Counterfactual MDPs: Planning Beyond Direct Control

Rui Silva

CMU-CS-20-122
August 2020

Computer Science Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Universidade de Lisboa
Instituto Superior Técnico
2744–016, Porto Salvo, Portugal

Thesis Committee:
Manuela Veloso (Co-Chair, CMU)
Francisco S. Melo (Co-Chair, Instituto Superior Técnico)
Ariel Procaccia
Reid Simmons
Daniel Borrajo (Universidad Carlos III de Madrid)
Pedro Lima (Instituto Superior Técnico)

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This research was sponsored by "Fundação para a Ciência e a Tecnologia" (Portuguese Foundation for Science and Technology) through project UIDB/50021/2020 (INESC-ID multi annual funding), and through the Carnegie Mellon Portugal Program and its Information and Communication Technology Institute, under project CMUP-ERI/HCI/0051/2013 and grant SFRH/BD/113695/2015. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Keywords: Counterfactual Markov Decision Processes, Planning Under Uncertainty

To those who, knowingly or unknowingly, made it all possible.

Abstract

Planning under uncertainty using Markov decision processes (MDPs) requires a model of the environment specifying the probabilistic effects of the actions that the agent is able to execute. The optimal policy of the agent is then computed from such model. As such, when planning, the agent assumes the environment may only change as the direct result of its actions. In this thesis we consider lifting that assumption, and allow the agent to reason over the counterfactual *"What if the world were different?"* Effectively, we allow the agent to reason over other possible configurations of the world, where more rewarding optimal policies may exist, and over the cost required in shifting the original environment to these modified worlds. Our goal is to endow the agent with the ability to plan over the possibility of actually operating in such configurations of the world, if beneficial. We introduce Counterfactual MDPs, a new class of MDPs that allows the agent to reason and plan over the aforementioned counterfactual. Solving a Counterfactual MDP consists in the maximization of the expected value/cost trade-off over possible changes to the world.

In the context of MDPs, the dynamics of the world are described in terms of transition probabilities. Our approach is thus to formulate the problem as a joint-optimization of the transition probabilities and optimal policy of the MDP. We analyze the complexity of the resulting problem, and formally prove it is NP-Hard. We then derive two gradient-based approaches for solving the problem. These approaches culminate in the contribution of an iterative gradient based algorithm, P-ITERATION, for solving Counterfactual MDPs. Additionally, we discuss methods for scaling up this algorithm to larger problems.

We demonstrate the applicability of Counterfactual MDPs and the performance of the algorithms proposed in multiple scenarios. We show, in particular, significant performance improvements that arise from allowing the agent to reason and plan over other possible worlds, and corresponding optimal policies.

In the process we realize, however, that Counterfactual MDPs implicitly assume that the specific world configuration the agent envisioned will be necessarily materialized. However, in many real-life scenarios there exists an underlying uncertainty in the outcome of applying changes to the world. We extend the Counterfactual MDP model to allow the agent to reason over this uncertainty, and dub the resulting model Stochastic Outcomes Counterfactual MDPs. This new model assumes the uncertainty associated with changes to the world follows a probabilistic distribution with parameters the agent can reason over and control, resulting in a new optimization problem. We show the gradient of this new optimization problem can be computed by solving an expectation, and thus propose a sampling based method for computing it. This allows us to extend P-ITERATION to this new class of problems with stochastic outcomes.

In the end we demonstrate the applicability of the new model in multiple scenarios with uncertainty in the outcome of changes to the world. We show that by reasoning over this uncertainty the agent is able to find more valuable world configurations.

Resumo

O planeamento sob incerteza com processos de decisão de Markov (MDPs) requer um modelo do ambiente que especifica os efeitos probabilísticos das acções que o agente pode executar. A política óptima do agente é depois calculada a partir deste modelo. Como tal, quando planeia, o agente assume que o ambiente apenas pode mudar como o resultado das suas acções. Nesta tese exploramos a ideia de levantar esta assumpção, e permitir que o agente raciocine acerca do contrafactual *"E se o mundo fosse diferente?"* Efectivamente, nós permitimos o agente raciocinar acerca de outras configurações possíveis do mundo—onde pode ser possível encontrar políticas óptimas mais valiosas—e acerca do custo necessário em mudar o mundo original para estes novos mundos. O nosso objectivo é dar ao agente a capacidade de planear com a possibilidade de operar nestas configurações do mundo alternativas, quando benéfico. Como tal, introduzimos os Counterfactual MDPs, uma nova classe de MDPs que permite o agente raciocinar acerca do contrafactual acima. Resolver um Counterfactual MDP consiste na maximização do compromisso entre o valor esperado e o custo das mudanças possíveis no mundo.

No contexto de MDPs, a dinâmica do mundo é descrita em termos de probabilidades de transição. A nossa abordagem consiste em formular o problema como uma optimização conjunta das probabilidades de transição e da política óptima do MDP. Analisamos a complexidade do problema e provamos formalmente que este é NP-Hard. De seguida derivamos duas abordagens de gradiente para resolver o problema. Estas abordagens culminam com a contribuição de um algoritmo de gradiente iterativo chamado P-ITERATION, para resolver Counterfactual MDPs. Adicionalmente, discutimos métodos para acelerar o algoritmo.

Demonstramos a aplicabilidade dos Counterfactual MDPs e a performance dos algoritmos propostos em múltiplos cenários. Mostramos, especificamente, as melhorias de performance significativas que surgem ao permitir o agente raciocinar, e planear, acerca de outros mundos e respectivas políticas óptimas.

Neste processo apercebemo-nos, no entanto, que os Counterfactual MDPs implicitamente assumem que a configuração do mundo idealizada pelo agente será necessariamente materializada. No entanto, em muitos cenários reais existe uma incerteza subjacente ao resultado de se aplicarem mudanças ao mundo. Estendemos o modelo de Counterfactual MDPs para permitir o agente raciocinar acerca desta incerteza, e apelidamos o modelo resultante de Stochastic Outcomes Counterfactual MDPs. Este novo modelo assume que a incerteza associada a mudanças ao mundo segue uma distribuição probabilística com parâmetros que o agente pode controlar, resultando num novo problema de optimização. Mostramos que o gradiente deste novo problema de optimização pode ser calculado resolvendo um valor esperado. Como tal, propomos utilizar um método de amostragem para o calcular, o que nos permite estender o algoritmo P-ITERATION para esta nova classe de problemas.

No final, mostramos a aplicabilidade do novo modelo em vários cenários com incerteza nas mudanças no mundo. Especificamente, mostramos os benefícios que advêm de permitir o agente raciocinar acerca desta incerteza, o que permite o agente descobrir outras configurações possíveis do mundo mais valiosas.

Acknowledgments

After five years into this journey it is now time to look back and acknowledge those who made it all possible. In hindsight, it is incredible to realize how many people greatly impacted my life throughout these years. I can only hope to do them some justice here.

First and foremost, I must thank my parents. It was their unwavering support that kept me going forward. They were always present, even when an entire ocean stood between us. No Skype call was too late, no Skype call was too early. They were always available. Whether to celebrate my achievements, or to support me in the hardest moments, they were there. For that, I cannot thank them enough.

This thesis would also not be possible without my two advisors, Francisco Melo and Manuela Veloso. I am grateful for their support and guidance over the years. This thesis is deeply shaped by the countless meetings, discussions and brainstorming sessions we shared. Additionally, their mentoring often extended beyond the realm of the PhD. The valuable lessons I learned from their experience will forever shape my personal and professional life.

I would also like to thank the other members of my thesis committee, Ariel Procaccia, Reid Simmons, Pedro Lima and Daniel Borrajo for their meaningful feedback and the insightful questions raised during my thesis proposal and defense. I am also grateful to the CMU Portugal Program staff, specially for their help in dealing with the bureaucracy related with transitioning between Lisbon and Pittsburgh.

My doctoral journey gave me the opportunity to work in two different groups, in two different countries. This allowed me to meet and work alongside incredible people. My journey started, and finished, in Portugal. There, I was lucky to share an office with Hang Yin, Filipa Correia, Diogo Carvalho, and Miguel Vasco. In different ways, and at different times, all four of them had an important role in my PhD and a positive impact in my life. They were a constant source of inspiration and motivation to work harder, but always while having fun. I also want to acknowledge João Pires Ribeiro, who was always available to discuss any problems or ideas over a cup of coffee, or during a walk around the IST building.

Pittsburgh was my home during the second and third years of my PhD. There, I was very fortunate to work in the CORAL group. I have very fond memories of the weekly group meetings and of the time spent in the lab. I would like to acknowledge the people I spent the most time with, both inside and outside the lab: Tiago Pereira, Vittorio Perera, Anahita Mohseni-Kabir, Kim Baraka, Devin Schwab, Philip Cooksey, Ashwin Khadke, and Arpit Agarwal. My life in Pittsburgh was also positively impacted by many other people. Gabriele Farina, who showed me a different way of thinking about optimization, and math in general, during long nightly walks around Pittsburgh. Luis Oliveira, Sheiliza Carmali and Tiago Pereira, my fellow Portuguese friends, who I consistently met for dinner every Friday. Finally, Pedro Paredes, who was always available to chat, and against whom I lost countless ping pong games.

Thank you all.

Contents

List of Figures

List of Tables

Chapter 1

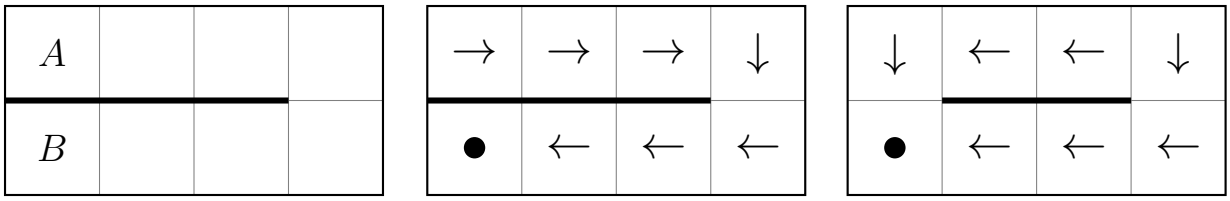
Introduction

Planning under uncertainty requires a model of the *dynamics* of the world, specifying the probabilistic effects of the actions that the agent is able to execute. Given this model, and a reward function describing the target task, the planner computes the optimal policy of the agent. This optimal policy specifies the actions to take at each state in order to maximize the reward collected in a world governed by the dynamics model provided. Models for planning under uncertainty have proved to be successful in a wide variety of tasks. In robotics, for example, these models have been shown to be particularly effective in allowing robots to plan over the uncertainty underlying their execution [52, 66, 73].

Despite showing a wide applicability, the existing approaches for planning under uncertainty make the implicit assumption that the dynamics of the world are fixed, and consequently, only allow the agent to reason over changes to the world that result directly from the execution of its actions. In this thesis we propose to lift this assumption, and allow agents to reason, plan, and act, over the counterfactual *"What if the world were different?"* That is, our goal is to allow agents to reason and plan over alternative configurations of the world where more rewarding optimal policies may be possible. As a motivating example, consider the following scenario.

> **Scenario** (CORRIDOR) *Consider the environment depicted in Figure 1.1(a). A mobile service robot docked at its charging station (located in A) receives a request from a user to pick up a package from the user's office (located in B). The robot operates in a 2×4 corridor located in an office setting, and is able to move in four directions (*UP*,* DOWN*,* LEFT*,* RIGHT*) or stay in place (*STAY*). Each move action moves the robot deterministically to an adjacent cell, factoring in obstacles. The robot is rewarded for navigating from A to B in the most efficient way.*

Given the layout of the corridor (Figure 1.1(a)), the most efficient plan is to traverse the entire corridor, by first travelling from A all the way to the right, move down, and then travel left all the way to B, as depicted in Figure 1.1(b). Assuming this scenario, the goal of this thesis is to endow the agent with the ability to reason over questions such as *"What if it was possible to have a direct passage between the top and bottom rows?"* Assuming that possibility, *"What would be the best location to have such passage?"* Having the passage in the best possible location, *"Would the benefit of being able to move across the passage surpass the cost of creating it?"* In



(a) (b) (c)

Figure 1.1: Robot departs from a docking station at A to attend a user request at B. (a) The original world configuration. (b) The optimal policy in the original world configuration, which requires the robot to traverse the entire corridor. (c) The optimal policy in the world configuration where the first door is open.

the particular case of the CORRIDOR scenario, this passage could correspond to a door separating the two sides of the corridor. As depicted in Figure 1.1(c), if there existed a door between states A and B that were to be opened, the agent would be able to follow a more rewarding policy. Even though the agent is not able to directly open such door, we still allow it to reason over the world configuration where the door is in fact open.

This scenario illustrates the class of problems that motivates this thesis. We advocate allowing agents to explicitly reason over other possible world configurations during their planning processes. In particular, we focus on configurations of the world that cannot be reached simply by the execution of the agent's actions, requiring instead some form of external intervention, possibly at a cost. In this setting, the goal of the agent becomes that of jointly optimizing its policy and the world configuration to operate at, by taking into account the value/cost trade-off over the changes to the environment.

In sum, this thesis proposes a different planning paradigm, where the agent is allowed to explicitly reason over alternative configurations of the world, and to plan over the possibility of actually operating in such configurations when beneficial. In other words, we endow agents with the ability to reason, plan, and act, over the counterfactual *"What if the world were different?"*, in order to allow the discovery of new, and possibly more rewarding, optimal policies.

1.1 Thesis Question

The discussion and motivating example introduced previously can be summarized in the following research question.

> **Research Question** *How can we endow agents with the ability to reason over alternative configurations of the world, and to plan over the possibility of operating in such configurations, in order to find more rewarding policies?*

Typical approaches for planning under uncertainty make the implicit assumption that the world only changes as the direct result of the actions of the agent. In this thesis, we propose to lift such assumption, and allow agents to reason, plan, and act over the counterfactual *"What if the world were different?"* We focus, in particular, in endowing agents with the ability to reason

over configurations of the world that are reachable through some form of external intervention, possibly at a cost. Allowing the agent to reason over these additional configurations of the world, and to plan over the possibility of operating there, may allow the discovery of new optimal policies that would not be possible in the original environment. The execution of these new optimal policies in the alternative worlds may actually allow the agent to collect more reward, improving its overall performance.

1.2 Thesis Approach

The approach followed in this thesis focuses on planning problems modeled as Markov decision processes (MDPs). In this model, the dynamics of the world are described in terms of a fixed probability transition function, which implicitly assumes that the world may only change as the direct result of the execution of the actions of the agent. This thesis considers lifting that restriction, allowing the agent to reason over additional configurations of the world—where more rewarding optimal policies may exist—and to plan over the possibility of operating in such configurations. In the context of problems described as MDPs, we allow the agent to reason over the impact of changes to the transition probabilities that govern the dynamics of the world. Our goal is to endow the agent with the ability to reason, and act, over the counterfactual *"What if the world were different?"*. Motivated by this goal, we contribute a new class of MDPs that we dubbed *Counterfactual* Markov decision processes.

Our approach starts with the formulation of Counterfactual MDPs as an optimization problem that maximizes the expected value/cost trade-off over possible changes to the world. The value of a given change to the world is measured as the expected discounted reward the agent is able to collect in the resulting world configuration. We assume, however, that changes to the world may come at a cost. This cost is task-specific, and, in general, will be a function of the effort involved in shifting the original world configuration to the new configuration. Additionally, our formulation assumes the aforementioned optimization problem is constrained to a set of *feasible* configurations of the world. This is useful in preventing the agent from reasoning over unrealistic world configurations.

We then extend the Counterfactual MDP model to account for the uncertainty associated with changing the world. We realize that the original model assumes any world configuration envisioned by the agent can be necessarily materialized. However, in real-life scenarios there exists an underlying uncertainty in the outcome of changing the world. As such, the actual resulting configuration may not yield the expected one. We dub the resulting model *Stochastic Outcomes Counterfactual* MDPs. This model is particularly interesting when it comes to modeling, for example, human-robot interaction scenarios. In practice, the reasoning over alternative configurations of the world can be interpreted as the agent planning over assistance requests to be posed to a nearby human. In general, the outcome of these assistance requests will be stochastic, since the human response may vary.

1.3 Contributions

Following the overall approach described in Section 1.2, we now go over the individual contributions of the thesis in greater detail.

1.3.1 Counterfactual MDPs

The first contribution of this thesis is the formulation of Counterfactual MDPs—a novel class of MDPs that allows the agent to plan over alternative configurations of the world, with the goal of allowing the discovery of more rewarding optimal policies. We formalize Counterfactual MDPs as an optimization problem that maximizes a benefit/cost trade-off over possible changes to the environment, in the framework of MDPs. In this setting, our formulation corresponds to a joint optimization over the transition probabilities and optimal policy of the MDP. Along with the formulation of the model, we contribute a complexity analysis of the problem. We show that, in general, solving Counterfactual MDPs is hard both in theory and in practice. In fact, we formally prove that the problem is NP-Hard. This thesis also makes important practical contributions for allowing the applicability of Counterfactual MDPs to different planning problems. We demonstrate how to model multiple scenarios, under different cost functions and sets of possible configurations of the world. In the process, we show that Counterfactual MDPs lead the agent to significant performance improvements in all tasks.

1.3.2 Algorithmic Approaches for Solving Counterfactual MDPs

This thesis contributes algorithms for solving Counterfactual MDPs. We propose a gradient-based approach for solving the optimization problem that characterizes our model, and formally derive two methods for computing the gradient. The first method derives the gradients directly from the Karush-Kuhn-Tucker conditions of the optimization problem. The second method exploits the linear programming structure of MDPs, leading to a more efficient approach. The approaches proposed for computing the gradient culminate in the P-ITERATION algorithm—an iterative gradient ascent algorithm that iteratively builds a sequence of world configurations that converge to a local maximum of the optimization problem. The non-convexity is tackled by performing random restarts starting from different possible initial world configurations.

This thesis also discusses mechanisms for scaling the methods proposed so they can be applied in larger problems. One of the approaches discussed is to adopt a factored representation of MDPs that is typically used for solving large state and action spaces scenarios. This representation compresses the transition dynamics and reward model of the MDP by factoring similar states, allowing for speed-ups in solving MDPs. We further discuss other approaches that exploit the iterative nature of P-ITERATION, and allow the reuse of computations performed in previous steps. Our experimental evaluation shows that these approaches can improve the overall performance of the P-ITERATION algorithm.

1.3.3 Stochastic Outcomes Counterfactual MDPs

The final contribution of this thesis regards the extension of the Counterfactual MDPs model to scenarios where there is uncertainty in the outcome of changes to the world. We dub the resulting

model Stochastic Outcomes Counterfactual MDPs, and our approach is to model the underlying uncertainty as following a probabilistic distribution with parameters that the agent can control. We show that the original model, Counterfactual MDPs, can be reduced to Stochastic Outcomes Counterfactual MDPs and, as such, the complexity analysis on the former extends to the latter. This allows us to conclude that, in general, Stochastic Outcomes Counterfactual MDPs are also hard to solve.

Following our approach in the original model, we also adopt a gradient-based method for solving Stochastic Outcomes Counterfactual MDPs. The new formulation that accounts for the stochastic outcomes in changes to the world, however, introduces some challenges in the computation of the gradient. We show that this gradient can actually be computed by solving an expectation, and propose to use sampling methods for estimating it. This allowed us to extend the P-ITERATION algorithm to the new class of problems with stochastic outcomes.

Finally, we demonstrate the applicability of this model in different scenarios, showing the improvements on the performance of the agent that arise when considering the uncertainty in the outcomes of changes in the world.

1.4 Reading Guide to the Thesis

The remainder of the thesis is structured as follows.

Chapter 2 reviews relevant background on the Markov decision processes framework, covering both the formulation of the model as well as methods for solving it.

Chapter 3 introduces the Counterfactual MDP model, which is formalized as an optimization problem. The model is introduced in iterative fashion, with each version allowing Counterfactual MDPs to model more complex problems. The chapter also contributes a complexity analysis of the problem, showing that solving Counterfactual MDPs is hard both in theory and in practice. Specifically, we formally establish that the problem is NP-Hard. Additionally, we discuss relevant practical considerations on the applicability of the model, namely, by introducing different mechanisms for parameterizing the transition probabilities. The chapter concludes with a discussion on alternative formulations to Counterfactual MDPs, with an application to a real-world human-robot interaction scenario.

Chapter 4 introduces P-ITERATION, a gradient-based algorithm for solving Counterfactual MDPs. The chapter starts with a formal derivation of two methods for computing the gradients of the optimization problem that characterizes Counterfactual MDPs. These methods culminate in the introduction of the P-ITERATION algorithm. The chapter also contributes techniques for scaling up the algorithm to larger problems, discussing other practical considerations on performance. The chapter concludes with an evaluation of the models and algorithms proposed. We show the applicability of Counterfactual MDPs in multiple planning scenarios, and the performance of the P-ITERATION therein.

Chapter 5 tackles the problem of reasoning over uncertain outcomes in changes to the world, in the context of Counterfactual MDPs. We formally derive a new model, dubbed Stochastic Outcomes Counterfactual MDPs, which allows the agent to reason, plan and act over the aforementioned stochastic outcomes. We show how to extend the previous gradient-based approaches to the new model. The chapter concludes by evaluating the applicability and performance of the new models and algorithms proposed.

Chapter 6 reviews literature relevant to the ideas followed in this thesis. This chapter focuses, in particular, on two lines of research. First, we review MDP models that allow some form of reasoning over the impact of changes to the transition probabilities of an MDP. Then, we review research on agents that request assistance to external entities at different steps of the agent loop—perception, decision, planning and execution.

Chapter 7 concludes the thesis by summarizing the main contributions, and discussing possible directions of future research.

Chapter 2

Background

The work in this thesis considers problems of planning under uncertainty modeled using *Markov Decision Processes*. This chapter provides the necessary background on this framework.

2.1 Markov Decision Processes

A Markov decision process (MDP) is a tuple $(\mathcal{X}, \mathcal{A}, P, r, \gamma)$ describing a sequential decision problem under uncertainty. The state space $\mathcal{X}$ denotes the set of possible states of the world, whereas the action space $\mathcal{A}$ is the set of actions available to the agent. When the agent takes an action $a \in \mathcal{A}$ while in state $x \in \mathcal{X}$, the world transitions to state $y \in \mathcal{X}$ with probability $P(y \mid x, a)$ and the agent receives a reward $r(x, a)$. The discount factor $\gamma \in [0, 1)$ determines the relative importance of present and future rewards. This decision process is depicted in Figure 2.1.

In a way, the reward function r can be seen as encoding the objectives the agent wants to achieve, whereas the transition probabilities P describe the dynamics of the world where the agent operates. In this setting, the goal of the agent is to identify a *policy*—a mapping $\pi : \mathcal{X} \times \mathcal{A} \to [0, 1]$ such that $\pi(a \mid x)$ is the probability of selecting action $a \in \mathcal{A}$ when in state $x \in \mathcal{X}$—that chooses the actions the agent should follow in order to achieve the task encoded by r, in the world described by P. Formally, the goal of the agent is to compute the policy π that maximizes the expected discounted reward at each state $x \in \mathcal{X}$:

$$v^\pi(x) \triangleq \mathbb{E}_{a_t \sim \pi(x_t)} \left[\sum_{t=0}^{\infty} \gamma^t r(x_t, a_t) \mid x_0 = x \right].$$

The vector $\boldsymbol{v}^\pi$ is called the *value function* associated with policy π and can be computed as the solution to the linear system

$$\boldsymbol{v}^\pi = \boldsymbol{r}^\pi + \gamma \mathbf{P}^\pi \boldsymbol{v}^\pi, \tag{2.1}$$

where $\boldsymbol{r}^\pi$ is a column vector with x-th entry

$$r^\pi(x) = \sum_{a \in \mathcal{A}} \pi(a \mid x) r(x, a),$$

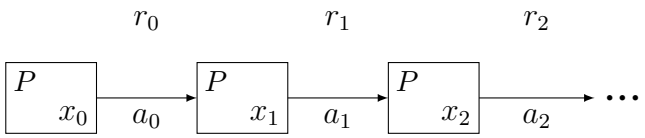


Figure 2.1: The decision process modeled by an MDP. The boxes denote the state (bottom right) and environment (top left) at each timestep. The decision process evolves as the agent takes actions. Specifically, when the agent is in state x_t, and executes action a_t, it receives a reward r_t and the state of the world transitions to state x_{t+1}. This transition is governed by the transition probabilities P.

and $\mathbf{P}^\pi$ is a matrix with element (x, y)

$$P^\pi(y \mid x) = \sum_{a \in \mathcal{A}} \pi(a \mid x) P(y \mid x, a).$$

In particular, we have that

$$\boldsymbol{v}^\pi = (\mathbf{I} - \gamma \mathbf{P}^\pi)^{-1} \boldsymbol{r}^\pi, \tag{2.2}$$

where $\mathbf{I}$ is the $|\mathcal{X}| \times |\mathcal{X}|$ identity matrix. The solution to (2.2) is well-defined, since matrix $(\mathbf{I} - \gamma \mathbf{P}^\pi)$ is non-singular [53]. Solving an MDP thus consists in computing an optimal policy π^* such that, for all $x \in \mathcal{X}$ and all policies π,

$$v^{\pi^*}(x) \geq v^\pi(x).$$

In general, all MDPs have at least one deterministic optimal policy. We introduced the more general case of stochastic optimal policies as it will be useful in later chapters.

2.2 Solving Markov Decision Processes

As discussed in the previous section, solving an MDP consists in computing an optimal policy π^*, *i.e.*, a mapping from states to actions which ensures that the robot, if selecting the actions as prescribed by π^*, collects as much reward as possible. There exist several methods for computing the optimal policy and the optimal value function, including dynamic programming, linear programming, stochastic approximation, among other approaches [53]. We describe two such methods, which will be useful in later chapters of this thesis.

