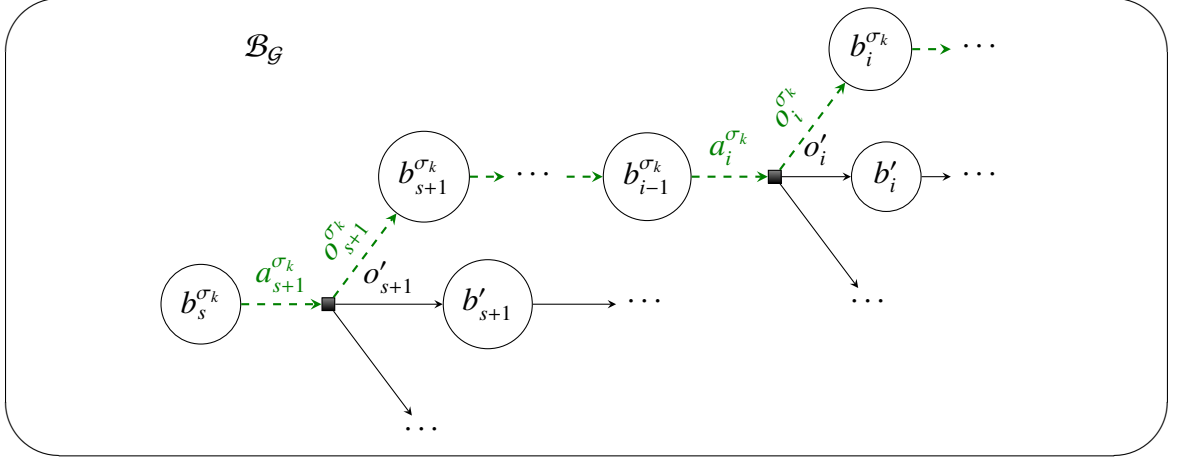


Figure 3.3 : An example run of BPS. The black box represents the goal-constrained belief space  $\mathcal{B}_G$  over the bounded horizon  $k$ . Circle nodes represent beliefs, the edges (e.g.,  $a_{s+1}^{\sigma_k}$ ,  $a_i^{\sigma_k}$ ) from circle nodes to rectangle nodes represent actions and the edges (e.g.,  $o'_{s+1}$ ,  $o'_i$ ) from rectangle nodes to circle nodes represent observations. The dashed green path represents one candidate plan  $\sigma_k$  found by the incremental SMT solver. BPS constructs a policy tree from this candidate plan by considering other branches following the rectangle node for each step.

If  $\Phi_k$  is satisfiable, the SMT solver returns a candidate plan (the dashed green path in Figure 3.3) and BPS tries to generate a valid policy from the candidate plan by considering all possible observations, i.e., other branches following the rectangle node at each step (line 14). If this policy generation succeeds, BPS finds a valid policy. Otherwise, BPS adds additional constraints that block this invalid plan (line 16) and forces the SMT solver to generate another better candidate.

If  $\Phi_k$  is unsatisfiable and thus there is no new plan for the current horizon, BPS increases the horizon by one (line 21) and repeats the above steps until a *valid* policy is found (line 18) or a given horizon bound  $h$  is reached (line 3).

This incremental SMT solver [19] can efficiently generate alternate candidate plans by maintaining a *stack* of *scopes*, where each scope is a container for a set of constraints and the corresponding “knowledge” learned from this set of constraints. For fast repeated sat-

isfiability checks, when updating constraints (lines 2, 6, 8, 16), rather than rebuilding the “knowledge” from scratch, the incremental SMT solver only changes the “knowledge” related to the updates by pushing (line 7) and popping (line 20) scopes. Thus the “knowledge” learned from previous satisfiability checks can be reused.

### 3.2.1 Constraint Generation

In the first step, BPS uses an encoding from Bounded Model Checking (BMC) [3] to construct the constraint  $\Phi_k$  representing the goal-constrained belief space  $\mathcal{B}_{\mathcal{G}}$  w.r.t. the POMDP  $P$ , the initial belief  $b_0$  and the safe-reachability objective  $\mathcal{G}$  over the bounded horizon  $k$ . The idea behind BMC is to find a finite plan with increasing horizon that satisfies the given safe-reachability objective.

The constraint  $\Phi_k$  contains three parts:

1. Starting from the initial belief (line 2):  $b_s = b_0$ .
2. Unfolding of the transition up to the horizon  $k$  (line 6):  $\bigwedge_{i=s+1}^k (b_i = \mathcal{T}_B(b_{i-1}, a_i, o_i))$ .
3. Satisfying the safe-reachability objective  $\mathcal{G}$  (line 8).

A safe-reachability objective can be represented as a constraint  $G(\sigma_k, \mathcal{G}, k)$  on bounded plans  $\sigma_k = (b_s, a_{s+1}, o_{s+1}, \dots, a_k, o_k, b_k)$  using the rules provided by BMC [3] as follows:

$$G(\sigma_k, \mathcal{G}, k) = \bigvee_{i=s}^k (b_i \in Dest \wedge (\bigwedge_{j=s}^{i-1} (b_j \in Safe))) \quad (3.1)$$

For a safe-reachability objective  $\mathcal{G}$  with a set  $Dest$  of goal beliefs and a set  $Safe$  of safe beliefs, a finite plan that visits a goal belief while staying in the safe region is sufficient to satisfy  $\mathcal{G}$ . Therefore, it is sufficient to specify that a bounded plan with length  $k$  eventually



visits a belief  $b_i \in Dest$  while staying in the safe region ( $\bigwedge_{j=s}^{i-1} (b_j \in Safe)$ ), as shown in Formula 3.1.

### 3.2.2 Plan Generation

The next step is to generate a candidate plan  $\sigma_k$  of length  $k$  that satisfies the constraint  $\Phi_k$ . BPS applies an incremental SMT solver to efficiently search for such a candidate in the *goal-constrained belief space*  $\mathcal{B}_{\mathcal{G}}$  defined by  $\Phi_k$  (line 10). If  $\Phi_k$  is unsatisfiable, there is no bounded plan  $\sigma_k$  for the current horizon. In this case, BPS increases the horizon (line 21). If  $\Phi_k$  is satisfiable, the SMT solver will return a satisfying model that assigns concrete values  $b_i^{\sigma_k}$ ,  $a_{i+1}^{\sigma_k}$  and  $o_{i+1}^{\sigma_k}$  for the belief  $b_i$ , action  $a_{i+1}$  and observation  $o_{i+1}$  at each step  $i$  respectively, which can be used to construct the candidate plan  $\sigma_k = (b_s^{\sigma_k}, a_{s+1}^{\sigma_k}, o_{s+1}^{\sigma_k}, b_{s+1}^{\sigma_k}, \dots, a_k^{\sigma_k}, o_k^{\sigma_k}, b_k^{\sigma_k})$ .

### 3.2.3 Policy Generation

Plan generation returns a candidate plan  $\sigma_k$  (the dashed green path in Figure 3.3) that satisfies the safe-reachability objective  $\mathcal{G}$ . This candidate plan is a single path that only covers a particular observation  $o_i^{\sigma_k}$  at each step  $i$ . A *valid* policy should also consider other possible observations  $o'_i \neq o_i^{\sigma_k}$ , i.e., other branches following the rectangle node for each step  $i$ . *Policy generation* (Algorithm 2) tries to construct a valid policy from a candidate plan by considering all possible observations at each step.

For a candidate plan  $\sigma_k$ , BPS processes each step of  $\sigma_k$ , starting from the last step (Algorithm 2, line 24). For each step  $i$ , since the set of observations  $\mathcal{O}$  is finite, BPS enumerates every possible observation  $o'_i \neq o_i^{\sigma_k}$  (line 25) and compute the next belief  $b'_i$  using the transition function (line 26). To ensure the action  $a_i^{\sigma_k}$  also works for this different observation  $o'_i$ , BPS computes a *valid* policy for the branch starting from  $b'_i$ , which is another

---

**Algorithm 2: PolicyGeneration**


---

**Input:**  
 POMDP  $P$   
 Safe-Reachability Objective  $\mathcal{G}$   
 Candidate Plan  $\sigma_k = (b_s^{\sigma_k}, a_{s+1}^{\sigma_k}, o_{s+1}^{\sigma_k}, b_{s+1}^{\sigma_k}, \dots, b_k^{\sigma_k})$   
 Start Step  $s$   
 Horizon Bound  $h$   
**Output:** A *Valid* Policy  $\pi$  and Constraints  $\phi$  for blocking invalid plans if the input candidate plan is invalid

```

23  $\pi \leftarrow \emptyset$ ;
24 for  $i = k$  downto  $s$  do
25   foreach observation  $o \in \mathcal{O} - \{o_i^{\sigma_k}\}$  do
26      $b'_i \leftarrow \mathcal{T}_{\mathcal{B}}(b_{i-1}^{\sigma_k}, a_i^{\sigma_k}, o)$ ;
27      $\pi' \leftarrow \text{BPS}(P, b'_i, \mathcal{G}, i, h)$ ;
28     if  $\emptyset = \pi'$  then
29        $\phi$  using Formula 3.2
30       return  $\emptyset, \phi$ ;
31      $\pi \leftarrow \pi \cup \pi'$ 
32      $\pi(b_{i-1}^{\sigma_k}) \leftarrow a_i^{\sigma_k}$ ;
33 return  $\pi, \emptyset$ ;

```

---

BPS problem and can be solved using Algorithm 1 (line 27).

If BPS successfully constructs the valid policy  $\pi'$  for this branch, BPS adds  $\pi'$  to the policy  $\pi$  for the original synthesis problem (line 31). Otherwise, this candidate plan  $\sigma_k$  can not be an element of a *valid* policy  $\sigma_k \notin \Omega_\pi$ . In this case, the prefix of the candidate plan  $(b_s^{\sigma_k}, a_{s+1}^{\sigma_k}, o_{s+1}^{\sigma_k}, \dots, b_{i-1}^{\sigma_k}, a_i^{\sigma_k})$  is invalid for current horizon  $k$  and BPS adds additional

constraints  $\phi$  to block all invalid plans that have this prefix (line 29):

$$\phi = \neg ((b_s = b_s^{\sigma_k}) \wedge (a_i = a_i^{\sigma_k}) \wedge \left( \bigwedge_{m=s+1}^{i-1} (a_m = a_m^{\sigma_k}) \wedge (o_m = o_m^{\sigma_k}) \wedge (b_m = b_m^{\sigma_k}) \right)) \quad (3.2)$$

Note that  $\phi$  is only valid for current horizon  $k$  and when the horizon increases, BPS *pops* the scope related to the additional constraints  $\phi$  from the stack of the SMT solver (line 20) so that those plans with this prefix with the increased horizon can be *revisit*. If BPS successfully constructs policies for all other branches at step  $i$ , the choice of action  $a_i^{\sigma_k}$  for belief  $b_{i-1}^{\sigma_k}$  is valid for all possible observations. Then BPS records this choice for belief  $b_{i-1}^{\sigma_k}$  in the policy (line 32). This policy generation terminates when it reaches the start step  $s$  as stated in the for-loop (line 24) or it fails to construct the valid policy  $\pi'$  for a branch (line 28).

### 3.3 Algorithm Analysis

#### 3.3.1 Algorithm Complexity

The *reachable belief space*  $\mathcal{B}_{b_0}$  can be seen as a tree where the root node is the initial belief  $b_0$  and at each node, the tree branches on every action and observation. The given horizon bound  $h$  limits the height of the tree. Therefore, the reachable belief space  $\mathcal{B}_{b_0}$  of height  $h$  contains  $O(|\mathcal{A}|^h |\mathcal{O}|^h)$  plans, where  $|\mathcal{A}|$  and  $|\mathcal{O}|$  are the size of the action set  $\mathcal{A}$  and the size of observation set  $\mathcal{O}$  respectively. To synthesize a *valid* policy, a naive approach that checks every plan in the reachable belief space  $\mathcal{B}_{b_0}$  requires  $O(|\mathcal{A}|^h |\mathcal{O}|^h)$  calls to the SMT solver. This exponential growth of the reachable belief space  $\mathcal{B}_{b_0}$  due to branches on both action and observation is a major challenge for synthesizing a *valid* policy.

In contrast, BPS exploits the notion of *goal-constrained belief space*  $\mathcal{B}_{\mathcal{G}}$  and efficiently explores the *goal-constrained belief space*  $\mathcal{B}_{\mathcal{G}}$  by leveraging an incremental SMT solver to generate a candidate plan  $\sigma$  of length at most  $h$ . This candidate plan fixes the choice of actions at each step and thus the policy generation process only needs to consider the branches on observations for each step, as shown in Figure 3.3. Therefore, BPS requires  $O(I|O|^h)$  calls to the SMT solver, where  $I$  is the number of interactions between plan generation and policy generation, while the naive approach described above requires  $O(|\mathcal{A}|^h|O|^h)$  SMT solver calls. In general,  $I$  is often much smaller than  $|\mathcal{A}|^h$ , which leads to much faster policy synthesis.

### 3.3.2 Observability

As discussed in the previous section (Section 3.3.1), the number of plans checked by BPS is  $O(I|O|^h)$  calls to the SMT solver, where  $I$  is the number of interactions between plan generation and policy generation. Thus, the observability of the domain, i.e., the capability of the robot’s perception, plays an important role in the performance of BPS.

In one extreme case, if the robot has perfect perception and the domain is fully observable, the robot knows the exact current state during execution. Then the underlying model becomes an MDP rather than a POMDP, and there is a large body of works [14, 15, 20, 24, 44, 46–48, 77, 82] that can deal with MDP with boolean objectives as discussed in Section 1.2.2). A fully observable domain can also be modeled as a POMDP  $P = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{O}, \mathcal{Z}, r)$  where the set of observations is equivalent to the state space, i.e.,  $\mathcal{O} = \mathcal{S}$ , and the probabilistic observation function is  $\mathcal{Z}(s', a, o) = 0$  for every  $s' \neq o$  and  $\mathcal{Z}(s', a, o) = 1$  when  $s' = o$ . To solve this fully observable domain, BPS requires  $O(I|\mathcal{S}|^h)$  calls to the SMT solver, where  $I$  is the number of interactions between plan generation and policy generation. For complex robot tasks, the state space  $\mathcal{S}$  is usually very large. As an

example, the size of the state space for the kitchen domain (Figure 3.1) with  $N$  regions and  $M$  obstacles is  $|S| = C(N, M) \cdot N$ , where  $C(N, M)$  is the number of  $M$ -combinations from the set of  $N$  regions. For  $N = 24$  and  $M = 4$ , there are more than  $10^5$  states. Therefore, BPS may check a huge number of plans in order to compute a valid policy due to a high dimensional observation space and thus may not scale well for fully observable domains.

In another extreme case, if the robot has no perception and the domain is unobservable, the robot has no information about the current state. Then the underlying model becomes an unobservable MDP (UMDP) [53] rather than a POMDP. A UMDP can be modeled as a POMDP  $P = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{O}, \mathcal{Z}, r)$  where the set of observations  $\mathcal{O} = \{o\}$  contains only one element (the same observation  $o$  is received for every state) and the probabilistic observation function is constant, i.e.,  $\mathcal{Z}(s', a, o) = \frac{1}{|\mathcal{S}|}$  for all  $s' \in \mathcal{S}$ . To solve this unobservable domain, BPS only requires  $O(I)$  calls to the SMT solver, where  $I$  is the number of interactions between plan generation and policy generation. This is because for unobservable domains, there is no branching on observations and a policy is just a plan. Therefore, BPS only checks a small number of plans in order to compute a valid policy and thus scales well for unobservable domains.

For general cases where the domain is partially observable, BPS requires  $O(I|\mathcal{O}|^h)$  calls to the SMT solver, where  $I$  is the number of interactions between plan generation and policy generation, as discussed in Section 3.3.1. Therefore, BPS is usually effective for POMDPs with a restricted partially observable component, but may not scale well for POMDPs with high-dimensional/continuous observation space.

### 3.4 Experiments

BPS is evaluated in a partially observable kitchen domain (Figure 3.1) with a PR2 robot and  $M$  uncertain obstacles placed in the yellow “shadow” region. The task for the robot is

to pass the yellow “shadow” region safely avoiding collisions with uncertain obstacles and eventually pick up a green cup from the black storage area.

The kitchen environment is first discretized into  $N$  regions. The locations of the obstacles are uniformly distributed among the regions in the yellow “shadow” region and there is at most one obstacle in each region. The robot starts at a known initial location. However, due to the robot’s imperfect perception, the locations of the robot, the locations of uncertain obstacles, and the location of the target cups are all partially observable during execution.

In this domain, the actuation and perception of the robot are imperfect. There are ten uncertain robot actions ( $|\mathcal{A}| = 10$ ):

1. Four *move* actions that move the robot to an adjacent region in four directions: including *move-north*, *move-south*, *move-west* and *move-east*. *Move* actions could fail with a probability  $p_{\text{fail}}$ , resulting in no change in the state.
2. Four *look* actions that observe a region to see whether there is an obstacle in that region, including *look-north*, *look-south*, *look-west*, *look-east* (look at the adjacent region in the corresponding direction). When the robot calls *look* to observe a particular region <sub>$i$</sub> , it may either make an observation  $o = o_{\text{pos}}$  representing the robot observes an obstacle in region <sub>$i$</sub>  or  $o = o_{\text{neg}}$  representing the robot observes no obstacle in region <sub>$i$</sub> . The probabilistic observation function  $\mathcal{Z}(s', a, o)$  for *look* actions is defined based on the false positive probability  $p_{\text{fp}}$  and the false negative probability  $p_{\text{fn}}$ .
3. Two *pick-up* actions that pick up an object from the black storage area: pick-up using the left hand  $a_L$  and pick-up using the right hand  $a_R$ . The model of *pick-up* actions is the same as that discussed in Section 2.2 (see Figure 2.2).

The task shown in Figure 3.1 can be specified as a safe-reachability objective with a set

*Dest* of goal beliefs and a set *Safe* of safe beliefs, defined as follows:

$$\begin{aligned} Dest &= \{b \in \mathcal{B} \mid \left(\sum b(\text{target cup in robot's hand})\right) > 1 - \delta_1\} \\ Safe &= \{b \in \mathcal{B} \mid \left(\sum b(\text{robot in collision})\right) < \delta_2\} \end{aligned} \quad (3.3)$$

where  $\delta_1$  and  $\delta_2$  are small values that represent tolerance. The reachability objective specifies that in a goal belief, the probability of having the target cup in the robot's hand should be greater than the threshold  $1 - \delta_1$ . The safety objective specifies that in a safe belief, the probability of the robot in collision (the robot and one obstacle in the same region) should be less than the tolerance  $\delta_2$ .

The goal is to generate a valid policy for the robot that is guaranteed to achieve the safe-reachability objective, even with uncertain actuation and perception.

### 3.4.1 Performance

The performance of BPS is evaluated using test cases of the kitchen domain with various numbers of obstacles. Z3 [19] is used as the backend incremental SMT solver. All experiments were conducted on a 3.0GHz Intel® processor with 32GB memory. For all the tests, the horizon bound is  $h = 20$  and the number of regions in the kitchen environment is  $N = 24$ .