Linear Programming The linear programming approach for solving MDPs computes the optimal value function by solving the following optimization problem

$$\begin{aligned} \min_{\boldsymbol{v}} \quad & \mathbf{1}^\top \boldsymbol{v} \\ \text{s.t.} \quad & v(x) \geq r(x, a) + \gamma \sum_{y \in \mathcal{X}} P(y \mid x, a) v(y), \quad \forall x \in \mathcal{X}, a \in \mathcal{A}, \end{aligned} \tag{2.3}$$

where $\mathbf{1}$ is a $|\mathcal{X}|$-dimensional vector of ones. The optimal policy π^* can then be retrieved from the optimal value function $\boldsymbol{v}^*$ computed. This is performed by checking for each state x the actions that maximize the expected value, *i.e.*,

$$\pi^*(x) \in \operatorname*{argmax}_{a \in \mathcal{A}} \left\{ r(x,a) + \gamma \sum_{y \in \mathcal{X}} P(y \mid x, a) v^*(y) \right\}. \tag{2.4}$$

Dynamic Programming MDPs can also be solved using a dynamic programming approach. One of the most popular algorithms in this class of approaches is *value iteration*, which works by iteratively refining an estimate of the optimal value function $\boldsymbol{v}^*$. Starting from an estimate $\boldsymbol{v}^0$, the method iteratively refines its estimate by computing

$$v^{t+1}(x) = \max_{a \in \mathcal{A}} \left\{ r(x,a) + \gamma \sum_{y \in \mathcal{X}} P(y \mid x, a) v^t(y) \right\}.$$

It is well-known in the literature that the estimates of value iteration converge to the optimal value function, *i.e.,* $\boldsymbol{v}^t \to \boldsymbol{v}^*$, as $t \to \infty$. The proof starts by showing that the value iteration updates consist of the Bellman backup operator $\mathrm{T} : \mathbb{R}^{|\mathcal{X}|} \to \mathbb{R}^{|\mathcal{X}|}$, which is a contraction [53]. The Bellman backup operator T is defined as,

$$\left(\mathrm{T}\boldsymbol{v}^t\right)(x) = \max_{a \in \mathcal{A}} \left\{ r(x,a) + \gamma \sum_{y \in \mathcal{X}} P(y \mid x, a) v^t(y) \right\}, \tag{2.5}$$

and we can prove it is a contraction by showing that for any estimates of the optimal value function $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ we have

$$\max_{x \in \mathcal{X}} \left| \left(\mathrm{T}\boldsymbol{v}_1\right)(x) - \left(\mathrm{T}\boldsymbol{v}_2\right)(x) \right| \leq \gamma \max_{x \in \mathcal{X}} \left| v_1(x) - v_2(x) \right|.$$

Since T is a contraction, by the Banach fixed-point theorem we conclude T must admit a unique fixed point solution $\boldsymbol{v}^* = \mathrm{T}\boldsymbol{v}^*$. Joining these two results together, we conclude

$$\max_{x \in \mathcal{X}} \left| \left(\mathrm{T}\boldsymbol{v}^t\right)(x) - v^*(x) \right| \leq \gamma \max_{x \in \mathcal{X}} \left| v^t(x) - v^*(x) \right|.$$

Finally, expanding the second term we observe

$$\max_{x \in \mathcal{X}} \left| v^{t+1}(x) - v^*(x) \right| \leq \gamma^{t+1} \max_{x \in \mathcal{X}} \left| v^0(x) - v^*(x) \right|. \tag{2.6}$$

Since $0 \leq \gamma < 1$, we have that $\gamma^t \to 0$ as $t \to \infty$. Consequently, we must have $\boldsymbol{v}^t \to \boldsymbol{v}^*$ as $t \to \infty$.

Chapter 3

Counterfactual MDPs

Planning under uncertainty typically assumes a fixed model of the world that specifies the probabilistic effects of the actions of the agent in terms of changes in the state of the world. Based on this model of the world, the planner proceeds by computing a policy the agent should follow in order to maximize the reward collected, according to some reward function. However, as discussed in Chapter 1, the world may be changed in more ways than those resulting from the direct execution of the actions of the agent. In fact, there may exist additional configurations of the world that allow for new policies that let the agent collect even more reward.

This thesis focuses, in particular, in planning problems modeled as Markov decision processes, where the dynamics of the world are described in terms of a fixed probability transition function. This thesis considers lifting that assumption of "fixed dynamics of the world", extending the Markov decision processes model to allow the agent to reason over additional configurations of the world, and to plan over the possibility of actually operating in such configurations. In the context of problems described as MDPs, our approach is to allow the agent to reason over the impact of changes to the transition probabilities that govern the dynamics of the world. In a way, our goal is to allow the agent to reason over the counterfactual *"What if the world were different?"* Consequently, we dubbed the resulting model as *Counterfactual* Markov decision processes.

In this chapter we introduce Counterfactual Markov decision processes, and we do so in an iterative fashion. We start with a simple version of the model, which despite its limited expressiveness, provides us the opportunity to discuss some important insights. These insights seed the formulation of the more expressive, but also complex, models. We then offer a complexity analysis on the model proposed, both from a practical perspective, and from a formal, computational point of view. The chapter concludes with a discussion on alternative formulations of our Counterfactual MDPs model.

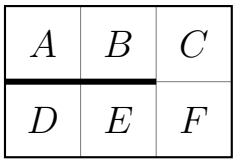
3.1 Model Formulation

Consider an agent facing a sequential decision problem described as an MDP $(\mathcal{X}, \mathcal{A}, P, r, \gamma)$, where the initial state, x_0, follows some known distribution μ_0. In the MDP model, the reward function r describes the *goal* of the agent, whereas the transition probabilities P govern the *dynamics* of the world. In this thesis, we are interested in investigating how different world configurations may impact the decision problem faced by the agent. In practice, different world configurations mostly impact the way the agent interacts with the world—*i.e.*, the outcome of the agent's actions. In the MDP model, the outcome of the agent's actions is described by the transition probabilities, P. As such, we describe different world configurations in terms of different transition probabilities. Formally, we write P_θ to denote the transition probabilities associated with the world configuration θ. Each configuration θ induces a different MDP $(\mathcal{X}, \mathcal{A}, P_\theta, r, \gamma)$, and we denote the corresponding optimal policy by π^*_θ.

For ease of understanding, throughout this chapter we will be considering a specific scenario. Namely, a 2×3 instance of the CORRIDOR scenario introduced in Chapter 1:

> **Scenario** (CORRIDOR) *Consider once again the problem faced by a mobile robot that must navigate from initial location A to goal location D in the grid world depicted in Fig. 3.1(a), where the obstacles between states A and D, and B and E, prevent the movement of the robot. This problem can be described as an MDP $(\mathcal{X}, \mathcal{A}, P_0, r, \gamma)$, where $\mathcal{X} = \{A, B, C, D, E, F\}$ and $\mathcal{A} = \{\text{UP}, \text{DOWN}, \text{LEFT}, \text{RIGHT}, \text{STAY}\}$. Each moving action moves the agent deterministically to the adjacent cell in the corresponding direction, except if an obstacle is found. Action* STAY *makes the agent stay in the same cell, and not move. The reward is -1 for all state-action pairs except (D, STAY), where it is 0. The initial distribution μ_0, in this case, is given by $\mu_0(x) = \mathbb{I}(x = A)$, where $\mathbb{I}(U)$ denotes the indicator for U, taking value 1 when U holds and 0 otherwise. Finally, we take a discount factor $\gamma = 0.9$.*

In this scenario, the original world configuration θ_0 corresponds to the world where the robot moves deterministically to adjacent cells, and where the movement between states A and D, and



(a)

	A	B	C	D	E	F
A	1					
B		1				
C						1
D				1		
E					1	
F						1

$P_{\theta_0}(y \mid x, \text{DOWN})$

(b)

	A	B	C	D	E	F
A				1		
B		1				
C						1
D				1		
E					1	
F						1

$P_{\theta_1}(y \mid x, \text{DOWN})$

(c)

	$v^*_{\theta_0}$	$v^*_{\theta_1}$
A	-4.1	-1
B	-3.4	-1.9
C	-2.7	-2.7
D	0	0
E	-1	-1
F	-1.9	-1.9

(d)

Figure 3.1: (a) The layout of the 2×3 corridor scenario. (b) and (c) The transition probabilities for action DOWN associated with the world configurations where the first obstacle exists/does not exist, respectively. (d) The optimal value functions for both worlds.

B and E, is prevented by the obstacles. Figure 3.1(b) depicts the transition probabilities associated with action DOWN of the original world configuration θ_0. On the other hand, Figure 3.1(c) depicts the same transition probabilities associated with a world configuration θ_1 where the first obstacle, between A and D, does not exist. Note that the difference between these two transition probabilities matrices regards the transitions starting from state A. In the former, when moving DOWN while in state A, the agent remains in the same state. However, in the latter, moving DOWN from state A immediately transitions the agent to the goal state D.

In Chapter 1, when the corridor scenario was introduced, we briefly discussed these exact same world configurations. At the time we concluded that, intuitively, world configuration θ_1 is "better", as it allows for a more efficient optimal policy. While the optimal policy in the original world configuration θ_0 lets the agent reach the goal state D in 5 steps, in world configuration θ_1 it takes the agent a single step. These optimal policies lead to the optimal value functions depicted in Figure 3.1(d). Given the reward function used, these values roughly correspond to the the (discounted) number of steps it takes to reach the goal position D, starting at each different state. We observe that the expected value of the states at the bottom row of the corridor—D, E and F—are the same in both configurations of the world. This is because following the optimal policy when starting at these states never takes the agent to the upper row of the corridor. As such, the presence of the obstacles is irrelevant. The expected value in state C is also the same in both worlds—in the optimal case, the agent always takes 3 steps to reach the goal state D. We observe, however, that the expected values in states A and B are clearly better in world configuration θ_1.

To allow the agent to reason about the benefits of different world configurations and corresponding optimal policies, we let $J(\theta)$ denote the *value* associated with world configuration θ

$$J(\theta) = \sum_{x \in \mathcal{X}} \mu_0(x) v^*_\theta(x), \tag{3.1}$$

where $v^*_\theta(x)$ denotes the value of the optimal policy π^*_θ in state $x \in \mathcal{X}$. In our 2×3 CORRIDOR scenario, for example, we observe the value of the world configurations θ_0 and θ_1 is computed as

$$J(\theta_0) = v^*_{\theta_0}(A) \approx -4.1,$$
$$J(\theta_1) = v^*_{\theta_1}(A) \approx -1.0,$$

by definition of the initial state distribution used and the optimal value functions in Figure 3.1(d). These values match our intuition that world configuration θ_1 is "better".

Being able to reason over the value of the optimal policy of any given world configuration allows us to define the simplest form of our model.

3.1.1 Unconstrained Counterfactual MDPs

In its simplest form, Counterfactual MDPs allow the agent to reason over the best possible world under no restrictions nor costs. In practice, this model allows the agent to reason over the counterfactual *"What is the best world I could possibly operate at?"*

The formulation of this counterfactual as the solution to an optimization problem follows naturally as

$$\max_\theta \; J(\theta), \tag{3.2}$$

where the goal of the agent is to find the world configuration θ, associated with the transition probabilities P_θ, that maximizes the value $J(\theta)$.

One may now wonder what is the solution of this Counterfactual MDP when applied to the aforementioned 2×3 CORRIDOR scenario. Intuitively, the world configuration θ_1 where the obstacle does not exist is an optimal solution. In this world configuration the agent can follow the optimal policy $\pi^*_{\theta_1}$ where in states A and D

$$\pi^*_{\theta_1}(A) = \text{DOWN}, \quad \pi^*_{\theta_1}(D) = \text{STAY}.$$

Following this policy, upon starting in state A, the agent can immediately execute action DOWN and reach the goal position D (at the expense of -1 reward), and afterwards the agent can execute action STAY (and collect reward 0). As a result, the value in state A is $v^*_{\theta_1}(A) = -1$. Furthermore, by our definition of the initial state distribution μ_0, we have that the world configuration θ_1 has value $J(\theta_1) = v^*_{\theta_1}(A) = -1$.

It turns out that, while this solution is optimal, it is not unique. In fact, there are infinite optimal solutions, comprising all world configuration with transition probabilities such that *the execution of some action transitions the agent from state A to state D*. Formally,

$$\theta^* \in \{\theta : \exists\, a \in \mathcal{A} : P_\theta(D \mid A, a) = 1\}\,.$$

As a result, we have, for instance, that the *"teleporting"* world configuration θ_{T} where the agent transitions between states A and D by executing action STAY, is an optimal solution.

One may argue this solution has an artificial flavour. We tackle this problem by constraining Counterfactual MDPs.

3.1.2 Constraining Counterfactual MDPs

We concluded the previous section with the observation that formulating Counterfactual MDPs as an unconstrained optimization problem may yield *artificial* optimal solutions in the sense of not being really feasible from a physical point of view. We discussed, in particular, a *"teleporting"* solution θ_{T} to the CORRIDOR scenario, where the agent moves between states by executing the STAY action, which is supposed to keep the agent in the same place.

We now extend our model to allow the specification of a set of possible/feasible world configurations Θ. Note that set Θ need not be countable, as the set of possible configurations may be continuous. The specification of this space of possible world configurations allows us to formalize Counterfactual MDPs as the optimization problem

$$\begin{array}{ll} \max\limits_{\theta} & J(\theta) \\ \text{s.t.} & \theta \in \Theta \end{array}\,, \tag{3.3}$$

where the optimization is now constrained to the set of possible world configurations Θ. The formulation in (3.3) is rather general, not relying on any particular parameterization of the transition probabilities for the specification of the set of possible world configurations Θ. We postpone to Section 3.3 a detailed discussion on different parameterizations and corresponding modeling impact. Let us now, instead, instantiate the framework proposed in the 2×3 CORRIDOR problem introduced previously.

In this case, and for simplicity of analysis, let us consider the obstacles actually correspond to doors, and define $\Theta = [0, 1]^2$ where a world configuration $\theta = [\theta_{A-D}, \theta_{B-E}]$ is such that θ_{x-y} corresponds to how much the door between states x and y is opened. Specifically, we let $\theta_{x-y} = 0$ correspond to a fully closed door, and $\theta_{x-y} = 1$ correspond to a fully open door. Additionally, we consider that the probability of the robot successfully moving through the door in either direction, $x \to y$ or $y \to x$, is proportional to θ_{x-y}.

Our configuration space Θ can be translated directly into a parameterization for the transition probabilities, where the transitions between states A and D take the form

$$\begin{aligned} P_\theta(D \mid A, \text{DOWN}) &= P_\theta(A \mid D, \text{UP}) = \theta_{A-D}, \\ P_\theta(A \mid A, \text{DOWN}) &= P_\theta(D \mid D, \text{UP}) = 1 - \theta_{A-D}, \end{aligned} \tag{3.4}$$

and, similarly, the transitions between B and E are defined as

$$\begin{aligned} P_\theta(E \mid B, \text{DOWN}) &= P_\theta(B \mid E, \text{UP}) = \theta_{B-E}, \\ P_\theta(B \mid B, \text{DOWN}) &= P_\theta(E \mid E, \text{UP}) = 1 - \theta_{B-E}. \end{aligned} \tag{3.5}$$

That is, when trying to move between states x and y, the robot has probability θ_{x-y} of moving to the intended state, and probability $1 - \theta_{x-y}$ of remaining in the same state.

In contrast with the unconstrained Counterfactual MDP from the previous section, a Counterfactual MDP with constraints only optimizes over a set of feasible configurations of the world—those corresponding to different openings of the two doors in the corridor. Consequently, the world configuration $\theta_1 = [1, 0]$, where the first door is fully open, would still be an optimal solution, with an associated value of -1. On the other hand, the teleporting world configuration θ_{T}, which allows the *"teleportation"* of the agent between states A and D by executing action STAY, would not be considered nor returned.

While this is an improvement over the previous formulation, we have that the world configuration θ_1 is still not a unique optimal solution. For example, the solution where both doors are open, $\theta_2 = [1, 1]$, is also an optimal solution. In fact, there are again infinite optimal solutions. This time, the optimal solutions comprise all world configurations where the first door is fully open, regardless of the opening of the second door. Formally,

$$\theta^* \in \{\theta : \theta_{A-D} = 1.0 \wedge \theta_{B-E} \in [0, 1]\}.$$

Intuitively, the world configuration θ_2 where both doors are open seems costlier to achieve, when compared to world configuration θ_1 where only the first door is open. This inherent cost results from the fact that this change to the world requires external intervention—since the agent cannot open the door itself—and more changes to the world should typically require more effort. We now extend the Counterfactual MDP model to include a *cost-aware component*.

3.1.3 Adding Costs to Counterfactual MDPs

The last section concluded with the observation that formulating Counterfactual MDPs as a constrained optimization problem prevents the agent from reasoning over infeasible/unrealistic configurations of the world. However, the solutions returned disregard the cost/effort associated in shifting the original world configuration to the new configuration. As a result, we observed that

according to that model, the world configuration θ_1 (where the first door gets fully open), and the world configuration θ_2 (where both doors get fully open) yield the same value. However, intuitively, we concluded that the world configuration with both doors open should be costlier, since the external intervention necessary in shifting the original world configuration seems to require a larger effort.

The final version of the model includes a cost component $C(\theta)$ that measures the cost associated in shifting the original world configuration θ_0 to a new world configuration θ. We formalize the resulting Counterfactual MDPs model as the optimization problem

$$\begin{array}{ll} \max\limits_{\theta} & F(\theta) = J(\theta) - C(\theta) \\ \text{s.t.} & \theta \in \Theta \end{array}, \tag{3.6}$$

where function $F(\theta)$ measures the value/cost tradeoff of a given world configuration θ. The definition of the cost function $C(\theta)$ is task-specific. Our formulation of Counterfactual MDPs is general and allows any definition. However, as we will see in Chapter 4, some solution methods for solving Counterfactual MDPs may impose some restrictions on the cost formulation.

In this case, and for simplicity of analysis, let us assume a simple cost function where the penalty for requesting a world configuration is proportional to the openings of both doors

$$C(\theta) = \theta_{A-D} + \theta_{B-E}.$$

The minimum cost function is achieved in the original world configuration θ_0 where both doors are closed. On the other hand, the cost function is maximized in world configuration θ_2, where both doors are open. More interesting observations ensue when adopting this cost function in the context of the Counterfactual MDP.

As discussed before, the world configurations with the first door and both doors open, θ_1 and θ_2, have the same expected value, *i.e.*, $J(\theta_1) = J(\theta_2)$. However, at the same time, we now have that $C(\theta_1) < C(\theta_2)$. Consequently, the solutions yielded by the resulting Counterfactual MDP now match our intuition that the world configuration were only the first door gets open is "better"—despite allowing the exact same optimal policy, the cost in shifting the original world configuration is smaller.

The only thing we are still missing is to check whether world configuration θ_1 is still better than the original world configuration θ_0. It is straightforward to check that

$$\begin{aligned} J(\theta_0) - C(\theta_0) &= v^*_{\theta_0}(A) \approx -4.1 - 0 \\ J(\theta_1) - C(\theta_1) &= v^*_{\theta_1}(A) = -1 - 1, \end{aligned}$$

and as such

$$F(\theta_0) \approx -4.1 < F(\theta_1) = -2.$$

We thus conclude that, in this scenario, the world configuration θ_1 is indeed better than the original world configuration. As such, it would be the solution to the Counterfactual MDP.

$\diamond$

In this section we introduced the Counterfactual MDP model. This model extends the standard MDP to allow the agent to reason over possible alternative configurations of the world, and

to plan over the possibility of actually operating in such configurations. In the process of introducing the model we discussed the importance of allowing a generic parameterization of the transition probabilities and cost function.

In the next section we contribute a complexity analysis of the Counterfactual MDP, and show that, in its general form, solving a Counterfactual MDP is hard. In the remainder of the chapter we provide an extensive discussion on different concrete parameterizations that can be used in the context of Counterfactual MDPs, as well as an alternative formulation for our model.

3.2 Complexity

We analyze the complexity of Counterfactual MDPs, both from a formal, computational point of view, and from a more practical standpoint. We show that the problem is hard both in theory and in practice.

3.2.1 Computational Complexity

Before analyzing the computational complexity of Counterfactual MDPs, we recall that for every optimization problem we can build an associated decision problem. In the case of the optimization problem (3.6), the associated decision problem is that of determining whether given an MDP $(\mathcal{X}, \mathcal{A}, P_0, r, \gamma, \mu_0)$, a cost function C and a lower-bound b, there is a probability transition function P such that the optimization problem has an optimal value larger than or equal to b.

The decision problem associated with problem (3.6) is NP-Hard, *even when no cost function is present*. This shows that the hardness in problem (3.6) is intrinsic in the selection of the transition function P that maximizes the expected value $J(P)$, and not due to the presence of a cost function. This observation is made formal in Theorem 1.

> **Theorem 1** *The decision problem associated with problem* (3.6) *is NP-Hard, even when* $C(P) \equiv 0$ *is the identically zero cost function.*

Proof. We reduce from 3-SAT-CNF, a well-known NP-Complete problem [30]. A boolean formula is said to be in 3-CNF if it is made up of a conjunction of clauses, and each clause is a disjunction of 3 literals. A literal corresponds to either a variable (*positive* literal) or the complement of a variable (*negative* literal). A 3-CNF formula is *satisfiable* if there is an assignment of truth values such that the formula evaluates to TRUE. For example, the formula at the bottom of Figure 3.2 is satisfiable since ($x_1 = 1$, $x_2 = 1$, $x_3 = 0$, $x_4 = 0$) is a satisfying assignment. The 3-SAT-CNF decision problem is that of assessing if a 3-CNF formula is satisfiable.

Given an instance of 3-SAT-CNF with n variables $\{x_1, \ldots, x_n\}$ and m clauses, we construct (in polynomial time in n and m) an instance of problem (3.6). This construction is depicted in Figure 3.2 for an example formula. The instance is constructed as follows. First, we let the state space $\mathcal{X}$ consist of $4m + 2$ states. For each clause i we create 3 states s_1^i, s_2^i, s_3^i, one for each of the 3 literals in the clause, and an initial state s_0^i. The two remaining states are absorbing and denoted as s_{fail} and s_{goal}. The action space $\mathcal{A}$ consists of a single action a. The reward function is always 0 except at state s_{fail} where it is -1. The probability transition function P_a

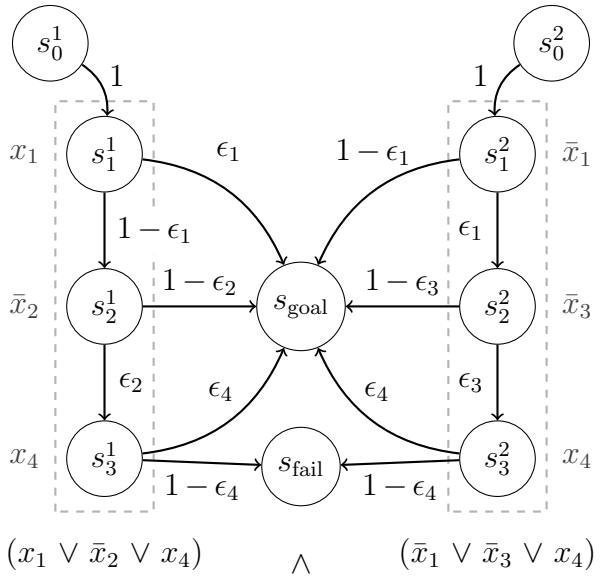


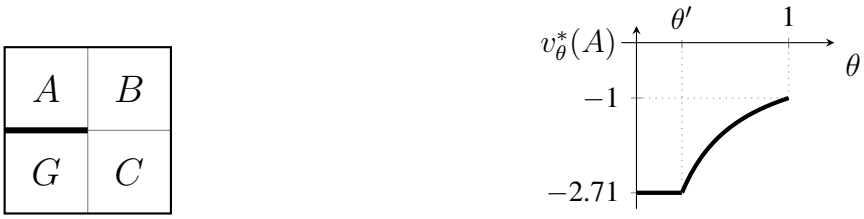
Figure 3.2: The MDP formulation of an instance of the 3-SAT-CNF problem with $n = 4$ variables and $m = 2$ clauses. Circles depict the $4m + 2$ states. Arrows depict the transition probabilities when executing the single action available. The transitions of the absorbing states were omitted for clarity.

is constructed as follows. For clause i, the initial state s_0^i transitions with probability 1 to the first literal in the same clause, s_1^i. Each state s_k^i ($k = 1, 2$) may transition to state s_{goal} or the next state in the clause s_{k+1}^i. State s_3^i may transition to s_{goal} or s_{fail}. Specifically, if state s_k^i is associated with a *positive* literal x_j, we let $P_a(s_{\text{goal}} \mid s_k^i) = \epsilon_j$. If s_k^i is associated with a *negative* literal, $\bar{x}_j$, we let $P_a(s_{\text{goal}} \mid s_k^i) = 1 - \epsilon_j$. The remaining transition probabilities are computed as $P_a(s_{k+1}^i \mid s_k^i) = 1 - P_a(s_{\text{goal}} \mid s_k^i)$ and $P_a(s_{\text{fail}} \mid s_3^i) = 1 - P_a(s_{\text{goal}} \mid s_3^i)$. In total we have n parameters ϵ_j (one for each variable), and use the same parameter ϵ_j on all states associated with variable $x_j \in \{x_1 \dots x_n\}$. The discount factor γ can be picked arbitrarily, as long as it is larger than 0. The initial distribution μ_0 is the uniform distribution over the initial states s_0^i. Finally, we let that $\epsilon_i \in [0, 1]$ for all $i \in \{1, \dots n\}$.

We now show that a solution to the 3-SAT-CNF problem exists if and only if problem (3.6) attains a value larger than or equal to $b \equiv 0$. This implies that the decision problem associated with problem (3.6) is at least as hard as the 3-SAT-CNF decision problem, completing the reduction.

($\Rightarrow$) We start with the *if* direction. Suppose the 3-SAT-CNF instance has a satisfying truth assignment $A = (A_1, \dots, A_n)$. We show our algorithm returns YES by constructing a probability transition function P such that the value of the optimization problem, $J(P)$, is larger than or equal to 0. To construct such probability transition function we simply let $\epsilon_i = A_i$ for all $i \in \{1, \dots, n\}$. Since a satisfying assignment makes at least one literal in every clause take value 1, by construction, in every clause there will be at least one state transitioning to s_{goal} with probability 1. Since s_{goal} is reached with probability 1, we must have $J(P) = 0$.

($\Leftarrow$) Moving to the *only if* direction, we start by supposing the decision problem associated with problem (3.6) returns YES, that is, there exists a transition probability P such that $J(P) \geq$



(a) (b)

Figure 3.3: (a) The layout of the simple version of the CORRIDOR scenario. (b) Plot of $v^*_\theta(A)$ as a function of θ, for this scenario.

0. By construction, no state can transition to s_{fail} with positive probability, or $J(P)$ would be negative. Hence, for every clause there must exist at least one state that transitions to s_{goal} with probability 1, and consequently the ϵ parameter associated with that transition is in $\{0, 1\}$. We can build a satisfying assignment as follows. For each variable $x_i \in \{x_1, \ldots, x_n\}$, we let $A_i = \epsilon_i$ if $\epsilon_i \in \{0, 1\}$, or we let $A_i = 0$ otherwise. Furthermore, by the way we constructed P, is is immediate to conclude that this truth assignment satisfies the logical formula. □

Theorem 1 immediately implies that that there cannot exist a polynomial algorithm that can solve all instance of problem (3.6). This remains true even if the cost function is constrained to be polynomial, linear or constant in P.

3.2.2 Practical Complexity

We provide evidence that solving Counterfactual MDPs is also hard in practice. We start with the observation that, in spite of its apparent simplicity, the optimization problem (3.6) associated with Counterfactual MDPs is a constrained non-convex optimization problem, since computing $\boldsymbol{v}^*_\theta$ involves solving the nonlinear recursion

$$v^*_\theta(x) = \max_{a \in \mathcal{A}} \left[r(x, a) + \gamma \sum_{y \in \mathcal{X}} P_\theta(y \mid x, a) v^*_\theta(y) \right]. \tag{3.7}$$

The dependence of the objective function F on θ is, therefore, non-trivial to express and optimize.

We show this in practice, by considering a simplified 2×2 version of the CORRIDOR scenario, depicted in Figure 3.3(a). In this simplified scenario there exists a single door, blocking the movement between initial state A from the goal state G. We adopt a formulation similar to that used in Section 3.1.3, and consider $\Theta = [0, 1]$, where configuration θ corresponds to how much a door is opened ($\theta = 0$ corresponds to a fully closed door, $\theta = 1$ corresponds to a fully open door). Moreover, we consider again that the probability of the robot successfully going through the door in either direction is proportional to θ. The configuration space Θ can be translated directly into a parameterization for the transition probabilities similar to that in (3.4)

$$\begin{aligned} P_\theta(G \mid A, \text{DOWN}) &= P_\theta(A \mid G, \text{UP}) = \theta \\ P_\theta(A \mid A, \text{DOWN}) &= P_\theta(G \mid G, \text{UP}) = 1 - \theta. \end{aligned} \tag{3.8}$$

Letting v^*_θ denote the optimal value function for the transition probabilities parameterized by θ, we arrive at the following system

$$\begin{cases} v^*_\theta(A) &= \max\{-1+\gamma v^*_\theta(B), -1+\gamma\left((1-\theta)v^*_\theta(A)+\theta v^*_\theta(G)\right)\} \\ &= \max\left\{-1-\gamma-\gamma^2,\ -\frac{1}{1-\gamma(1-\theta)}\right\} \\ v^*_\theta(B) &= -1-\gamma \\ v^*_\theta(C) &= -1 \\ v^*_\theta(G) &= 0, \end{cases} .$$

where we assumed a reward -1 for all state-action pairs, except (G, STAY), where we take reward 0. In the computation of $v^*_\theta(A)$ we only consider the actions RIGHT and DOWN as all other action result in the agent remaining in A, which is suboptimal. Moreover, $v^*_\theta(B)$ does not depend on θ because moving from B to G always takes 2 steps in the optimal case.

Figure 3.3(b) depicts $v^*_\theta(A)$ as a function of θ, when assuming a discount factor $\gamma = 0.9$. Note that this function is flat in the region $[0, \theta')$, with $\theta' \approx 0.3$. This makes it hard to use methods based on local information. Unfortunately, these flat regions may be common in general. In fact, these flat regions occur when the optimal policy selects actions associated with transition probabilities that are not affected by the changes to the world. In the aforementioned CORRIDOR scenario, the flat region occurred while the optimal policy was to select action RIGHT when in state A. For $\theta \in (\theta', 1]$ the optimal policy becomes to select action DOWN in that state, and consequently, the value v^*_θ grows with θ since the chances of this action moving the agent to state G are increasing. Finally, note that when $\theta = \theta'$ there are two deterministic optimal policies that differ on the action selected in state A. One selects action DOWN, the other RIGHT. Consequently, there are infinitely many stochastic optimal policies (any convex combination of those two deterministic optimal policies).

3.3 Specification of set of possible worlds

We formalized Counterfactual MDPs as a constrained optimization problem over the transition probabilities P, which maximizes the trade-off between the expected rewards in a world governed by P, and the cost of getting such world. This optimization constrains the search for world configurations to a set of "valid" configurations, thus preventing the agent from reasoning over unrealistic worlds. As discussed previously, in Section 3.1.2, this is necessary since unfeasible worlds may potentially allow the most rewarding optimal policies. Figure 3.4 depicts an arbitrary example of this phenomenon.

As anticipated in previous sections with the corridor example, our approach is to specify this set of possible worlds through a parameterization $\theta \in \Theta$ of the transition probabilities. In the CORRIDOR scenario we adopted the parameterization specified in (3.4) and (3.5). This parameterization constrains the optimization to only search over different transition probabilities between states A and D, and B and E—the states actually separated by a door. If the optimization was not constrained the agent could potentially request the (unrealistic) world where every action teleports the agent to the goal state.

We propose to consider two classes of parameterizations: *local* and *global*. Local parameterizations allow the parameterization of specific elements of the transition probabilities, providing a finer control over the definition of possible changes to the world (see Figure 3.5(a)). Global parameterizations, on the other hand, focus on the specification of a space of possible world configurations, by parameterizing the transition probabilities as a combination of possible world configurations known in advance (see Figure 3.5(b)). In practice, both types of parameterizations can be used for modeling a planning problem. The choice, however, comes with consequences in terms of modeling and algorithmic complexity.

3.3.1 Local Parameterizations

A *local* parameterization allows us to parameterize specific elements of the probability transition function. The simplest parameterization of this kind takes the form

$$\begin{aligned} P_\theta(y \mid x, a) &= \theta \\ P_\theta(z \mid x, a) &= 1 - \theta, \end{aligned} \tag{3.9}$$

for $\theta \in [0, 1]$, arbitrary states x, y, z, and arbitrary action a. The transition probabilities remain stochastic, since both elements $P_\theta(y \mid x, a)$ and $P_\theta(z \mid x, a)$ remain necessarily non-negative and sum up to 1. More precisely, this is only guaranteed if there is a probability 1 of moving from state x to one of y or z in the original world configuration P_{θ_0}. In order to avoid this additional assumption, this parameterization can be extended as

$$\begin{aligned} P_\theta(y \mid x, a) &= \xi_{x,a}\, \theta \\ P_\theta(z \mid x, a) &= \xi_{x,a}\, (1 - \theta), \end{aligned}$$

where $\xi_{x,a} = P_{\theta_0}(y \mid x, a) + P_{\theta_0}(z \mid x, a)$ is a normalization constant.

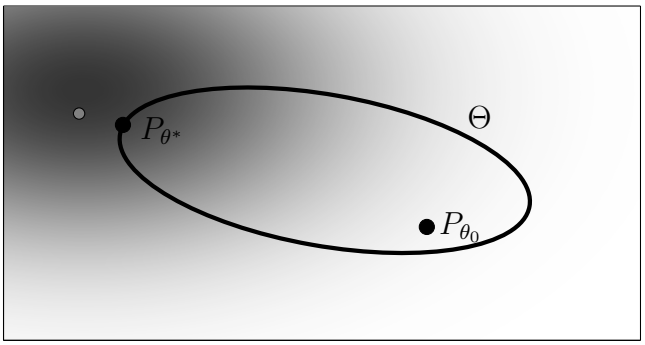
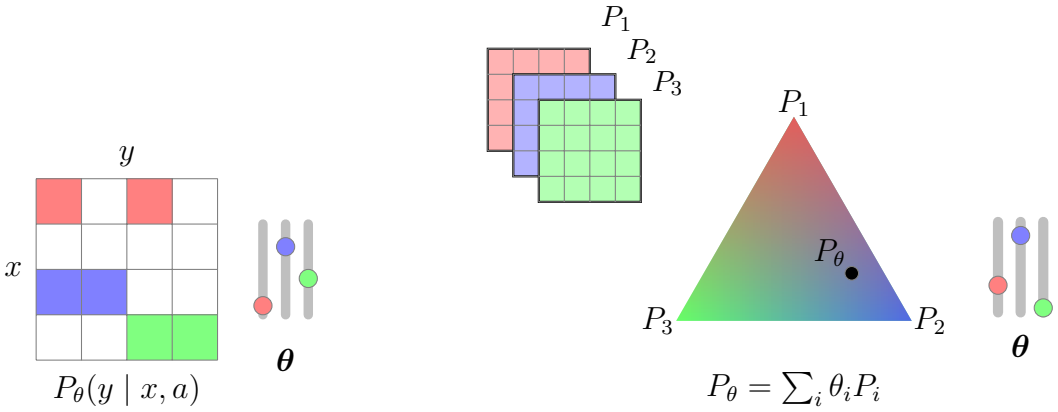


Figure 3.4: Illustration of an arbitrary space of possible world configurations Θ. The values associated with the world configurations are depicted by the shaded region, with darker colors corresponding to higher values. Note how the configuration associated with the highest value (gray circle) may not be possible, *i.e.*, it is outside the space Θ and corresponds to an unrealistic configuration of the world that the agent should not consider. P_{θ^*} is the most valuable feasible configuration of the world.



(a) Local parameterization

(b) Global parameterization

Figure 3.5: The two classes of parameterizations considered. (a) Abstract example of a local parameterization that controls specific elements of the transition probabilities. Each color corresponds to a different parameter that can be controlled. (b) Abstract example of a global parameterization with multipe vertex configurations. The resulting world P_θ is computed as a convex combination of these vertex configurations.

The CORRIDOR scenario presented in previous sections illustrates an application of this parameterization. Two remarks are in order. First, for longer versions of the CORRIDOR scenario, on $2 \times L$ grids, the number of parameters grows linearly with the number of doors. Secondly, this parameterization has the disadvantage of imposing a bounded domain on θ, which may require more complex solution methods. For example, a projection operator may be necessary to ensure the configurations are within the valid domain.

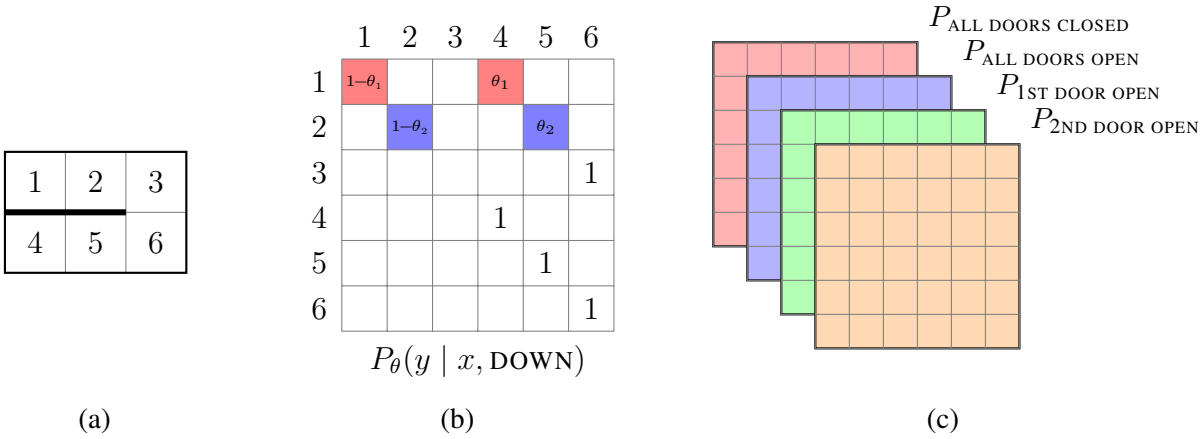
This can be avoided using a softmax operator. We consider a parameterization where entries (x, a, y) of the transition probabilities are associated with parameters $\theta_{x,a,y}$. For an arbitrary state-action pair (x, a), we define $\mathcal{X}_{x,a}$ as the set of entries $(x, a, y), \forall y \in \mathcal{X}$ that are associated with parameters. The transition probabilities are then formulated as

$$P_\theta(y \mid x, a) = \xi_{x,a} \frac{\exp(\theta_{x,a,y})}{\sum_{z \in \mathcal{X}_{x,a}} \exp(\theta_{x,a,z})}, \tag{3.10}$$

where $\xi_{x,a} = \sum_{z \in \mathcal{X}_{x,a}} P_{\theta_0}(z \mid x, a)$, is a normalization constant. This parameterization does not impose a bounded domain on parameters θ. The disadvantage of this parameterization is that it cannot represent sparse solutions, since the softmax function is never 0.

3.3.2 Global Parameterization

We consider a *global* parameterization that which parameterizes the transition probabilities as a combination of possible world configurations known in advance. An example is when the transition probabilities are parameterized as a convex combination of a finite set of M world configurations known *a priori*, each represented as a probability transition matrix $P_i, i = 1 \dots M$



(a) (b) (c)

Figure 3.6: Application of local and global parameterizations to the corridor scenario. (a) A 2×3 corridor with 6 states. (b) The local parameterization only requires 2 parameters, one per door that can be open. (c) The global parameterization requires 4 parameters, one per possible combination of doors open/closed.

$$P_\theta = \sum_{i=1}^{M} \theta_i P_i, \tag{3.11}$$

where parameters θ lie on the $M-1$ simplex, that is, all elements of θ are non-negative and sum up to 1. This parameterization also requires a bounded domain for the parameters. It is possible to avoid this by extending the parameterization, similarly to what we did before:

$$P_\theta = \sum_{i=1}^{M} u_i(\theta) P_i, \qquad u_i(\theta) = \frac{\exp(\theta_i)}{\sum_{j=1}^{M} \exp(\theta_j)}. \tag{3.12}$$

Let us now now illustrate an application of a global parameterization to the 2×3 CORRIDOR scenario depicted in Figure 3.6(a), where two doors can be opened. As depicted in Figure 3.6(c), the global parameterization requires the specification of 4 world configurations—configuration with all the doors open, $P_{\text{ALL DOORS OPEN}}$, all the doors closed $P_{\text{ALL DOORS CLOSED}}$, and with each door open, $P_{\text{1ST DOOR OPEN}}$ and $P_{\text{2ND DOOR OPEN}}$, respectively. This translates to a parameterization of the transition probabilities

$$P_\theta = \theta_0 P_{\text{ALL DOORS OPEN}} + \theta_1 P_{\text{ALL DOORS CLOSED}} + \theta_2 P_{\text{1ST DOOR OPEN}} + \theta_3 P_{\text{2ND DOOR OPEN}},$$

where θ lies on the 3-simplex. More generally, and assuming a corridor with N doors that can be opened, a global parameterization will require the specification of $M = 2^N$ transition probabilities matrices. We conclude that the number of vertex configurations to be specified grows exponentially with the number of environmental features that can be tuned.

3.3.3 Modeling Considerations

When modeling a planning problem, the choice between a local or global parameterization should take into account the nature of the possible changes to the world. If the possible changes to the world are specific to certain elements of the transition probabilities and mostly independent among themselves, it should be easier to build a local parameterization. For example, on the CORRIDOR scenario the local parameterization required a single parameter for each possible change—the opening of a door—and this parameter had an expressive meaning—how much the door is opened (see Figure 3.6(b)). On the other hand, a global parameterization would require defining the transition probabilities for all combinations of possible worlds—all doors closed, only one door opened, only two doors opened, and so on. This combinatorial explosion of models that need to be specified can become cumbersome.

In scenarios where environmental features impact multiple entries of the transition probabilities, it tends to be easier to build a global parameterization instead. For example, consider a variation of the CORRIDOR scenario where there is an environmental feature associated with the accuracy of the robot in moving RIGHT—the wheels of the robot are now skewed, thus, when moving right, the robot now moves to the cell to its right with probability θ_{R}, or remains in the same cell with probability $1 - \theta_{\text{R}}$. This feature impacts multiple entries of the transition probabilities in order to introduce the stochastic accuracy component when moving right. Modeling this feature with a local parameterization requires an accuracy parameter θ_{R} to be considered in all states. Additionally, when multiple environmental features impact the same states, complex parameterizations may be necessary in order to encode the desired stochastic transitions.

3.3.4 Algorithmic Considerations

The nature of the parameterization also has an impact at an algorithmic level. When the possible changes to the world are independent it can be more efficient to use a local parameterization. As discussed before, in the CORRIDOR scenario the local parameterization leads to a linear number of parameters with the number of possible changes to the world, while in the global parameterization we observe a combinatorial explosion. Computing and storing all the possible combinations can quickly become intractable. Additionally, optimizing over a larger number of parameters is typically harder. In the next Chapter we will conclude that the parameterization used may also impact the solution methods.

3.4 Alternative Formulations

In this chapter we introduced Counterfactual MDPs—a novel planning model that allows the agent to reason over additional possible configurations of the word, and to plan over the possibility of actually operating in such configurations. The decision process associated with this novel model can be interpreted as depicted in Figure 3.7. When compared with the decision process of a standard MDP (previously depicted in Figure 2.1), the main difference is that the agent starts by considering the possibility of operating in a new world configuration governed by transition probabilities P. After that step, the decision process takes the form of a standard MDP governed

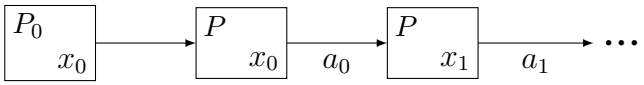


Figure 3.7: The decision process in Counterfactual MDPs. The boxes denotes the state (bottom right) and environment (top left) at each timestep. In this decision process, in the first step the agent may choose to operate in a different world configuration P, but afterwards only the state may change.

by that exact same world P.

A different formulation arises when we allow the agent to solicit different configurations of the world at each decision step, *i.e.*, during *execution time*. One possible approach for implementing this formulation consists in enriching the repertoire of actions of the agent with a new set of *soliciting* actions, each soliciting a different environment. Formally, we let $\mathcal{A}_\Theta = \{a_\theta,\ \forall \theta \in \Theta\}$ denote the set of actions for soliciting possible environments. These additional actions lead to an extended action set for the agent

$$\bar{\mathcal{A}} = \mathcal{A} \cup \mathcal{A}_\Theta. \tag{3.13}$$

In practice, introducing these new actions results in the new decision process depicted in Figure 3.8. At each timestep t, depending on the class of actions executed, the decision process can branch into either a new state x_t and the previous environment P_{t-1}, or to the previous state x_{t-1} and a new environment P_t.

Allowing the agent to solicit different environments requires changes to the agent's decision process. Specifically, as depicted in Figure 3.8, it is necessary to keep track of the current environment at each step. This can be modeled by extending the state space of the decision process as

$$\bar{\mathcal{X}} = \mathcal{X} \times \mathcal{X}_\Theta, \tag{3.14}$$

where $\mathcal{X}_\Theta$ denotes a set of states, each referring to a different environment. Notice that the extended state space is computed as the cartesian product between the original state space and

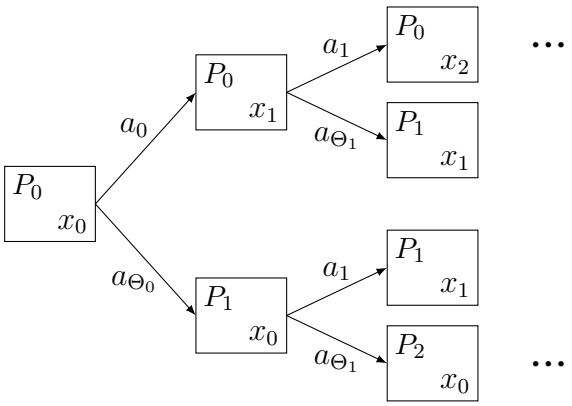


Figure 3.8: The decision process associated with the action-based formulation. At each timestep either the state or the environment may change, but not both. In the figure we assume actions $a_t \in \mathcal{A}$ and $a_{\Theta_t} \in \mathcal{A}_\Theta$.

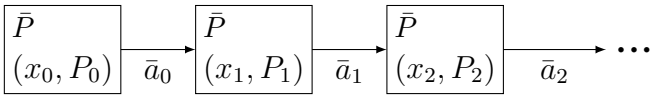


Figure 3.9: The decision process in the action-based formulation, modeled as an extended MDP. The boxes denotes the state (bottom right) and environment (top left) at each timestep. In this decision process, both the state and environment may change at each timestep.

$\mathcal{X}_\Theta$, since we assumed all the environments share the same state space. The transitions between extended states are modeled by an extended transition dynamics function $\bar{P} : \bar{\mathcal{X}} \times \bar{\mathcal{A}} \to \bar{\mathcal{X}}$.

The cost for soliciting a different environment is modeled with a *soliciting* cost function $c : \mathcal{X}_\Theta \times \mathcal{A}_\Theta \to \mathbb{R}$, where $c(x_\theta, a_{\theta'})$ denotes the cost of requesting environment $P_{\theta'}$ while in environment P_θ. This lets us define the extended reward function $\bar{r} : \bar{\mathcal{X}} \times \bar{\mathcal{A}} \to \mathbb{R}$

$$\bar{r}((x, x_\theta), \bar{a}) = r(x, \bar{a})\mathbb{I}(\bar{a} \in \mathcal{A}) - c(x_\theta, \bar{a})\mathbb{I}(\bar{a} \in \mathcal{A}_\Theta), \tag{3.15}$$

where $\mathbb{I}(a \in A)$ is the indicator function, taking value 1 when $a \in A$ and 0 otherwise. In words, for soliciting actions $\bar{a} \in \mathcal{A}_\Theta$, the extended reward returns the soliciting cost, independently of the current state x. For control actions $\bar{a} \in \mathcal{A}$, the extended reward returns the immediate reward, independently of the environment x_θ.

This formulation results in the decision process depicted in Figure 3.9, where at each step the world is governed by a fixed transition probabilities function $\bar{P}$. The goal of the agent is thus to compute the policy over the extended action set $\bar{\mathcal{A}}$ that maximizes the tradeoff between the expected rewards in the task and the costs associated with changes to the environment. This can be modeled as an optimization problem that maximizes the expected discounted extended rewards $\bar{r}$

$$\max_{\bar{\pi}} \sum_{t=0}^{\infty} \mathbb{E}_{\bar{a}_t \sim \bar{\pi}(\bar{x}_t),\ (x_{t+1}, P_{t+1}) \sim \bar{P}_t(\bar{x}_t, \bar{a}_t)} \left[\gamma^t \bar{r}(\bar{x}_t, \bar{a}_t) \mid \bar{x}_0 = (x_0, P) \right], \tag{3.16}$$

where $\bar{\pi}$ is an extended policy and $\bar{\pi}(\bar{a} \mid \bar{x})$ denotes the probability of selecting extended action $\bar{a}$ while in extended state $\bar{x}$. Problem (3.16) thus results in a maximization over extended policies.

In practice, this optimization problem corresponds to (large) MDP $\mathcal{M} = (\bar{\mathcal{X}}, \bar{\mathcal{A}}, \bar{P}, \bar{r}, \gamma)$, and as such can be solved using traditional MDP solving methods, as those previously discussed in Chapter 2. There are, however, drawbacks to this approach as discussed in the following sections. Before delving into those drawbacks, we first show a practical application to a human-robot interaction scenario.

3.4.1 Application to human-robot interaction scenario

We instantiate the alternative formulation to the following human-robot collaborative scenario.

> **Scenario** (1 STRAP BACKPACK ASSISTANCE) *A 2-arm manipulator assists a human to put a backpack, as depicted in Figure 3.10. The robot is trained to move a strap of the backpack through the arm of the human user up to the shoulder, and drop it there.*

(a) (b) (c)

Figure 3.10: Sequence of steps in the task where the robot helps a human user putting on the right strap of a backpack. (a) The human user starts with his right arm in a lowered position. (b) The human user raises his arm by request of the robot. (c) The robot putting the backpack on the human user.

> *The robot is provided with a unique control action* $\mathcal{A} = \{\text{EXECUTE}\}$*, which triggers the execution of a movement primitive previously taught to the robot by demonstration. The state space* $\mathcal{X} = \{I, S, F\}$ *includes initial, success and failure states. An execution has two possible outcomes. The agent transitions to* S *if it succeeds in the task, and to* F *otherwise. The agent receives a reward* 1 *when transitioning to state* S*, or a reward* -5 *when transitioning to state* F*.*

In this scenario, the execution success depends on the pose $\bar{x}$ adopted by the human user. There is a larger probability of success, when the pose adopted by the user is similar to those observed during training. We consider the pose of the user is represented by a 9-dimensional vector, consisting on the 3D position of the shoulder, elbow and hand joints of its right arm. Consequently, we estimate the transition probabilities of the two execution outcomes using a support vector machine (SVM) classifier trained to predict whether the robot can successfully execute the task for a given target pose of the human user.

We formulate this scenario as follows. We consider the extended state space $\bar{\mathcal{X}}$ as the 9-dimensional continuous space that spans all possible configurations of the three joints of the human user.

The set of soliciting actions $\mathcal{A}_\Theta$ includes voice requests that the robot can pose to the user. We consider a total of 5 voice requests that allow the robot to request the human user to MOVE FORWARD, MOVE BACKWARD, MOVE LEFT, MOVE RIGHT or RAISE RIGHT ARM. This set $\mathcal{A}_\Theta$ does not span all possible world configurations, but offers instead *relative* soliciting actions.

The extended transition probabilities $\bar{P}$ were learned from data. Since the transition probabilities depend how the human user responds to the robot's requests, we do not have, beforehand, a reasonable model of how the soliciting actions of the robot impact the target pose. We adopted a simple dynamic model of the form

$$\bar{y} = \bar{x} + d(a),$$

where a was the request issued by the robot and $d(a)$ was the observed displacement in the target

pose of the human user. For each soliciting action a we built a distribution $p_d(\cdot \mid a)$ and adopted

$$\bar{P}(\bar{y} \mid \bar{x}, a) = p_d(\bar{y} - \bar{x} \mid a).$$

Specifically, for each action $a \in \mathcal{A}_\Theta$ we collected samples from different human users responding to the robot's requests. Then, we used Gaussian kernel density estimation to represent $p_d(\cdot \mid a)$.

The extended reward function was designed so as to take into consideration the cost-benefit trade-off involved in requesting human assistance. In particular, we guided the design of the reward function so as to reward successful executions and penalize large number of requests. Guided by these general principles, we adopted

$$\bar{r}(\bar{x}, \bar{a}) = \begin{cases} -5 & \text{if } \bar{a} = \textsc{execute} \text{ and failed} \\ -1 & \text{if } \bar{a} \in \mathcal{A}_\Theta \\ 1 & \text{if } \bar{a} = \textsc{execute} \text{ and succeeded} \end{cases}$$

The resulting extended MDP was solved using the Q-learning algorithm [71]. We evaluated our approach on a real robot. Specifically, we performed 10 trials with a human user who was given the freedom to select the starting position. On all trials the robot was able to successfully complete the task, despite the different starting positions adopted by the user. We refer to [63] for more details on the approach and results here presented. It is worth noting we explored preliminary versions of this idea in previous works. We first explored an approach where the requests for changes to the world posed by the robot followed a set of manually coded rules [38]. That approach was applied and evaluated in a scenario where the robot helps a human user putting on a hat. Later, we proposed an improved approach suited for collaborative manipulation, between a human user and a robot, that followed a decision-theoretic framework [62]. This improved approach started the development of what resulted in our Counterfactual MDP model.

3.4.2 Discussion

This section introduced an alternative formulation to Counterfactual MDPs, based on the assumption that the agent is allowed to request different world configurations *during* execution time. This formulation culminates in the optimization problem (3.16), which consists of an extended MDP that can potentially be solved using traditional MDP solving methods. However, this alternative formulation comes with some computational drawbacks.

The scenarios envisioned in this thesis are characterized by continuous spaces of possible configurations of the world. Under the alternative formulation, this class of scenarios leads to a continuous extended state space $\bar{\mathcal{X}}$—this is required in order to keep track of the current world at any given timestep. Additionally, this infinite set of possible worlds also leads to an infinite set of soliciting actions $\mathcal{A}_\Theta$—one action per configuration that the agent may request. This results in a continuous state and action space MDP, and it is known in the literature that this class of MDPs are hard to solve.