To evaluate the gains from incremental solving, BPS is tested in two settings: with and without incremental solving. Note that when incremental solving is disabled, each call to the SMT solver requires solving the SMT constraints from scratch, rather than reusing the results from the previous SMT solver calls. Figure 3.4 shows the performance results of BPS with and without incremental solving. As shown in Figure 3.4, enabling incremental solving in BPS leads to a performance improvement in policy synthesis. This is because

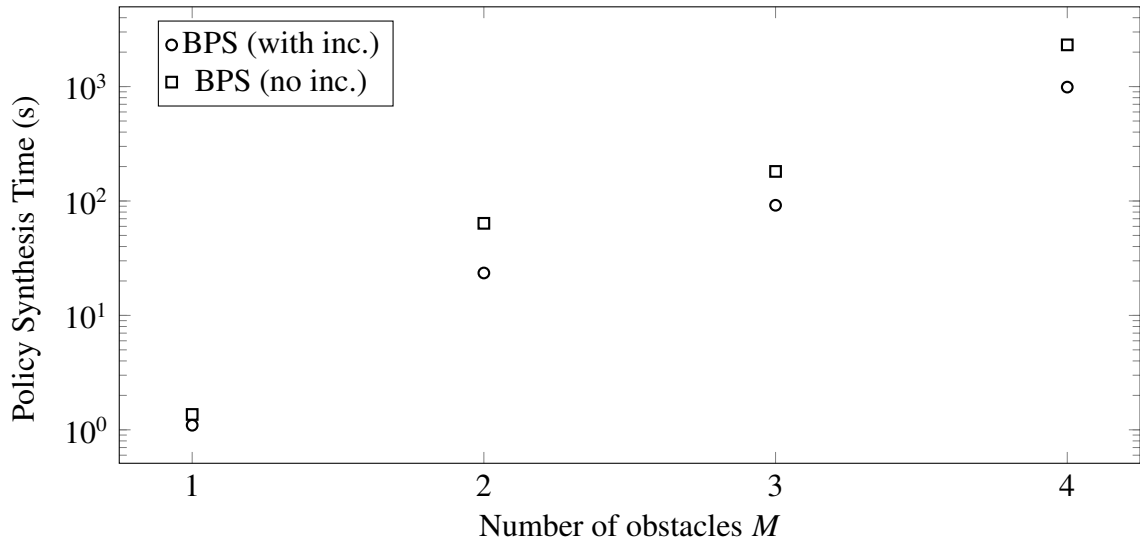


Figure 3.4 : Performance of BPS as the number of obstacles  $M$  varies. The plot of circles shows the performance of BPS with incremental solving and the plot of squares shows the performance of BPS without incremental solving.

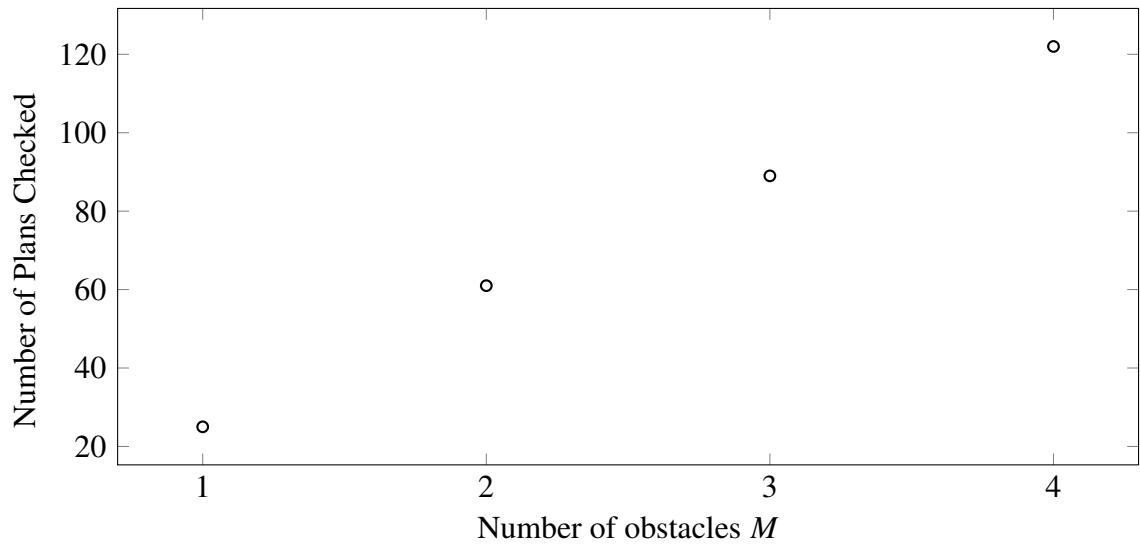


Figure 3.5 : The number of plans checked (i.e., the number of SMT calls) by BPS during policy synthesis as the number of obstacles  $M$  varies.

the BMC encoding [3] used in BPS is particularly suitable for incremental solving since increasing horizon and blocking invalid plans correspond to pushing/popping constraints.



To demonstrate the gains from utilizing the goal-constrained belief space compared to a simple exhaustive search in the reachable belief space, this thesis first estimates the number of plans in the reachable belief space. There is no observation branching for the four *move* actions, and there are two observation branches for the four *look* actions. The two *pick-up* actions can be ignored since these two actions are not available in every step and can only be performed when the robot is fairly confident that it is in the position where it is ready to pick up a cup from the black storage area. Therefore, the approximate lower bound of the number of plans in the reachable belief space with at most  $h = 20$  steps is  $(4 + 4 \times 2)^{20} \approx 10^{21}$ . However, as shown in Figure 3.5, for the largest test, the number of plans checked (around 120) in BPS is very small compared to the number of plans in the reachable belief space. These results show that BPS can solve problems in large reachable belief spaces with a small number of SMT solver calls by focusing on the goal-constrained belief space.

However, Figure 3.4 also shows that the synthesis time grows exponentially as the number of obstacles increases, which matches the complexity analysis in Section 3.3.1. This is because the current implementation of BPS operates on an *exact* tree representation of policies with all the observation branches. As the number of obstacles increases, both the horizon bound (the height of the policy tree) and the size of the state space (the belief space dimension) increase, which leads to exponential growth of plans in the policy tree and makes the policy synthesis problem much harder.

### 3.4.2 Horizon Bound

To evaluate how BPS performs with different horizon bounds, BPS is tested in the kitchen domain with  $M = 2$  obstacles. Figure 3.6 shows the performance results of BPS as the horizon bound  $h$  increases. For the kitchen domain with  $M = 2$  obstacles, it requires at

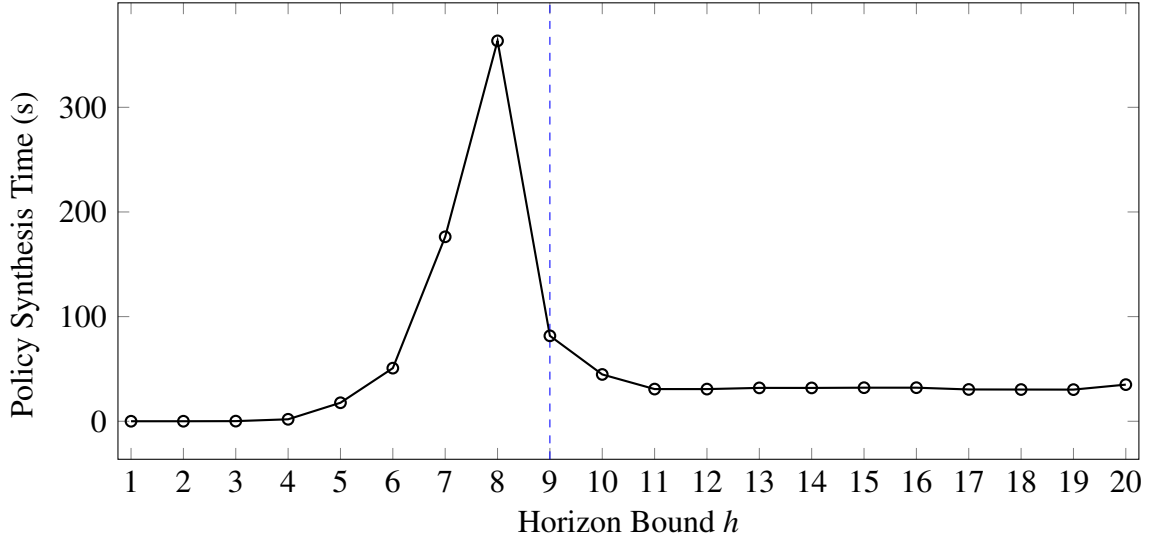


Figure 3.6 : Performance of BPS for the kitchen domain with  $M = 2$  obstacles as the horizon bound  $h$  increases. The blue dashed line is the plot  $h_{\min} = 9$ .

least  $h_{\min} = 9$  (the blue dashed line in Figure 3.6) steps to complete the tasks. As shown in Figure 3.6, when the horizon bound  $h < h_{\min}$ , there is no valid policy within the horizon bound, and BPS needs to explore every plan in the goal-constrained belief space in order to confirm non-existence of valid policies. Therefore, the computation time of BPS grows rapidly as the horizon bound  $h$  increases when the horizon bound  $h < h_{\min}$ . When the horizon bound  $h \geq h_{\min}$ , the computation time of BPS does not change much as the horizon bound  $h$  increases. This is because BPS stops increasing the horizon after a valid policy is found as shown in Algorithm 1. Therefore, when the horizon bound  $h \geq h_{\min}$ , increasing the horizon bound does not affect the performance of BPS.

### 3.4.3 Physical Validation

BPS is validated on a Fetch robot for the domain shown in Figure 3.7. The setup of this domain is similar to the kitchen domain. The Fetch needs to pick up a target object (the blue can on the table) while avoiding collisions with uncertain obstacles such as floor signs and

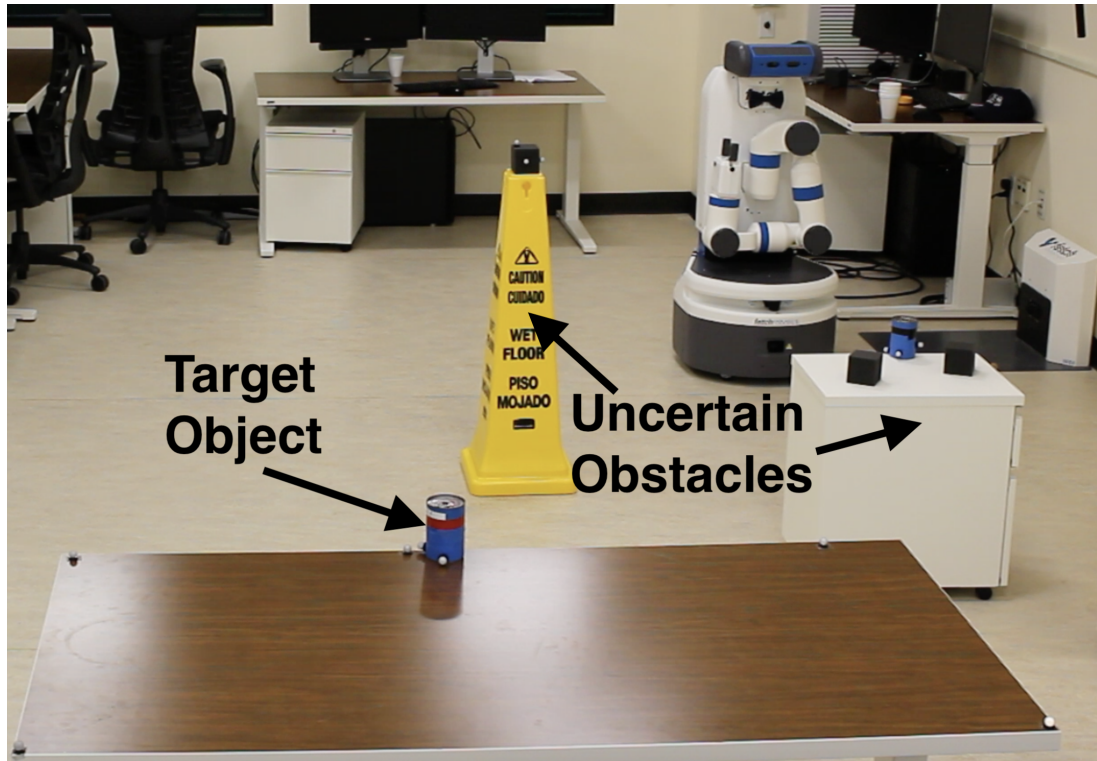


Figure 3.7 : An example uncertain domain with safe-reachability objective: a robot with imperfect actuation and perception needs to navigate through an office to pick up the blue can from the table, while avoiding collisions with uncertain obstacles such as floor signs and file cabinets.

file cabinets, which can be placed in different locations. The POMDP's state space consists of robot locations and object locations. A Vicon system is used to detect object locations, which is usually accurate but can still produce false negative and false positive due to occlusion or inappropriate Vicon marker configurations on objects. The false negative and false positive probabilities can be estimated by counting the false negative and false positive events during 100 Vicon detections. The POMDP's probabilistic observation function is defined based on the false negative and false positive probabilities. Sometimes the Fetch may fail to move its base when given a *move* action command and stay in the same place. The failure probability of these move actions can be estimated by counting the failure

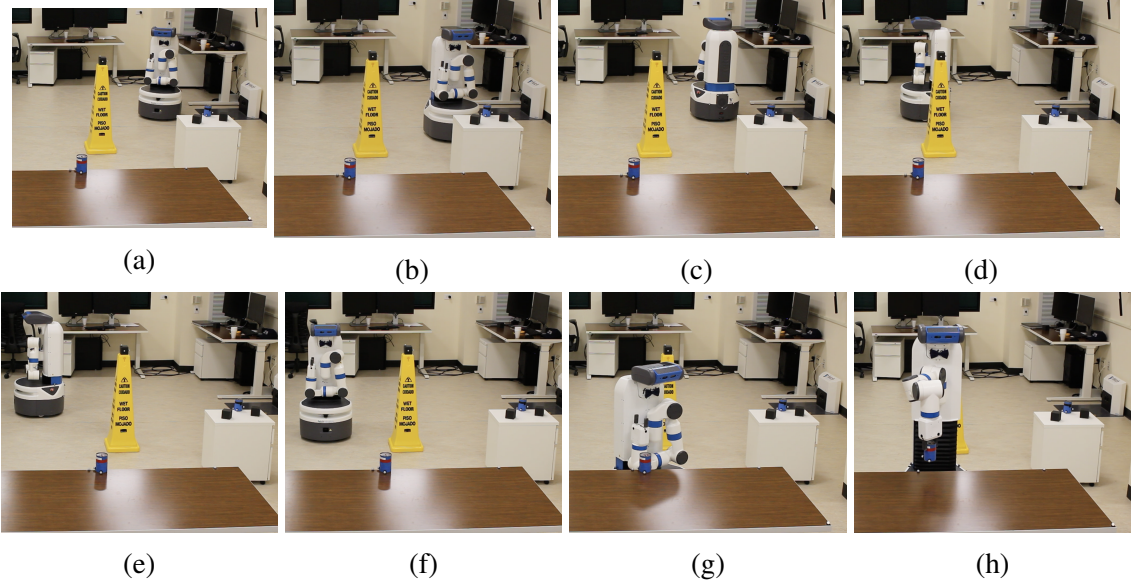


Figure 3.8 : Physical validation of BPS for the domain shown in Figure 3.7.

events during 100 *move* action executions. The POMDP's probabilistic transition function is defined based on this failure probability.

Figure 3.8a, 3.8b, 3.8c, 3.8d, 3.8e, 3.8f, 3.8g, and 3.8h show the execution of the policy constructed by BPS. BPS produces a policy that achieves the given safe-reachability objective.

### 3.5 Discussion

In this chapter, an offline policy synthesis method called BPS is presented for POMDPs with safe-reachability objectives. BPS exploits the notion of a goal-constrained belief space to improve computational efficiency. BPS constructs constraints in a way similar to Bounded Model Checking [3] to compactly represent the goal-constrained belief space, which can be efficiently explored through an incremental Satisfiability Modulo Theories solver [19]. BPS is evaluated in an uncertain robotic domain, and the results show that

BPS can synthesize policies for large problems by focusing on the goal-constrained belief space.

The current implementation of BPS operates on an *exact* representation of the policy (the tree structure shown in Figure 3.3). As a result, BPS suffers from the exponential growth as the horizon increases. An important ongoing question is how to represent the policy approximately while preserving correctness. Another issue arises from the discrete representations (discrete POMDPs) used in BPS. While many robot tasks can be modeled using these representations, discretization often suffers from the “curse of dimensionality”. Investigating how to deal with continuous state spaces and continuous observations directly without discretization is another promising future direction for this work and its application in robotics.

## Chapter 4

### Online Planning for POMDPs with Safe-Reachability Objectives

This chapter presents an online policy synthesis algorithm called Online Partial Policy Synthesis (OPPS) for POMDPs with safe-reachability objectives, first published in [79].

The BPS method presented in Chapter 3 is an *offline* synthesis method that computes a full policy before execution. Another category of approaches to planning under uncertainty is *online planning* that interleaves planning and execution [6, 31, 38, 39, 49, 64, 69]. The choice of the approach depends on the problem at hand. Offline synthesis provides a strong correctness guarantee, but it is difficult to scale. Online planning is much more scalable but makes it hard to ensure correctness. Online planning works well for domains where replanning is likely to succeed but often fails for domains where replanning is difficult or infeasible in some states.

This chapter presents an online planning method, OPPS, to scale up the previous BPS approach further through online planning and achieve a good balance between efficiency and correctness. OPPS is based on the notion of *partial policies*, which only contain a sampled subset of all observation branches at each step and approximate full policies. This idea of partial policies resembles the state-of-the-art online POMDP algorithm based on Determinized Sparse Partially Observable Tree (DESPOT) [6, 69]. Both DESPOT and partial policies contain a subset of all possible observations to improve efficiency. There are two major differences between OPPS and DESPOT: first, DESPOT handles POMDPs with (discounted) rewards while OPPS solves POMDPs with safe-reachability objectives. Sec-

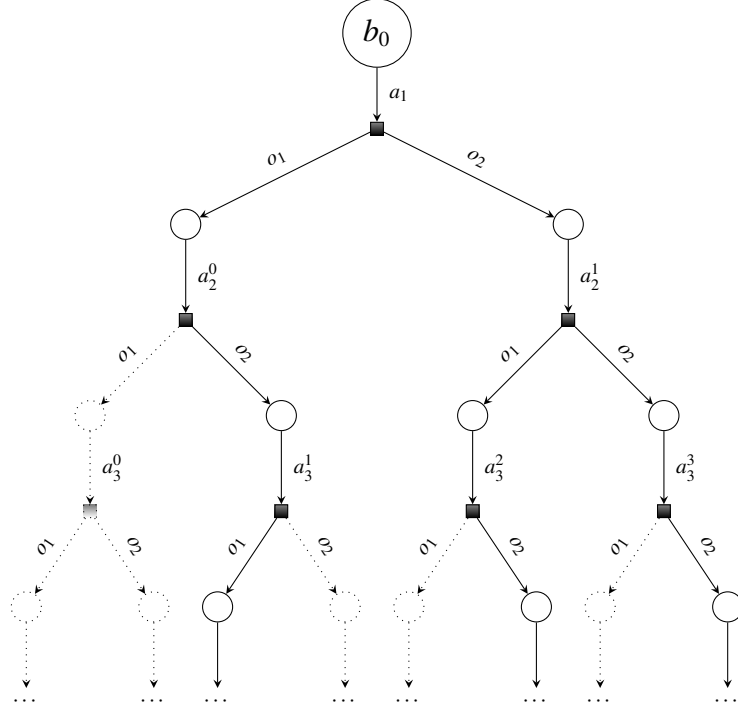


Figure 4.1 : A  $k$ -step partial policy  $\pi^p$  for an uncertain domain with 2 observations ( $o_1$  and  $o_2$ ), represented as a partial tree rooted at the initial belief  $b_0$  (including only solid branches). Circle nodes represent beliefs, the edges (e.g.,  $a_1, a_2^0, a_2^1, \dots$ ) from circle nodes to rectangle nodes represent actions, and the edges ( $o_1$  and  $o_2$ ) from rectangle nodes to circle nodes represent observations.

ond, DESPOT contains all action branches while OPPS constructs partial policies (Figure 4.1) that only contains one action per step, which is part of the desired execution satisfying the safe-reachability objective.