One may consider an approach where these continuous spaces are discretized. However, this approach will suffer from the *curse of dimensionality* as we increase the number of parameters associated with the space of possible configurations of the world. Additionally, the quality of the solutions computed will also necessarily decrease, due to the loss of "resolution" caused

by the discretization. We note that in the human-robot interaction scenario considered before we assumed soliciting actions of a *relative nature*. That is, the soliciting actions voiced by the robot requested a relative change to the environment—for example, a raise of the human's arm. Consequently, we were able to keep a discrete action set. In general, however, there is no reason to assume this is always the case.

Finally, it is worth highlighting once more that Counterfactual MDPs assume the agent reasons over different configurations of the world at planning time, and that these worlds are then available at execution time. Despite this assumption, in practice, Counterfactual MDPs can be used to model human-robot interaction scenarios—we show multiple examples in the next chapter. However, in scenarios that assume continued interactions, it may be interesting to allow the agent to reason over the different configurations of the world at execution time. Counterfactual MDPs could potentially be used in this class of domains by adopting a *replanning* approach—at execution time the agent would solve new Counterfactual MDPs as necessary. However, this replanning approach can quickly become computationally expensive. As such, for this class of scenarios, the alternative formulation discussed may be more suitable.

3.5 Summary of the Chapter

This thesis focuses on endowing agents to reason, plan and act over the counterfactual *"What if the world were different?"* We focus, in particular, on planning problems modeled as Markov decision Processes, where the dynamics of the world are described in terms of transition probabilities. This chapter introduced Counterfactual MDPs—a more general MDP model that allows the agent to reason over the impact of different transition probabilities (*i.e.*, worlds) on the optimal policy and rewards the agent may collect, and to plan over the possibility of actually operating in a world governed by such transition probabilities.

We formulated Counterfactual MDPs as a constrained optimization problem (3.6) that maximizes the expected value/cost trade-off of changes to the transition probabilities. In introducing the general form of the model we motivate the need for the optimization problem to be constrained, and for the objective function to account for a cost.

The chapter proceeds by analyzing the complexity of the Counterfactual MDP model, showing that solving its general form is hard both in theory and in practice. We prove, in particular, that Counterfactual MDPs are actually NP-Hard. This complexity analysis provides important insights that motivate the algorithmic approaches developed in the next chapter. The next chapter contributes an evaluation on both the performance of the algorithms developed, and on the applicability of Counterfactual MDPs to different planning problems. That evaluation builds upon relevant practical contributions introduced in this chapter—namely, the two classes of parameterizations that we analyzed in Section 3.3.

The chapter concluded by introducing an alternative formulation to Counterfactual MDPs. This alternative formulation is particularly well-suited to human-robot interaction scenarios, in that it assumes the agent is allowed to request different configurations of the world during execution time. This formulation leads to a decision process that can actually be modeled as a large MDP, where the state and action spaces are extended to account for the possible configurations of the world—the state space needs to keep track of the current configuration at all times, whereas

the action space provides additional actions for requesting each possible world. We show how this formulation can be applied to a real-life scenario, where a robot assists a human user putting on a backpack. We also discuss, however, the disadvantages of this formulation. Namely, that it is not particularly well-suited to planning problems with a continuous set of possible configurations of the world. This is in contrast with our main formulation of Counterfactual MDPs, which, due to its formulation as an optimization problem, can handle this more general set of possible configurations of the world.

Chapter 4

Algorithms for Solving Counterfactual MDPs

In the previous chapter we introduced a model for planning under uncertainty that allows agents to reason over the counterfactual *"What if the world were different?"* This model extends Markov decision processes by lifting the assumption that the dynamics of the world are described in terms of a fixed probability transition function. The resulting model, dubbed Counterfactual MDP, was formulated as an optimization problem that jointly optimizes the optimal policy and transition probabilities of the MDP. We analyzed the complexity of solving this optimization problem, both from a formal, computational point of view, and from a more practical standpoint. We showed that solving this optimization problem is hard both in theory and in practice.

In this chapter we introduce approaches for solving Counterfactual MDPs. We start by proposing a gradient-based approach for solving the optimization problem (3.6). This approach builds upon the gradient of the objective function F in (3.6), with respect to the world configurations. We start by deriving two methods for computing the gradients with respect to the transition probabilities. The first method derives the gradient directly from the Karush-Kuhn-Tucker conditions of the optimization problem, whereas the second method exploits the linear programming formulation of MDPs, leading to a better performance. We then provide a detailed discussion that links the gradient computation to the parameterizations introduced in the previous chapter, in Section 3.3. Having the ability to compute the gradient allows us to introduce P-ITERATION—an iterative gradient-based algorithm for solving Counterfactual MDPs—and to discuss mechanisms for scaling it up.

4.1 Gradient Approach

We now propose methods for solving the optimization problem in (3.6). The objective function of this optimization problem is, in general, non-convex and therefore potentially hard to solve exactly. Consequently, we propose an iterative gradient-based approach.

Specifically, our approach starts from an arbitrary feasible initial configuration $\theta^{(0)} \in \Theta$ and iteratively builds a sequence $\left\{\theta^{(k)}, k = 1, \dots\right\}$ that will converge to a local maximum of $F(\theta)$

as

$$\theta^{(k+1)} = \theta^{(k)} + \alpha \nabla_\theta F(\theta^{(k)}). \tag{4.1}$$

We start by observing that

$$\nabla_\theta F(\theta) = \nabla_\theta J(\theta) - \nabla_\theta C(\theta). \tag{4.2}$$

We focus on the gradient of the expected discounted rewards, $J(\theta)$, since the cost function is task dependent. For a general parameterization of the transition probabilities, the gradient can be computed through the chain rule as

$$\frac{\partial J(\theta)}{\partial \theta_i} = \sum_{\substack{x,y \in \mathcal{X} \\ a \in \mathcal{A}}} \frac{\partial J(\theta)}{\partial P_\theta(y \mid x, a)} \frac{\partial P_\theta(y \mid x, a)}{\partial \theta_i}, \tag{4.3}$$

where two derivatives need computing. First, the derivative of the expected discounted rewards $J(\theta)$ with respect to the transition probabilities. Then, the derivative of the transition probabilities with respect to parameters θ.

We start in Sections 4.1.1 and 4.1.2 by formally deriving two methods for computing the former gradient. Then, in Section 4.1.3 we discuss the computation of the latter.

4.1.1 KKT-Based Method

We formally derive a method for computing the gradient of the expected discounted rewards of the optimal policy in an MDP, as a function of the transition probabilities, $\nabla_P J(P)$. This follows from the differentiation of a linear program with respect to P, using the Karush-Kuhn-Tucker (KKT) conditions.

As anticipated in Section 2.2, the value function $\boldsymbol{v}^*$ associated with the optimal policy can be computed as the solution to the linear program in (2.3). Any primal-dual solution $(\boldsymbol{v}^*, \boldsymbol{\lambda}^*)$ of problem (2.3) minimizes the Lagrangian

$$\mathcal{L}(\boldsymbol{v}^*, \boldsymbol{\lambda}^*) = \mathbf{1}^\top \boldsymbol{v}^* - \sum_{\substack{x \in \mathcal{X} \\ a \in \mathcal{A}}} \left(v^*(x) - r(x,a) - \gamma \sum_{y \in \mathcal{X}} P(y \mid x, a) v^*(y) \right) \lambda^*_{x,a},$$

and respects the stationary-complementary slackness (SCS) conditions—a subset of the Karush-Kuhn-Tucker conditions [7]. In this case the SCS conditions are

$$\begin{cases} 1 - \sum_{a \in \mathcal{A}} \lambda^*_{x,a} + \gamma \sum_{y \in \mathcal{X}} \sum_{a \in \mathcal{A}} \lambda^*_{y,a} P(x \mid y, a) = 0 & \forall x \in \mathcal{X} \\ \lambda^*_{x,a} \left(v^*(x) - r(x,a) - \gamma \sum_{y \in \mathcal{X}} P(y \mid x, a) v^*(y) \right) = 0 & \forall x \in \mathcal{X}, a \in \mathcal{A} \end{cases}$$

where $\lambda^*_{x,a} \in \mathbb{R}_+$ is a non-negative scalar for each state $x \in \mathcal{X}$ and action $a \in \mathcal{A}$. These SCS conditions can be seen as a level set $g(P, \boldsymbol{v}^*, \boldsymbol{\lambda}^*) = \mathbf{0} \in \mathbb{R}^{|\mathcal{X}| + |\mathcal{X}||\mathcal{A}|}$. This in turn defines an implicit function $h : P \mapsto (\boldsymbol{v}^*, \boldsymbol{\lambda}^*)$.

Let us fix a transition probability function $\bar{P}$ and let $(\bar{\boldsymbol{v}}^*, \bar{\boldsymbol{\lambda}}^*) = h(\bar{P})$ be the corresponding primal-dual solution of problem (2.3). From the implicit function theorem, we know that the derivative of h at point $\bar{P}$ is given by

$$\left[\left.\frac{\partial h}{\partial P}\right|_{\bar{P}}\right] = -\left[\left.\frac{\partial g}{\partial(\boldsymbol{v}^*, \boldsymbol{\lambda}^*)}\right|_{\bar{P},\bar{\boldsymbol{v}}^*,\bar{\boldsymbol{\lambda}}^*}\right]^{-1}\left[\left.\frac{\partial g}{\partial P}\right|_{\bar{P},\bar{\boldsymbol{v}}^*,\bar{\boldsymbol{\lambda}}^*}\right], \tag{4.4}$$

where $[\partial g/\partial(\boldsymbol{v}^*, \boldsymbol{\lambda}^*)|_{\bar{P},\bar{\boldsymbol{v}}^*,\bar{\boldsymbol{\lambda}}^*}]$ is the *Jacobian* of g relative to variable $(\boldsymbol{v}^*, \boldsymbol{\lambda}^*)$, evaluated at the point $(\bar{P}, \bar{\boldsymbol{v}}^*, \bar{\boldsymbol{\lambda}}^*)$. We consider $(\boldsymbol{v}^*, \boldsymbol{\lambda}^*)$ as the vector of "dependent variables" in g, and as such the Jacobian is a square matrix of dimension $|\mathcal{X}| + |\mathcal{X}||\mathcal{A}|$. Similarly, $[\partial g/\partial P|_{\bar{P},\bar{\boldsymbol{v}}^*,\bar{\boldsymbol{\lambda}}^*}]$ is the Jacobian of g relative to variable P, evaluated at point $(\bar{P}, \bar{\boldsymbol{v}}^*, \bar{\boldsymbol{\lambda}}^*)$. This Jacobian is a matrix of dimensions $(|\mathcal{X}| + |\mathcal{X}||\mathcal{A}|) \times |\mathcal{X}||\mathcal{A}||\mathcal{X}|$.

We note that Equation (4.4) is equivalent to solving the linear system $\mathbf{AX} = \mathbf{B}$, where

$$\mathbf{A} = \left[\left.\frac{\partial g}{\partial(\boldsymbol{v}^*, \boldsymbol{\lambda}^*)}\right|_{\bar{P},\bar{\boldsymbol{v}}^*,\bar{\boldsymbol{\lambda}}^*}\right], \quad \mathbf{B} = -\left[\left.\frac{\partial g}{\partial P}\right|_{\bar{P},\bar{\boldsymbol{v}}^*,\bar{\boldsymbol{\lambda}}^*}\right].$$

Specifically, $\mathbf{A}$ is a square matrix of dimension $|\mathcal{X}| + |\mathcal{X}||\mathcal{A}|$, $\mathbf{B}$ is a $(|\mathcal{X}| + |\mathcal{X}||\mathcal{A}|) \times |\mathcal{X}||\mathcal{A}||\mathcal{X}|$ matrix, and $\mathbf{X} = \partial h/\partial P|_{\bar{P}}$ is the Jacobian we wish to compute. This Jacobian includes the partial derivatives of $\boldsymbol{v}^*$ with respect to the transition probabilities P, $[\partial v^*/\partial P(y \mid x, a)]$. These partial derivatives can, subsequently, be used in the computation of the gradient of the expected rewards, $J(P)$, with respect to the transition probabilities. Computing matrices $\mathbf{A}$ and $\mathbf{B}$ is easy, since g has a simple representation, involving only sums and some monomials. However, a drawback of this method is that these matrices grow fast with the size of the problem.

4.1.2 Fixed Policy Differentiation

While the KKT-based method computes the gradient as a function of the transition probabilities, in practice, the Jacobians required to solve (4.4) become very large for MDPs with big state and action spaces.

We introduce a more computationally efficient method that exploits the linear program structure of problem (2.3), allowing us to reduce the size of the Jacobians required to compute the gradient $\nabla_P J(P)$. This method starts with the observation that

$$\nabla_P J(P) = \nabla_P \left[\boldsymbol{\mu}_0^\top \boldsymbol{v}_P^*\right]. \tag{4.5}$$

Since the initial distribution $\boldsymbol{\mu}_0$ does not depend on P we only have to compute $\nabla_P \boldsymbol{v}_P^*$. As discussed in Section 3.1.3, $\boldsymbol{v}_P^*$ is the solution of the fixed-point equation in (3.7), and consequently, the derivative cannot be directly computed. Another way of seeing this starts with the observation that the computation of $\boldsymbol{v}_P^*$ depends on both the optimal policy π_P^* and transition probabilities P. However, π_P^* also depends on P, and as seen in (2.4), this dependency is non-differentiable.

However, we observe that, except in a small subset of transition probabilities, the optimal policy π_P^* remains optimal in a neighborhood $B(P, \varepsilon)$ of P, for small enough positive constant

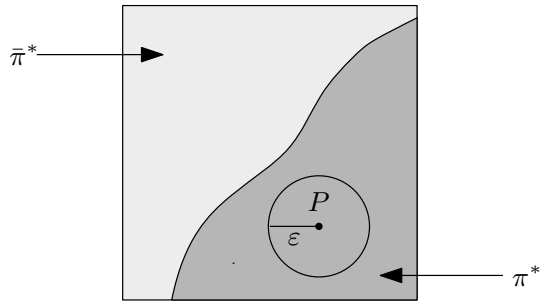


Figure 4.1: Close-up on an arbitrary space of transition probabilities. In the dark gray region, the optimal policy is π^*. In the light gray region, the optimal policy is $\bar{\pi}^*$. Depicts how policy π^* remains optimal in a neighborhood of probability transition function P.

ε[1] (see Figure 4.1). As a result, we have that

$$\boldsymbol{v}^*_{P'} = \boldsymbol{v}^{\pi^*_P}_{P'}, \quad \forall P' \in B(P, \varepsilon), \tag{4.6}$$

and consequently we can compute the exact gradient of $\boldsymbol{v}^*_{P'}$, the optimal value function in world $P' \in B(P, \varepsilon)$, by fixing π^*_P and ignoring its dependency with respect to the transition probabilities. Having that, from (2.2) we conclude

$$\begin{aligned} \frac{\partial \boldsymbol{v}^{\pi^*_P}_P}{\partial P(y \mid x, a)} &= \gamma \left(\mathbf{I} - \gamma \mathbf{P}^{\pi^*_P}\right)^{-1} \frac{\partial \mathbf{P}^{\pi^*_P}}{\partial P(y \mid x, a)} \left(\mathbf{I} - \gamma \mathbf{P}^{\pi^*_P}\right)^{-1} \boldsymbol{r}^{\pi^*_P} \\ &= \gamma \left(\mathbf{I} - \gamma \mathbf{P}^{\pi^*_P}\right)^{-1} \frac{\partial \mathbf{P}^{\pi^*_P}}{\partial P(y \mid x, a)} \boldsymbol{v}^{\pi^*_P}_P. \end{aligned} \tag{4.7}$$

By definition of $\mathbf{P}^\pi$, it follows that the Jacobian $\left[\partial \mathbf{P}^{\pi^*_P} / \partial P(y \mid x, a)\right]$ is a $|\mathcal{X}| \times |\mathcal{X}|$ sparse matrix where entry (x, y) has value $\pi^*_P(a \mid x)$. As such, we can extend the result in (4.7) as

$$\frac{\partial \boldsymbol{v}^{\pi^*_P}_P}{\partial P(y \mid x, a)} = \gamma \pi^*_P(a \mid x) v^{\pi^*_P}_P(y) \left(\mathbf{I} - \gamma \mathbf{P}^{\pi^*_P}\right)^{-1} \mathbb{1}_x \tag{4.8}$$

where $\mathbb{1}_x \in \mathbb{R}^{|\mathcal{X}|}$ denotes the indicator vector with x-th entry as 1. The Jacobian $\left[\partial \boldsymbol{v}^{\pi^*_P}_P / \partial P(y \mid x, a)\right]$ can thus be computed by solving a linear system $\mathbf{AX} = \mathbf{B}$, where

$$\mathbf{A} = \left(\mathbf{I} - \gamma \mathbf{P}^{\pi^*_P}\right), \qquad \mathbf{B} = \gamma \pi^*_P(a \mid x) v^{\pi^*_P}_P(y) \mathbb{1}_x, \tag{4.9}$$

and consequently, $\mathbf{A}$ is a square matrix of dimension $|\mathcal{X}|$ and $\mathbf{B}$ a vector of dimension $|\mathcal{X}|$. The computation of the full Jacobian $\left[\partial \boldsymbol{v}^{\pi^*_P}_P / \partial P\right]$ follows a similar flavour, with $\mathbf{B}$ extended as the horizontal stacking of (4.8) for all states $x, y \in \mathcal{X}$ and actions $a \in \mathcal{A}$, resulting in a $|\mathcal{X}| \times |\mathcal{X}||\mathcal{A}||\mathcal{X}|$ matrix. We note the size of the linear system that needs to be solved is significantly smaller than that of the KKT-based method. Moreover, as we will see in the next section, when

[1] This results from the continuity of the problem with respect to each entry of the transition probabilities, and can be formally established by replicating the proof of Proposition 4 in [48].

adopting certain parameterizations of the transition probabilities, in practice we may only need to compute the partial derivatives with respect to some (x, a, y) triplets.

Finally, one may wonder what happens when π^*_P does not remain optimal in a neighborhood of P. Unfortunately, the value function is no longer guaranteed to be differentiable when this is the case. This is depicted in Figure 3.3(b), which plots the value function as a function of the transition probabilities P, on the corridor scenario. As discussed before, at $\theta = \theta'$ there are multiple optimal policies, and we can observe that the function is not differentiable in that point. In practice, the set of points where the function is non-differentiable has null Lebesgue measure [48]. Furthermore, one can always resort to a subgradient method when these points are problematic [6].

4.1.3 Gradient of Parameterizations

From Equation (4.3) we observed that computing the gradient of the expected discounted rewards $J(\theta)$ with respect to the world configuration θ requires the computation of two partial derivatives. First, the derivative of the expected discounted rewards with respect to the transition probabilities. Then, the derivative of the transition probabilities with respect to the parameterization used. In Sections 4.1.1 and 4.1.2 we derived two methods for computing the former gradient. We now focus on the computation of the latter, *i.e.*, the gradient of the transition probabilities with respect to local and global parameterizations.

Local Parameterizations

For convenience, let us start by recalling the local parameterization previously defined in (3.9)

$$P_\theta(y \mid x, a) = \theta \qquad\qquad P_\theta(z \mid x, a) = 1 - \theta,$$

The gradient of this local parameterization can be computed as

$$\begin{aligned}\frac{\partial J(P_\theta)}{\partial \theta_i} &= \frac{\partial J(P_\theta)}{\partial P_\theta(y \mid x, a)} - \frac{\partial J(P_\theta)}{\partial P_\theta(z \mid x, a)} \\ &= \gamma \pi^*_\theta(a \mid x) \boldsymbol{\mu}_0^\top \left(\mathbf{I} - \gamma \mathbf{P}_\theta^{\pi^*_\theta}\right)^{-1} \mathbb{1}_x \left(v_\theta^{\pi^*_\theta}(y) - v_\theta^{\pi^*_\theta}(z)\right),\end{aligned}$$

where we used the chain rule (4.3) and gradient in (4.8).

Computing the gradient of a general *local* parameterization may, however, become expensive when the same parameter is used in multiple entries of the transition probabilities. This can be observed from the chain rule in (4.3), since the partial derivative $[\partial P_\theta(y \mid x, a)/\partial \theta_i]$ will be non-zero for all entries of the transition probabilities that θ_i impacts, and consequently needs to be computed. This can be problematic in the softmax parameterization in (3.10), since multiple parameters are used in the computation of the normalization denominator. In particular, for a state-action pair (x, a) the gradient for this parameterization is

$$\frac{\partial J(\theta)}{\partial \theta_{x,a,y}} = \gamma \pi^*_\theta(a \mid x) \boldsymbol{\mu}_0^\top \left(\mathbf{I} - \gamma \mathbf{P}_\theta^{\pi^*_\theta}\right)^{-1} \mathbb{1}_x \sum_{z \in \mathcal{X}_{x,a}} v_\theta^{\pi^*_\theta}(z) \frac{\partial P_\theta(z \mid x, a)}{\partial \theta_{x,a,y}},$$

which can become expensive to compute as $|\mathcal{X}_{x,a}|$ grows. A good indicator that a *global* parameterization is better suited is actually when a single parameter impacts many entries of the transition probabilities.

Global Parameterizations

In *global* parameterizations the impact of a parameter is bounded by the number of vertex world configurations of the feasibility model space. In fact, for the softmax version of the *global* parameterization in (3.12), we can analytically compute the gradient as

$$\frac{\partial J(P_\theta)}{\partial \theta_i} = \gamma u_i(\theta) \boldsymbol{\mu}_0^\top \left(\mathbf{I} - \gamma \mathbf{P}_\theta^{\pi_\theta^*}\right)^{-1} \left(\mathbf{P}_i^{\pi_\theta^*} - \mathbf{P}_\theta^{\pi_\theta^*}\right) \boldsymbol{v}_\theta^{\pi_\theta^*}, \tag{4.10}$$

since

$$\begin{aligned}
\frac{\partial u_j(\theta)}{\theta_i} &= u_i(\theta)\mathbb{I}(i=j) - u_i(\theta)u_j(\theta), \\
\frac{\partial P_\theta}{\partial \theta_i} &= \sum_{j=1}^{M} \frac{\partial u_j(\theta)}{\partial \theta_i} P_j = u_i(\theta)\left(P_i - P_\theta\right),
\end{aligned}$$

where $\mathbb{I}(y = x)$ is an indicator function, taking value 1 when $y = x$ and 0 otherwise. The gradient of this parameterization may be more efficient to compute in problems with large state spaces, and small number of vertex world configurations.

4.2 P-Iteration

We propose a gradient-based algorithm for solving Counterfactual MDPs, dubbed P-ITERATION. The algorithm is summarized in Algorithm 1. It includes an outer cycle that performs N random restarts, to avoid local maxima. In particular, each iteration of this outer cycle computes a solution starting from a random initial configuration $\theta^{(0)}$, and in the end, P-ITERATION returns the best solution found. Within the outer cycle, the algorithm performs standard projected gradient updates following the gradients computed with either the KKT or fixed policy differentiation methods. The updates continue until a stopping condition is met. This may involve either the norm of the gradient, the distance between consecutive estimates, or even a predefined maximum number of iterations. The projection operator Proj_Θ ensures that the iterates of the algorithm remain within the set Θ of admissible configurations. As such, the projection operator to be used depends on the parameterization of the transition probabilities considered. As discussed in Section 3.3, however, it is possible to formulate parameterizations of the transition probabilities with unbounded domain, which do not require this projection operator.

Algorithm 1 P-ITERATION algorithm

Require: MDP, $\mathcal{M} = (\mathcal{X}, \mathcal{A}, P_{\theta_0}, r, \gamma)$
Require: Initial state distribution, μ_0
Require: Configuration space, Θ
Require: Cost function, C
Require: Number of restarts, N
Require: Stopping condition, ϵ
1: $\theta^* = \theta_0$
2: $F^* = J(\theta^*) - C(\theta^*)$
3: **for** $n = 1$ **to** N **do**
4: $k \leftarrow 0$
5: Randomly select $\theta^{(0)}$
6: **repeat**
7: $\boldsymbol{v}^*_{\theta^{(k)}} \leftarrow \text{SOLVE\_MDP}(\mathcal{X}, \mathcal{A}, P_{\theta^{(k)}}, r, \gamma, \mu_0)$
8: $J(\theta^{(k)}) \leftarrow \boldsymbol{\mu}_0^\top \boldsymbol{v}^*_{\theta^{(k)}}$
9: $\nabla_\theta F(\theta^{(k)}) = \nabla_\theta J(\theta^{(k)}) - \nabla_\theta C(\theta^{(k)})$
10: Update

$$\theta^{(k+1)} \leftarrow \text{Proj}_\Theta[\theta^{(k)} + \alpha \nabla_\theta F(\theta^{(k)})]$$

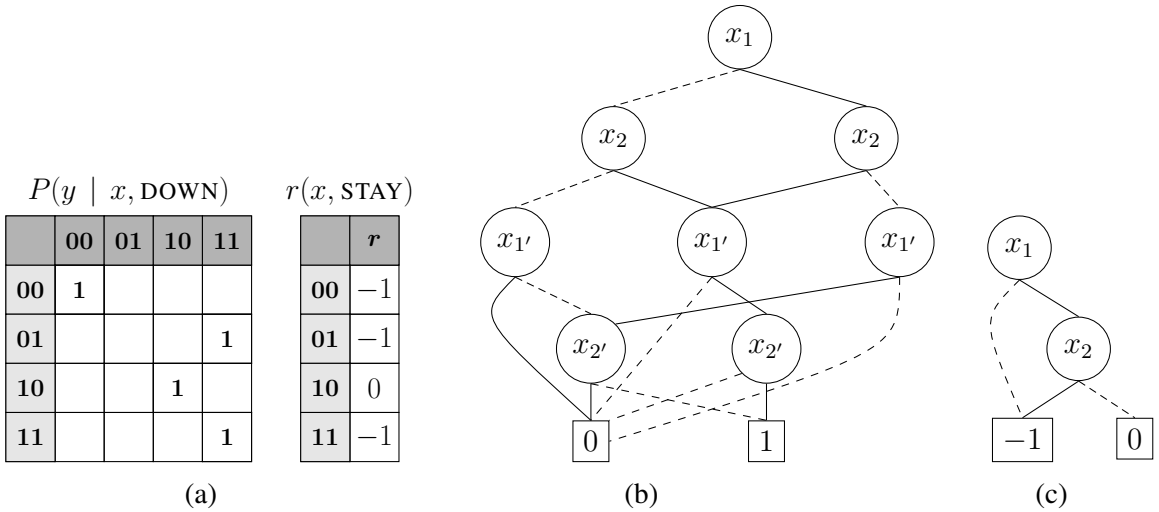
11: $k \leftarrow k + 1$
12: **until** $\left\|\theta^{(k)} - \theta^{(k-1)}\right\|_\infty < \epsilon$
13: $F^{(n)} = J(\theta^{(k)}) - C(\theta^{(k)})$
14: **if** $F^{(n)} > F^*$ **then**
15: $\theta^* = \theta^{(k)}$
16: $F^* = F^{(n)}$
17: **return** θ^*

4.3 Scaling Up

In previous sections we anticipated some degradation in our approach's performance as the problem size increases. In particular, due to the need of solving an MDP at each step of the gradient search (line 7 of Algorithm 1). This section discusses mechanisms for speeding up our approach, by alleviating the computations required in consecutive MDP solving steps. We start by exploring the application of factored and sparse MDP representations. Then, we consider approaches for reusing computations in solving intermediate MDPs.