OPPS computes a partial policy parameterized by a *replanning probability*, which is the probability of the event that the robot receives an observation not covered by the partial policy, thus requiring replanning. Moreover, this chapter provides a theoretical analysis of this replanning probability framework, showing that the probability of the constructed partial policy failing is bounded by the replanning probability. OPPS allows users to specify an appropriate bound on the replanning probability to balance efficiency and correctness:

for domains where replanning is likely to succeed, users can increase the bound for efficiency; on the other hand, for domains where replanning is difficult or infeasible in some states, users can decrease the bound and allocate more computational resources to achieve a higher success rate.

To further improve performance, OPPS updates the replanning probability bound instead of using the same bound during partial policy construction. This bound update step enables quicker detection of the current partial policy meeting the bound and avoids unnecessary computation. To offer better safety guarantees, once a partial policy is constructed, OPPS checks whether the immediate successor belief of every uncovered observation of the constructed partial policy satisfies the safety requirement. Thus OPPS guarantees that the robot still satisfies the safety requirement when replanning fails. Section 4.2.2 has more details on the bound update step and safety guarantees.

OPPS is evaluated in the uncertain mobile manipulation domain presented in [78] and the Tag domain [58]. OPPS is also validated on a Fetch robot for the domain shown in Figure 3.7. The results demonstrate that OPPS scales better than BPS and can solve problems that are beyond the capabilities of BPS within the time limit.

## 4.1 Problem Formulation

### 4.1.1 Partial Policy

Computing an exact policy that selects an action for every belief is intractable, due to the curse of dimensionality [56]: the belief space  $\mathcal{B}$  is a high-dimensional, continuous space with an infinite number of beliefs. One way to make the problem tractable is to focus on the *reachable belief space*  $\mathcal{B}_{b_0}$  [42, 58], which only contains beliefs reachable from the initial belief  $b_0$  and is generally much smaller than the original belief space  $\mathcal{B}$ . Therefore, instead



of computing a policy  $\pi : \mathcal{B} \mapsto \mathcal{A}$  over the entire belief space, these methods [42, 58] only compute a policy  $\pi_{\mathcal{B}_{b_0}} : \mathcal{B}_{b_0} \mapsto \mathcal{A}$  over the reachable belief space.

Chapter 3 has shown that the performance of policy synthesis for POMDPs with safe-reachability objectives can be further improved based on the notion of a *goal-constrained belief space*  $\mathcal{B}_{\mathcal{G}}$ . The goal-constrained belief space  $\mathcal{B}_{\mathcal{G}}$  combines the reachable belief space  $\mathcal{B}_{b_0}$  and the set  $\Omega_{\mathcal{G}}$  of valid plans defined by the safe-reachability objective  $\mathcal{G}$ , which only contains beliefs reachable from the initial belief  $b_0$  under a valid plan  $\sigma \in \Omega_{\mathcal{G}}$ . In general,  $\mathcal{B}_{\mathcal{G}}$  is much smaller than the reachable belief space  $\mathcal{B}_{b_0}$ . Thus, BPS only computes a policy  $\pi_{\mathcal{B}_{\mathcal{G}}} : \mathcal{B}_{\mathcal{G}} \mapsto \mathcal{A}$  over the *goal-constrained belief space* to improve efficiency. This policy  $\pi_{\mathcal{B}_{\mathcal{G}}}$  is essentially a set  $\Gamma$  of conditional plans [33] in Definition 2.1.5. Figure 2.1 shows an example of a conditional plan  $\gamma = (a_1, \nu)$  represented as a tree rooted at the initial belief  $b_0$ .

It is clear that the number of valid plans in a valid conditional plan  $\gamma$  grows exponentially as the horizon  $h_{\gamma}$  increases. To address this challenge, this thesis introduces *partial* conditional plans that only contains a small number of valid plans to approximate full conditional plans:

**Definition 4.1.1** (Partial Conditional Plan).

A  $k$ -step *partial* conditional plan is a tuple  $\gamma^p = (a, O^p, \nu^p)$ , where  $a \in \mathcal{A}$  is an action,  $O^p \subseteq \mathcal{O}$  is a subset of the observation set  $\mathcal{O}$ , and  $\nu^p$  is a *partial observation strategy* that maps an observation  $o \in O^p$  to a partial conditional plan  $\gamma^{p'}$ .

Similar to a full conditional plan defined in Section 2.1.3, a partial conditional plan  $\gamma^p = (a_1, O^p, \nu^p)$  together with a belief  $b$  defines a set  $\Omega_{\gamma^p, b}$  of  $k$ -step plans  $\sigma_k = (b, a_1, o_1, \dots, a_k, o_k, b_k)$  in belief space. For each plan  $\sigma_k \in \Omega_{\gamma^p, b}$ , the execution is the following process: initially, the execution starts at the belief  $b$  and take the action  $a_1$  specified by the partial conditional plan  $\gamma^p$ . Upon receiving the observation  $o_1 \in O^p$ , the execution moves to the successor belief  $b_1 = \mathcal{T}_{\mathcal{B}}(b, a_1, o_1)$  and start executing the conditional plan

$\gamma_{o_1}^p$  for the observation  $o_1$  specified by the observation strategy  $\nu^p$  of the partial conditional plan  $\gamma^p$ . This process repeats until the execution reaches a terminal belief. The horizon  $h_{\gamma^p} = \max_{\sigma \in \Omega_{\gamma^p, b}} |\sigma|$  of a partial conditional plan  $\gamma^p$  is defined as the maximum length of the plans in  $\Omega_{\gamma^p, b}$ .

**Definition 4.1.2** (Valid Partial Conditional Plan).

A partial conditional plan  $\gamma^p$  is *valid* w.r.t. a safe-reachability objective  $\mathcal{G}$  and a belief  $b \in \mathcal{B}$  if every plan in the set  $\Omega_{\gamma^p, b}$  is a valid plan, i.e.,  $\Omega_{\gamma^p, b} \subseteq \Omega_{\mathcal{G}}$ .

**Definition 4.1.3** (Partial Policy).

A  $k$ -step partial policy  $\pi^p = \{\gamma_1^p, \gamma_2^p, \dots\}$  is a set of partial conditional plans.  $\pi^p$  is associated with a set of beliefs  $\mathcal{B}_{\pi^p} \subseteq \mathcal{B}$ . Each partial conditional plan  $\gamma^p \in \pi^p$  is associated with a belief  $b \in \mathcal{B}_{\pi^p}$  and  $\pi^p(b) = \gamma^p$  specifies this partial conditional plan  $\gamma^p$  for this belief  $b$ . For a  $k$ -step partial policy  $\pi^p$ , the maximum horizon  $h_{\max} = \max_{\gamma^p \in \pi^p} h_{\gamma^p}$  is  $k$ .

Similarly, the  $k$ -step partial policy  $\pi^p = \{\gamma_1^p, \gamma_2^p, \dots\}$  defines a set  $\Omega_{\pi^p} = \bigcup_{b \in \mathcal{B}_{\pi^p}} \Omega_{\gamma^p, b}$  of plans, where  $\gamma^p = \pi^p(b)$  is the partial conditional plan specified for the belief  $b$  by the partial policy  $\pi^p$ .

**Definition 4.1.4** (Valid Partial Policy).

A  $k$ -step partial policy  $\pi^p$  is *valid* w.r.t. a safe-reachability objective  $\mathcal{G}$  if every plan in the set  $\Omega_{\pi^p}$  is valid ( $\Omega_{\pi^p} \subseteq \Omega_{\mathcal{G}}$ ).

### 4.1.2 Replanning Probability

Since a partial policy  $\pi^p$  only contains a subset of all observation branches at each step (see Figure 4.1), during online execution, it is possible that a dotted observation branch  $o$  that is not part of the partial policy is visited. In this case, the new successor belief  $b'$  of this branch  $o$  is not covered by the partial policy ( $b' \notin \mathcal{B}_{\pi^p}$ ) and it is required to recursively

compute a new partial policy for this new belief  $b'$ . However, since  $\pi^p$  does not consider all possible observation branches, it is possible that the partial conditional plan  $\gamma^p$  chosen by  $\pi^p$  is *invalid* for the new observation branch  $o$ , even for a valid partial policy. As a result, there is no partial policy for the new belief  $b'$  and execution fails.

To preserve correctness, it is ideal to bound the execution failure probability  $p_{\text{fail}}(\pi^p)$  of a valid partial policy  $\pi^p$ , which is equivalent to the execution failure probability measured from the initial belief  $b_0$  under the valid partial conditional plan  $\gamma^p = \pi^p(b_0)$  specified for the belief  $b_0$  by the partial policy  $\pi^p$ , i.e.,  $p_{\text{fail}}(\pi^p) = p_{\text{fail}}(b_0, \gamma^p) = \Pr(\text{failure}|b_0, \gamma^p)$ . The challenge is that  $p_{\text{fail}}(\pi^p)$  is difficult to compute because it requires checking whether every valid partial conditional plan  $\gamma^p = (a, O^p, \nu^p) \in \pi^p$  is also valid for every uncovered observation  $o \notin O^p$ , i.e., there exists a valid partial policy for the successor belief  $b'$  of the uncovered observation branch  $o$ , which essentially computes a valid *full* policy  $\pi$ .

However, the replanning probability  $p_{\text{replan}}(\pi^p) = p_{\text{replan}}(b_0, \gamma^p) = \Pr(\text{replanning}|b_0, \gamma^p)$  ( $\gamma^p = \pi^p(b_0) = (a, O^p, \nu^p)$ ) is the partial conditional plan for the initial belief  $b_0$  specified by the partial policy  $\pi^p$  of visiting a dotted observation branch that is not part of the partial policy and *replanning* a new partial policy is easy to compute recursively:

$$p_{\text{replan}}(\pi^p) = p_{\text{replan}}(b_0, \gamma^p) = \sum_{o \in O^p} \Pr(o|b, a) p_{\text{replan}}(b', \pi^p(b')) + \sum_{o \notin O^p} \Pr(o|b, a) \quad (4.1)$$

where  $\Pr(o|b, a) = \sum_{s' \in S} \mathcal{Z}(s', a, o) \sum_{s \in S} \mathcal{T}(s, a, s') b(s)$  is the probability of receiving observation  $o$  after performing action  $a$  in belief  $b$ . For the base case where the belief  $b$  is associated with a partial conditional plan  $\gamma^p = (a, O^p, \emptyset)$  with empty observation strategies,  $p_{\text{replan}}(b, \gamma^p) = \sum_{o \notin O^p} \Pr(o|b, a)$ .

The following theorem states that for a *valid* partial policy  $\pi^p$ , the failure probability  $p_{\text{fail}}(\pi^p)$  is bounded by the replanning probability  $p_{\text{replan}}(\pi^p)$ :

**Theorem 4.1.1.** For any  $k$ -step valid partial policy  $\pi^p = \{\gamma_1^p, \gamma_2^p, \dots\}$ ,  $p_{\text{fail}}(\pi^p) \leq p_{\text{replan}}(\pi^p)$ .

*Proof.* Theorem 4.1.1 can be proved by induction. Let  $\delta_{\text{fail}}(b) : \mathcal{B} \mapsto \{0, 1\}$  be a failure indicator and when  $\delta_{\text{fail}}(b) = 1$ , there is no valid partial policy for belief  $b$  and execution fails.

- Base case: consider those beliefs  $b$  associated with partial conditional plans  $\gamma^p = (a, O^p, \emptyset) = \pi^p(b)$  with empty observation strategies. Since  $\pi^p$  is valid,  $\gamma^p$  is also valid according to Definition 4.1.2 and 4.1.4. Therefore, for every observation  $o \in O^p$ ,  $b' = \mathcal{T}_{\mathcal{B}}(b, a, o) \in \text{Dest}$  is the terminal goal belief and  $\delta_{\text{fail}}(b') = 0$ . Thus,

$$\begin{aligned} p_{\text{fail}}(b, \gamma^p) &= \sum_{o \notin O^p} \Pr(o|b, a) \delta_{\text{fail}}(b') \\ &\leq \sum_{o \notin O^p} \Pr(o|b, a) = p_{\text{replan}}(b, \gamma^p) \end{aligned}$$

since  $\delta_{\text{fail}}(b') \leq 1$ .

- Inductive case: consider those beliefs  $b$  associated with partial conditional plans  $\gamma^p = (a, O^p, \nu^p) = \pi^p(b)$  with non-empty observation strategies. For every  $o \in O_k^p$ , let  $b' = \mathcal{T}_{\mathcal{B}}(b, a, o)$  be the successor belief and assume  $p_{\text{fail}}(b', \pi^p(b')) \leq p_{\text{replan}}(b', \pi^p(b'))$ , then

$$\begin{aligned} p_{\text{fail}}(b, \gamma^p) &= \sum_{o \in O^p} \Pr(o|b, a) p_{\text{fail}}(b', \pi^p(b')) + \sum_{o \notin O^p} \Pr(o|b, a) \delta_{\text{fail}}(b') \\ &\leq \sum_{o \in O^p} \Pr(o|b, a) p_{\text{replan}}(b', \pi^p(b')) + \sum_{o \notin O^p} \Pr(o|b, a) \\ &= p_{\text{replan}}(b, \gamma^p) \end{aligned}$$

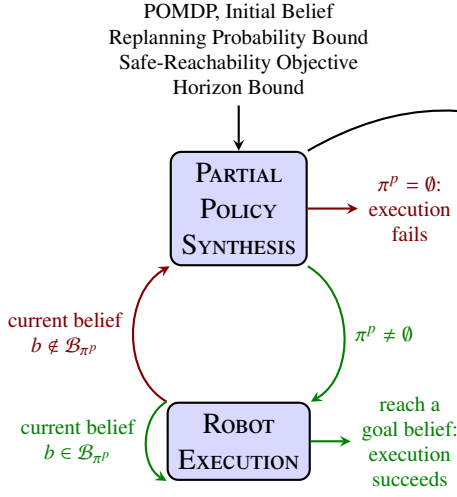


Figure 4.2 : OPPS

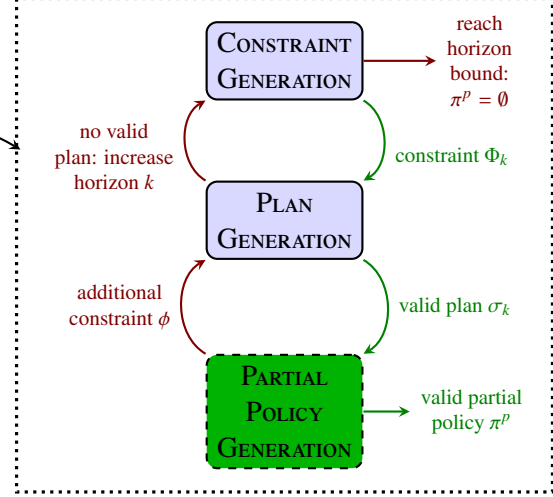


Figure 4.3 : Partial policy synthesis

since  $\delta_{\text{fail}}(b') \leq 1$ .

Therefore, For any belief  $b \in \mathcal{B}_{\pi^P}$ ,  $p_{\text{fail}}(b, \pi^P(b)) \leq p_{\text{replan}}(b, \pi^P(b))$ . Thus  $p_{\text{fail}}(\pi^P) = p_{\text{fail}}(b_0, \pi^P(b_0)) \leq p_{\text{replan}}(b_0, \pi^P(b_0)) = p_{\text{replan}}(\pi^P)$ .  $\square$

Since  $p_{\text{fail}}(\pi^P) \geq 0$ , by Theorem 4.1.1,  $p_{\text{fail}}(\pi^P) = 0$  when the replanning probability for a valid partial policy  $p_{\text{replan}}(\pi^P) = 0$ . In this case,  $\pi^P$  is a full policy.

### 4.1.3 Problem Statement

Given a POMDP  $P$ , an initial belief  $b_0$ , a replanning probability bound  $\delta_{p_{\text{replan}}}$ , a safe-reachability objective  $\mathcal{G}$  and a horizon bound  $h$ , the goal is to synthesize a *valid*  $k$ -step ( $k \leq h$ ) partial policy  $\pi^P$  with a replanning probability  $p_{\text{replan}}(\pi^P) \leq \delta_{p_{\text{replan}}}$ .

Since the replanning probability  $p_{\text{replan}}(\pi^P)$  is bounded by  $\delta_{p_{\text{replan}}}$ , by Theorem 4.1.1,  $\pi^P$  provides the guarantee of achieving the given safe-reachability objective with probability at least  $1 - \delta_{p_{\text{replan}}}$ .

---

**Algorithm 3: OPPS**


---

**Input:**  
 POMDP  $P$   
 Initial Belief  $b_0$   
 Replanning Probability Bound  $\delta_{p_{\text{replan}}}$   
 Safe-Reachability Objective  $\mathcal{G} = (Dest, Safe)$   
 Horizon Bound  $h$

**Output:**  
 A boolean: true - success, false - failure

```

/* Generate partial policy */
34  $\gamma^p \leftarrow \text{PartialPolicySynthesis}(P, b_0, \mathcal{G}, \delta_{p_{\text{replan}}}, 0, h)$ 
35 if  $\pi^p = \emptyset$  then
    /* No partial policy: failure */
36     return false
    /* Track current belief */
37  $b \leftarrow b_0$ 
38 repeat
    /* Get the partial conditional plan for current belief */
39      $\gamma^p \leftarrow \pi^p(b)$ 
    /* Unpack the partial conditional plan for current belief */
40      $(a, O^p, v^p) \leftarrow \gamma^p$ 
41     Execute action  $a$ 
42     Receive observation  $o$ 
    /* Update belief */
43      $b \leftarrow \mathcal{T}_{\mathcal{B}}(b, a, o)$ 
44     if  $b \in Dest$  then
        /* reach a goal belief: success */
45         return true
46 until  $b \notin \mathcal{B}_{\pi^p}$ ;
    /* recursively perform OPPS */
47 return  $\text{OPPS}(P, b, \delta_{p_{\text{replan}}}, \mathcal{G}, h)$ 

```

---

## 4.2 Online Partial Policy Synthesis

Figure 4.2 shows the overall structure of OPPS (Algorithm 3). OPPS follows the typical online planning paradigm [62] that interleaves synthesis of valid partial policies (line 34) and execution (lines 41, 42, 43). If there are no valid partial policies within the horizon bound (line 35), execution fails. Otherwise, OPPS follows the generated partial policy

until a goal belief is reached (line 44: execution succeeds) or a new belief  $b \notin \mathcal{B}_{\pi^p}$  is visited (line 46). In the latter case, OPPS recursively replans for this belief  $b$ . Next section describes the component of partial policy synthesis shown in Figure 4.3.

#### 4.2.1 Partial Policy Synthesis

The core steps of partial policy synthesis (Algorithm 4) are shown in Figure 4.3. OPPS replaces the component of policy generation in BPS with a new component of partial policy generation (the green dashed component).

For a given POMDP  $P$ , an initial belief  $b_0$  and a safe-reachability objective  $\mathcal{G} = (Dest, Safe)$ , OPPS first symbolically encodes the goal-constrained belief space over a bounded horizon  $k$  as a logical formula  $\Phi_k$  (lines 48, 52, 54).  $\Phi_k$  compactly represents the requirement of reaching a goal belief  $b \in Dest$  safely in  $k$  steps. In constraint generation (Figure 4.3), OPPS uses the Bounded Model Checking [3] encoding to construct  $\Phi_k$ , which contains three parts:

- start from the initial belief (line 48) :  $b_s = b_0$ .
- unfold the transition up to horizon  $k$  (line 52) :  $\bigwedge_{i=s+1}^k (b_i = \mathcal{T}_B(b_{i-1}, a_i, o_i))$ .
- satisfy the objective  $\mathcal{G}$  (line 54):  $G(\sigma_k, \mathcal{G}, k) = \bigvee_{i=s}^k (b_i \in Dest \wedge (\bigwedge_{j=s}^{i-1} (b_j \in Safe)))$ .

Then in plan generation (Figure 4.3), OPPS compute a valid *plan*  $\sigma_k$  by checking the satisfiability of  $\Phi_k$  (line 56) through an SMT solver [19]. Note that the horizon  $k$  restricts the plan length and thus the robot can only execute  $k$  actions before reaching a goal belief  $b \in Dest$ .