4.3.1 Factored MDP representations

Our approach to speeding up P-ITERATION focuses on reducing the computational effort required in the consecutive MDP solving steps. To achieve this, we start by considering the adoption of an MDP representation typically used for solving large state and action spaces scenarios. Specifically, we consider a factored representation based on *Algebraic Decision Diagrams* (ADDs)—a



(a) (b) (c)

Figure 4.2: Application of ADDs to the simplified 2×2 corridor depicted in Figure 3.1(a). The scenario has 4 states. These can be encoded in binary format with 2 boolean variables—$x_1, x_2 \in \mathbb{B}$. (a) The transition probabilities and reward function for a given action. (b) The transition probabilities represented as an ADD. Dashed edges represent the boolean variable taking value 0. Full edges represent the boolean variable taking value 1. Note that we need 4 boolean variables—2 for indexing the rows (x_1, x_2), 2 for indexing the columns $(x_{1'}, x_{2'})$. (c) The reward function represented as an ADD.

data structure that allows for efficient computations of matrix-vector operations [4].

ADDs offer a compact representation for functions that map a boolean space to the space of real numbers: $f : \mathbb{B}^k \to \mathbb{R}$. When it comes to MDPs, ADDs can represent the reward function associated with an action $a \in \mathcal{A}$, $\boldsymbol{r}_a$, by encoding the state space in a boolean format. Similarly, ADDs can represent the transition probabilities associated with an action $a \in \mathcal{A}$, $\mathbf{P}_a$, by encoding in boolean format state and next-state pairs (x, y). This can be achieved by enumerating the states in $\mathcal{X}$ and assigning each the corresponding binary number. Encoding the states in boolean format requires $\log_2(|\mathcal{X}|)$ bits (boolean variables). Similarly, encoding the state and next-state pairs requires $2 \times \log_2(|\mathcal{X}|)$ bits. Figure 4.2 depicts a concrete example on the corridor scenario. ADDs allow us to compactly represent large reward and transition probability functions by factoring inputs with similar values. They are specially advantageous in problems with a lot of "structure" that can be compressed/factored (see Table 4.1).

This compact representation allows for efficient computations of matrix-vector operations, such as sums, multiplications, among others [4]. In fact, these are the operations involved in the value iteration algorithm, thus allowing the successful application of ADDs to solving large MDPs [22].

The main challenge in applying ADDs to the P-ITERATION algorithm is that ADDs are not particularly efficient at computing matrix inverses or solving linear equation systems. This is a key step of the P-ITERATION algorithm in computing the gradient, as discussed in Sections 4.1.1 and 4.1.2. Another challenge regards the overheads associated with building large functions using ADDs, when compared to the typical matrix representations—while matrices allow direct

Table 4.1: The approximate size of the transition probabilities function in the CORRIDOR scenario, assuming the dense matrix or ADD representation. The matrix size is measured as rows times columns. The ADD size is measured in terms of tree nodes.

Length	Matrix	ADD
2	16	10
4	64	22
8	256	46
16	1024	94
512	1M	3070

access to each entry, building an ADD typically requires composing multiple subtrees. These overheads are exacerbated by the fact that many functions need to be recomputed at every step of our algorithm, due to its iterative nature. However, as we will observe in Section 4.4, despite the additional computation effort required for dealing with these two challenges, the performance improvements achieved on the MDP solving steps can make it worthwhile to adopt the ADD representation.

Computing $\frac{\partial J(\theta)}{\partial P_\theta(y|x,a)}$ with ADDs

As mentioned above, ADDs are not efficient at computing matrix inverses or solving equation systems. However, we realize it is possible to expand the linear system in (4.9) as

$$(\mathbf{I} - \gamma \mathbf{P}_\theta^{\pi_\theta^*})\boldsymbol{s} = \mathbb{1}_x \iff \boldsymbol{s} = \mathrm{T}(\boldsymbol{s}) = \mathbb{1}_x + \gamma \mathbf{P}_\theta^{\pi_\theta^*}\boldsymbol{s}, \tag{4.11}$$

where operator T is a contraction mapping. This follows from

$$|\mathrm{T}(\boldsymbol{s}_t) - \mathrm{T}(\boldsymbol{s}^*)| = |\mathbb{1}_x + \gamma \mathbf{P}_\theta^{\pi_\theta^*}\boldsymbol{s}_t - \mathbb{1}_x - \gamma \mathbf{P}_\theta^{\pi_\theta^*}\boldsymbol{s}^*| = \gamma |\mathbf{P}_\theta^{\pi_\theta^*}(\boldsymbol{s}_t - \boldsymbol{s}^*)| \leq \gamma |\boldsymbol{s}_t - \boldsymbol{s}^*|, \tag{4.12}$$

since $\mathbf{P}_\theta^{\pi_\theta^*}$ is a stochastic matrix and its eigenvalues are bounded by 1.

As a result, it is possible to compute $\boldsymbol{s}^*$ through fixed-point iteration (FPI). Furthermore, operator T consists of matrix-vector operations, which ADDs are efficient at. This allows us to compute the gradient as

$$\frac{\partial \boldsymbol{v}_{P_\theta}^*}{\partial P_\theta(y \mid x, a)} = \gamma \pi_\theta^*(a \mid x) v_{P_\theta}^{\pi_\theta^*}(y) \boldsymbol{s}^*,$$

and consequently $\frac{\partial J(\theta)}{\partial P_\theta(y|x,a)} = \boldsymbol{\mu}_0^\top \frac{\partial \boldsymbol{v}_{P_\theta}^*}{\partial P_\theta(y|x,a)}$. The remaining steps of the P-ITERATION algorithm continue the same.

Computing $\nabla_\theta J(\theta)$ with ADDs

One disadvantage of the previous approach is that it may require solving multiple fixed point iteration computations in order to compute the final gradient $\nabla_\theta J(\theta)$. Specifically, it solves an FPI computation for all $x \in \mathcal{X}$ such that $\exists y \in \mathcal{X}, a \in \mathcal{A} : \frac{\partial P_\theta(y|x,a)}{\partial \theta_i} \neq 0$. As a result, this approach works best in problems where the possible modifications to the world are local to a small number of starting states x, *i.e.*, rows in the transition probabilities.

However, we can improve the performance of this approach by exploiting the FPI to compute the full gradient $\nabla_\theta J(\theta) = \nabla_\theta \left[\boldsymbol{\mu}_0^\top \boldsymbol{v}_\theta^*\right]$ at once. To achieve this, we start by noticing that

$$\nabla_{\theta_i} \boldsymbol{v}_\theta^* = \sum_{\substack{x,y \in \mathcal{X} \\ a \in \mathcal{A}}} \left(\gamma \pi_\theta^*(a \mid x) v_\theta^{\pi_\theta^*}(y) \left(\mathbf{I} - \gamma \mathbf{P}_\theta^{\pi_\theta^*}\right)^{-1} \mathbb{1}_x \frac{\partial P_\theta(y \mid x, a)}{\partial \theta_i} \right),$$

where we used the chain and gradient rules in (4.3) and (4.7), respectively. Now, for a parameter $1 \leq i \leq K$ and for each state $x \in \mathcal{X}$ we define

$$\mathcal{C}_{x,i} = \gamma \sum_{\substack{y \in \mathcal{X} \\ a \in \mathcal{A}}} \pi_\theta^*(a \mid x) v_\theta^{\pi_\theta^*}(y) \frac{\partial P_\theta(y \mid x, a)}{\partial \theta_i}. \tag{4.13}$$

As a result,

$$\left(\mathbf{I} - \gamma \mathbf{P}_\theta^{\pi_\theta^*}\right) \boldsymbol{s}_i = \begin{bmatrix} \mathcal{C}_{1,i} \\ \vdots \\ \mathcal{C}_{|\mathcal{X}|,i} \end{bmatrix} \iff \boldsymbol{s}_i = \begin{bmatrix} \mathcal{C}_{1,i} \\ \vdots \\ \mathcal{C}_{|\mathcal{X}|,i} \end{bmatrix} + \gamma \mathbf{P}_\theta^{\pi_\theta^*} \boldsymbol{s}_i,$$

where $\frac{\partial J(\theta)}{\partial \theta_i} = \boldsymbol{\mu}_0^\top \boldsymbol{s}_i^*$, is the gradient with respect to a single parameter θ_i. This is still a contraction, and can be proved following the steps in the previous proof.

The approach can be extended to the full gradient $\nabla_\theta J(\theta)$ by horizontally stacking the vectors above for each parameter K

$$\left(\mathbf{I} - \gamma \mathbf{P}_\theta^{\pi_\theta^*}\right) \boldsymbol{s} = \begin{bmatrix} \mathcal{C}_{1,1} & \dots & \mathcal{C}_{1,K} \\ \vdots & & \vdots \\ \mathcal{C}_{|\mathcal{X}|,1} & \dots & \mathcal{C}_{|\mathcal{X}|,K} \end{bmatrix} \iff \boldsymbol{s} = \begin{bmatrix} \mathcal{C}_{1,1} & \dots & \mathcal{C}_{1,K} \\ \vdots & & \vdots \\ \mathcal{C}_{|\mathcal{X}|,1} & \dots & \mathcal{C}_{|\mathcal{X}|,K} \end{bmatrix} + \gamma \mathbf{P}_\theta^{\pi_\theta^*} \boldsymbol{s}, \tag{4.14}$$

where $\boldsymbol{s}$ is a $|\mathcal{X}| \times K$ matrix. Then, the full gradient is computed as $\nabla_\theta J(\theta) = \boldsymbol{\mu}_0^\top \boldsymbol{s}^*$.

We note this gradient computation can be further specialized depending on the class of parameterizations used. Assuming a local parameterization such as that in (3.9), where a parameter θ_i controls the transition from x to either y or z, we have that each $\mathcal{C}_{\cdot,i}$ can be written as

$$\mathcal{C}_{\cdot,i} = \gamma \pi_\theta^*(a \mid x) \left(v_\theta^{\pi_\theta^*}(y) - v_\theta^{\pi_\theta^*}(z) \right) \mathbb{1}_x. \tag{4.15}$$

which turns out to be a sparse vector. On the other hand, assuming now a global parameterization, such as that in (3.11), where a parameter θ_i is associated with a world configuration P_i, we can specialize each $C_{\cdot,i}$ as

$$\mathcal{C}_{\cdot,i} = \gamma u_i(\theta) \left(P_i^{\pi_\theta^*} - P_\theta^{\pi_\theta^*} \right) \boldsymbol{v}_\theta^*. \tag{4.16}$$

Further speed-ups with sparse representations

In the previous section we discussed how adopting an ADD-based MDP representation could speed-up the MDP solving step of our algorithm. This is due to the ADD's factored nature, which may allow for more efficient matrix-vector operations, as those used in value iteration. In fact, past work has shown significant performance improvements in solving large MDPs using ADDs [22]. However, we also observed that computing the gradients using the ADD representation may end up being more computationally expensive, since an FPI method needs to be used. Additionally, the cost of computing the gradients may be further exacerbated by the aforementioned larger overheads associated with building the ADD trees, when compared to the matrix representation. As a result, reducing the impact of these overheads may speed up P-ITERATION even further when using ADDs.

One mechanism we found to work well in practice for reducing these overheads is to exploit the sparsity of planning problems. For example, the transition probabilities tend to be sparse due to *state locality*—that is, most states tend to transition to nearby states (according to some distance metric). As the problem size increases, building the ADD associated with the transition probabilities becomes increasingly costlier, in part, due to the need to iterate over the (x, a, y) state-action-next state triplets. We observed performance improvements when using instead data structures that allow the iteration over non-zero entries of the transition probabilities. For example, when it comes to the transition probabilities, for each state-action pair (x, a), we can just store a list of states for which the transition probability is positive. Formally, we only store transitions where

$$\mathcal{T} = \{y : y \in \mathcal{X} \wedge P(y \mid x, a) > 0\}.$$

Naturally, the performance improvements when adopting this technique will be the better the sparser the planning problem. In the last section of this chapter we show the benefits of this technique in practice.

4.3.2 Seeding the MDP solver

At each step t the P-ITERATION algorithm modifies the current transition probabilities P_{θ_t}, following the gradient $\nabla_{\theta_t} J(\theta_t)$. This results in a new MDP with transition probabilities $P_{\theta_{t+1}}$ that needs to be solved. However, the changes in the transition probabilities will typically be small, since, in general, we will be using small learning rates α. As such, for consecutive iterations t and $t+1$, we should roughly have $P_{\theta_t} \approx P_{\theta_{t+1}}$. As a result, it is reasonable to assume that the optimal values at consecutive iterations are also similar—*i.e.*, $\boldsymbol{v}_{\theta_t}^* \approx \boldsymbol{v}_{\theta_{t+1}}^*$.

Building upon this observation, it seems plausible that we can improve the performance of P-ITERATION by seeding each MDP solving step (line 7 of the algorithm), with the solution computed in the previous iteration. For instance, in the case of the *value iteration* method discussed in Section 2.2, this seeding could be performed by initializing the new estimate for the optimal value function as the previous optimal value function—*i.e.*, $\boldsymbol{v}^0_{\theta_{t+1}} = \boldsymbol{v}^*_{\theta_t}$.

We note, however, that the discount factor γ used may influence the impact of this technique in speeding up P-ITERATION. This follows from our previous discussion on the convergence of value iteration, and specifically, Equation (2.6). We observe that, in general, value iteration converges faster when using a smaller discount factor γ. As such, it is plausible that this seeding technique offers more significant performance improvements when adopting a larger γ. In the next section we show this technique in practice.

4.4 Results

In this section we evaluate the models and algorithms proposed. We show the applicability of Counterfactual MDPs in multiple scenarios, using different parameterizations of the transition probabilities and cost functions. The scenarios were selected in order to enable an analysis from different perspectives. On one hand, we consider *navigation* scenarios—these are well-structured scenarios that can be easily parameterized in terms of size. The benefits of these scenarios are twofold. First, they allow an intuitive interpretation of the resulting solutions. Second, they facilitate an analysis of the scalability of the methods proposed, both in terms of execution time and quality of solutions achieved. Besides the navigation scenarios, we also test our approach in a real-world robotic water pouring task. These tasks demonstrate the applicability of our models and methods in more complex scenarios, where the world configurations lead to complex parameterizations of the transition probabilities.

4.4.1 Navigation Scenarios

We start the analysis of Counterfactual MDPs with three different navigation scenarios, corresponding to increasingly more complex versions of the CORRIDOR scenario previously introduced, and depicted in Figure 1.1(a). For the purpose of our evaluation, we assume the corridor is now parameterized with length L, where the top and bottom rows are separated by an obstacle, except in the L-th column (see Figure 3.1(a) for $L = 3$). The second navigation scenario considered is the MAZE scenario that extends the corridor scenario to grids:

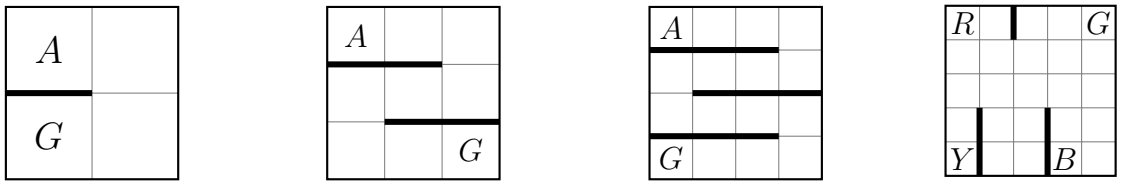
> **Scenario** (MAZE) *Consider the environment depicted in Figures 4.3(a) to 4.3(c). The* MAZE *scenario extends the* CORRIDOR *to $L \times L$ grids in which every pair of adjacent rows is separated by obstacles except in one of the ends (see Figure 4.3(c) for $L = 4$). The goal of the agent is to reach the goal position G. The state space $\mathcal{X}$ includes all the locations on the grid, and the action space includes the actions* UP, DOWN, LEFT, RIGHT, STAY. *Each moving action moves the agent deterministically to the adjacent cell in the corresponding direction, except if an obstacle is found. The reward is -1 for all state-action pairs except (G, STAY), where it is 0.*

Finally, we consider the TAXI scenario—a larger navigation scenario that serves as a popular benchmark problem from the reinforcement learning literature [14].

Scenario (TAXI) *Consider the environment depicted in Figure 4.3(d). An agent must navigate a* 5×5 *maze-like environment, picking up and dropping off a passenger from/to a set of pre-designated locations (marked as* R*,* G*,* Y *and* B *in the figure). The state space* $\mathcal{X}$ *includes the possible positions of the agent, the location of the passenger (in one of the marked sites or inside the taxi) and its destination (one of the marked sites), yielding up to* 500 *states. The action space* $\mathcal{A}$ *includes four moving actions (*UP*,* DOWN*,* LEFT*,* RIGHT*), as well as actions for picking up and dropping off the passenger—*PICK UP *and* DROP OFF*, respectively. The navigation actions move the agent deterministically in the corresponding direction, except if in the presence of the boundaries of the grid, or one of the five gates—Figure 4.3(d) depicts the five gates with thicker lines. The reward function penalizes the agent with* -1 *for every navigation action. Additionally, the reward for a successful drop-off is* 20*, while an unsuccessful drop-off (in the wrong location) penalizes the agent with* -10*.*

We adopt a local parameterization in all scenarios. Specifically, a parameterization similar to that in (3.4). In the case of the CORRIDOR and TAXI scenarios this results in a parameterized space of valid world configurations $\Theta = [0, 1]^K$, where we admit that K obstacles correspond to doors/gates, and θ_k indicates how much the k-th door/gate is open. In the MAZE scenario, this parameterization results in a fixed space of valid world configurations $\Theta = [0, 1]^{L-1}$, where we admit all the obstacles separating rows are doors, and θ_k indicates how much the k-th door is open. As before, these parameterizations assume that the probability of the robot moving through the door/gate is proportional to its openness.

In all scenarios, we consider there is a cost in changing the world so that the doors become open. Specifically, we adopt a cost function C that (roughly) represents a fixed cost for a door to be opened—*i.e.*, the cost of the door getting slightly open is the same as the cost for fully opening the door. In a way, this models the idea that the cost for changing the world mainly results from requiring the external intervention, and not necessarily from the request itself. For example, if we assume the external intervention takes the form of requests to open the door that are posed to human users, this cost function suggests that the underlying cost is in bothering the user in the first place. In order to model this cost function we consider a smooth and differentiable



(a) Maze scenario ($L = 2$) (b) Maze scenario ($L = 3$) (c) Maze scenario ($L = 4$) (d) Taxi scenario.

Figure 4.3: Example navigation scenarios. (a) to (c) The MAZE scenario for different values of L. (d) The TAXI domain. The letters correspond to pick-up/drop-off points for the passengers, and the bold lines correspond to the 5 gates that can be configured.

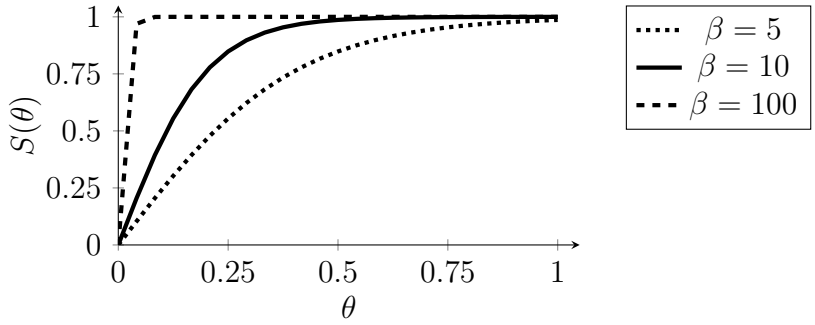


Figure 4.4: The smooth step function $S(\theta)$, for different values of β.

approximation of a step function

$$S(\theta) = \frac{2}{1 + e^{-\beta\theta}} - 1 \tag{4.17}$$

and let the cost function $C(\theta)$ average the step function in each component of θ over all states

$$C(\theta) = \frac{1}{|\mathcal{X}|} \sum_{k=1}^{K} S(\theta_k), \tag{4.18}$$

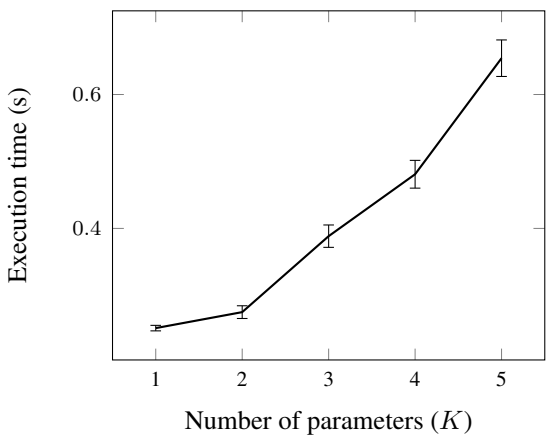
with $\beta = 100$. The smooth step-size function $S(\theta)$ is depicted in Figure 4.4 for different values of β. Note that larger values of β provide a better approximation of the true step function.

This Counterfactual MDP allows the agent to reason over different configurations of the world, which may be achievable by reaching out to a nearby human user at a (fixed) cost. The CORRIDOR scenario, in particular, is heavily inspired by recent research involving the deployment of real service robots in an office setting, with the robots requesting the help of nearby humans for calling the elevator [57].

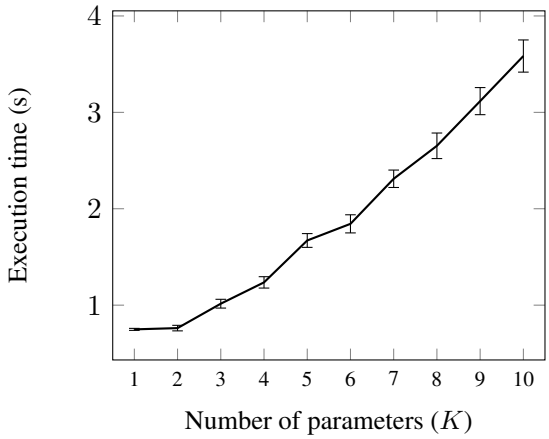
Results

Figure 4.5 and Table 4.2 summarize the performance of P-ITERATION in the different navigation scenarios, for different values of K and problem sizes. These experiments assume a uniform distribution μ_0 over the initial states and $N = 50$ random restarts. The Adam gradient algorithm is used as described in [37]. All results are averaged over 30 randomly seeded runs. Experiments performed on a ThinkPad W550s laptop, with 16GB of RAM and an Intel i7-5600U CPU.

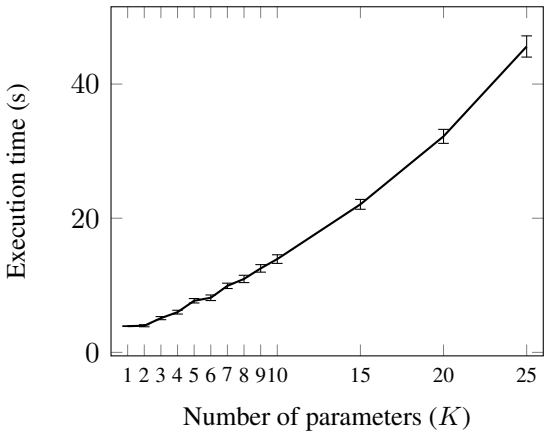
Figures 4.5(a) and 4.5(d) plot the execution time of P-ITERATION as the number of parameters K increases, in the CORRIDOR and TAXI scenarios. The four plots seem to suggest that the execution time degrades gracefully as the number of parameters increases. The two remaining Figures, 4.5(e) and 4.5(f), plot the execution time as a function of the problem size $(\mathcal{X} \times \mathcal{A} \times \mathcal{X} \times K)$. Figure 4.5(e) plots the execution time as a function of the corridor length L, assuming a fixed number of parameters $K = 1$. In Figure 4.5(f), both the state-space and number of parameters is growing, since in the MAZE scenario the number of parameters is linearly proportional to the state-space size. As such, these execution times seem to match our expectations.



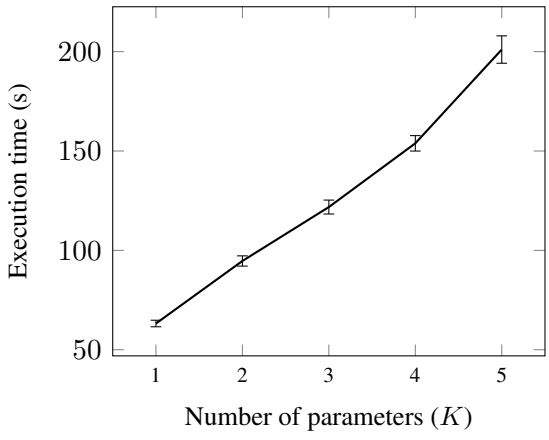
(a) Corridor of length $L = 10$



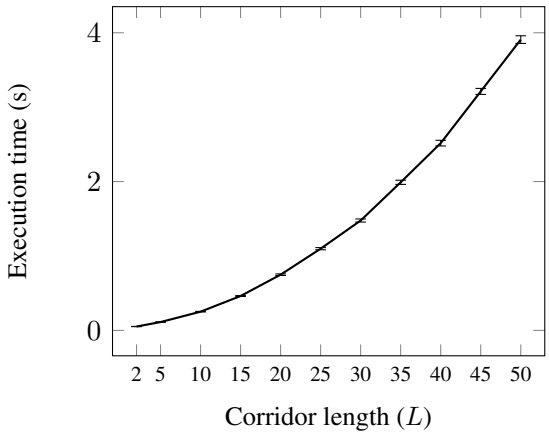
(b) Corridor of length $L = 20$



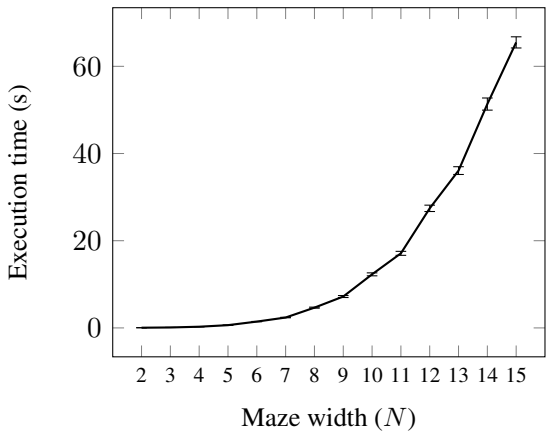
(c) Corridor of length $L = 50$



(d) Taxi



(e) Corridor



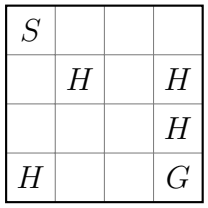
(f) Maze

Figure 4.5: Performance of the P-ITERATION algorithm in terms of execution time. Results are averaged over 30 runs. Bars represent the standard error. (a) to (d) Plots of the execution time as a function of the number of parameters in the corridor and taxi scenarios. (e) Plot of the execution time as a function of the corridor length and a single parameter. (f) Plot of the execution time as a function of the problem size of the maze scenario.