If  $\Phi_k$  is satisfiable, the SMT solver returns a valid plan  $\sigma_k = (b_0^{\sigma_k}, a_1^{\sigma_k}, o_1^{\sigma_k}, \dots, b_k^{\sigma_k})$ . This valid plan only covers a particular observation  $o_i^{\sigma_k}$  at step  $i$ . In partial policy generation (Figure 4.3), OPPS generates a valid partial policy  $\pi^p$  with replanning probability

---

**Algorithm 4:** PartialPolicySynthesis
 

---

**Input:** POMDP  $P$   
 Initial Belief  $b_0$   
 Replanning Probability Bound  $\delta_{p_{\text{replan}}}$   
 Safe-Reachability Objective  $\mathcal{G} = (Dest, Safe)$   
 Start Step  $s$   
 Horizon Bound  $h$   
**Output:** *Valid* partial policy  $\pi^p$  with a replanning probability  $p_{\text{replan}}(\pi^p) \leq \delta_{p_{\text{replan}}}$

```

/* Start from initial belief */
48  $\Phi_k \leftarrow (b_s = b_0)$ 
/* Initial horizon */
49  $k \leftarrow s$ 
50 while  $k \leq h$  do
    /* Add transition at step  $k$  if  $k > s$  */
    51 if  $k > s$  then
    52      $\Phi_k \leftarrow \Phi_k \wedge (b_k = \mathcal{T}_B(b_{k-1}, a_k, o_k))$ 
    /* Push scope */
    53     push( $\Phi_k$ )
    /* Add goal constraints at step  $k$  */
    54      $\Phi_k \leftarrow \Phi_k \wedge G(\sigma_k, \mathcal{G}, k)$ 
    55     repeat
        /* Generate valid plan */
        56      $\sigma_k \leftarrow \text{IncrementalSMT}(\Phi_k);$ 
        57     if  $\sigma_k \neq \emptyset$  then /* Find valid plan */
            /* Generate partial policy */
            58      $\pi^p, \phi = \text{PartialPolicyGeneration}(P, \delta_{p_{\text{replan}}}, \mathcal{G}, \sigma_k, s + 1, h)$ 
            59     if  $\pi^p = \emptyset$  then /* Generation failed */
                /* Blocking invalid plans */
                60      $\Phi_k \leftarrow \Phi_k \wedge \phi$ 
            61     else
                62     return  $\pi^p$ 
        63     until  $\sigma_k = \emptyset;$ 
    /* Pop goal and  $\phi$  at step  $k$  */
    64     pop( $\Phi_k$ )
    /* Increase horizon */
    65      $k \leftarrow k + 1$ 
66 return  $\emptyset$ 
  
```

---



$p_{\text{replan}}(\pi^p) \leq \delta_{p_{\text{replan}}}$  from  $\sigma_k$  by sampling a subset  $\mathcal{O}^p \subseteq \mathcal{O}$  of observations (solid branches in Figure 4.1) at each step, where  $\delta_{p_{\text{replan}}}$  is the given replanning probability bound.

If this partial policy generation succeeds, OPPS finds a valid partial policy  $\pi^p$ . Otherwise, OPPS constructs an additional constraint  $\phi$  to block invalid plans (line 60) and force the SMT solver to generate another plan. Note that  $\phi$  is only valid for the current horizon  $k$  and when the horizon increases, OPPS should *pop* the scope related to the additional constraint  $\phi$  from the stack of the SMT solver (line 64) so that OPPS can *revisit*  $\sigma_k$  with an increased horizon. The incremental SMT solver can efficiently generate alternate valid plans by maintaining a *stack* of *scopes* for the “knowledge” learned from previous satisfiability checks [19] [17, 78].

If  $\Phi_k$  is unsatisfiable and there are no valid plans for the current horizon, OPPS increases the horizon (line 65) and repeat the above steps until a *valid* partial policy is found (line 62) or a given horizon bound is reached (line 50). Next section describes the new component of partial policy generation.

#### 4.2.2 Partial Policy Generation

In partial policy generation (Algorithm 5), OPPS constructs a valid partial policy  $\pi^p$  that satisfies the given bound  $\delta_{p_{\text{replan}}}$  from a valid plan  $\sigma_k$ . For each step  $i$ , OPPS recursively constructs a partial policy for the branch  $o_i^{\sigma_k}$  (line 71). If the replanning probability of  $\pi^p$  is greater than the given bound  $\delta_{p_{\text{replan}}}$ , OPPS needs to add more observation branches to  $\pi^p$  by sampling a new observation  $o'$  (line 77) and recursively constructing a partial policy  $\pi^{p'}$  for  $o'$ . This is another partial policy synthesis problem with a new initial belief (line 78), and can be solved recursively using Algorithm 4 (line 79).

If OPPS successfully constructs a valid  $\pi^{p'}$  for  $o'$ , OPPS can add the observation  $o'$  to the partial conditional  $\gamma^p$  for the current belief  $b$  (line 74 or 83). Otherwise, this input plan

---

**Algorithm 5:** PartialPolicyGeneration
 

---

**Input:** POMDP  $P$   
 Replanning Probability Bound  $\delta_{p_{\text{replan}}}$   
 Safe-Reachability Objective  $\mathcal{G} = (Dest, Safe)$   
 Valid Plan  $\sigma_k = (b_0^{\sigma_k}, a_1^{\sigma_k}, o_1^{\sigma_k}, \dots, b_k^{\sigma_k})$   
 Step  $i$   
 Horizon Bound  $h$   
**Output:** Valid partial policy  $\pi^p$  with replanning probability  $p_{\text{replan}}(\pi^p) \leq \delta_{p_{\text{replan}}}$ ,  
 Constraint  $\phi$  for blocking invalid plans

```

67 if  $i > k$  then /* Reach end of the plan */
    /* Terminal belief:  $\gamma^p$  is an empty partial conditional plan */
68      $\gamma^p \leftarrow \emptyset, \pi^p \leftarrow \{\gamma^p\}$ 
69     return  $\pi^p, \emptyset$ 
70  $O^p \leftarrow \emptyset, v^p \leftarrow \emptyset, \delta'_{p_{\text{replan}}} \leftarrow \delta_{p_{\text{replan}}}, b \leftarrow b_{i-1}^{\sigma_k}, a \leftarrow a_i^{\sigma_k}, o' \leftarrow o_i^{\sigma_k}$  /* Initialize */
    /* Recursively process next step */
71  $\pi^p, \phi \leftarrow \text{PartialPolicyGeneration}(P, \delta'_{p_{\text{replan}}}, \mathcal{G}, \sigma_k, i + 1, h)$ 
72 if  $\pi^p = \emptyset$  then
73     return  $\emptyset, \phi$ 
74  $b' \leftarrow b_i^{\sigma_k}, O^p \leftarrow O^p \cup \{o'\}, v^p(o') \leftarrow \pi^p(b'), \gamma^p \leftarrow (a, O^p, v^p)$  /* Add  $o'$  to  $\gamma^p$  */
75 while  $p_{\text{replan}}(\pi^p) > \delta_{p_{\text{replan}}}$  do
    /* Update replanning probability bound */
76      $\delta'_{p_{\text{replan}}} \leftarrow \delta'_{p_{\text{replan}}} + \frac{\Pr(o'|b,a)(\delta'_{p_{\text{replan}}} - p_{\text{replan}}(b', v^p(o')))}{\sum_{o \in O - O^p - \{o'\}} \Pr(o|b,a)}$ 
77      $o' \leftarrow$  sampled observation in  $O - O^p$  according to the probability of occurrence
78      $b' \leftarrow \mathcal{T}_{\mathcal{B}}(b, a, o')$  /* Get new initial belief */
    /* Recursively construct partial policy */
79      $\pi^{p'} \leftarrow \text{PartialPolicySynthesis}(P, b'_i, \delta'_{p_{\text{replan}}}, \mathcal{G}, i, h)$ 
80     if  $\pi^{p'} = \emptyset$  then /* Construction failed */
81         Construct  $\phi$  using Formula 3.2
82         return  $\emptyset, \phi$ 
83      $O^p \leftarrow O^p \cup \{o'\}, v^p(o') \leftarrow \pi^{p'}(b'_i), \pi^p \leftarrow \pi^p \cup \pi^{p'}$  /* Add  $o'$  to  $\gamma^p$  */
84 foreach observation  $o \in O - O^p$  do /* Final safety check */
85      $b'_i \leftarrow \mathcal{T}_{\mathcal{B}}(b, a, o);$  /* Try observation  $o$  */
86     if  $b'_i \notin Safe$  then /* Violates safety */
87         Construct  $\phi$  using Formula 3.2
88         return  $\emptyset, \phi$ 
89  $\pi^p(b) \leftarrow \gamma^p$  /* Record partial conditional plan for  $b$  in  $\pi^p$  */
90 return  $\pi^p, \emptyset$ 
  
```

---

$\sigma_k$  cannot be an element of a *valid* partial policy  $\pi^p$ , i.e.,  $\sigma_k \notin \Omega_{\pi^p}$ . In this case, the prefix of this plan  $(b_0^{\sigma_k}, a_1^{\sigma_k}, o_1^{\sigma_k}, \dots, b_{i-1}^{\sigma_k}, a_i^{\sigma_k})$  is invalid for current horizon  $k$  and OPPS can construct the additional constraint  $\phi$  to block invalid plans based on Formula 3.2.

$\phi$  blocks the invalid plans that have this prefix (line 81) and avoids unnecessary checks of these plans (checking  $\sigma_k$  has already shown that these plans are invalid).

### Updating Replanning Probability Bound

As OPPS adds more observation branches to the partial conditional plan  $\gamma^p = (a, O^p, \nu^p)$  for the current belief  $b$ , the replanning probability bound  $\delta'_{p_{\text{replan}}}$  for the remaining uncovered observation branches  $O - O^p$  needs to be updated (line 76) in order to avoid unnecessary computation.

Initially,  $O^p$  is empty and  $\delta'_{p_{\text{replan}}}$  is the input bound  $\delta_{p_{\text{replan}}}$  (line 70).  $\delta'_{p_{\text{replan}}}$  bounds the replanning probability  $p_{\text{replan}}(b', \nu^p(o))$  of the next-step partial conditional plan  $\nu^p(o)$  for every remaining uncovered observation  $o \in O - O^p$ , where  $b' = \mathcal{T}_{\mathcal{B}}(b, a, o)$  is the corresponding successor belief for the observation  $o$ .  $\delta'_{p_{\text{replan}}} = \delta_{p_{\text{replan}}}$  guarantees that the replanning probability  $p_{\text{replan}}(\pi^p)$  of the current partial policy  $\pi^p$  satisfies the original bound  $\delta_{p_{\text{replan}}}$ , i.e.,  $p_{\text{replan}}(\pi^p) = p_{\text{replan}}(b, \gamma^p) = \sum_{o \in O} \Pr(o|b, a) p_{\text{replan}}(b', \nu^p(o)) \leq \sum_{o \in O} \Pr(o|b, a) \delta'_{p_{\text{replan}}} \leq \delta'_{p_{\text{replan}}} = \delta_{p_{\text{replan}}}$  since  $p_{\text{replan}}(b', \nu^p(o)) \leq \delta'_{p_{\text{replan}}}$  based on the definition of  $\delta'_{p_{\text{replan}}}$ .

During partial policy generation, after adding a new observation  $o' \in O - O^p$  to the partial conditional plan  $\gamma^p$  (line 74 or 83), the replanning probability bound  $\delta'_{p_{\text{replan}}}$  is updated in order to avoid unnecessary computation. Suppose a new next-step partial conditional plan  $\nu^p(o')$  with the same replanning probability  $\alpha$  is constructed for every remaining uncovered observation  $o \in O - O^p - \{o'\}$ . Then the replanning probability of the observation branches  $O - O^p$  is  $\Pr(o'|b, a) p_{\text{replan}}(b', \nu^p(o')) + \alpha \sum_{o \in O - O^p - \{o'\}} \Pr(o|b, a) \leq \sum_{o \in O - O^p} \Pr(o|b, a) \delta'_{p_{\text{replan}}}$ , where  $b' = \mathcal{T}_{\mathcal{B}}(b, a, o')$  is the corresponding successor belief for the observation  $o'$ . Therefore

$\alpha \leq \delta'_{p_{\text{replan}}} + \frac{\Pr(o'|b,a)(\delta'_{p_{\text{replan}}} - p_{\text{replan}}(b', v^p(o')))}{\sum_{o \in O - O^p - \{o'\}} \Pr(o|b,a)}$ . Then the new bound for the remaining uncovered observation  $o \in O - O^p - \{o'\}$  should be  $\delta'_{p_{\text{replan}}} + \frac{\Pr(o'|b,a)(\delta'_{p_{\text{replan}}} - p_{\text{replan}}(b', v^p(o')))}{\sum_{o \in O - O^p - \{o'\}} \Pr(o|b,a)}$  and this new bound is at least  $\delta'_{p_{\text{replan}}}$  since  $p_{\text{replan}}(b', v^p(o')) \leq \delta'_{p_{\text{replan}}}$  according to the definition of  $\delta'_{p_{\text{replan}}}$ . When the replanning probability bound becomes larger, computing a partial conditional plan is usually less expensive. Therefore, updating the replanning probability bound (line 76) improves efficiency and still makes the current partial policy  $\pi^p$  satisfy the original bound  $\delta_{p_{\text{replan}}}$ .

### Safety Guarantee

OPPS constructs a valid partial policy  $\pi^p$  with a replanning probability bounded by the given bound  $\delta_{p_{\text{replan}}}$ . By Theorem 4.1.1, the execution failure probability  $p_{\text{fail}}(\pi^p)$  is also bounded by  $\delta_{p_{\text{replan}}}$ . If the replanning probability  $p_{\text{replan}}(\pi^p) > 0$ , in the worst case, the robot might visit a belief  $b \notin \mathcal{B}_{\pi^p}$  and find that there is no valid partial policy for this belief  $b$ , and then execution fails due to unsuccessful replanning. In this case, though OPPS cannot achieve the original safe-reachability objective, a guarantee of the robot still satisfying the safety requirement is preferable to the situation where the robot violates the safety requirement. OPPS can provide this safety guarantee by checking whether the successor belief of every uncovered observation at every step is *safe* (lines 84, 85, 86, 87, 88).

## 4.3 Experiments

OPPS is tested on the kitchen domain (horizon bound  $h = 30$ ) presented in Chapter 3 and the classic Tag domain [58] ( $h = 100$ ). OPPS is also validated on a Fetch robot for the scenario shown in Figure 3.7 ( $h = 20$ ). Z3 [19] is used as the backend SMT solver. All experiments were conducted on a 3.0 GHz Intel® processor with 32 GB of memory. The

time-out is set to be 1800 seconds. For all the tests of the kitchen and Tag domains, the results are averaged over 50 independent runs.

In the uncertain kitchen domain [78], a robot needs to eventually pick up a cup from the storage while avoiding collisions with  $M$  uncertain obstacles. This kitchen domain is an example scenario that requires a correctness guarantee of accomplishing tasks, and POMDPs with boolean objectives provide a better safe-reachability guarantee than the quantitative POMDP models [78].

The kitchen environment is discretized into  $N = 36$  regions. The actuation and perception of the robot are imperfect, modeled as ten uncertain robot actions: *move* and *look* in four directions, pick-up using the left hand and pick-up using the right hand. The robot starts at a known initial location. However, due to the robot’s imperfect perception, the location of the robot and the locations of uncertain obstacles are all partially observable during execution. This kitchen domain has a large state space  $|S| = C(N, M) \cdot N$ , where  $C(N, M)$  is the number of  $M$ -combinations from the set of  $N$  regions. In the largest test ( $M = 7$ ) there are more than  $10^8$  states.

#### 4.3.1 Performance

The previous method BPS and OPPS are evaluated with different replanning probability bounds (from 0.1 to 0.9) in this kitchen domain for test cases with various numbers of obstacles. BPS computes a full policy that covers all observation branches and is equivalent to OPPS with replanning probability bound  $\delta_{p_{\text{replan}}} = 0$ .

Figure 4.4a, 4.4b, 4.4c and 4.4d show the average computation time of one synthesis call, the average number of synthesis calls, the average total computation time and the average computation time per step as the bound  $\delta_{p_{\text{replan}}}$  increases, respectively. As shown in Figure 4.4a (semi-log scale) and 4.4b, as the bound  $\delta_{p_{\text{replan}}}$  increases, the computation

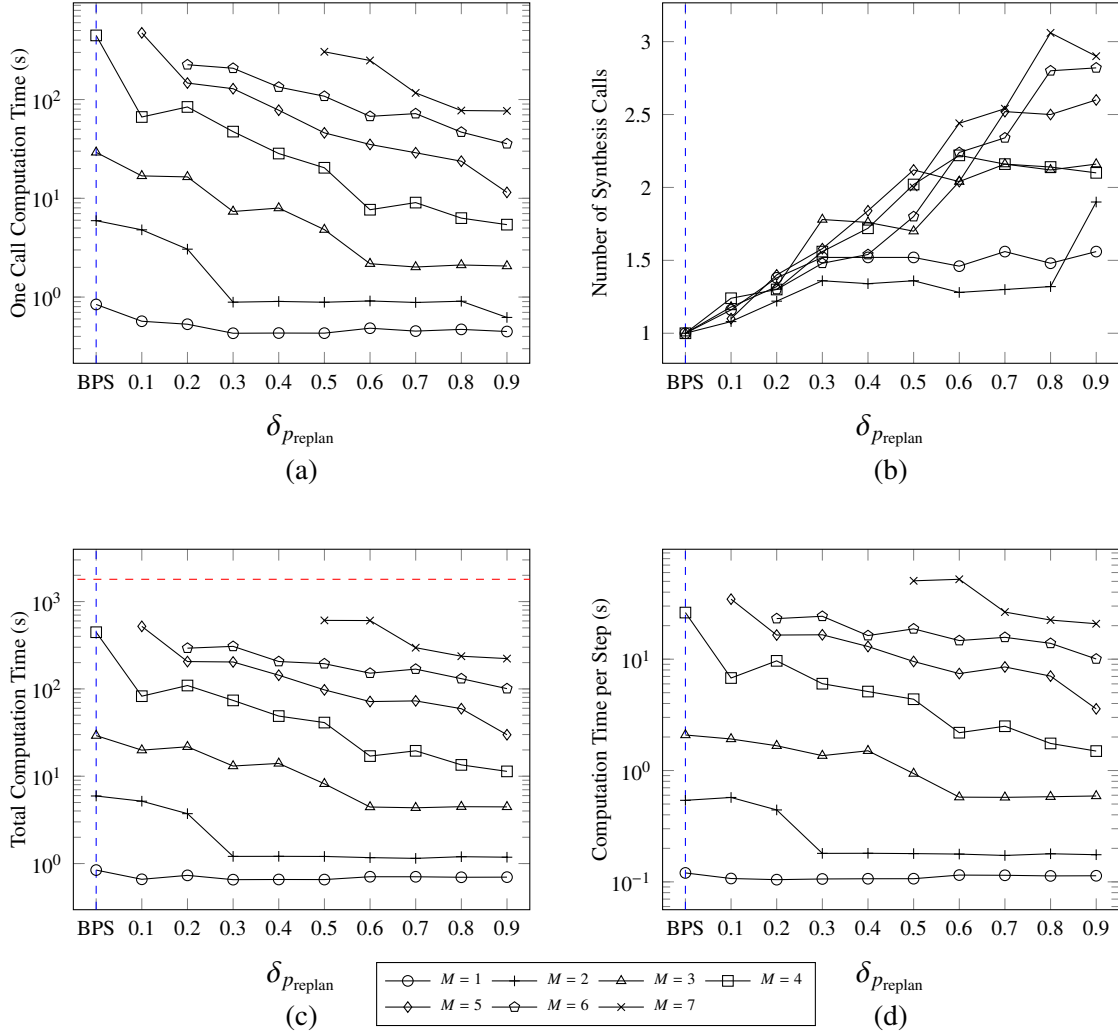


Figure 4.4 : Performance results for the kitchen domain as the bound  $\delta_{p_{\text{replan}}}$  increases. Different plots correspond to tests with different numbers  $M$  of obstacles. Missing data points in a plot indicate the timeout. The red dashed line is the timeout (time = 1800 seconds). The blue dashed line passes through the data points generated by BPS. All the results are averaged over 50 independent runs.

time of one synthesis call decreases very quickly while the number of calls to partial policy synthesis does not increase much. Therefore, the total computation time (Figure 4.4c) keeps decreasing as the bound  $\delta_{p_{\text{replan}}}$  increases. Additionally, shown in Figure 4.4c (semi-log scale), with a small bound  $\delta_{p_{\text{replan}}} = 0.1$ , OPSS achieves a big performance gain compared

to BPS: for the test case with  $M = 4$  obstacles, the speedup is around 5 times, and for the test case with  $M = 5$  obstacles, BPS cannot solve this within the time limit while OPPS with  $\delta_{p_{\text{replan}}} = 0.1$  can solve this in around 9 minutes. Therefore OPPS achieves better performance than BPS in the tests by computing valid *partial* policies instead of valid *full* policies. The same trend is also shown in the results of the average computation time per step (Figure 4.4d). These results suggest that for some robotic domains where replanning is easy, such as the tested mobile manipulation domain, users can increase the replanning probability bound for better scalability.

#### 4.3.2 Success Rate

For all the previous performance tests, the constructed partial policies by OPPS with different bounds  $\delta_{p_{\text{replan}}}$  always achieve the given safe-reachability objective, i.e., the success rate is 100%. This is because the robot can move in four directions. When the robot enters a region surrounded by obstacles in three directions, the robot can always move back to its previous position, which means replanning is always possible. However, for some domains such as autonomous driving cars and robot chefs, once the robot commits to an action and finds something wrong, it is difficult or impossible to reverse the effect of the action and replan. To evaluate how OPPS performs in these scenarios, OPPS is tested in the kitchen domain with different numbers  $M$  of obstacles ( $M \leq 4$  since BPS cannot solve tests with more than 4 obstacles within the time limit), but the robot's power is restricted by disabling the *move-north* action. Therefore, when the robot performs *move-south* and enters a region surrounded by obstacles in three directions, replanning fails. However, the robot still satisfies the safety requirement, thanks to the safety guarantee of OPPS.

Figure 4.5 shows the success rate as the bound  $\delta_{p_{\text{replan}}}$  increases. For all the tests, the success rate is always greater than  $1.0 - \delta_{p_{\text{replan}}}$  (all data points are above the plot of

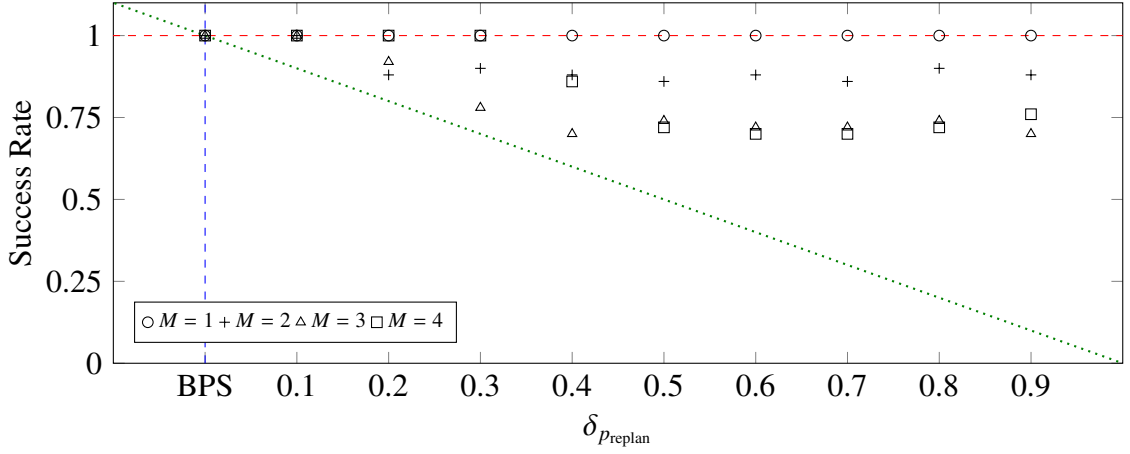


Figure 4.5 : Success rate as  $\delta_{p_{\text{replan}}}$  increases. The green dotted line shows the plot of success rate =  $1.0 - \delta_{p_{\text{replan}}}$ . The red dashed line is the plot of success rate = 1.0. The blue dashed line passes through the data points generated by BPS.

success rate =  $1.0 - \delta_{p_{\text{replan}}}$ ). This matches Theorem 4.1.1: the failure probability of a valid partial policy is bounded by the replanning probability. Moreover, as the bound  $\delta_{p_{\text{replan}}}$  decreases to 0, OPPS produces a valid full policy with 100% success rate. These results suggest that for some domains where replanning is difficult, users can decrease the bound  $\delta_{p_{\text{replan}}}$  and allocate computational resources for a high success rate.

Note that the replanning probability bound is a conservative upper bound of the failure probability since it pessimistically assumes all the uncovered observation branches that require replanning will fail, which is a rare case in practice. As shown in Figure 4.5, even with a high replanning probability bound  $\delta_{p_{\text{replan}}} = 0.9$ , the success rate is still at least 70% and the failure rate is at most 30%, which is much smaller than the given bound  $\delta_{p_{\text{replan}}} = 0.9$ .

### 4.3.3 Gains from Updating Replanning Probability Bound

As discussed in Section 4.2.2, updating the replanning probability bound during partial policy generation is important for avoiding unnecessary computation and improving effi-



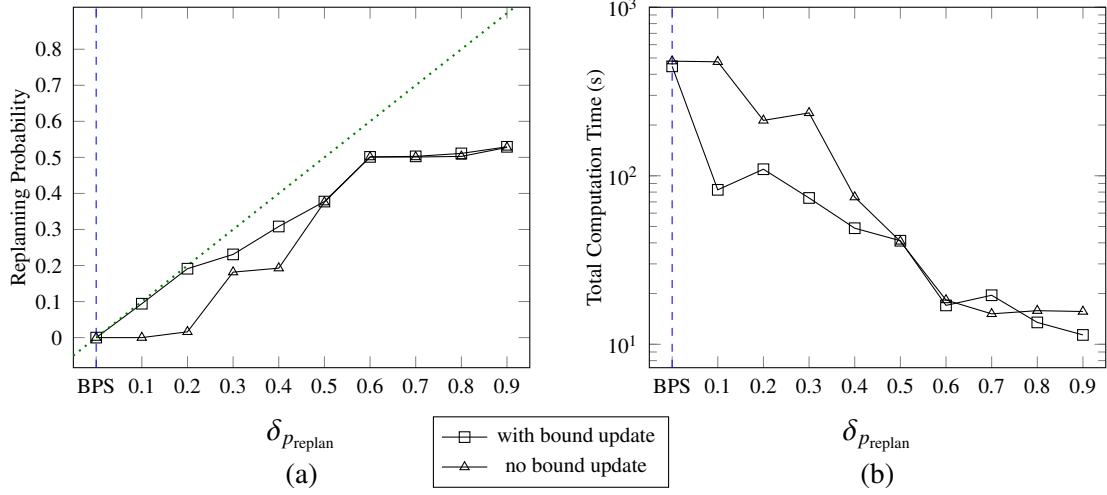


Figure 4.6 : Replanning probability and total computation time as the bound  $\delta_{p_{\text{replan}}}$  increases ( $M = 4$ ). The green dotted line shows the plot of replanning probability =  $\delta_{p_{\text{replan}}}$ . The blue dashed line passes through the data points generated by BPS.

ciency. To evaluate the gains from this bound update step, OPPS is tested with and without bound update in the kitchen domain with  $M = 4$  obstacles.

Figure 4.6a and 4.6b (semi-log scale) show the average replanning probability of the constructed partial policies and the average total computation time as the bound  $\delta_{p_{\text{replan}}}$  increases, respectively. As shown in Figure 4.6a, with both settings (with and without bound update) OPPS constructs a partial policy with a replanning probability smaller than the bound  $\delta_{p_{\text{replan}}}$ . However, OPPS without bound update constructs a partial policy with a lower replanning probability than that constructed by OPPS with bound update. Therefore, OPPS without bound update performs unnecessary computation and constructs a partial policy with more branches and thus spends more time than OPPS with bound update, as shown in Figure 4.6b. For the tests with the bound  $\delta_{p_{\text{replan}}} = 0.1, 0.2, 0.3$  that take more time to solve than those with  $\delta_{p_{\text{replan}}} > 0.3$ , OPPS with bound update achieves a 2-5 times speedup.

#### 4.3.4 Physical Validation

OPPS is validated on a Fetch robot for the domain shown in Figure 3.7. The setup of this domain is similar to the kitchen domain. The Fetch needs to pick up a target object (the blue can on the table) while avoiding collisions with uncertain obstacles such as floor signs and file cabinets, which can be placed in different locations. The POMDP’s state space consists of robot locations and object locations. A Vicon system is used to detect object locations, which is usually accurate but can still produce false negative and false positive due to occlusion or inappropriate Vicon marker configurations on objects. The false negative and false positive probabilities can be estimated by counting the false negative and false positive events during 100 Vicon detections. The POMDP’s probabilistic observation function is defined based on the false negative and false positive probabilities. To test the effects of different replanning probability bounds, the Fetch is only allowed to move in three directions (west, east and south), similar to the setup in the previous success rate tests. Sometimes the Fetch may fail to move its base when given a *move* action command and stay in the same place. The failure probability of these move actions can be estimated by counting the failure events during 100 *move* action executions. The POMDP’s probabilistic transition function is defined based on this failure probability. Figure 4.7a shows the initial state. There are two uncertain obstacles (a wet-floor sign and a file cabinet). OPPS is tested with two bounds  $\delta_{p_{\text{replan}}} = 0.9$  and  $\delta_{p_{\text{replan}}} = 0.1$ .

With  $\delta_{p_{\text{replan}}} = 0.9$ , after observing no obstacle in the south direction, the Fetch decides to move south (Figure 4.7b) because the partial policy constructed with a high replanning probability bound does not cover the case where the Fetch is surrounded by obstacles and the wall. Then replanning fails, but the Fetch still satisfies the safety requirement as shown in Figure 4.7b, thanks to the safety guarantee of OPPS.

However, with  $\delta_{p_{\text{replan}}} = 0.1$ , after observing no obstacles in the south direction, the

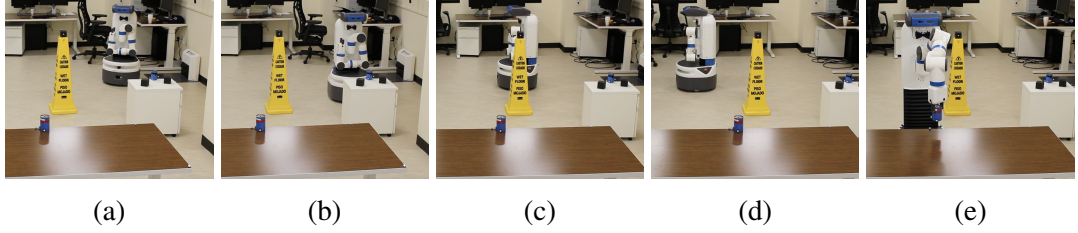


Figure 4.7 : Physical validation of OPPS for the domain shown in Figure 3.7.

Fetch decides to move west (Figure 4.7c) because the partial policy constructed with a low replanning probability bound covers the case where obstacles surround the robot. In order to avoid this situation, the Fetch needs to move west and gather more information. Then the Fetch observes an obstacle in the south direction and decides to move west again (Figure 4.7d). Next, the Fetch observes no obstacle in the south direction, and now it can move south safely. Unlike the case shown in Figure 4.7b where the robot is surrounded by two obstacles and the wall, in the situation shown in Figure 4.7d, if there is another obstacle in the south direction, the Fetch can still move west since there are only two obstacles. Finally, the Fetch moves to the table and picks up the target object (Figure 4.7e).

#### 4.3.5 Tag Domain

To further demonstrate the advantage of OPPS over the previous BPS method, OPPS is evaluated on a classic POMDP benchmark [58]. The task for the robot is to search for and tag a moving agent in a grid with 29 locations. The agent follows a fixed strategy that intentionally moves away from the robot. Both the robot and the agent can move in four directions or stay. The robot's location is fully observable while the agent's location is unobservable unless the robot and the agent are in the same location.

This Tag domain is challenging for BPS because of a large number of observations ( $|O| = 30$ ) and more importantly, a huge planning horizon for computing a full policy.

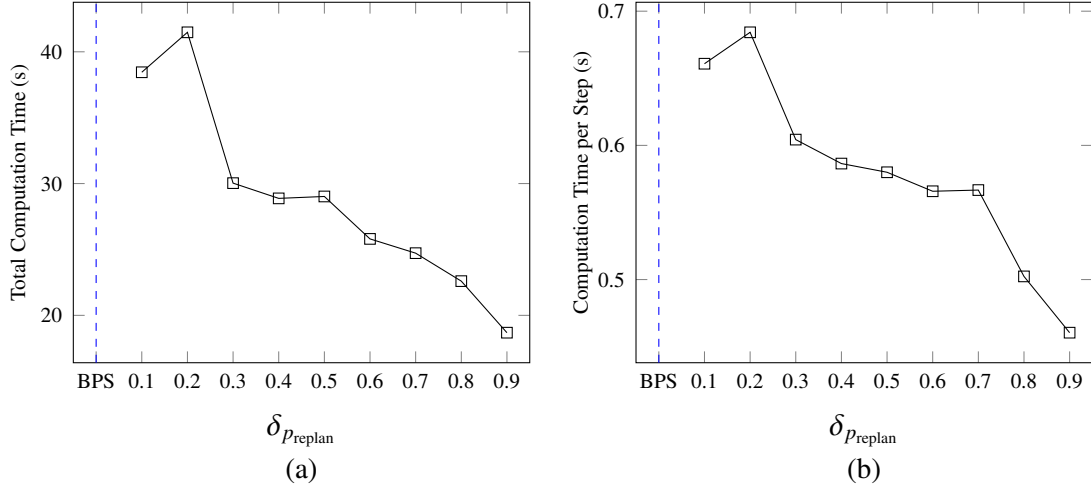


Figure 4.8 : Performance results for the Tag domain as the replanning probability bound  $\delta_{p_{\text{replan}}}$  increases. All the results are averaged over 50 independent runs.

However, computing a full policy is unnecessary since replanning is easy in this domain. Figure 4.8a and 4.8b show the average total computation time and the average computation time per step for the Tag domain as  $\delta_{p_{\text{replan}}}$  increases. These results show a similar trend to the previous kitchen domain tests: with a small bound  $\delta_{p_{\text{replan}}} = 0.1$ , OPPS achieves a big performance gain compared to BPS. BPS cannot solve this test within the 1800-second time limit while OPPS with  $\delta_{p_{\text{replan}}} = 0.1$  can solve this test in around 40 seconds and the computation time per step is less than 1 second.

#### 4.4 Discussion

This chapter presented a new approach, called OPPS, to policy synthesis for POMDPs with safe-reachability objectives. OPPS introduces the notion of a *partial policies* to improve computational efficiency. Rather than explicitly enumerating all possible observations to construct a *full* policy, OPPS samples a subset of all observations at each step to ensure bounded replanning probability. The theoretical and empirical results both show that the

failure probability of a valid partial policy is bounded by the replanning probability. Moreover, OPPS guarantees that the robot still satisfies the safety requirement when replanning fails. The results show that compared to the previous BPS approach, OPPS with a proper replanning probability bound scales better in the tested benchmarks and can solve problems beyond the capabilities of BPS within the time limit. Moreover, the results also suggest that for domains where replanning is easy, users can increase the replanning probability bound for efficiency. On the other hand, for domains where replanning is difficult, users can decrease the replanning probability bound and allocate more computation time in order to achieve a higher success rate. The results also indicate that by updating the replanning probability bound during partial policy generation, OPPS can quickly detect if the current partial policy satisfies the bound and avoid unnecessary computation.

The current implementation of OPPS is restricted to discrete POMDPs. While many robot applications can be modeled using this discrete representation, discretization often suffers from the “curse of dimensionality”. Investigating how to deal with continuous POMDPs [1, 33, 64] directly without discretization is a promising future direction for this work and its application in robotics. OPPS constructs partial conditional plans by sampling observations according to the probability of occurrence (Algorithm 5, line 77), which does not consider the importance of observations [49]. How to extend OPPS to handle critical observations is another important ongoing question.

## Chapter 5

### Combining Safe-Reachability and Quantitative Objectives

This chapter presents an offline policy synthesis algorithm, called Point-Based Policy Synthesis (PBPS) for POMDPs with both safe-reachability objective and quantitative objectives.

Chapter 3 and 4 investigates practical policy synthesis approaches for POMDPs with only boolean (safe-reachability) objectives. POMDPs with boolean objectives provides a strong correctness guarantee of completing tasks [78], but the constructed policy may not be optimal. On the other hand, POMDPs with quantitative objectives provide an optimality guarantee but may lead to overly conservative or overly risky behaviors [75], depending on the particular reward function chosen. For the example domain shown in Figure 1.1, there are many valid policies that can achieve the task objective, i.e., satisfying the boolean objective. Among these valid policies, the preferable policy is the one that passes the smallest number of red regions that the robot should avoid. Therefore, for domains that require both correctness and optimality, POMDPs with both boolean and quantitative objectives are natural formulations.

Policy synthesis for POMDPs with both boolean and quantitative objectives has been studied before [9]. In their work, the goal is to find an optimal policy that also ensures a goal state is reached with probability 1 (almost-sure satisfaction). A more general policy synthesis problem of POMDPs with both boolean and quantitative objectives is to synthesize an optimal policy that satisfies the boolean objective with a probability above a threshold. This thesis studies this problem for the special case of *safe-reachability* objec-

tives, which require that with a probability above a threshold, a goal state is eventually reached while keeping the probability of visiting unsafe states below a different threshold. Many robot tasks such as the one shown in Figure 1.1 can be formulated as POMDPs with safe-reachability and quantitative objectives.

The previous chapters consider POMDPs with only safe-reachability objectives. For offline synthesis, an approach called Bounded Policy Synthesis (BPS) has been presented in Chapter 3. BPS computes a valid policy over a *goal-constrained belief space* rather than the entire belief space to improve scalability. The goal-constrained belief space only contains beliefs visited by desired executions with a bounded horizon that can achieve the safe-reachability objective and is generally much smaller than the original belief space. On the other hand, for POMDPs with only quantitative objectives, point-based POMDP solvers [1, 6, 36, 42, 49, 58, 69, 71] have become quite successful in recent years. Point-based POMDP solvers can solve large POMDPs by producing approximated policies over a finite set of representative beliefs instead of the entire belief space.

Ideally, an *exact* policy for POMDPs with safe-reachability and quantitative objectives can be constructed by enumerating all beliefs in the goal-constrained belief space and performing the value iteration on these beliefs. However, this enumeration is generally very expensive [76] even though the goal-constrained belief space is finite when restricted to a bounded horizon. To improve efficiency, this thesis selects representative beliefs from the goal-constrained belief space and produce an *approximate* policy by performing the point-based backup [42, 58] on these representative beliefs that approximate the goal-constrained belief space. This selection process essentially asks whether there exists a valid policy starting from a belief and can be solved by BPS. Since policies need to be constructed in order to approximate the goal-constrained belief space, policy iteration is chosen to handle the quantitative objective because policy iteration typically converges faster than value

iteration [36].