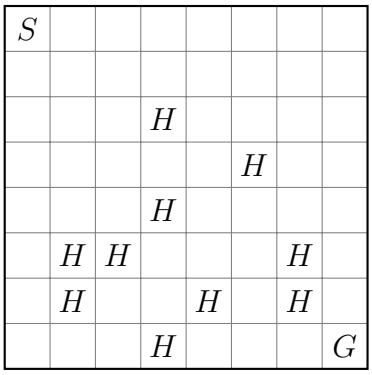
Table 4.2 suggests the algorithm scales satisfactorily in terms of the quality of the solutions achieved. In all three scenarios P-ITERATION is always able to find a configuration of the world θ at least as good as the original one, *i.e.*, $F(\theta) > J(\theta_0)$. In some cases the algorithm is actually able to consistently compute the optimal solution (in boldface). Since the scenarios are well structured, their optimal solutions are intuitive. In the case of the CORRIDOR scenario the best solution corresponds to fully opening the first door. In the MAZE scenario, in general, the optimal solution corresponds to opening all the doors on the first column. However, when the maze length is odd, opening all the doors on the rightmost column is also an optimal solution. In the TAXI scenario the optimal solution is to open all the gates. This is because, by definition, the cost of partially opening the door is the same as opening it completely.

Table 4.2: The performance of P-ITERATION in terms of the quality of solutions, in navigation scenarios. The results are rounded to 2 decimal places. Scenarios where the optimal solution was computed in more than 50% of the runs are in bold.

Scenario	N	$J(\theta_0)$	K	$F(\theta)$
CORRIDOR	2	-1.40	1	$\mathbf{-1.23}$
	5	-3.49	1	$\mathbf{-2.32}$
	10	-5.61	1	$\mathbf{-3.86}$
			2	$\mathbf{-3.86}$
			3	$\mathbf{-3.86}$
			4	$\mathbf{-3.87}$
			5	-3.90
	20	-7.54	1	$\mathbf{-5.85}$
			3	$\mathbf{-5.85}$
			5	-5.87
			7	-5.89
			10	-5.92
	50	-9.00	1	$\mathbf{-8.12}$
			5	-8.13
			10	-8.15
			15	-8.17
			25	-8.23
MAZE	2	-1.40	1	$\mathbf{-1.23}$
	6	-7.28	5	$\mathbf{-3.98}$
	7	-7.97	6	$\mathbf{-4.51}$
	11	-9.17	10	-6.16
	15	-9.56	14	-7.24
TAXI	—	80.13	1	$\mathbf{82.66}$
			2	$\mathbf{83.43}$
			3	$\mathbf{86.57}$
			4	$\mathbf{87.04}$
			5	$\mathbf{89.88}$



(a) FROZEN LAKE 4×4



(b) FROZEN LAKE 8×8

Figure 4.6: Layout of the FROZEN LAKE scenario for increasing sizes, with the start and goal states S and G, respectively, and the holes H that the robot must avoid.

4.4.2 Scenarios with complex parameterizations

To assess the applicability of our approach in a broader class of problems, we consider three additional domains that require more complex parameterizations of the transition probabilities.

FROZEN LAKE

We consider the FROZEN LAKE domain, a scenario that has recently become popular among the reinforcement learning community [8], and will serve as an application of global parameterizations in Counterfactual MDPs.

> **Scenario** (FROZEN LAKE) *Consider the scenarios depicted in Figures 4.6(a) and 4.6(b). A robot must traverse an icy grid, from an initial state S to a goal state G, while avoiding the holes H where the ice has melted. The movement of the robot is uncertain due to the frozen ice—when moving in a given direction, the wheels of the robot may slip, causing the robot to move in a different way. The state space $\mathcal{X}$ includes all the cells of the grid, and the action space is $\mathcal{A} = \{\text{UP}, \text{DOWN}, \text{LEFT}, \text{RIGHT}, \text{STAY}\}$. All moving actions are uncertain. For example, when moving left or right, there is a probability the robot moves up or down instead. Similarly, when moving up or down, the robot may end up actually moving left or right. As a result, the transition probabilities associated with a given moving action are sampled uniformly between the three possible directions (barring obstacles). Upon falling on a hole, the robot will stay in that state forever. Action* STAY *is deterministic and keeps the robot in the same state. The reward function takes value -1 in all state-action pairs, except (G, STAY) where it is 0.*

We assume the world can be configured in terms of the grip of the robot's wheels. Specifically, we assume the grip profile can be set between NO-GRIP and FULL-GRIP. The former denotes the original scenario, where the robot may slip and move in either the intended or perpendicular directions. The latter denotes a world where the robot always moves deterministically in the intended direction.

Table 4.3: The performance of P-ITERATION in terms of the quality of solutions, in the FROZEN LAKE scenario. The results are averaged over 30 runs.

FROZEN LAKE				
Size	$J(\theta_{\text{ng}})$	$F(\theta_{\text{g}})$	$F(\theta)$	$u_{\text{g}}(\theta)$
4×4	-46.34	-20.85	-14.55	0.930
8×8	-58.95	-28.13	-21.59	0.927

For this scenario, we adopt the softmax variant of the global parameterization, as defined in (3.12). We start by denoting the transition probabilities associated with the two world configurations. We let P_{ng} and P_{g} describe the dynamics of the worlds with no-grip and full-grip, respectively. This leads to a parameterization of the form

$$P_\theta = u_{\text{g}}(\boldsymbol{\theta}) P_{\text{g}} + u_{\text{ng}}(\boldsymbol{\theta}) P_{\text{ng}},$$

where for optimization purposes, we bound $\boldsymbol{\theta} \in [-\theta_{\max}, \theta_{\max}]^2$, with $\theta_{\max} = 4$.

Finally, we assume that changes to the grip profile of the robot require some engineering effort. We let the engineering effort required grow exponentially with the grip profile

$$C(\boldsymbol{\theta}) = K \exp\Big(-\beta \left(1 - u_{\text{g}}(\boldsymbol{\theta})\right) \Big),$$

where we take $K = 15$ and $\beta = -20$. According to this cost function, we assume a large effort is required in changing the configuration of the world to high grip performance, whereas achieving lower grip performance is not as costly.

Results Table 4.3 summarizes the performance of our approach on the FROZEN LAKE domain. First, column $J(\theta_{\text{ng}})$ denotes the value in the original world where the robot has no grip. Column $F(\theta_{\text{g}})$ denotes the value of the world configuration where the robot has full grip, while taking into account the engineering effort associated in shifting the original world. Finally, the last two columns regard the solutions achieved by our approach. Modeling the problem as a Counterfactual MDP and then solving it through P-ITERATION, we were able to find the solution that better establishes the value/cost tradeoff in changing the grip profile of the robot. Perhaps surprisingly, we observe that the most valuable world configuration is not P_{g}—that where the robot has full-grip. In fact, the best solution seems to occur at a grip profile set roughly to 93%. This solution suggests that the additional engineering effort required in achieving the full-grip configuration is not worth it.

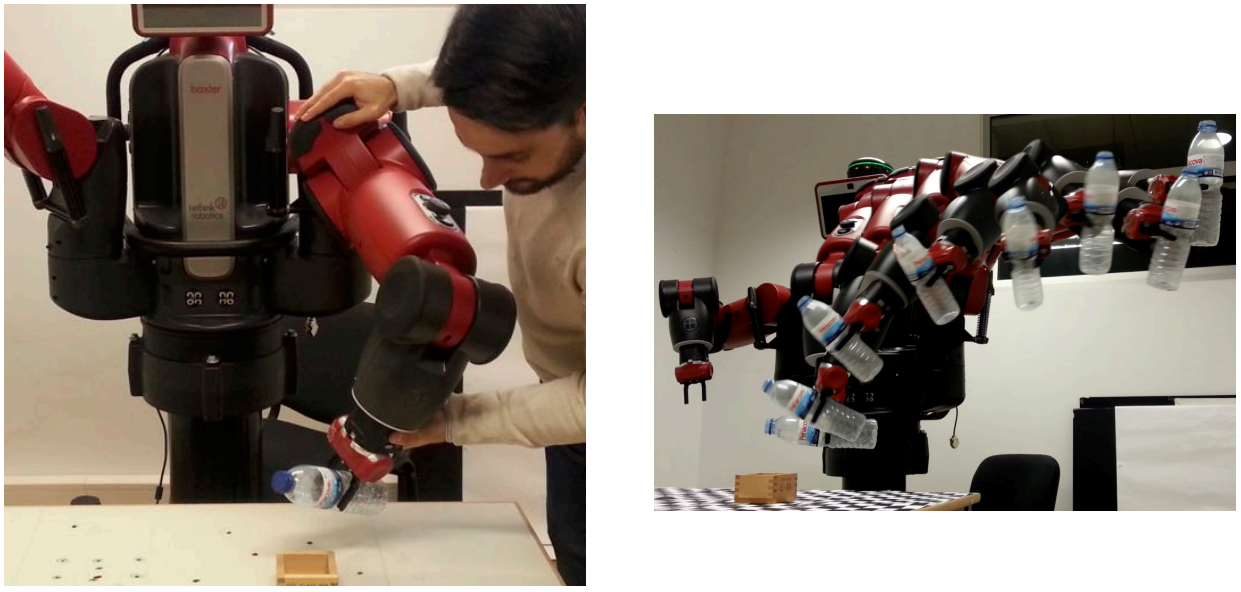
This scenario is interesting as it depicts the applicability of Counterfactual MDPs to engineering related problems, where it may be possible to estimate/model the value and costs of certain changes to the world.

ROBOT WATER POURING

Finally, we depict the applicability of Counterfactual MDPs on a fairly unstructured scenario. We consider a robotic water pouring task defined as follows.

Scenario (WATER POURING) *A robot is trained to pour water in cups located on top of a table, as depicted in Figure 4.7(a). Specifically, the robot is trained by demonstration how to pour water around a specific position $\bar{\boldsymbol{\theta}}$. After training, the robot is then tasked with pouring water in (possibly) different cup locations $\boldsymbol{\theta}_0$, as depicted in Figure 4.7(b). The robot is provided two actions $\mathcal{A} = \{x, q\}$, including the execution of the trained pouring motion, or "quit", signaling that the robot is not confident in executing the task for the current configuration of the world $\boldsymbol{\theta}_0$. The state space $\mathcal{X} = \{I, S, F, A\}$ includes* initial, success, failure *and* absorbing *states. The reward function penalizes the agent with -5 for executing q in state I. In that case, the agent also transitions immediately to state A. An execution x is penalized with reward -1 and has two possible outcomes. The agent transitions to S if it succeeds the task, and to F otherwise. In both outcomes, the agent ultimately transitions to state A, but when transitioning from F receives a penalty of -9.*

The transition probabilities of the two execution outcomes (success and failure) were estimated using 6 pouring trajectories collected on a real BAXTER robot. Specifically, the probability of success of an execution (moving from state I to state S) was modeled as a radial kernel centered in $\bar{\boldsymbol{\theta}}$, the average final position of the trajectories collected. The radial kernel used is depicted in Figure 4.8, with the shaded region centered around point $\bar{\boldsymbol{\theta}}$. We note this simple model is actually often used in practice, for example, in Probabilistic Movement Primitives [51].



(a) (b)

Figure 4.7: Instance of the robotic water pouring task. (a) The robot being taught by demonstration how to pour water in a cup. (b) The robot executing the task to a new pouring location.

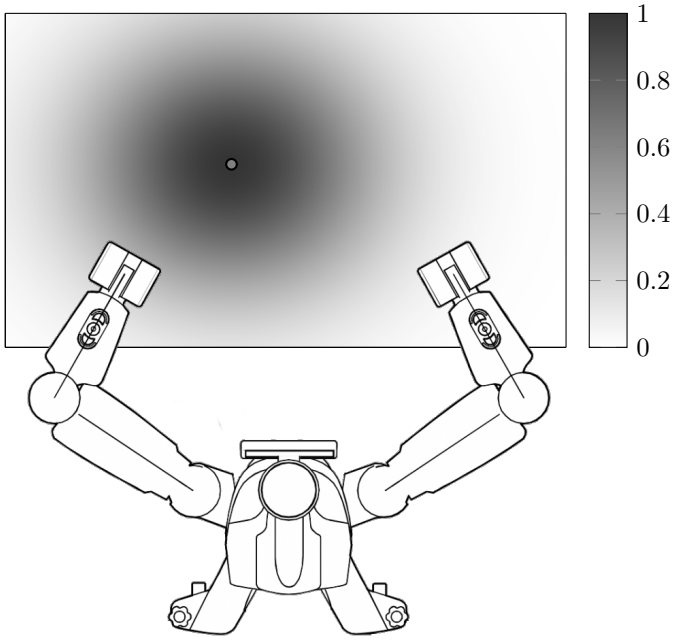


Figure 4.8: The setup of the WATER POURING scenario. The BAXTER robot has to pour water in cups located on a table in front. The execution success probability is modeled as a radial kernel, depicted by the shaded region. Darker corresponds to higher success probability in executing the pouring motion. The colorbar depicts the colors associated with different probabilities of success.

Formally, for a new position of the cup $\boldsymbol{\theta}$, the execution success probability is modeled as

$$P_\theta(S \mid I, x) = e^{-\xi\|\boldsymbol{\theta}-\bar{\boldsymbol{\theta}}\|_2^2}$$
$$P_\theta(F \mid I, x) = 1 - P_\theta(S \mid I, x),$$

where $\xi = 22.017$ is the width of the kernel, and was estimated empirically.

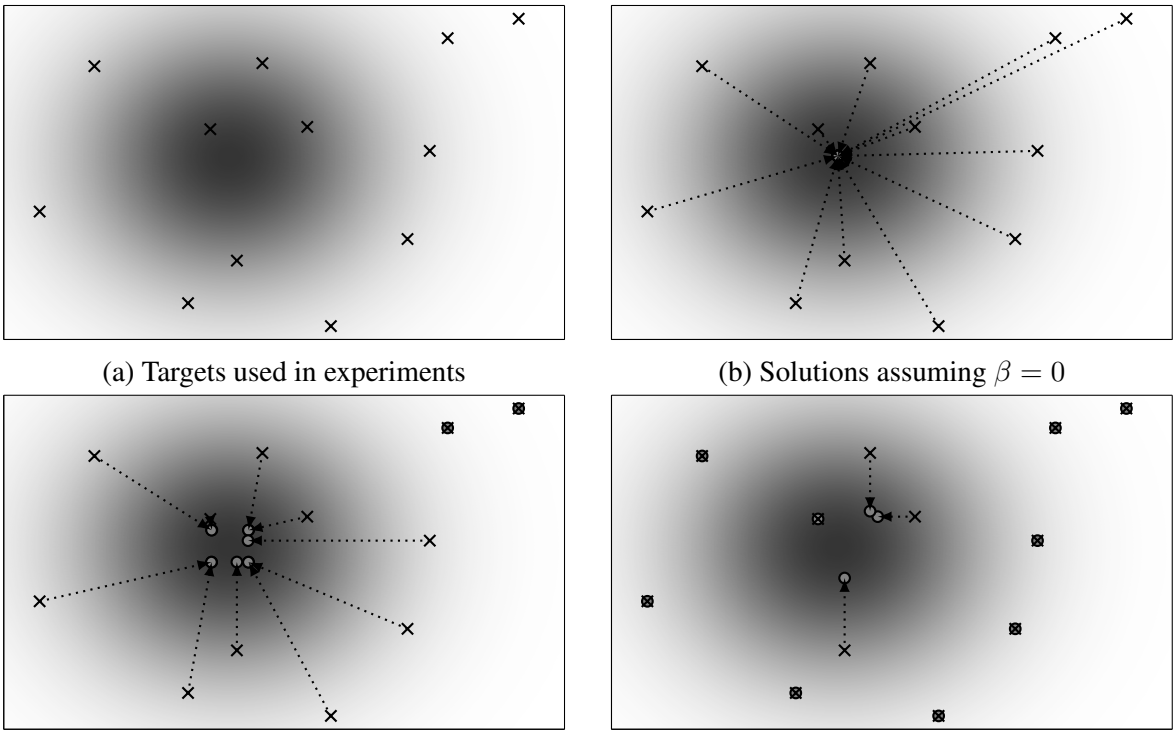
In this scenario we allow the agent to reason over changes to the target location of the cup. We assume that changes to the location of the cup follow a cost function that penalizes large displacements of the cup

$$C(\boldsymbol{\theta}) = \beta\|\boldsymbol{\theta} - \boldsymbol{\theta}_0\|_1, \tag{4.19}$$

for constant β and initial cup position $\boldsymbol{\theta}_0$. This cost can be interpreted, for example, as the burden of bothering the human user to move his cup.

Results We ran experiments starting from the 12 different initial configurations $\boldsymbol{\theta}_0$, depicted as crosses in Figure 4.9(a). These 12 configurations of the world correspond to different initial locations of the cup, where the robot is tasked to pour water. The average value in these original world configurations, $J(\boldsymbol{\theta}_0)$, was approximately -4.57. By the definition of the reward function, this suggests that the agent typically executed the "quit" action when faced with these initial locations of the cup.

We then tested our algorithm assuming different values of cost parameter β. Assuming $\beta = 0$, *i.e.* no cost in changing the target cup position, P-ITERATION always returned the solution where



(a) Targets used in experiments

(b) Solutions assuming $\beta = 0$

(c) Solutions assuming $\beta = 10$

(d) Solutions assuming $\beta = 20$

Figure 4.9: Performance in the water pouring scenario. (a) The 12 targets used in the experiments. (b) to (d) The solutions computed for different values of β.

the cup is to be moved to the center of the radial kernel, thus guaranteeing a successful execution, and an average $J(\boldsymbol{\theta}) = -1$. Intuitively, these solutions make sense, because if there is no cost in changing the world, one might as well request the best possible version of the world—in this case, having the cup positioned at the location where the robot is guaranteed to succeed in the task. The solutions returned are depicted in Figure 4.9(b).

For increasing values of β, *i.e.* increasing penalties for moving the cup, P-ITERATION computed solutions that maximized the tradeoff between the increase in success likelihood, and the cost for changing the world. From Table 4.4 we observe that the solutions returned under different costs were always better than the original world configuration $\boldsymbol{\theta}_0$. These values suggest that the solutions computed tend to request the cup to be moved closer to the center of the radial kernel, where there is a higher chance of the robot succeeding in pouring water. This is supported by Figures 4.9(b) to 4.9(d), which depict the changes to the world requested for different values of β. For example, assuming $\beta = 10$, as depicted in Figure 4.9(c), we observe there were two target positions for which no change to the world was requested. This is because it would be too costly to move those cups to a new configuration of the world where it is worth trying to execute the pouring action.

This scenario is interesting as it depicts the applicability of Counterfactual MDPs in Human-Robot Interaction scenarios. In this class of scenarios, the solution returned by the Counterfactual MDP can be interpreted quite naturally as a request that is posed to the human user. In the

Table 4.4: The performance of P-ITERATION in terms of the quality of solutions, in the WATER POURING scenario. The results are rounded to 2 decimal places, and averaged over 30 runs.

WATER POURING				
$J(\theta_0)$	$F(\theta)$			
	$\beta = 0$	$\beta = 5$	$\beta = 10$	$\beta = 20$
-4.57	-1.00	-2.55	-3.73	-4.37

specific case of this WATER POURING scenario, the agent is allowed to reason over different target locations of the cup. Even though the robot cannot move the cup itself, the cup can be indirectly moved by reaching out to the nearby human user. The goal of the agent thus became that of determining the configuration of the world that maximizes the tradeoff between the increase in success likelihood, and the burden of the request posed to the user.

4.4.3 Scaling up

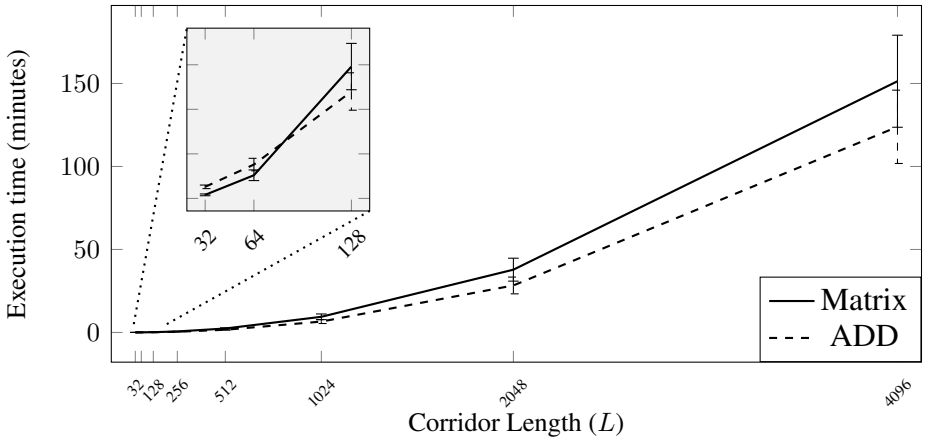
In Section 4.3 we discussed mechanisms for speeding up P-ITERATION, by reducing the computational effort required in the MDP solving step (line 7 of the algorithm). We now assess the applicability and performance of these mechanisms.

Different Representations

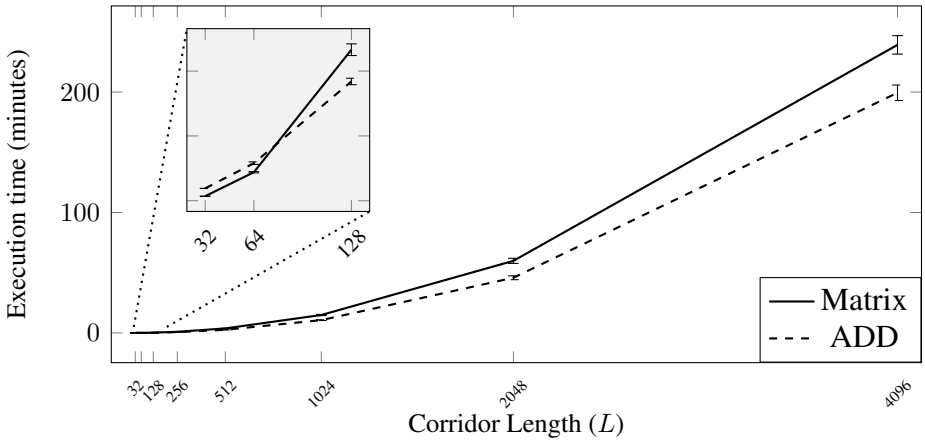
We start by analyzing the first method proposed, which regarded the adoption of different MDP representations. For these experiments, we consider the CORRIDOR scenario as it can be easily parameterized in terms of size, facilitating an analysis on the scalability of the methods. Moreover, we assume an initial state distribution $\mu_0(I) = 1$, *i.e.*, similar to that used in the motivating scenario, and a cost function that sums the smooth function defined in (4.17) over all parameters θ_k and parameterized by $\beta = 10$.

One of the mechanisms we considered for speeding up our algorithms regarded the adoption of a factored representation of MDPs, based on Algebraic Decision Diagrams. We discussed the benefits of this representation in solving MDPs with large state and action spaces, but also the challenges in adopting them in the context of Counterfactual MDPs—namely, the challenges in the computation of the gradients, due to the need to compute some matrix inverses. In order to overcome these challenges we proposed an alternative method for computing the gradients based on Fixed Point Iteration, and discussed that in adopting the ADD representation we would be trading off the benefits of solving MDPs faster, with possibly costlier gradient computations. Our results suggest that the this tradeoff is worth it, and that adopting an ADD representation can be useful in solving larger problems.

Figures 4.10(a) and 4.10(b) compare the performance of the P-ITERATION algorithm under two MDP representations, sparse matrices and ADDs, for different problem sizes and number of parameters. The results for the ADD representation, in particular, assume the method that computes the gradient $\nabla_\theta J(\theta)$ with a single FPI. From the figures, two observations ensue.



(a) $K = 1$

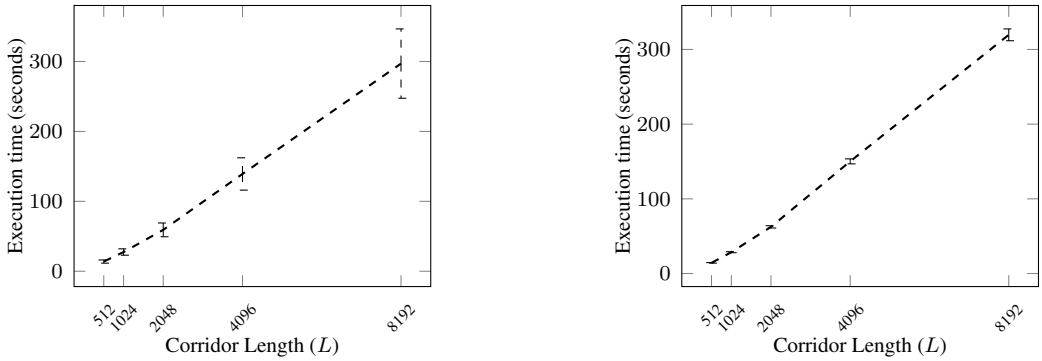


(b) $K = 5$

Figure 4.10: Comparison of the performance of P-ITERATION when adopting a matrix representation *vs.* a factored representation based on ADDs. The performance is measured in the CORRIDOR scenario for increasing lengths. Figures (a) and (b) assume $K = 1$ and $K = 5$, respectively. Results are averaged over 5 runs.

First, we observe that for smaller problems (up to 64 states), the sparse matrix representation actually seems to outperform the ADD representation. On the other hand, we also observe that as the problem size increases the ADD representation starts to consistently outperform the sparse matrix representation. In fact, the figures suggest that the gap in performance increases with the problem size as well. These results are plausible, since in smaller problems the speeds-ups achieved in solving the MDPs may not compensate for the computational overheads associated in building the ADD representation. However, as the problem size increases and solving the MDPs becomes the bottleneck, then the ADD representation becomes the better choice. In sum, we observe the tradeoff anticipated previously in Section 4.3.1.