This chapter presents an offline policy synthesis approach, *Point-Based Policy Synthesis* (PBPS), for POMDPs with safe-reachability and quantitative objectives. PBPS combines BPS [78] and Point-Based Policy Iteration (PBPI) [36] to synthesize good approximate policies that satisfy the safe-reachability objective. At a high level, PBPS applies BPS to efficiently explore the goal-constrained belief space for finding a valid policy  $\pi$  that satisfies the safe-reachability objective. Then PBPS adapts PBPI to transform  $\pi$  into an improved policy  $\pi'$ . This improved policy  $\pi'$  may reach some belief  $b'$  that is not visited by the current policy. Therefore, PBPS invokes BPS again to check whether there exists a valid policy starting from  $b'$ . By doing this, new belief regions can be explored, and the set of representative beliefs can be expanded, which is crucial to the quality of the constructed policy [36, 42, 58]. PBPS alternates between the computation of valid policies and policy iteration until the termination condition is satisfied.

The theoretical analysis shows that PBPS inherits many desirable properties of PBPI. First, PBPS maintains validity and is monotonic: at each iteration before termination, PBPS produces a valid policy for which the values increase for at least one belief of the representative belief set and decrease for none of these beliefs. Second, the error introduced by PBPS due to approximation is bounded. PBPS is evaluated in the kitchen domain [78] and the Tag domain [58]. PBPS is also validated on a Fetch robot for the domain in Figure 1.1. The results demonstrate that PBPS produces good approximate policies that achieve the given safe-reachability objective.



## 5.1 Problem Formulation

### 5.1.1 Quantitative Objectives

Each conditional plan  $\gamma = (a, \nu) \in \pi$  in a  $k$ -step policy  $\pi = \{\gamma_1, \gamma_2, \dots\}$  induces a value function  $V_\gamma(b)$  that specifies the expected total reward of executing the conditional plan  $\gamma$  starting from the belief  $b$ :

$$V_\gamma(b) = \sum_{s \in \mathcal{S}} r(s, a) b(s) + \sum_{o \in \mathcal{O}} \Pr(o|b, a) V_{\gamma_o}(b_a^o) \quad (5.1)$$

where  $b_a^o = \mathcal{T}_{\mathcal{B}}(b, a, o)$  is the successor belief and  $\gamma_o$  is the conditional plan for the observation  $o$  specified by the observation strategy  $\nu$  of the conditional plan  $\gamma$ .

Since the value function  $V_\gamma$  is linear with respect to the belief space [67], Formula 5.1 can be rewritten as:

$$V_\gamma(b) = \alpha_\gamma \cdot b \quad (5.2)$$

where  $\alpha_\gamma$  is the  $\alpha$ -vector that specifies the reward for every state  $s \in \mathcal{S}$  following the conditional plan  $\gamma = (a, \nu)$ :

$$\alpha_\gamma(s) = r(s, a) + \sum_{o \in \mathcal{O}} \sum_{s' \in \mathcal{S}} \mathcal{Z}(s', a, o) \mathcal{T}(s, a, s') \alpha_{\gamma_o}(s') \quad (5.3)$$

Therefore, the value function  $V_\pi$  of the policy  $\pi$  can be represented as a set of  $\alpha$ -vectors  $V_\pi = \{\alpha_{\gamma_1}, \alpha_{\gamma_2}, \dots\}$ . For every belief  $b \in \mathcal{B}_\pi$ ,  $V_\pi(b) = V_\gamma(b) = \alpha_\gamma \cdot b$ , where  $\gamma$  is the conditional plan for the belief  $b$  specified by the policy  $\pi$ .

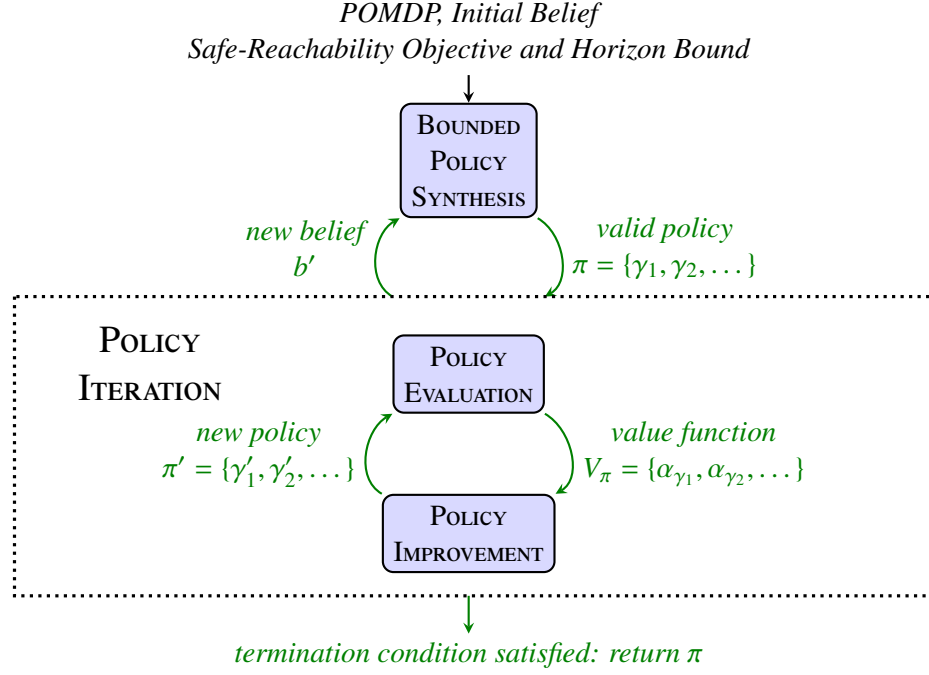


Figure 5.1 : Overview of the PBPS algorithm. PBPS interleaves computation of valid policies and policy iteration.

### 5.1.2 Problem Statement

Given a POMDP  $P = (\mathcal{S}, \mathcal{A}, \mathcal{T}, O, \mathcal{Z}, r)$ , an initial belief  $b_0$ , a safe-reachability objective  $\mathcal{G}$ , and a horizon bound  $h$ , the goal is to synthesize a *valid*  $k$ -step ( $k \leq h$ ) policy  $\pi_{\mathcal{G}}^*$  that maximizes the expected reward from the initial belief  $b_0$

$$\pi_{\mathcal{G}}^* = \underset{\text{valid } \pi}{\operatorname{argmax}} V_{\pi}(b_0) \quad (5.4)$$

Note that  $\pi_{\mathcal{G}}^*$  is different from the optimal policy  $\pi^* = \underset{\pi}{\operatorname{argmax}} V_{\pi}(b_0)$  without the requirement of satisfying the safe-reachability objective ( $\pi^*$  may not be valid).

---

**Algorithm 6: PBPS**


---

**Input:**  
 POMDP  $P$ , Initial Belief  $b_0$ , Safe-Reachability Objective  $\mathcal{G}$ , Horizon Bound  $h$   
**Output:** A *Valid*  $k$ -Step Policy  $\pi$

```

91  $\pi \leftarrow \text{BPS}(P, b_0, \mathcal{G}, 0, h);$                                 /* Compute an initial valid policy */
92 while true do
93   Compute  $V_\pi = \{\alpha_{\gamma_1}, \alpha_{\gamma_2}, \dots\};$                       /* Policy evaluation */
94   /* Start policy Improvement */
95    $\pi' \leftarrow \emptyset$ 
96   /* Point-based backup */
97   foreach  $b \in \mathcal{B}_\pi$  do
98     foreach  $a \in \mathcal{A}$  do
99        $v_a \leftarrow \emptyset$ 
100      foreach  $o \in \mathcal{O}$  do
101         $b_a^o \leftarrow \mathcal{T}_\mathcal{B}(b, a, o);$                                 /* Compute successor belief */
102         $\Gamma \leftarrow \emptyset$ 
103        foreach  $\gamma \in \pi$  do
104          if  $\Omega_{\gamma, b_a^o} \subseteq \Omega_\mathcal{G}$  then
105            /*  $\gamma$  is valid */
106             $\Gamma \leftarrow \Gamma \cup \{\gamma\}$ 
107          if  $\Gamma = \emptyset$  then
108            /* Every  $\gamma \in \pi$  is invalid, invoke BPS to explore */
109             $\pi_a^o \leftarrow \text{BPS}(P, b_a^o, \mathcal{G}, 0, h)$ 
110            /* Add this new conditional plan to  $\pi'$  for improvement */
111             $\pi' \leftarrow \pi' \cup \pi_a^o$ 
112             $\gamma_a^o \leftarrow \pi_a^o(b_a^o)$ 
113          else
114             $\gamma_a^o \leftarrow \underset{\gamma \in \Gamma}{\text{argmax}} \alpha_\gamma \cdot b_a^o$ 
115            /* Record observation strategy */
116             $v_a(o) = \gamma_a^o$ 
117          foreach  $s \in \mathcal{S}$  do
118             $\alpha_a(s) \leftarrow r(s, a) + \sum_{o \in \mathcal{O}} \sum_{s' \in \mathcal{S}} \mathcal{Z}(s', a, o) \mathcal{T}(s, a, s') \alpha_{\gamma_a^o}(s')$ 
119           $a' \leftarrow \underset{a \in \mathcal{A}}{\text{argmax}} \alpha_a \cdot b$ 
120          /* Construct a new conditional plan */
121           $\gamma' = (a', v_{a'})$ 
122           $\pi' \leftarrow \pi' \cup \{\gamma'\}$ 
123        if  $\left( \frac{1}{|\mathcal{B}_\pi|} \sum_{b \in \mathcal{B}_\pi} (V_{\pi'}(b) - V_\pi(b)) \right) \leq \epsilon$  then
124          /* termination condition satisfied */
125          return  $\pi'$ 
126         $\pi = \pi'$ 

```

---

## 5.2 Point-Based Policy Synthesis

Figure 5.1 shows the overview of the Point-Based Policy Synthesis (PBPS) approach (Algorithm 6). PBPS combines Bounded Policy Synthesis (BPS) [78] and Point-Based Policy Iteration (PBPI) [36] to compute a good approximate policy  $\pi$  that satisfies the given safe-reachability objective..

At a high level, PBPS aims to achieve both boolean (safe-reachability) and quantitative (maximizing rewards) objectives. In order to satisfy the given safe-reachability objective, PBPS applies BPS to efficiently explore the goal-constrained belief space  $\mathcal{B}_G$  and generate a valid  $k$ -step policy  $\pi$  (line 91). This valid policy  $\pi$  may not be optimal. Therefore, PBPS applies PBPI to compute an improved policy  $\pi'$  with respect to the value function  $V_\pi$  of the current policy  $\pi$ . Although the goal-constrained belief space over a bounded horizon is finite, performing the backup on the entire goal-constrained belief space is still very expensive [76]. Therefore, the point-based backup [42, 58] is performed on the representative set  $\mathcal{B}_\pi$  of beliefs visited by the current policy  $\pi$  to improve efficiency.

This improved policy  $\pi'$  may reach some successor belief  $b_a^o = \mathcal{T}_B(b, a, o)$  for a belief  $b \in \mathcal{B}_\pi$  (line 99), and none of the existing conditional plans in  $\pi$  are valid starting from  $b_a^o$  (line 104). In this case, PBPS invokes BPS to check whether  $b_a^o$  belongs to the goal-constrained belief space (line 105). PBPS alternates between the computation of valid policies and policy iteration until the policy improvement  $\frac{1}{|\mathcal{B}_\pi|} \sum_{b \in \mathcal{B}_\pi} (V_{\pi'}(b) - V_\pi(b))$  meets the threshold  $\epsilon$  (line 116).

Next section describes each component (Figure 5.1) of the PBPS algorithm (Algorithm 6).

### 5.2.1 Bounded Policy Synthesis

For a given POMDP  $P$ , an initial belief  $b_0$ , a safe-reachability objective  $\mathcal{G} = (Dest, Safe)$  and a horizon bound  $h$ , first the same BPS algorithm presented in Chapter 3 is applied to compute a valid  $k$ -step ( $k \leq h$ ) policy  $\pi$ . For completeness, a brief summary of BPS is provided here. See Chapter 3 for more details.

BPS first symbolically encodes the goal-constrained belief space over a bounded horizon  $k$  as a compact logical formula  $\Phi_k$ .  $\Phi_k$  compactly represents the requirement of reaching a goal belief  $b \in Dest$  safely in  $k$  steps, based on the encoding from Bounded Model Checking [3]. Then BPS computes a valid plan by checking the satisfiability of the constraint  $\Phi_k$  through an SMT solver [19].

If  $\Phi_k$  is satisfiable, the SMT solver returns a valid plan  $\sigma_k = (b_0, a_1, o_1, \dots, b_k)$  (the dashed green path in Figure 2.1).  $\sigma_k$  only covers a particular observation  $o_i$  at step  $i$ . BPS tries to generate a valid policy  $\pi$  based on this valid plan  $\sigma_k$  by considering all other observations, i.e., other branches following the rectangle node at each step.

If  $\Phi_k$  is unsatisfiable and there is no new valid plan for the current horizon, BPS increases the horizon by one and repeat the above steps until a valid policy is found or a given horizon bound is reached.

### 5.2.2 Policy Iteration

Once a valid  $k$ -step policy  $\pi$  is found, PBPS tries to improve  $\pi$  by adapting the PBPI algorithm presented in [36]. PBPI considers infinite-horizon policies represented as finite-state controllers while PBPS focuses on bounded-horizon policies represented as a set of conditional plans. Moreover, PBPI only deals with quantitative objectives while PBPS needs to consider both boolean (safe-reachability) and quantitative objectives. Therefore, PBPS cannot directly apply PBPI to compute an improved policy. Instead, PBPS adapts the policy

evaluation and the policy improvement steps in PBPI for policy iteration.

### Policy Evaluation

For a valid  $k$ -step policy  $\pi = \{\gamma_1, \gamma_2, \dots\}$ , PBPS recursively computes the  $\alpha$ -vector  $\alpha_\gamma$  for each conditional plan  $\gamma = (a, v)$ :

- If  $\gamma$  is a conditional plan associated with a terminal belief,  $\alpha_\gamma(s) = r(s, a)$ .
- If  $\gamma$  is a conditional plan associated with a non-terminal belief, PBPS computes the  $\alpha$ -vector  $\alpha_\gamma$  based on Formula 5.3.

Then the value function of  $\pi$  can be represented as a set of  $\alpha$ -vectors  $V_\pi = \{\alpha_{\gamma_1}, \alpha_{\gamma_2}, \dots\}$  (line 93).

### Policy Improvement

In the policy improvement step, PBPS computes an improved policy  $\pi'$  based on the value function  $V_\pi$  of the current policy  $\pi$ . This policy improvement step applies the point-based backup algorithm [42, 58] to the finite set  $\mathcal{B}_\pi$  of beliefs visited by the current policy  $\pi$ .

Since PBPS needs to consider not only a quantitative objective as in the standard point-based backup, but also a boolean objective that defines the validity of a policy, PBPS can not directly apply the standard point-based backup in PBPS. There are two important differences between PBPS and the standard point-based backup.

First, the standard point-based backup loops over all  $\alpha$ -vectors in the value function  $V_\pi$  of the current policy  $\pi$  to find the best  $\alpha$ -vector  $\alpha_a^o = \underset{\alpha \in V_\pi}{\operatorname{argmax}} \alpha \cdot b_a^o$  for the successor belief  $b_a^o = \mathcal{T}_B(b, a, o)$ . Conceptually, this step applies each conditional plan  $\gamma$  in the policy  $\pi$  to the belief  $b_a^o$  and finds the best conditional plan for  $b_a^o$ . However, not every conditional plan  $\gamma$  in the policy  $\pi$  is valid starting from the belief  $b_a^o$ . Therefore, PBPS performs an additional

filtering step (lines 101, 102, 103) to construct a subset  $\Gamma \subseteq \pi$  of valid conditional plans starting from the belief  $b_a^o$ . Then PBPS selects the best conditional plan from these valid ones (line 109).

Second, after PBPS performs the filtering step described above, it is possible that the current policy  $\pi$  does not contain a valid conditional plan for  $b_a^o$ , i.e.,  $\Gamma = \emptyset$  (line 104). In this case, PBPS invokes BPS again to compute a valid policy  $\pi_a^o$  for the belief  $b_a^o$  (line 105) and add  $\pi_a^o$  to the new policy  $\pi'$  so that in later iterations, the value of  $\pi_a^o$  (line 106) can be improved. By constructing this new policy  $\pi_a^o$ , new belief regions that have not been reached by previous policies can be explored. This exploration step together with the point-based backup (when a different action becomes more optimal) expands the belief set  $\mathcal{B}_\pi$ . As pointed out by previous works [36, 42, 58], this expansion of the representative belief set  $\mathcal{B}_\pi$  is crucial to the quality of the constructed policy.

### 5.2.3 Algorithm Analysis

This section provides two theorems to address the important properties of PBPS. Theorem 5.2.1 shows that PBPS maintains validity and keeps improving the value of the policy at each iteration before termination. Theorem 5.2.2 shows that the error introduced by PBPS due to approximation is bounded.

**Theorem 5.2.1.** At each iteration before termination, PBPS transforms a policy  $\pi$  to a new policy  $\pi'$ . For each belief  $b \in \mathcal{B}_\pi$ ,  $\gamma' = \pi'(b)$  is valid and  $V_{\pi'}(b) \geq V_\pi(b)$ . For at least one belief  $b \in \mathcal{B}_\pi$ ,  $V_{\pi'}(b) > V_\pi(b)$ .

*Proof.* Each iteration of PBPS consists of two steps:

- In the policy evaluation step, PBPS computes the exact value function  $V_\pi$  of the current policy  $\pi$  (line 93).

- In the point-based backup step, PBPS constructs a conditional plan  $\gamma$  for each belief  $b \in \mathcal{B}_\pi$ . According to Algorithm 6, for each observation  $o$ , PBPS selects the best conditional plan  $\gamma_a^o$  from the subset  $\Gamma \subseteq \pi$  of valid conditional plans starting from the successor belief  $b_a^o$  (line 109). When  $\Gamma = \emptyset$ , PBPS invokes BPS to construct a new valid conditional plan  $\gamma_a^o$  for the belief  $b_a^o$  (lines 105, 106, 107). Then PBPS selects the best action  $a$  for the belief  $b$  (line 113) and construct the best conditional plan  $\gamma$  (line 114) with respect to the value function  $V_\pi$  of the current policy  $\pi$ . By construction,  $\gamma$  is also valid starting from the belief  $b$ .

Therefore, PBPS can not cause a reduction in the value of any belief  $b \in \mathcal{B}_\pi$  and PBPS always produce a valid policy  $\pi'$ . According to the termination condition (line 116),  $\pi'$  improves the value for at least one belief  $b \in \mathcal{B}_\pi$  before termination.  $\square$

Theorem 5.2.1 states that PBPS maintains validity and keeps improving the value of the policy after each iteration.

PBPS is an approximation method that performs the point-based backup on the representative set  $\mathcal{B}_\pi$  of beliefs visited by the policy  $\pi$  rather than the entire goal-constrained belief space  $\mathcal{B}_\mathcal{G}$ . As a result, PBPS implicitly prunes conditional plans for every belief  $b \in \mathcal{B}_\mathcal{G} - \mathcal{B}_\pi$ . This implicit pruning may remove conditional plans that are part of the optimal policy  $\pi_\mathcal{G}^*$  that is also valid with respect to the safe-reachability objective  $\mathcal{G}$ , producing a suboptimal policy.

Note that  $\pi_\mathcal{G}^*$  is different from the optimal policy  $\pi^*$  without the requirement of satisfying the safe-reachability objective ( $\pi^*$  may not be valid). As PBPS continues improving  $\pi$ , the value  $V_\pi$  is getting closer to the value  $V_{\pi_\mathcal{G}^*}$ , but  $V_\pi$  may not converge to the optimal value  $V_{\pi^*}$  due to the additional safe-reachability objective. Let  $\delta_\pi = \max_{b \in \mathcal{B}_\pi} (V_{\pi^*}(b) - V_\pi(b))$  to be the measurement of the difference between  $\pi^*$  and  $\pi$ .  $\delta_\pi$  can also be seen as the



measurement of the difference between the quantitative objective and the safe-reachability objective. Intuitively, if proper rewards for goal states and unsafe states are set,  $\delta_\pi$  should not be too large since the reward function also encodes the safe-reachability objectives to some extent, and the difference between the quantitative objective and the safe-reachability objective is small. However, as discussed in [78], there exist domains where no reward function exactly encodes the safe-reachability objective and  $\delta_\pi$  may always be greater than 0. How to design a proper reward function that minimizes  $\delta_\pi$  is beyond the scope of this work.