A second mechanism discussed for scaling up our algorithms to larger problems involved exploiting the sparse structure of many planning problems in the construction of the ADD data structures. In particular, we discussed how the overheads associated with building the ADD data



(a) $K = 1$ (b) $K = 5$

Figure 4.11: Performance of P-ITERATION using a sparse factored MDP representation. Measured in the CORRIDOR scenario for increasing lengths. Figures (a) and (b) assume $K = 1$ and $K = 5$, respectively. Results are averaged over 5 runs.

structures could potentially be alleviated by adopting data structures that allow the iteration over non-zero entries of the transition probabilities (in contrast with iterating over all (x, a, y) triplets). We empirically assessed the performance of this mechanism in the CORRIDOR scenario, applying it both in the computation of the gradients and in the MDP solving step. Figures 4.11(a) and 4.11(b) depict the performance for different problem sizes and number of parameters. These figures suggest a great improvement in performance when exploiting the sparsity of the problem. Specifically, when compared to the performance achieved previously in Figures 4.10(a) and 4.10(b), where the sparsity was not exploited. We note, however, that this large improvement may result from the intrinsic sparse nature of the CORRIDOR scenario's transition probabilities. These result are still relevant, since many problems show similar sparsity in the transition probabilities.

All experiments assume $N = 5$ random restarts, learning rate $\alpha = 0.05$ and discount factor $\gamma = 0.9$. Results were averaged over 5 runs. We adopted the value iteration method for solving MDPs in both representations. For the ADD representation, in particular, we followed a version previously introduced in the literature [22]. The fixed point iteration methods used in solving MDPs and in computing the gradients, adopted a termination condition following an l_∞ norm with 10^{-3} and 10^{-6} thresholds, respectively. Our implementation using the ADD representation made extensive use of the CUDD C++ library [67]. All experiments were performed on a ThinkPad W550s laptop, with 16GB of RAM and an Intel i7-5600U CPU at 2.60GHz.

Seeding the MDP Solver

In Section 4.3.2 we discussed a second method for speeding up P-ITERATION. This method aims at exploiting the iterative nature of P-ITERATION, in order to reduce the computational effort required in solving the MDP at a given step t. In particular, by reusing information computed in the previous iteration $t - 1$.

We now assess the impact of adopting this mechanism. We assume that the MDP solving step follows the value iteration (VI) method. As such, we reuse information computed in the previous

Table 4.5: Comparison of the number of iterations performed by value iteration in solving the MDP, when initializing the value estimate to zero (No Seed), or with the optimal value function computed in the previous step (Seed). The results are averaged over 10 randomly seeded runs.

	Size	Number of VI Iterations	
		No seed	Seed
CORRIDOR	$L = 10$	22.14 ± 0.08	17.22 ± 0.06
	$L = 20$	32.53 ± 0.09	27.39 ± 0.07
	$L = 50$	66.63 ± 0.13	59.48 ± 0.11
FROZEN LAKE	4×4	1215 ± 0.00	80.14 ± 1.09
	8×8	1215 ± 0.00	149.50 ± 0.97

step, by seeding the computation of the optimal value $\boldsymbol{v}^*_{\theta_t}$ with the optimal value computed in the previous iteration. We refer to Section 4.3.2 for more details, but essentially, we let $\boldsymbol{v}^{(0)}_{\theta_t} = \boldsymbol{v}^*_{\theta_{t-1}}$, where $\boldsymbol{v}^{(0)}_{\theta_t}$ is the initial estimate of the optimal value function in world configuration θ_t.

For these experiments we consider the CORRIDOR and FROZEN LAKE scenarios. As we will see, in these domains the effectiveness of the seeding approach varies, which allows for insightful discussions. The scenarios follow the costs and parameterizations described previously, with the CORRIDOR assuming one parameter $K = 1$. All experiments adopt a discount factor $\gamma = 0.99$, learning rate $\alpha = 0.001$, and termination condition threshold

$$\left|\boldsymbol{v}^*_{\theta_{t+1}} - \boldsymbol{v}^*_{\theta_t}\right|_\infty < \epsilon \frac{1-\gamma}{2\gamma},$$

with $\epsilon = 10^{-3}$. These hyperparameters follow the discussion at the end of Section 4.3.2, where we discussed why the seeding technique should provide more significant performance improvements when adopting smaller learning rates and larger discount factors.

Table 4.5 describes the average number of iterations performed by value iteration (VI) in the MDP solving step of P-ITERATION (line 7), for different scenarios and problem sizes. We consider, in particular, two initialization methods for value iteration. The first method (No seed) initializes the optimal value estimate to zero. The second method (Seed) initializes the optimal value estimate to the optimal value computed in the previous step. The results are averaged over 10 randomly seeded runs. From the table, several observations are in order.

First, we observe that seeding the value iteration method, on average, reduces the number of iterations required in the MDP solving step of the algorithm. This leads to a better performance of P-ITERATION as the overhead associated in the seeding procedure is negligible.

The second observation is that the performance improvements in the FROZEN LAKE scenario were superior to those achieved in the CORRIDOR scenario. By inspecting the consecutive iterates $\boldsymbol{v}^{(t)}$ in the MDP solving of the CORRIDOR, we conclude this results from the "linear" shape of the scenario, and from the way the values are updated in VI. As discussed before, the best solution

in this scenario is to fully open the door, and as such, in general, each gradient step will open the door a bit more, *i.e.*, increase θ. By increasing θ, we observe an immediate increase of the expected value of the initial state (top left), in the first iteration of VI. This is because there is now a larger probability of the agent moving directly to the goal state by moving down. Then, in the second iteration of VI, the expected values of the first and second states are updated (the value of the first state is updated as now the value of remaining in the same state increased). VI proceeds by updating the values of the states in the top row in this linear fashion, thus requiring more iterations the longer the corridor. The updated values are propagated faster in the FROZEN LAKE scenario, since the states are more connected.

Finally, we observe that the number of iterations of the non-seeded VI in the FROZEN LAKE scenario is the same for both problem sizes. This occurs because after $T = 1215$ steps the VI updates become smaller than the termination threshold specified above. This follows from (2.6).

4.4.4 Comparing against baseline

We conclude the experimental evaluation by comparing the performance of the P-ITERATION algorithm against a baseline approach. In particular, we consider a baseline that first discretizes the space of possible configurations of the world Θ, and then performs an exhaustive search for the best solution (also known as grid search). Due to the discretization of the parameter space, this baseline approach may not always arrive at the optimal solution. This is specially true in scenarios with complex optimal solutions that are not at the boundaries of the parameter space—for example, the FROZEN LAKE and WATER POURING scenarios we considered before. In any case, it is interesting to compare its execution time performance against P-ITERATION.

For this purpose we consider again the CORRIDOR scenario and adopt the same hyperparameters used in the previous experiment, including the space of possible configurations of the world $\Theta = [0, 1]^K$. We perform the experiments for increasing lengths L of the corridor, and increasing number of parameters K (doors that can be opened). Additionally, we evaluate the P-ITERATION algorithm under different random restarts $N \in \{10, 20, 40\}$. The baseline approach is evaluated assuming different discretization steps $h = \{10^{-1}, 10^{-2}, 10^{-3}\}$. For each parameter, and given the space of possible configurations of the world Θ considered, a step h leads to $r = h^{-1} + 1$ *discretization candidates* $\{0, h, 2h, \ldots, 1\}$. As a result, the baseline approach works by solving r^K MDPs. This suggests that the execution time performance of the baseline approach will quickly deteriorate as we increase the candidates and the number of parameters used.

Table 4.6 compares the execution time performance of P-ITERATION against the baseline approach. The results of the baseline approach are averaged over 3 runs, whereas the results of the P-ITERATION approach are averaged over 30 randomly seeded runs. We perform more runs with P-ITERATION to account for the random restarts used. Multiple observations ensue from the table. First, we observe that, as expected, as the problem size increases (length L and parameters K), both approaches take longer to solve the problem. We observe a similar effect as we decrease the step size h in the baseline approach, and as we increase the number of random restarts N in the P-ITERATION algorithm. Secondly, we observe that as we decrease the step size, the performance of the baseline approach deteriorates faster, when compared to that of P-ITERATION with the increase in number of random restarts. Finally, the results suggest that the execution time performance of the baseline approach is only competitive with P-ITERATION

Table 4.6: Comparison of the execution time performance (in seconds) of P-ITERATION against the baseline method, in the CORRIDOR scenario, under different lengths L and number of parameters K. The baseline approach results are averaged over 3 runs. P-ITERATION results are averaged over 30 randomly seeded runs. All experiments ran with a 90 minutes time limit, after which the process was stopped. The experiments that surpassed this limit are denoted by a dash.

Execution time performance (seconds) in the CORRIDOR scenario

		Baseline			P-ITERATION		
L	K	$h = 10^{-1}$	$h = 10^{-2}$	$h = 10^{-3}$	$N = 10$	$N = 20$	$N = 40$
10	1	0.01 ± 0.00	0.14 ± 0.00	0.71 ± 0.05	0.06 ± 0.00	0.12 ± 0.00	0.24 ± 0.00
	2	0.05 ± 0.00	5.23 ± 0.43	452.75 ± 6.13	0.05 ± 0.00	0.11 ± 0.01	0.24 ± 0.01
	3	0.40 ± 0.01	438.88 ± 3.58	—	0.10 ± 0.02	0.18 ± 0.02	0.39 ± 0.03
20	1	0.02 ± 0.00	0.27 ± 0.02	2.78 ± 0.19	0.27 ± 0.01	0.50 ± 0.01	1.03 ± 0.02
	2	0.17 ± 0.00	18.18 ± 0.01	1823.32 ± 0.92	0.21 ± 0.01	0.45 ± 0.04	1.02 ± 0.07
	3	1.57 ± 0.00	1650.39 ± 1.83	—	0.41 ± 0.07	0.76 ± 0.09	1.63 ± 0.03
30	1	0.06 ± 0.00	0.64 ± 0.04	6.81 ± 0.54	0.68 ± 0.03	1.28 ± 0.04	2.63 ± 0.05
	2	0.43 ± 0.00	44.81 ± 0.05	4480.15 ± 4.43	0.55 ± 0.03	1.18 ± 0.11	2.67 ± 0.18
	3	3.92 ± 0.00	4062.92 ± 5.59	—	1.08 ± 0.20	2.02 ± 0.25	4.33 ± 0.35

when assuming, either the largest step size $h = 10^{-1}$ or a single parameter $K = 1$. In fact, we observe that the execution time of the baseline approach quickly becomes several orders of magnitude larger than that of P-ITERATION.

One may now wonder how both approaches compare in terms of the quality of the solutions computed. As discussed before, in the particular case of our CORRIDOR scenario, the optimal world configuration corresponds to that where the first door is fully open, $\theta_1 = 1$, and all other doors are closed $\theta_k = 0$. Given our approach for discretizing the parameter space, in this case, we have that the baseline approach is always able to find this optimal solution. The quality of the solutions computed by P-ITERATION is reported in Table 4.7, with the results averaged over 30 randomly seeded runs. From the table we conclude that P-ITERATION typically computes the optimal solution, or a solution very close to the optimal one. Moreover, we observe that the more random restarts we allow the algorithm to perform, on average, the better the solutions returned.

In sum, the results suggest that P-ITERATION outperforms the baseline approach. In terms of execution time performance, the baseline approach is only competitive under very low resolutions (large step sizes for the grid search), or when assuming small number of parameters. In fact, our results show that as we increase the resolution and the number of parameters, P-ITERATION can be several orders of magnitude faster than the baseline approach. Additionally, we observed that the better execution time performance does not come at the expense of the quality of the solutions computed. In fact, in the scenarios considered, P-ITERATION was typically able to compute, either the optimal solution, or a solution very close to the optimal one.

Table 4.7: The quality of the solutions computed by P-ITERATION in the CORRIDOR scenario, assuming different corridor lengths L and number of parameters K. The results are averaged over 30 randomly seeded runs. Note that for a given length L the optimal solution is independent of the number of parameters K, since the optimal solution consists in the world configuration where the first door is open and all others are closed.

Quality of solutions computed in the CORRIDOR scenario

		Optimal	P-ITERATION		
L	K	Value	$N = 10$	$N = 20$	$N = 40$
10	1	-3.862	-3.862 ± 0.000	-3.862 ± 0.000	-3.862 ± 0.000
	2		-3.867 ± 0.003	-3.862 ± 0.000	-3.862 ± 0.000
	3		-3.881 ± 0.004	-3.872 ± 0.004	-3.864 ± 0.002
20	1	-5.852	-5.852 ± 0.000	-5.852 ± 0.000	-5.852 ± 0.000
	2		-5.855 ± 0.001	-5.852 ± 0.000	-5.852 ± 0.000
	3		-5.862 ± 0.002	-5.857 ± 0.002	-5.853 ± 0.001
30	1	-6.984	-6.984 ± 0.000	-6.984 ± 0.000	-6.984 ± 0.000
	2		-6.986 ± 0.001	-6.984 ± 0.000	-6.984 ± 0.000
	3		-6.990 ± 0.001	-6.988 ± 0.001	-6.985 ± 0.001

4.5 Summary of the Chapter

This chapter contributed an iterative gradient-based approach for solving Counterfactual MDPs. This approach builds upon the computation of the gradient of the objective function F with respect to the transition probabilities. We proposed two methods for computing these gradients. In the first method the gradients are formally derived from the KKT conditions of the optimization problem. In the second method the gradients are computed more efficiently by exploiting the linear programming structure of MDPs. The computation of these gradients is non-trivial, due to the intricate dependencies of the objective function F on the transition probabilities P_θ. Specifically, the computation of F depends both on the transition probabilities P_θ and on the optimal policy π_{P_θ}. However, π_{P_θ} also depends on P_θ, and this dependency is non-differentiable, as seen in (2.4). The key idea behind the more efficient method for computing the gradient builds upon the observation that, in general, for small changes in the transition probabilities, the optimal policy does not to change. As such, we can approximate the gradient at a given transition probability P_θ by first computing its optimal policy $\pi^*_{P_\theta}$, and then, when computing the gradient, ignore the dependency of $\pi^*_{P_\theta}$ on P_θ.

Having a method for computing the gradient allowed us to introduce P-ITERATION, an iterative gradient-based algorithm for solving Counterfactual MDPs. This algorithm iteratively builds a sequence of world configurations that converges to a local maximum of the objective

function F, by following the gradient steps. In order to tackle the non-convexity of the problem, the algorithm is restarted M times, starting at different initial configurations.

In this chapter we also discussed the performance of the algorithm, and observed that one of the bottlenecks regards the need for solving multiple MDPs during the gradient iterations. We discussed different mechanisms for speeding up the algorithm by alleviating the computational effort in solving the intermediate MDPs. In particular, we discussed the adoption of a different MDP representation, and a seeding mechanism for the MDP solving process that reuses information computed in the previous iteration. Even though not considered in this thesis, it could also be interesting to explore other methods commonly used on the motion and classical planning literature. For example, *model switching* [70] or *variable-resolution* [42, 83] planning techniques, where the planner may adjust the level of detail used, by considering abstract representations of the state that hide some information. In the context of Counterfactual MDPs, this concept of variable-resolution could also be explored by using abstract state representations, but also by considering smaller spaces of possible configurations of the world. While these techniques may allow for a faster planning, it will necessarily come at the expense of the quality of the solutions returned.

Finally, the chapter concluded with an extensive evaluation of Counterfactual MDPs and the algorithms proposed for solving them. The scenarios considered in the experimental evaluation were selected in order to allow an evaluation from different perspectives. The results show the applicability of Counterfactual MDPs on multiple different scenarios, modeled using different cost functions and parameterizations of the transition probabilities. Additionally, the results also show the performance of the algorithms developed in scenarios easily parameterized in terms of problem size and number of parameters. Most importantly, however, the experimental evaluation performed demonstrates that Counterfactual MDPs can effectively allow agents to reason over changes to the world, and to plan over the possibility of operating in such modified worlds, where new and possibly more rewarding optimal policies are available.

Chapter 5

Stochastic Outcomes Counterfactual MDPs

In Chapter 3 we introduced Counterfactual MDPs, a model that allows the agent to reason, plan and act over the counterfactual *What if the world were different?* This model effectively allows the agent to reason over possible configurations of the world, and plan over the possibility of actually operating in such configurations. We assume that shifting the original world configuration to these alternative worlds comes at a cost. As such, the agent plans over the value/cost trade-off of operating in the alternative worlds, only considering them when beneficial. The resulting model, however, makes the implicit assumption that the world configuration envisioned by the agent can be satisfied. However, in many real-world scenarios there exists an underlying uncertainty in changes to the world.

This Chapter tackles the problem of uncertain outcomes in the context of Counterfactual MDPs, introducing a new model that allows the agent to reason over the possibility that the world configuration yielded after its request may not be the one expected. We dub the resulting model as *Stochastic Outcomes* Counterfactual MDPs. Upon introducing the model, we discuss connections between standard and Stochastic Outcomes Counterfactual MDPs. In particular, we show that Counterfactual MDPs can actually be reduced to Stochastic Outcomes Counterfactual MDPs. As such, the complexity analysis introduced for the former model, can be extended to the latter, allowing us to conclude that solving the general form of Stochastic Outcomes Counterfactual MDPs is also hard. This motivates the proposal of a gradient-based approach. In the process of introducing the gradient method, we discover the interesting insight that the gradient of the new optimization problem now takes the form of an expectation. We propose a sampling based method for estimating this new gradient and dub the new algorithm STOCHASTIC OUTCOMES P-ITERATION. We conclude with an evaluation of the models and algorithms proposed. In particular, this evaluation demonstrates the improvements on the performance of the agent that arise when we consider the aforementioned uncertainty in the outcomes of changes to the world.

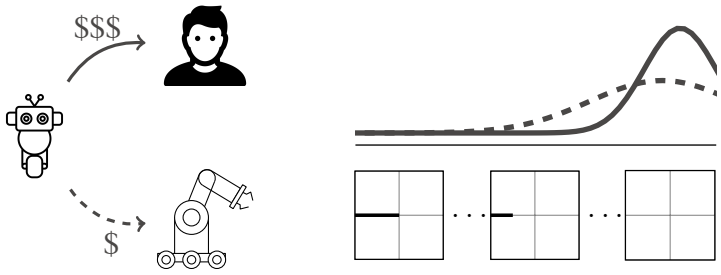


Figure 5.1: The agent reasons over possible configurations of the corridor, which can only be achieved through some external intervention. The external intervention may take the form of assistance in opening the door, provided by either a human user or a mobile manipulator nearby. Tasking the mobile manipulator with opening the door is less costly than disturbing the user. However, it comes at the expense of a higher uncertainty in the outcome of the changes to the world.

5.1 Model Formulation

The original Counterfactual MDP model allows the agent to reason over alternative configurations of the world, and to plan over the possibility of actually operating in such configurations. Shifting the original world configuration to these alternative worlds will generally involve a cost. This model makes the implicit assumption that the world configuration envisioned by the agent is necessarily satisfied. In many real-world scenarios, however, there exists an underlying uncertainty in the process of modifying the world.

Figure 5.1 depicts an example of this uncertainty in the CORRIDOR scenario. In this example our service robot reasons over alternative possible configurations of the world where the doors of the corridor may be open. Since the robot does not have arms it can use to open the doors, achieving such configurations requires necessarily some form of external intervention. In this example, let us assume the external intervention is provided by either a mobile manipulator robot, or by a human user nearby. The human user should be able to open the door as expected, and make sure our robot can go through. However, this external intervention comes at the high cost of disturbing the user. Tasking the mobile manipulator robot with opening the door may be less costly. However, this is at the expense of some uncertainty in the outcome of the changes to the world, as there is a chance that this mobile manipulator may sometimes fail to open the door.

We model the uncertainty in shifting the original world configuration θ_0 to a new configuration θ as a generic probability distribution f. Formally, we let $f(\theta' \mid \theta, w)$ denote the probability of a request for a world configuration θ resulting in a different configuration θ'. $w \in \mathcal{W}$ is an additional parameter the agent controls, which can tune different properties of the distribution—for instance, the dispersion. In practice, this parameter w can be interpreted as the type of external intervention solicited. This allows us to model the uncertainty associated with different types of external intervention—recall the example in Figure 5.1 where we assumed the external intervention consisted on assistance by either a manipulator robot or a nearby human user. A continuous set $\mathcal{W}$ allows us to further model complex combinations of external intervention—for example, a combined external intervention from the human and the manipulator robot at the same time.

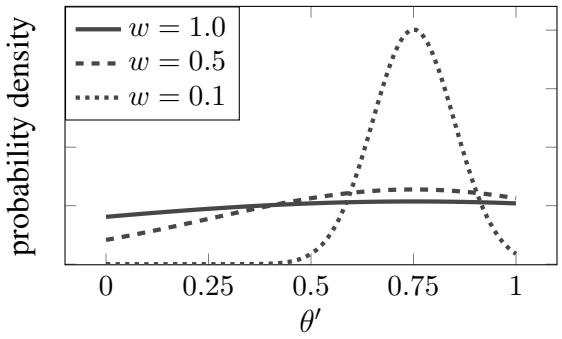


Figure 5.2: Probability density function of the truncated normal distribution centered at request $\theta = 0.75$ and truncated to $[0, 1]$, assuming different dispersion parameters w. Illustrates the probability of outcomes θ' under different requests θ and dispersion parameters w.

Similarly to Counterfactual MDPs, we assume $J(\theta)$ denotes the value associated with a world configuration θ. However, we now need to allow the agent to reason over the stochastic outcomes of an external intervention that changes the world. Formally, we let $J_{\mathbb{E}}(\theta, w)$ denote the expected value resulting from requesting a world configuration θ through an external intervention characterized by w:

$$J_{\mathbb{E}}(\theta, w) = \mathbb{E}_{\theta' \sim f(\cdot|\theta,w)} \left[J(\theta') \right].$$

Additionally, we let $C(\theta, w)$ denote the cost associated with the external intervention for shifting the original world configuration θ_0 to configuration θ, when such intervention is characterized by w. Note we assume the cost regards the world configuration requested, θ, and not necessarily the one resulting from the stochastic outcomes.

The problem of determining the best world configuration, under stochastic outcomes modeled by f, can be formalized as the optimization problem

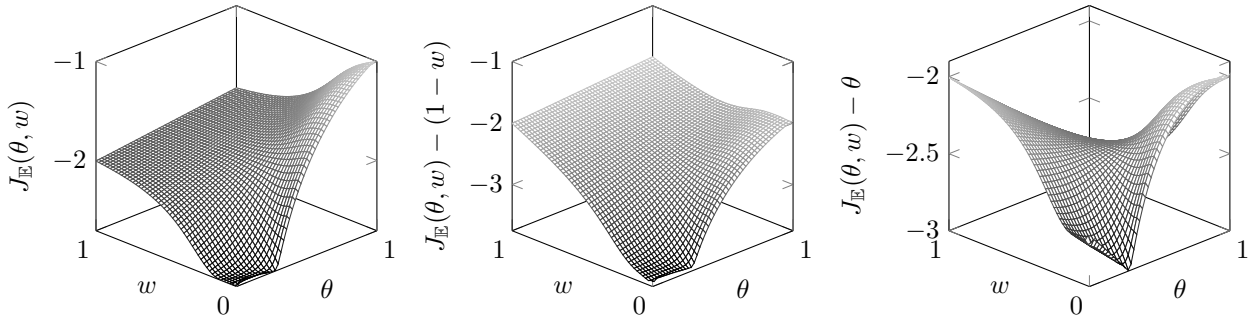
$$\begin{aligned} \max_{\theta, w} \quad & F(\theta, w) = J_{\mathbb{E}}(\theta, w) - C(\theta, w) \\ \text{s.t.} \quad & \theta \in \Theta \\ & w \in \mathcal{W} \end{aligned} \quad , \tag{5.1}$$

We dub this problem as *Stochastic Outcomes* Counterfactual MDPs.

Let us now take a moment to appreciate how the CORRIDOR scenario depicted in Figure 5.1 can be formalized as above.

5.1.1 Application to CORRIDOR scenario

Let us start by considering the uncertainty in the outcome of trying to shift the original world to a new configuration $\theta \in \Theta$. We let $f(\theta' \mid \theta, w)$ follow a truncated normal distribution centered at the configuration θ, with standard deviation controlled by parameter w, and truncated to the interval $[0, 1]$. The truncated normal ensures a peak probability at the requested configuration θ, with some probability of the outcome either under- or over-shooting the desired openness of the door. This is depicted in Figure 5.2. More importantly, the outcomes are guaranteed to remain in the set of valid world configurations $\Theta = [0, 1]$.



(a) $K_\theta = K_w = 0$ (b) $K_w = 1$ (c) $K_\theta = 1$

Figure 5.3: Objective function for $\gamma = 0.9$, under different requests for world configurations θ and dispersion parameters w. (a) Plot assuming no cost function. (b) Plot assuming cost on dispersion parameter. (c) Plot assuming cost function on world configurations.

We assume the external intervention may consist on assistance provided by either the human user or the nearby manipulator robot. For simplicity, we let $\mathcal{W} = [0, 1]$[1], where $w = 0$ and $w = 1$ denote external interventions that change the world with no/maximum uncertainty, respectively. Figure 5.2 plots the truncated normal distribution centered at a requested world configuration $\theta = 0.75$, for external interventions with different levels of precision. In this case, small/large values of w correspond to a narrower/broader outcome uncertainty distribution, which translates in the agent posing the request to a more/less precise external intervention, respectively.

Finally, we assume a cost function

$$C(\theta, w) = K_\theta \theta + K_w(1 - w),$$

which linearly penalizes changes to the world and uncertainty parameters, and is parameterized by two constants K_θ and K_w. The optimization problem in (5.1) instantiates in

$$\max_{\theta \in [0,1], w \in [0,1]} F(\theta, w) = J_{\mathbb{E}}(\theta, w) - C(\theta, w).$$

Figure 5.3 plots the objective function $F(\theta, w)$ as a function of requests θ and parameters w, in three different scenarios. First, Figure 5.3(a) depicts the objective function assuming no costs in changes to the world (*i.e.*, $K_\theta = K_w = 0$). In this case, as expected, the maximum value is attained when requesting an external intervention that fully opens the door ($\theta = 1$) with no uncertainty in the outcomes ($w = 0$). Perhaps surprising, is the observation that when requesting world configurations associated with smaller values of θ, it is actually better to request an imprecise external intervention (larger w). This hints at the possibility of non-trivial solutions when considering costs for specific world configurations and/or uncertainty parameters. In fact, this is also supported by the two remaining figures. Figure 5.3(b) depicts the objective function under a cost parameter $K_w = 1$, *i.e.*, requests for precise external interventions are penalized. Interestingly, in this case the objective function is maximized by requesting $(\theta = 1, w = 1)$. That

[1]We introduce a closed set $\mathcal{W}$ for simplicity. However, the truncated distribution is not defined for null standard deviations. Formally, one should consider the left-open set $\mathcal{W} =]0, 1]$ instead.

is, by requesting an imprecise external intervention to fully open the door. Finally, Figure 5.3(c) depicts the objective function under a cost for requesting changes to the world, assuming $K_\theta = 1$. In this case, the maximum value is attained once again by requesting a precise external intervention to fully open door. However, we observe that the value at $(\theta = 0, w = 1)$ is very close. Arguably, this last example may have an artificial taste, as it seems to suggest that requesting an imprecise external intervention to keep the door closed may lead to the door getting open sometimes. This results from the simple formulation we adopted in this example for the dispersion parameter. Note that a maximum dispersion $w = 1$ lets our uncertainty distribution approach a uniform distribution (see Figure 5.2). In Section 5.3 we discuss a formulation that prevents these artificial solutions, rendering more realistic settings.