The following theorem shows the error introduced by PBPS is bounded.

**Theorem 5.2.2.** PBPS produces a policy  $\pi$  with error  $\eta = V_{\pi_{\mathcal{G}}^*}(b_0) - V_\pi(b_0)$  bounded by

$$\eta \leq h(r_{\max} - r_{\min})d_{\mathcal{B}_\pi} + \delta_\pi \quad (5.5)$$

where  $b_0$  is the initial belief,  $h$  is the horizon bound,  $r_{\max} = \max_{s,a} r(s, a)$ ,  $r_{\min} = \min_{s,a} r(s, a)$ ,  $d_{\mathcal{B}_\pi} = \max_{b' \in \mathcal{B}_\mathcal{G}} \min_{b \in \mathcal{B}_\pi} \|b - b'\|$  is the maximum distance from any  $b' \in \mathcal{B}_\mathcal{G}$  to  $\mathcal{B}_\pi$ .

*Proof.* Let  $\gamma^* = (a^*, v^*)$  be the optimal conditional plan specified by  $\pi_{\mathcal{G}}^*$  for the belief  $b_0$

and  $\gamma = (a, \nu)$  be the conditional plan specified by  $\pi$  for the belief  $b_0$ .

$$\begin{aligned}
\eta &= V_{\pi_{\mathcal{G}}^*}(b_0) - V_{\pi}(b_0) = V_{\gamma^*}(b_0) - V_{\gamma}(b_0) \\
&= \left( \sum_{s \in \mathcal{S}} r(s, a^*)b(s) + \sum_{o \in \mathcal{O}} \Pr(o|b_0, a^*)V_{\pi_{\mathcal{G}}^*}(b_{a^*}^o) \right) \\
&\quad - \left( \sum_{s \in \mathcal{S}} r(s, a)b(s) + \sum_{o \in \mathcal{O}} \Pr(o|b_0, a)V_{\pi}(b_a^o) \right) \quad \text{Formula 5.1} \\
&= \left( \sum_{s \in \mathcal{S}} r(s, a^*)b(s) + \sum_{o \in \mathcal{O}} \Pr(o|b_0, a^*)V_{\pi_{\mathcal{G}}^*}(b_{a^*}^o) \right) \\
&\quad - \left( \sum_{s \in \mathcal{S}} r(s, a)b(s) + \sum_{o \in \mathcal{O}} \Pr(o|b_0, a)V_{\pi}(b_a^o) \right) \\
&\quad + \left( \sum_{s \in \mathcal{S}} r(s, a^*)b(s) + \sum_{o \in \mathcal{O}} \Pr(o|b_0, a^*)V_{\pi}(b_{a^*}^o) \right) \\
&\quad - \left( \sum_{s \in \mathcal{S}} r(s, a^*)b(s) + \sum_{o \in \mathcal{O}} \Pr(o|b_0, a^*)V_{\pi}(b_{a^*}^o) \right) \quad \text{add 0} \\
&= \sum_{o \in \mathcal{O}} \Pr(o|b_0, a^*) \left( V_{\pi_{\mathcal{G}}^*}(b_{a^*}^o) - V_{\pi}(b_{a^*}^o) \right) \\
&\quad + \left( \sum_{s \in \mathcal{S}} r(s, a^*)b(s) + \sum_{o \in \mathcal{O}} \Pr(o|b_0, a^*)V_{\pi}(b_{a^*}^o) \right) \\
&\quad - \left( \sum_{s \in \mathcal{S}} r(s, a)b(s) + \sum_{o \in \mathcal{O}} \Pr(o|b_0, a)V_{\pi}(b_a^o) \right) \quad (5.6)
\end{aligned}$$

Since the policy  $\pi$  specify the conditional plan  $\gamma$  for the belief  $b_0$ ,  $\gamma$  should be the best conditional plan for  $b_0$  w.r.t. the value function  $V_{\pi}$  and thus the sum of last two terms in Formula 5.6:  $\left( \sum_{s \in \mathcal{S}} r(s, a^*)b(s) + \sum_{o \in \mathcal{O}} \Pr(o|b_0, a^*)V_{\pi}(b_{a^*}^o) \right) - \left( \sum_{s \in \mathcal{S}} r(s, a)b(s) + \sum_{o \in \mathcal{O}} \Pr(o|b_0, a)V_{\pi}(b_a^o) \right) \leq 0$ . Therefore,  $\eta \leq \sum_{o \in \mathcal{O}} \Pr(o|b_0, a^*) \left( V_{\pi_{\mathcal{G}}^*}(b_{a^*}^o) - V_{\pi}(b_{a^*}^o) \right)$ . Let  $b' \in \bigcup_{o \in \mathcal{O}} \{b_{a^*}^o\}$  be the successor belief where PBPS makes its worst error,  $\gamma_o^*$  be the optimal conditional plan specified by the policy  $\pi_{\mathcal{G}}^*$  for  $b'$ ,  $\gamma_o$  be the best conditional plan in  $\pi$  for  $b'$  and  $b \in \mathcal{B}_{\pi}$  be the belief

associated with  $\gamma_o$ . Then

$$\begin{aligned}
 \eta &\leq \sum_{o \in \mathcal{O}} \Pr(o|b_0, a^*) (\alpha_{\gamma_o^*} \cdot b' - \alpha_{\gamma_o} \cdot b') \\
 &\leq \alpha_{\gamma_o^*} \cdot b' - \alpha_{\gamma_o} \cdot b' \quad \sum_{o \in \mathcal{O}} \Pr(o|b_0, a^*) = 1 \\
 &= (\alpha_{\gamma_o^*} - \alpha_{\gamma_o}) \cdot (b' - b) + (\alpha_{\gamma_o^*} - \alpha_{\gamma_o}) \cdot b \quad \text{add 0}
 \end{aligned} \tag{5.7}$$

Following the derivations in [58], for the first term in Formula 5.7

$$\begin{aligned}
 &(\alpha_{\gamma_o^*} - \alpha_{\gamma_o}) \cdot (b' - b) \\
 &\leq \|\alpha_{\gamma_o^*} - \alpha_{\gamma_o}\|_{\infty} \|b' - b\|_1 && \text{Hölder's inequality} \\
 &\leq \|\alpha_{\gamma_o^*} - \alpha_{\gamma_o}\|_{\infty} d_{\mathcal{B}_{\pi}} && \text{definition of } d_{\mathcal{B}_{\pi}} \\
 &\leq h(r_{\max} - r_{\min}) d_{\mathcal{B}_{\pi}}
 \end{aligned}$$

The last inequality holds because  $\alpha$ -vector represents the reward starting from some state within the horizon bound  $h$ .

For the second term  $(\alpha_{\gamma_o^*} - \alpha_{\gamma_o}) \cdot b$  in Formula 5.7, since  $\gamma_o^*$  may not be valid starting from  $b$ ,  $\alpha_{\gamma_o^*} \cdot b$  is at most  $V_{\pi^*}(b)$  where  $\pi^*$  is an optimal policy without the requirement of satisfying the safe-reachability objective and may not be valid. According to the definition of  $\delta_{\pi}$ ,  $(\alpha_{\gamma_o^*} - \alpha_{\gamma_o}) \cdot b \leq \delta_{\pi}$ . Based on the above analysis,  $\eta \leq h(r_{\max} - r_{\min}) d_{\mathcal{B}_{\pi}} + \delta_{\pi}$ .  $\square$

As the set  $\mathcal{B}_{\pi}$  is expanded through the exploration step (line 105) and the point-based backup,  $\mathcal{B}_{\pi}$  covers more beliefs from the goal-constrained belief space  $\mathcal{B}_{\mathcal{G}}$ . Therefore, the first term  $h(r_{\max} - r_{\min}) d_{\mathcal{B}_{\pi}}$  in the error bound converges to 0 since the distance  $d_{\mathcal{B}_{\pi}}$  converges to 0. As discussed before, the value of the second term  $\delta_{\pi}$  in the error bound is closely related to the reward function. How to design a proper reward function that minimizes the

second term  $\delta_r$  of the error bound is still an open question.

### 5.3 Experiments

PBPS is tested on two domains: the kitchen domain (horizon bound  $h = 20$ ) presented in [78] and the Tag domain (horizon bound  $h = 30$ ) presented in [58]. PBPS is also validated on a Fetch robot for the scenario shown in Figure 1.1 ( $h = 20$ ). Z3 [19] is used as the backend SMT solver. All experiments were conducted on a 2.9 GHz Intel® processor with 16 GB memory.

#### Kitchen Domain

In the kitchen domain [78], a robot needs to eventually pick up a cup from the storage while avoiding collisions with  $M$  uncertain obstacles. This kitchen domain is an example scenario that requires a correctness guarantee of accomplishing tasks, and POMDPs with boolean objectives provide a better guarantee than the purely quantitative POMDP formulations [78].

The kitchen environment is discretized into  $N = 36$  regions. The actuation and perception of the robot are imperfect, modeled as ten uncertain robot actions: *move* and *look* in four directions, pick-up using the left hand and pick-up using the right hand. The robot starts at a known initial location. However, due to the robot’s imperfect perception, the location of the robot and the locations of uncertain obstacles are all partially observable during execution. The same safe-reachability objective  $\mathcal{G} = (Dest, Safe)$  in [78] is used to

specify the task:

$$Dest = \{b \in \mathcal{B} \mid \left( \sum b(\text{target cup in hand}) \right) > 1 - \delta_1\}$$

$$Safe = \{b \in \mathcal{B} \mid \left( \sum b(\text{robot in collision}) \right) < \delta_2\}$$

where  $\delta_1$  and  $\delta_2$  are small values that represent tolerance.

For the quantitative objective, it is desired that the robot avoid certain regions as much as possible by assigning the reward of  $-10$  for states where the robot is in these regions. A reward of  $-1$  is assigned for each action.

### Tag Domain

In the Tag domain [58], the task for the robot is to search for and tag a moving agent in a grid with 29 locations. The agent follows a fixed strategy that intentionally moves away from the robot. Both the robot and the agent can move in four directions or stay. The robot's location is fully observable while the agent's location is unobservable unless the robot and the agent are in the same location. The safe-reachability objective  $\mathcal{G} = (Dest, Safe)$  for this domain is: (1) *Dest* contains beliefs where the robot can tag the agent with a probability at least  $\delta$ ; (2) *Safe* =  $\mathcal{B}$  since there are no unsafe states and all beliefs are safe beliefs.

#### 5.3.1 Results

The performance results of PBPS are summarized in Table 5.1. To evaluate how the quality of constructed policies is affected by the exploration step (line 105) of PBPS (Algorithm 6) where BPS is invoked to explore an uncovered belief region, PBPS is tested in two settings: with and without the exploration step.

As shown in Table 5.1, PBPS with exploration achieves much better reward compared

Table 5.1 : Performance of PBPS with and without the exploration step for different problems.

Domain	Reward		Time (s)		$ \mathcal{B}_\pi $	
	w. exp.	no exp.	w. exp.	no exp.	w. exp.	no exp.
Kitchen ( $M = 1$ )	-9.350	-9.350	18.166	10.321	147	94
Kitchen ( $M = 2$ )	-13.040	-24.296	347.281	33.798	396	73
Kitchen ( $M = 3$ )	-16.882	-31.801	3611.684	290.019	474	64
Tag ( $\delta = 0.5$ )	-6.987	-9.070	509.235	21.352	370	109
Tag ( $\delta = 0.6$ )	-6.871	-9.108	1132.996	21.985	811	107

to PBPS without exploration for large problems. However, this exploration step is expensive and requires more computation time. The results also demonstrate the advantage of approximate policies against exact policies: computing exact policies requires exhaustively exploring the whole goal-constrained belief space, which is intractable in practice since exploration is costly as shown in Table 5.1. On the contrary, PBPS produces approximate policies by performing the point-based backup, and only explores belief regions of the goal-constrained belief space that are promising based on the current value function.

To compare with solvers for POMDPs with only quantitative objectives, a state-of-the-art POMDP solver SARSOP [42] is tested on the Tag domain. The policy produced by SARSOP achieves an expected reward of  $-6.02$  but only visits beliefs where the probability of tagging the agent is at most  $0.539$ . The policy generated by PBPS for the Tag domain with  $\delta = 0.6$  guarantees that the robot eventually visits a belief where the probability of tagging the agent is at least  $0.6$ . For the kitchen domain, since PBPS considers both safe-reachability and quantitative objectives, PBPS also provides a better guarantee of accomplishing tasks than solvers for purely quantitative POMDP formulations as discussed in [78].

### 5.3.2 Physical Validation

PBPS is validated on a Fetch robot for the domain shown in Figure 1.1. The setup of this domain is similar to the kitchen domain. The Fetch needs to pick up a target object (the blue can on the table) while avoiding collisions with uncertain obstacles such as floor signs and file cabinets, which can be placed in different locations. There are two regions marked with red tape, and it is desired that the robot avoid these red regions as much as possible by assigning the reward of  $-10$  for states where the robot is in these regions. A reward of  $-1$  is assigned for each action.

The POMDP's state space consists of robot locations and object locations. A Vicon system is used to detect object locations, which is usually accurate but can still produce false negative and false positive due to occlusion or inappropriate Vicon marker configurations on objects. The false negative and false positive probabilities can be estimated by counting the false negative and false positive events during 100 Vicon detections. The POMDP's probabilistic observation function is defined based on the false negative and false positive probabilities. Sometimes the Fetch may fail to move its base when given a *move* action command and stay in the same place. The failure probability of these move actions can be estimated by counting the failure events during 100 *move* action executions. The POMDP's probabilistic transition function is defined based on this failure probability.

Both BPS and PBPS are evaluated in this office domain. Figure 5.2a, 5.2b, 5.2c, 5.2d, and 5.2e show the execution of the policy constructed by BPS. Figure 5.2f, 5.2g, 5.2h, 5.2i, and 5.2j show the execution of the policy constructed by PBPS. As shown in these figures, both executions accomplish the task safely. However, the execution for BPS visits both red regions (Figure 5.2b and 5.2d) that the robot should avoid while the execution for PBPS visits zero red regions. Therefore, PBPS produces a policy that is more optimal than that produced by BPS by considering both boolean and quantitative objectives.

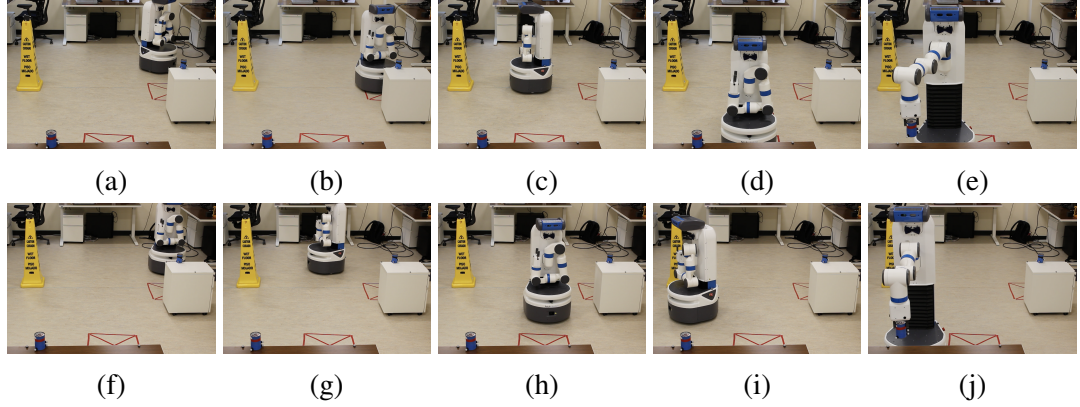


Figure 5.2 : Executions of policies constructed by BPS (figures in the first row) and PBPS (figures in the second row) for the domain shown in Figure 1.1.

## 5.4 Discussion

This chapter presented a new policy synthesis approach called PBPS for POMDPs with both safe-reachability and quantitative objectives. PBPS combines BPS [78] and Point-Based Policy Iteration [36]: BPS is applied to efficiently explore the goal-constrained belief space, and the point-based backup is performed on a finite subset of representative beliefs from the goal-constrained belief space to compute good approximate policies that satisfy safe-reachability objectives. The theoretical analysis shows that PBPS maintains validity and guarantees improved policies at each iteration before termination. Moreover, the error introduced by PBPS is bounded. Both simulation and physical experiments demonstrate that the policies constructed by PBPS achieve the safe-reachability objective and are of high quality with respect to the quantitative objective.

This thesis focuses on a common boolean objective: safe-reachability. While many robot tasks can be formulated using safe-reachability objectives, there are settings in robotics such as patrolling and surveillance that require general temporal properties to specify the tasks. Investigating how to extend PBPS to deal with general temporal properties is a



promising future direction for this work and its application in robotics. The current implementation of PBPS is restricted to discrete POMDPs. How to extend PBPS for continuous POMDPs is another important ongoing question.

## Chapter 6

### Conclusions and Future Work

Planning robust executions under uncertainty is a fundamental challenge for building autonomous robots, especially in domains like disaster rescue, self-driving vehicles, health care, and even around the home. POMDPs [67,70] provide a standard framework for modeling a variety of robot applications in the face of uncertainty. Traditionally, POMDPs have been posed with respect to quantitative objectives such as maximizing rewards. However, for many robot applications that desire a strong correctness guarantee of completing tasks, POMDPs with boolean requirements are natural formulations. This thesis investigates POMDPs with a common boolean requirement: safe-reachability, which requires that with a probability above a threshold, a goal state is eventually reached while keeping the probability of visiting unsafe states below a different threshold. Many robot tasks such as the one shown in Figure 3.7 can be formulated as a safe-reachability objective. Moreover, this thesis discusses an example POMDP domain in Section 2.2, showing that for certain robot domains that require accomplishing tasks safely, POMDPs with safe-reachability objectives provide a better guarantee of both safety and reachability than existing quantitative POMDP formulations.

The central challenge in solving POMDPs is the requirement of reasoning over a vast space of probability distributions called the belief space. What’s more, it has been shown that the policy synthesis problem of POMDPs with reachability objectives is undecidable in general [10,50,57]. To address these challenges, this thesis introduces the notion of a goal-constrained belief space, which only contains beliefs (probability distributions over states)

reachable from the initial belief under desired executions that can achieve the given safe-reachability objective. In general, this constrained space is much smaller than the original belief space. Based on this notion, this thesis presents an offline synthesis approach BPS, which compactly represents the goal-constrained belief space over a bounded horizon using a set of symbolic constraints, and employs an incremental Satisfiability Modulo Theories (SMT) solver [19] to efficiently search for a valid policy over it. BPS is evaluated using a case study involving a partially observable robotic domain with uncertain obstacles. The results show that BPS can synthesize policies over large belief spaces with a small number of SMT solver calls by focusing on the goal-constrained belief space.

The solutions to POMDPs are policies that specify the action to take contingent on an observation received. A full policy that covers all possible events is generally expensive to compute. To improve computational efficiency, this thesis introduces the notion of *partial policies* that only cover a sampled subset of all possible observations at each step and approximates full policies. Based on this notion, this thesis presents an online planning approach OPPS, which constructs a partial policy parameterized by a *replanning probability*. In addition, this thesis proves that the probability of the constructed partial policy failing is bounded by the replanning probability. Therefore, this replanning probability measures the approximation quality of a partial policy, and OPPS allows users to specify an appropriate bound on the replanning probability to balance efficiency and correctness. Moreover, OPPS updates this bound during partial policy construction to quickly detect if the current partial policy meets the bound and avoid unnecessary computation. OPPS is validated in several robotic domains, and the results show that OPPS outperforms the offline approach BPS and can solve problems that are beyond the capabilities of BPS within the time limit.

On the one hand, Safe-reachability objectives provide a strong correctness guarantee of accomplishing tasks safely. On the other hand, the traditional quantitative objectives offer

a strong optimality guarantee. For robotic domains that require both correctness and optimality, the POMDP model with both safe-reachability and quantitative objects are natural formulations. Based on this formulation, this thesis presents a policy synthesis approach PBPS, which computes good approximate policies that also satisfy the safe-reachability objective by combining the BPS method for POMDPs with only safe-reachability objectives and policy iteration. To improve efficiency, PBPS produces approximated policies by performing the point-based backup [42, 58] on a representative belief subset of all beliefs that are reachable from initial beliefs under desired executions that can achieve the safe-reachability objectives. This thesis proves that PBPS maintains validity and guarantees improved policies at each iteration. Moreover, the error introduced by this approximation approach PBPS is bounded. PBPS is evaluated in several robotic domains under uncertainty. Both simulation and physical experiments demonstrate that the policies constructed by PBPS achieve the safe-reachability objective and are of high quality with respect to the quantitative objective.

## 6.1 Open Questions

This thesis analyzes an example domain in Section 2.2 to qualitatively demonstrate the difference between POMDPs with boolean objectives and the traditional quantitative POMDP formulations. How to determine which POMDP formulation should be used for a particular domain is still an open question. Analyzing a wide range of situations and performing a quantitative analysis on the difference between POMDPs with boolean objectives and the quantitative POMDP formulations is an interesting future topic. Moreover, the theoretical analysis of the approximation method PBPS for POMDPs with both boolean and quantitative objectives reveals that the difference between boolean and quantitative objectives plays an important role on the optimality guarantee. The quantitative analysis of these two

objectives may provide an answer on how to minimize the error due to considering both objectives at the same time in PBPS.

All the policy synthesis approaches presented in this thesis are restricted to discrete POMDPs. While many robot applications can be modeled using this discrete representation, discretization often suffers from the “curse of dimensionality”. Investigating how to deal with continuous POMDPs [1, 33, 64] directly without discretization is a promising future direction for the work presented in this thesis and its application in robotics.

This thesis focuses on a common boolean objective: safe-reachability. While many robot tasks can be formulated using safe-reachability objectives, there are settings in robotics such as patrolling and surveillance that require general temporal properties to specify the tasks. Investigating how to extend the policy synthesis approaches presented in this thesis to deal with general temporal properties is another critical ongoing question.

## Bibliography

- [1] H. Bai, D. Hsu, W. S. Lee, and V. A. Ngo, “Monte Carlo value iteration for continuous-state POMDPs,” in *Algorithmic Foundations of Robotics IX: Selected Contributions of the Ninth International Workshop on the Algorithmic Foundations of Robotics*, D. Hsu, V. Isler, J.-C. Latombe, and M. C. Lin, Eds. Berlin, Heidelberg: Springer, 2011, pp. 175–191.
- [2] J. Bidot, L. Karlsson, F. Lagriffoul, and A. Saffiotti, “Geometric backtracking for combined task and motion planning in robotic systems,” *Artificial Intelligence*, vol. 247, pp. 229–265, 2017.
- [3] A. Biere, A. Cimatti, E. M. Clarke, O. Strichman, and Y. Zhu, “Bounded model checking,” *Advances in Computers*, vol. 58, pp. 117–148, 2003.
- [4] N. Bjørner, A. Phan, and L. Fleckenstein, “vz - an optimizing SMT solver,” in *Proceedings of the 21st International Conference on Tools and Algorithms for the Construction and Analysis of Systems - Volume 9035*. Berlin, Heidelberg: Springer-Verlag, 2015, pp. 194–199.
- [5] R. Bloem, K. Chatterjee, T. A. Henzinger, and B. Jobstmann, “Better quality in synthesis through quantitative objectives,” in *Proceedings of the 21st International Conference on Computer Aided Verification*, ser. CAV ’09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 140–156.
- [6] P. Cai, Y. Luo, D. Hsu, and W. S. Lee, “HyP-DESPOT: A hybrid parallel algorithm for

- online planning under uncertainty,” in *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018.
- [7] K. Chatterjee, M. Chmelík, R. Gupta, and A. Kanodia, “Qualitative analysis of POMDPs with temporal logic specifications for robotics applications,” in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 325–330.
- [8] K. Chatterjee, M. Chmelík, and J. Davies, “A symbolic SAT-based algorithm for almost-sure reachability with small strategies in POMDPs,” in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, ser. AAAI’16. AAAI Press, 2016, pp. 3225–3232.
- [9] K. Chatterjee, M. Chmelík, R. Gupta, and A. Kanodia, “Optimal cost almost-sure reachability in POMDPs,” *Artificial Intelligence*, vol. 234, no. C, pp. 26–48, May 2016.
- [10] K. Chatterjee, M. Chmelík, and M. Tracol, “What is decidable about partially observable Markov decision processes with  $\omega$ -regular objectives,” *Journal of Computer and System Sciences*, vol. 82, no. 5, pp. 878–911, Aug. 2016.
- [11] K. Chatterjee, Z. Komarkova, and J. Kretinsky, “Unifying two views on multiple mean-payoff objectives in Markov decision processes,” in *Proceedings of the 2015 30th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, ser. LICS ’15. Washington, DC, USA: IEEE Computer Society, 2015, pp. 244–256.
- [12] K. Chatterjee, R. Majumdar, and T. A. Henzinger, “Markov decision processes with multiple objectives,” in *Proceedings of the 23rd Annual Conference on Theoretical*

- Aspects of Computer Science*, ser. STACS'06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 325–336.
- [13] S. Chaudhuri, M. Clochard, and A. Solar-Lezama, “Bridging boolean and quantitative synthesis using smoothed proof search,” in *Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, ser. POPL '14. New York, NY, USA: ACM, 2014, pp. 207–220.
  - [14] C. Courcoubetis and M. Yannakakis, “Markov decision processes and regular events,” *IEEE Transactions on Automatic Control*, vol. 43, no. 10, pp. 1399–1418, Oct 1998.
  - [15] —, “The complexity of probabilistic verification,” *Journal of the ACM*, vol. 42, no. 4, pp. 857–907, July 1995.
  - [16] N. T. Dantam, S. Chaudhuri, and L. E. Kavraki, “The task-motion kit: An open source, general-purpose task and motion-planning framework,” *IEEE Robotics Automation Magazine*, vol. 25, no. 3, pp. 61–70, Sept 2018.
  - [17] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. E. Kavraki, “Incremental task and motion planning: A constraint-based approach,” in *Robotics: Science and Systems*, 2016.
  - [18] —, “An incremental constraint-based framework for task and motion planning,” *International Journal of Robotics Research*, 2018.
  - [19] L. De Moura and N. Bjørner, “Z3: An efficient SMT solver,” in *Proceedings of the Theory and Practice of Software, 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, ser. TACAS'08/ETAPS'08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 337–340.



- [20] X. C. Ding, S. L. Smith, C. Belta, and D. Rus, “MDP optimal control under temporal logic constraints,” in *2011 50th IEEE Conference on Decision and Control and European Control Conference*, Dec 2011, pp. 532–538.
- [21] C. Dornhege, P. Eyerich, T. Keller, S. Trüg, M. Brenner, and B. Nebel, *Semantic Attachments for Domain-Independent Planning Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 99–115.
- [22] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison, *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge University Press, 1998.
- [23] E. Erdem, K. Haspalamutgil, C. Palaz, V. Patoglu, and T. Uras, “Combining high-level causal reasoning with low-level geometric reasoning and motion planning for robotic manipulation,” in *2011 IEEE International Conference on Robotics and Automation (ICRA)*, May 2011, pp. 4575–4581.
- [24] K. Etessami, M. Kwiatkowska, M. Y. Vardi, and M. Yannakakis, “Multi-objective model checking of Markov decision processes,” *Logical Methods in Computer Science*, vol. Volume 4, Issue 4, Nov. 2008.
- [25] C. Garrett, T. Lozano-Perez, and L. Kaelbling, “Sample-based methods for factored task and motion planning,” in *Proceedings of Robotics: Science and Systems*, Cambridge, Massachusetts, July 2017.
- [26] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, “FFRob: An efficient heuristic for task and motion planning,” *Algorithmic Foundations of Robotics XI*, vol. 107, p. 179, 2015.

- [27] M. Gharbi, R. Lallement, and R. Alami, “Combining symbolic and geometric planning to synthesize human-aware plans: toward more efficient combined search.” in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2015, pp. 6360–6365.
- [28] M. K. Ghosh, “Markov decision processes with multiple costs,” *Operations Research Letters*, vol. 9, no. 4, pp. 257–260, July 1990.
- [29] D. K. Grady, M. Moll, and L. E. Kavraki, “Extending the applicability of POMDP solutions to robotic tasks,” *IEEE Transactions on Robotics*, vol. 31, no. 4, pp. 948–961, Aug 2015.
- [30] S. Gulwani, “Dimensions in program synthesis,” in *Proceedings of the 12th International ACM SIGPLAN Symposium on Principles and Practice of Declarative Programming*, ser. PPDP ’10. New York, NY, USA: ACM, 2010, pp. 13–24.
- [31] D. Hadfield-Menell, E. Groshev, R. Chitnis, and P. Abbeel, “Modular task and motion planning in belief space,” in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2015, pp. 4991–4998.
- [32] K. He, M. Lahijanian, L. E. Kavraki, and M. Y. Vardi, “Towards manipulation planning with temporal logic specifications,” in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 346–352.
- [33] J. Hoey and P. Poupart, “Solving POMDPs with continuous or large discrete observation spaces,” in *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, ser. IJCAI’05. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005, pp. 1332–1338.

- [34] P. Hou, W. Yeoh, and P. Varakantham, “Solving risk-sensitive POMDPs with and without cost observations,” in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, ser. AAAI’16. AAAI Press, 2016, pp. 3138–3144.
- [35] J. D. Isom, S. P. Meyn, and R. D. Braatz, “Piecewise linear dynamic programming for constrained POMDPs,” in *Proceedings of the 23rd National Conference on Artificial Intelligence*, ser. AAAI’08. AAAI Press, 2008, pp. 291–296.
- [36] S. Ji, R. Parr, H. Li, X. Liao, and L. Carin, “Point-based policy iteration,” in AAAI. AAAI Press, 2007, pp. 1243–1249.
- [37] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, “Planning and acting in partially observable stochastic domains,” *Artificial Intelligence*, vol. 101, no. 1-2, pp. 99–134, May 1998.
- [38] L. P. Kaelbling and T. Lozano-Pérez, “Integrated task and motion planning in belief space,” *International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1194–1227, Aug. 2013.
- [39] ———, “Implicit belief-space pre-images for hierarchical planning and execution,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 5455–5462.
- [40] D. Kim, J. Lee, K.-E. Kim, and P. Poupart, “Point-based value iteration for constrained POMDPs,” in *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, ser. IJCAI’11. AAAI Press, 2011, pp. 1968–1974.
- [41] A. Krontiris and K. Bekris, “Dealing with difficult instances of object rearrangement,” in *Proceedings of Robotics: Science and Systems*, Rome, Italy, July 2015.

- [42] H. Kurniawati, D. Hsu, and W. S. Lee, “SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces,” in *Robotics: Science and Systems*, 2008.
- [43] J. Křetínský and T. Meggendorfer, “Conditional value-at-risk for reachability and mean payoff in Markov decision processes,” in *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, ser. LICS ’18. New York, NY, USA: ACM, 2018, pp. 609–618.
- [44] M. Kwiatkowska, G. Norman, and D. Parker, “PRISM 4.0: Verification of probabilistic real-time systems,” in *Proc. 23rd International Conference on Computer Aided Verification (CAV’11)*, ser. LNCS, G. Gopalakrishnan and S. Qadeer, Eds., vol. 6806. Springer, 2011, pp. 585–591.
- [45] F. Lagriffoul and B. Andres, “Combining task and motion planning,” *International Journal of Robotics Research*, vol. 35, no. 8, pp. 890–927, July 2016.
- [46] M. Lahijanian, S. B. Andersson, and C. Belta, “Control of Markov decision processes from PCTL specifications,” in *Proceedings of the 2011 American Control Conference*, June 2011, pp. 311–316.
- [47] ———, “Temporal logic motion planning and control with probabilistic satisfaction guarantees,” *IEEE Transactions on Robotics*, vol. 28, no. 2, pp. 396–409, April 2012.
- [48] R. Luna, M. Lahijanian, M. Moll, and L. E. Kavraki, “Asymptotically optimal stochastic motion planning with temporal goals,” in *Workshop on the Algorithmic Foundations of Robotics*, Istanbul, Turkey, Mar. 2014.
- [49] Y. Luo, H. Bai, D. Hsu, and W. S. Lee, “Importance sampling for online planning under uncertainty,” in *Workshop on Algorithmic Foundations of Robotics*, 2016.

- [50] O. Madani, S. Hanks, and A. Condon, “On the undecidability of probabilistic planning and related stochastic optimization problems,” *Artificial Intelligence*, vol. 147, no. 1-2, pp. 5–34, July 2003.
- [51] J. Marecki and P. Varakantham, “Risk-sensitive planning in partially observable environments,” in *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, ser. AAMAS ’10. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2010, pp. 1357–1368.
- [52] M. Mohri, “Finite-state transducers in language and speech processing,” *Computational Linguistics*, vol. 23, no. 2, pp. 269–311, June 1997.
- [53] M. Mundhenk, J. Goldsmith, C. Lusena, and E. Allender, “Complexity of finite-horizon Markov decision process problems,” *Journal of the ACM*, vol. 47, no. 4, pp. 681–720, July 2000.
- [54] S. Nedunuri, S. Prabhu, M. Moll, S. Chaudhuri, and L. E. Kavraki, “SMT-based synthesis of integrated task and motion plans from plan outlines,” in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 655–662.
- [55] S. Nedunuri, Y. Wang, S. Prabhu, M. Moll, S. Chaudhuri, and L. E. Kavraki, “Synthesis of integrated task and motion plans from plan outline using SMT solvers,” Rice University, Tech. Rep., 2014.
- [56] C. Papadimitriou and J. N. Tsitsiklis, “The complexity of Markov decision processes,” *Mathematics of Operations Research*, vol. 12, no. 3, pp. 441–450, 1987.
- [57] A. Paz, *Introduction to Probabilistic Automata*. Orlando, FL, USA: Academic Press, Inc., 1971.

- [58] J. Pineau, G. Gordon, and S. Thrun, “Point-based value iteration: An anytime algorithm for POMDPs,” in *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, ser. IJCAI’03. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003, pp. 1025–1030.
- [59] R. Platt Jr, R. Tedrake, L. Kaelbling, and T. Lozano-Pérez, “Belief space planning assuming maximum likelihood observations,” in *Robotics: Science and Systems*, 2010.
- [60] P. Poupart, K. Kim, and D. Kim, “Closing the gap: Improved bounds on optimal POMDP solutions,” in *Proceedings of the Twenty-First International Conference on International Conference on Automated Planning and Scheduling*, ser. ICAPS’11. AAAI Press, 2011, pp. 194–201.
- [61] P. Poupart, A. Malhotra, P. Pei, K. Kim, B. Goh, and M. Bowling, “Approximate linear programming for constrained partially observable Markov decision processes,” in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, ser. AAAI’15. AAAI Press, 2015, pp. 3342–3348.
- [62] S. Ross, J. Pineau, S. Paquet, and B. Chaib-draa, “Online planning algorithms for POMDPs,” *Journal of Artificial Intelligence Research*, vol. 32, no. 1, pp. 663–704, 2008.
- [63] P. Santana, S. Thiébaux, and B. Williams, “RAO\*: An algorithm for chance-constrained POMDP’s,” in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, ser. AAAI’16. AAAI Press, 2016, pp. 3308–3314.
- [64] K. M. Seiler, H. Kurniawati, and S. P. N. Singh, “An online and approximate solver for POMDPs with continuous action space,” in *ICRA*, 2015, pp. 2290–2297.

- [65] G. Shani, R. I. Brafman, and S. E. Shimony, “Prioritizing point-based POMDP solvers,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 38, no. 6, pp. 1592–1605, Dec 2008.
- [66] G. Shani, J. Pineau, and R. Kaplow, “A survey of point-based POMDP solvers,” *Autonomous Agents and Multi-Agent Systems*, vol. 27, no. 1, pp. 1–51, July 2013.
- [67] R. D. Smallwood and E. J. Sondik, “The optimal control of partially observable Markov processes over a finite horizon,” *Operations Research*, vol. 21, no. 5, pp. 1071–1088, 1973.
- [68] T. Smith and R. Simmons, “Point-based POMDP algorithms: Improved analysis and implementation,” in *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, ser. UAI’05. Arlington, Virginia, United States: AUAI Press, 2005, pp. 542–549.
- [69] A. Somani, N. Ye, D. Hsu, and W. S. Lee, “DESPOT: Online POMDP planning with regularization,” in *Proceedings of the 26th International Conference on Neural Information Processing Systems*, ser. NIPS’13. USA: Curran Associates Inc., 2013, pp. 1772–1780.
- [70] E. J. Sondik, “The optimal control of partially observable Markov processes over the infinite horizon: Discounted costs,” *Oper. Res.*, vol. 26, no. 2, pp. 282–304, Apr. 1978.
- [71] M. T. J. Spaan and N. Vlassis, “A point-based POMDP algorithm for robot planning,” in *2004 IEEE International Conference on Robotics and Automation (ICRA)*, vol. 3, April 2004, pp. 2399–2404 Vol.3.
- [72] —, “Perseus: Randomized point-based value iteration for POMDPs,” *Journal of Artificial Intelligence Research*, vol. 24, no. 1, pp. 195–220, Aug. 2005.

- [73] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel, “Combined task and motion planning through an extensible planner-independent interface layer,” in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 639–646.
- [74] M. Svoreňová, M. Chmelík, K. Leahy, H. F. Eniser, K. Chatterjee, I. Černá, and C. Belta, “Temporal logic motion planning using POMDPs with parity objectives: Case study paper,” in *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, ser. HSCC ’15. New York, NY, USA: ACM, 2015, pp. 233–238.
- [75] A. Undurti and J. P. How, “An online algorithm for constrained POMDPs,” in *2010 IEEE International Conference on Robotics and Automation (ICRA)*, May 2010, pp. 3966–3973.
- [76] L. Valiant, “The complexity of enumeration and reliability problems,” *SIAM Journal on Computing*, vol. 8, no. 3, pp. 410–421, 1979.
- [77] M. Y. Vardi, “Automatic verification of probabilistic concurrent finite state programs,” in *Proceedings of the 26th Annual Symposium on Foundations of Computer Science*, ser. SFCS ’85. Washington, DC, USA: IEEE Computer Society, 1985, pp. 327–338.
- [78] Y. Wang, S. Chaudhuri, and L. E. Kavraki, “Bounded policy synthesis for POMDPs with safe-reachability objectives,” in *AAMAS*, 2018, pp. 238–246.
- [79] —, “Online partial conditional plan synthesis for POMDPs with safe-reachability objectives,” in *WAFR*, 2018, to appear.
- [80] Y. Wang, N. T. Dantam, S. Chaudhuri, and L. E. Kavraki, “Task and motion policy synthesis as liveness games,” in *Proceedings of the Twenty-Sixth International*



*Conference on International Conference on Automated Planning and Scheduling*, ser. ICAPS'16. AAAI Press, 2016, pp. 536–540.

- [81] D. White, “Multi-objective infinite-horizon discounted Markov decision processes,” *Journal of Mathematical Analysis and Applications*, vol. 89, no. 2, pp. 639–647, 1982.
- [82] E. M. Wolff, U. Topcu, and R. M. Murray, “Robust control of uncertain markov decision processes with temporal logic specifications,” in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, Dec 2012, pp. 3372–3379.