5.1.2 Complexity

In Section 3.2 we discussed the complexity of Counterfactual MDPs. In particular, we showed that solving the general form of the associated optimization problem (3.6) is hard both in theory and in practice. We now develop a similar analysis for the case of Stochastic Outcomes Counterfactual MDPs.

Let us start with a complexity analysis from a computational perspective. The analysis starts with the observation that Counterfactual MDPs can be reduced to Stochastic Outcomes Counterfactual MDPs. Given a Counterfactual MDP we can create a Stochastic Outcomes Counterfactual MDP where the uncertainty distribution f takes the form of a Dirac delta centered in the requested configuration θ—*i.e.*, we take an uncertainty distribution where the request θ has probability 1. As such, an efficient algorithm for solving Stochastic Outcomes Counterfactual MDPs would necessarily be able to efficiently solve standard Counterfactual MDPs. Since Counterfactual MDPs have been proved to be NP-Hard, we conclude we should not expect an efficient algorithm that computes the exact solution for Stochastic Outcomes Counterfactual MDPs in its general form.

From a more practical perspective, it follows from Figure 5.3 that the objective function of the optimization problem associated with Stochastic Outcomes Counterfactual MDPs (5.1) is non-convex in both θ and w. In practice, this also renders the optimization problem in (5.1) hard to solve.

5.2 Approaches for Stochastic Outcomes Counterfactual MDPs

In this section we develop approaches for solving Stochastic Outcomes Counterfactual MDPs. We follow an approach similar to that proposed for Counterfactual MDPs, and consider a gradient-based method for solving the optimization problem in (5.1). Our approach builds upon the gradient of the objective function with respect to the world configuration θ and parameter w:

$$\nabla_{\theta,w} F(\theta, w) \; = \nabla_{\theta,w} \Big[\mathbb{E}_{\theta' \sim f(\cdot|\theta,w)} \left[J(\theta') \right] \Big] - \nabla_{\theta,w} \, C(\theta, w). \tag{5.2}$$

We focus on the gradient of the expected values over the stochastic outcomes, since the cost function is task dependent. We assume, however, that the dependence of C on θ and w is differentiable and easy to compute.

5.2.1 Gradient of Expected Stochastic Outcomes

We start by observing that, for a given θ and w

$$\mathbb{E}_{\theta' \sim f(\cdot|\theta,w)} \left[J(\theta') \right] = \int J(\theta') f(\theta' \mid \theta, w) \, \mathrm{d}\theta'. \tag{5.3}$$

As a result, the gradient follows

$$\begin{aligned}
&\nabla_{\theta,w} \, \mathbb{E}_{\theta' \sim f(\cdot|\theta,w)} \Big[J(\theta') \Big] \\
&\quad = \nabla_{\theta,w} \int J(\theta') f(\theta' \mid \theta, w) \, \mathrm{d}\theta' \\
&\quad = \int J(\theta') \nabla_{\theta,w} f(\theta' \mid \theta, w) \, \mathrm{d}\theta' \\
&\quad = \int J(\theta') \frac{\nabla_{\theta,w} f(\theta' \mid \theta, w)}{f(\theta' \mid \theta, w)} f(\theta' \mid \theta, w) \, \mathrm{d}\theta' \\
&\quad = \int J(\theta') \nabla_{\theta,w} \Big[\log f(\theta' \mid \theta, w) \Big] f(\theta' \mid \theta, w) \, \mathrm{d}\theta' \\
&\quad = \mathbb{E}_{\theta' \sim f(\cdot|\theta,w)} \Big[J(\theta') \nabla_{\theta,w} \log f(\theta' \mid \theta, w) \Big],
\end{aligned} \tag{5.4}$$

where in the second step we use the general form of the Leibniz integral rule to move the gradient operator inside the integral, and in the third and fourth steps we use the $\log$ derivative trick.

We conclude that the exact solution to the gradient of the expected stochastic outcomes can be computed by solving an expectation. Solving this expectation quickly becomes intractable as the dimensionality of the configuration space, Θ and $\mathcal{W}$, increases. We propose instead to approximate this gradient through sampling mechanisms. Different sampling methods can be used in estimating this gradient.

For a desired world configuration θ and dispersion w, we can estimate the gradient in (5.4) by first sampling M outcomes $\{\theta'_m, \; m = 1, \ldots, M\}$ from $f(\, \cdot \mid \theta, w)$, and then following

$$\nabla_{\theta,w} \, \mathbb{E}_{\theta' \sim f(\cdot|\theta,w)} \Big[J(\theta') \Big] \approx \frac{1}{M} \sum_{m=1}^{M} J(\theta'_m) \nabla_{\theta,w} \log f(\theta'_m \mid \theta, w). \tag{5.5}$$

The quality of the approximation in (5.5) is related with the number of samples M used. In general, larger values of M should yield better approximations.

Importance Sampling

One disadvantage of using the naive sampling method described above is that every gradient computation will require solving M MDPs. Note that for each sample θ'_m we need to compute $J(\theta'_m)$. This can be problematic as the number of samples M used increases.

In order to potentially alleviate this issue we may consider an alternative sampling method—importance sampling. Importance sampling is a common technique for estimating expected values, when the distribution f of the expectation is hard to sample from. While this is not necessarily our case (recall f is problem specific), the ideas underlying importance sampling may be suitable for alleviating our expensive J computations.

In this method, the expectation over distribution f is estimated from samples generated from a different distribution q over world configurations. In our case, this follows

$$\begin{aligned}
&\mathbb{E}_{\theta' \sim f(\cdot \mid \theta, w)} \Big[J(\theta') \nabla_{\theta, w} \log f(\theta' \mid \theta, w) \Big] = \\
&\quad = \int J(\theta') \nabla_{\theta, w} \Big[\log f(\theta' \mid \theta, w) \Big] f(\theta' \mid \theta, w) \frac{q(\theta')}{q(\theta')} \, \mathrm{d}\theta' = \\
&\quad = \mathbb{E}_{\theta' \sim q} \Big[J(\theta') \nabla_{\theta, w} \left[\log f(\theta' \mid \theta, w) \right] \frac{f(\theta' \mid \theta, w)}{q(\theta')} \Big] = \\
&\quad = \mathbb{E}_{\theta' \sim q} \Big[J(\theta') \frac{\nabla_{\theta, w} f(\theta' \mid \theta, w)}{q(\theta')} \Big].
\end{aligned} \tag{5.6}$$

In the first step we expanded the expectation, and simultaneously multiplied and divided by $q(\theta')$. The multiplicative term q allows us to specify a new expectation, and thus in the second step we have an expectation over distribution q. Finally, the third step follows from the $\log$ gradient.

As a result, we can estimate the gradient from M samples $\{\theta'_m,\ m = 1, \ldots, M\}$ generated from the alternative distribution q—for example, a uniform distribution covering the configuration space Θ. Formally,

$$\begin{aligned}
\mathbb{E}_{\theta' \sim f(\cdot \mid \theta, w)} \Big[J(\theta') \nabla_{\theta, w} \log f(\theta' \mid \theta, w) \Big] &= \mathbb{E}_{\theta' \sim q} \Big[J(\theta') \frac{\nabla_{\theta, w} f(\theta' \mid \theta, w)}{q(\theta')} \Big] \\
&\approx \frac{1}{M} \sum_{m=1}^{M} J(\theta'_m) \frac{\nabla_{\theta, w} f(\theta'_m \mid \theta, w)}{q(\theta')}.
\end{aligned} \tag{5.7}$$

What is interesting is that we get to specify distribution q beforehand, and it is fixed throughout all gradient iterations. As such, after sampling the M possible outcomes from distribution q, we can precompute and store all solutions $J(\theta'_m)$. The true expectation in (5.4) can then be estimated as in (5.7), while reusing the computation of values J. In practice, the only new computations required are $\nabla_{\theta, w} f(\theta'_m \mid \theta, w)$, which in general should be easy to compute. As in (5.5), larger values of M should typically yield better estimates of the gradient.

$\diamond$

We conclude by noting that there exists an underlying trade-off in using these two sampling methods. On one hand, the naive sampling method may be more computationally expensive since each gradient computation requires solving multiple MDPs. However, the gradient estimates may be more accurate than those computed by importance sampling since the outcomes used are sampled directly from the current distribution $f(\ \cdot \mid \theta, w)$. This may allow the naive sampling method to reach better solutions, at the expense of a higher computational cost.

Additionally, we note that both methods are susceptible to the *curse of dimensionality*—as the parameter space increases, the M samples tend to become sparser. The importance sampling method, however, is likely to be more susceptible, since it generates the samples from a different distribution. As such, the quality of the its gradient estimates may decrease faster as the number of parameters increases.

5.2.2 Stochastic Outcomes P-Iteration Algorithm

Algorithm 2 summarizes STOCHASTIC OUTCOMES P-ITERATION, a gradient-based algorithm for solving the optimization problem in (5.1). The outer cycle in lines 3–15 performs N random restarts, in order to tackle the non-convexity of the optimization problem. Each iteration of this outer cycle starts by randomly selecting an initial world configuration θ_0 and dispersion parameter w_0. These initial points seed the gradient search performed in the inner cycle in lines 6–11. The parameters are updated following the gradient ascent update rules, until a stopping condition is met. While Algorithm 2 adopts a standard stochastic gradient update rule, other more complex gradient updates can be used instead, such as Adam gradient [37]. The gradients are approximated via M samples, using some sampling method, for example importance sampling. The algorithm always returns a solution at least as valuable as the original world configuration θ_0, with no requests for changes.

Algorithm 2 STOCHASTIC OUTCOMES P-ITERATION algorithm

Require: MDP, $\mathcal{M} = (\mathcal{X}, \mathcal{A}, P_{\theta_0}, r, \gamma)$
Require: Initial state distribution, μ_0
Require: Configuration space, Θ
Require: Request space, $\mathcal{W}$
Require: Cost function, C
Require: Number of restarts, N
Require: Outcome distribution, f
Require: Stopping condition, ϵ

1: $\theta^* = \theta_0$
2: $F^* = J(\theta_0)$
3: **for** $n = 1$ **to** N **do**
4: $\quad k \leftarrow 0$
5: $\quad$ Randomly select $\theta^{(0)} \in \Theta$ and $w^{(0)} \in \mathcal{W}$
6: $\quad$ **repeat**
7: $\quad\quad$ Estimate $\nabla_{\theta,w} \mathbb{E}_{\theta' \sim f(\cdot|\theta,w)} \Big[J(\theta') \Big]$ using sampling
8: $\quad\quad$ Estimate $\nabla_{\theta,w} F(\theta^{(k)}, w^{(k)})$ using (5.4)
9: $\quad\quad$ Update

$$(\theta^{(k+1)}, w^{(k+1)}) \leftarrow (\theta^{(k)}, w^{(k)}) + \alpha \nabla_{\theta,w} F(\theta^{(k)}, w^{(k)})$$

10: $\quad\quad k \leftarrow k + 1$
11: $\quad$ **until** $\left\| (\theta^{(k)}, w^{(k)}) - (\theta^{(k+1)}, w^{(k+1)}) \right\| < \epsilon$
12: $\quad F^{(n)} = \mathbb{E}_{\theta' \sim f(\cdot|\theta^{(k)}, w^{(k)})} [J(\theta')] - C(\theta^{(k)}, w^{(k)})$
13: $\quad$ **if** $F^{(n)} > F^*$ **then**
14: $\quad\quad \left(\theta^*, w^*\right) = \left(\theta^{(k)}, w^{(k)}\right)$
15: $\quad\quad F^* = F^{(n)}$
16: **return** (θ^*, w^*)

5.2.3 Discussion

We conclude this section with a discussion on relevant connections with the algorithms proposed for solving standard Counterfactual MDPs, as well as some considerations on performance.

Connection to P-ITERATION

STOCHASTIC OUTCOMES P-ITERATION actually corresponds to a more general version of P-ITERATION. We conclude this by analyzing the gradients computed by STOCHASTIC OUTCOMES P-ITERATION as f approaches the Dirac delta function. Similar to what we did in the corridor example, consider now f follows a normal distribution centered on a requested configuration θ and with standard deviation w. It is well known that the normal distribution approaches the Dirac delta function in the limit as w approaches 0. Intuitively, it follows that

$$\begin{aligned}
&\lim_{w \to 0} \mathbb{E}_{\theta' \sim f(\cdot|\theta,w)} \left[J(\theta') \right] \\
&= \lim_{w \to 0} \int_{-\infty}^{+\infty} J(\theta') \frac{1}{\sqrt{2\pi} w} \exp\left(-\frac{(\theta' - \theta)^2}{2w^2} \right) \mathrm{d}\theta' \\
&= \int_{-\infty}^{+\infty} \lim_{w \to 0} J(wz + \theta) \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{z^2}{2} \right) \mathrm{d}z \\
&= J(\theta) \int_{-\infty}^{+\infty} \exp\left(-\frac{z^2}{2} \right) \mathrm{d}z = J(\theta),
\end{aligned}$$

where in the first step we apply the definition of expected value and expand the normal distribution probability density function; in the second step we perform the substitution $z = (\theta' - \theta)/w$, and move the limit inside the integral by considering the dominated convergence theorem; and in the third step we observe the integral corresponds to the cumulative distribution function of a standard normal, thus taking value 1.

We conclude that, in the limit, the gradients computed by the new algorithm actually correspond to those computed in the original P-ITERATION algorithm. This suggests we can solve Counterfactual MDPs with the new STOCHASTIC OUTCOMES P-ITERATION algorithm. This may be interesting since, unlike the approaches developed in Chapter 4, the methods developed here do not require the gradient of the optimal value function with respect to the transition probabilities. As we saw before in Sections 4.1.1 and 4.1.2, computing such gradient is not trivial and may require some expensive computations, such as matrix inverses. Applying the methods developed in this chapter, however, comes at the expense of having to solve multiple MDPs in each gradient iteration.

Performance

As discussed before, from a computational perspective the problem addressed in this paper is hard, and as such we should not expect an efficient algorithm for computing the exact solution of the problem in its general form. We proposed instead an iterative gradient-based algorithm for solving the underlying optimization problem, where, at each step t, we update the desired world configuration θ and dispersion parameter w following a gradient update.

From a practical perspective, the most computationally expensive step of our algorithm regards estimating each gradient step, which requires the solution of multiple MDPs. In some cases, however, it should be possible to speed up the solving of these MDPs. For example, when the outcome distribution f is narrow, it is likely that sampled world configurations are associated with similar transition probabilities. In that case, it is reasonable to assume that the value functions of different sampled world configurations are also similar. Specifically, for two consecutive world configurations samples θ_m and θ_n, we expect $\boldsymbol{v}^*_{\theta_m} \approx \boldsymbol{v}^*_{\theta_n}$. Building upon this observation, we can speed up our algorithm by seeding the MDP solving of configuration θ_n with the optimal value function computed for the previous sample θ_m. That is, we can seed the value iteration algorithm with $\boldsymbol{v}^{(0)}_n = \boldsymbol{v}^*_m$. This approach follows the seeding mechanism discussed in Section 4.3.2, which we showed to work well in practice in the experimental evaluation in Section 4.4.3.

5.3 Results

We evaluate the applicability of Stochastic Outcomes Counterfactual MDPs in multiple scenarios, with different parameterizations of the transition probabilities, uncertainty distributions and cost functions. Moreover, we assess the performance of STOCHASTIC OUTCOMES P-ITERATION both in terms of execution time and the quality of the solutions computed. In some experiments we adopt the two sampling methods proposed for estimating the gradient—naive and importance sampling—and discuss their performance.

5.3.1 Corridor

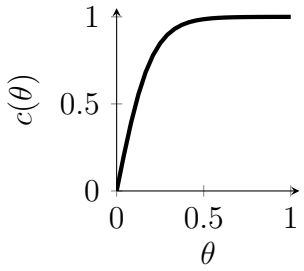
We consider the CORRIDOR scenario described previously in Section 5.1, and depicted in Figure 5.1. For this experimental evaluation, we let the corridor be parameterized by a length L, with the agent always starting at the top left position, and aiming to reach the bottom left position.

As before, we assume the agent is allowed to reason over alternative configurations of the world, where the doors of the corridor may be open/closed. We adopt a parameterization of the transition probabilities similar to that used in the original corridor example (3.4), where we denote a world configuration θ as a vector in $\Theta = [0, 1]^K$, and admit that K obstacles correspond to doors, with θ_k indicating how much the k-th door is open. Again, this parameterization assumes that the probability of the robot moving up or down is proportional to the openness of the door.

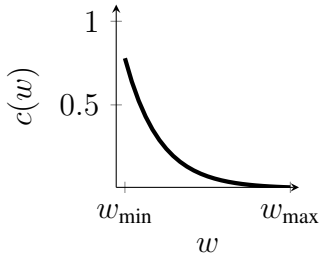
Since our robot does not have arms, the opening/closing of the doors requires necessarily some form of external intervention, which may be provided with different levels of precision. We model the uncertainty in the outcome of an external intervention as an independent multivariate truncated normal distribution $f(\theta' \mid \theta, w)$, centered at the desired configuration $\boldsymbol{\theta}$, truncated to the interval $[0, 1]$, and with dispersion $\boldsymbol{\sigma}$. Specifically, we let the dispersion depend on two components

$$\sigma_i = w_i S(\theta_i). \tag{5.8}$$

The first component regards the set of possible external interventions. We let $w \in \mathcal{W} = [w_{\min}, w_{\max}]^K$, where w_k denotes the level of expertise requested in opening the k-th door. For



(a) Cost for opening a door.



(b)

Figure 5.4: Illustration of the cost functions used in the corridor scenario. 5.4(a) The cost for requesting a door to be open. This is a smooth step size function, where the cost for slightly opening the door is similar to that of fully opening it. 5.4(b) The cost for controlling the outcome uncertainty. This cost is proportional to the precision requested.

these experiments we assume $w_{\text{min}} = 0.05$ and $w_{\text{max}} = 1.0$, denoting respectively external interventions that change the world with no/maximum uncertainty. The second component depends on the request θ_k itself, following the smooth step function S depicted in Figure 4.4. This component forces the dispersion to be small in requests for worlds with closed doors, *i.e.*, $\theta_k \approx 0$. We note that since the original world configuration θ_0 assumes all the doors are closed, even the most imprecise external intervention can perfectly fulfill that request by doing nothing. We note, as well, the interesting connection with the discussion raised at the end of Section 5.1. Recall that in Section 5.1 we adopted a simpler formulation where the dispersion only depended on the external entity w_i, and we observed, however, that this simple model could potentially lead to unrealistic solutions, due to the outcome uncertainty distribution f converging to a uniform distribution. Specifically, we observed that in some cases the best solution would be to request an imprecise external intervention to keep the door shut, because there would be a high probability it would end up opening it anyway. The formulation adopted for this experimental evaluation is more realistic and will not have this problem.

The robot is penalized for requesting an external intervention for opening the doors of the corridor. In particular, the higher the precision requested, the larger the penalty incurred. Specifically, we consider a cost function

$$C(\theta, w) = K_\theta c(\theta) + K_w c(w), \tag{5.9}$$

where the costs associated with requesting a world configuration θ take the form of a smooth (and differentiable) step size function:

$$c(\theta) = \sum_{k=1}^{K} \left(\frac{2}{1 + e^{-\beta\theta_k}} - 1 \right),$$

with $\beta = 10$. This makes the cost of requesting a door to be slightly open, approximately the same as that of requesting it to be fully open. This cost is depicted in Figure 5.4(a). On the other hand, we assume the costs associated with the outcome uncertainty grow exponentially

with respect to the precision requested

$$c(w) = \sum_{k=1}^{K} \exp(-\beta_w w_k),$$

with $\beta_w = 5$. In this scenario, this translates in the cost of bothering the most precise external entities (for example humans) being exponentially larger than the cost of bothering less precise external entities (for example robots).

Finally, for these experiments we will assume the following instance of the problem

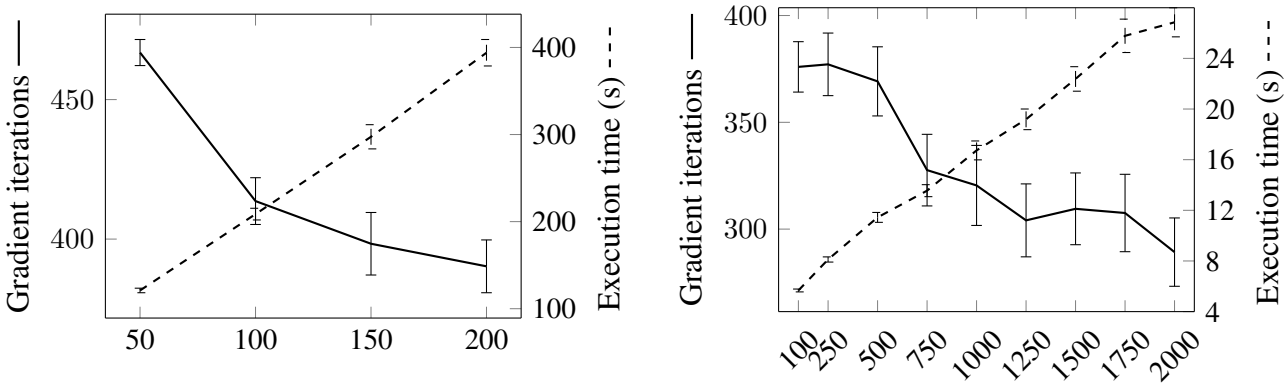
$$\mathcal{I} : K_\theta = 2.0 \qquad K_w = 1.0, \tag{5.10}$$

where the agent incurs a cost both when requesting changes to the world, and when controlling the outcome uncertainty.

All experiments assume a discount factor $\gamma = 0.9$, and use the Adam gradient algorithm as described in [37], with a learning rate $\alpha = 0.01$. We note that in the experiments with Stochastic Outcomes Counterfactual MDPs, in general, we will use smaller learning rates. Since the gradient is approximated via sampling methods, larger learning rates may lead to some instability in the optimization.

Execution Time

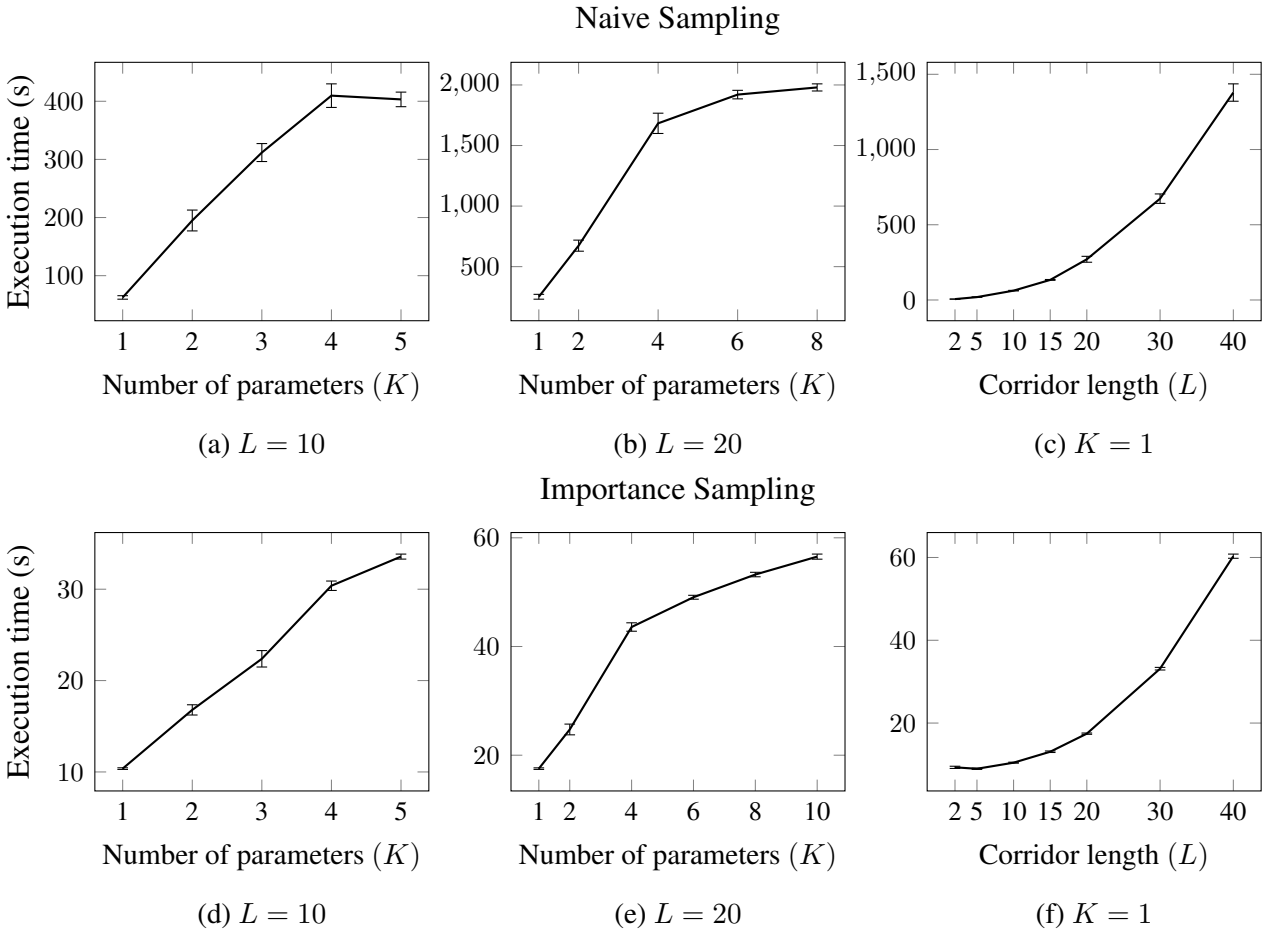
We consider instance $\mathcal{I}$ when analyzing the performance of our algorithm in terms of execution time. We started by analyzing the impact of increasing the number of samples M in the performance of the algorithm. Figure 5.5 plots the impact of the number of samples M in terms



(a) Naive sampling

(b) Importance sampling

Figure 5.5: Tradeoff between number of gradient iterations and execution time, for increasing number of samples M. (a) and (b) The results assuming naive and importance sampling strategies, respectively. The results are averaged over 15 runs of the CORRIDOR scenario. The bars correspond to the standard error of the mean.



(a) $L = 10$ (b) $L = 20$ (c) $K = 1$

(d) $L = 10$ (e) $L = 20$ (f) $K = 1$

Figure 5.6: Execution time performance of STOCHASTIC OUTCOMES P-ITERATION algorithm in the CORRIDOR scenario. Results averaged over 7 runs. Bars represent the standard error of the mean. The top row assumes the naive sampling strategy, whereas the bottom row assumes importance sampling. (a), (b), (d) and (e) The execution time as a function of the number of parameters, on corridors of different lengths L. (c) and (f) The execution time as a function of the corridor length and a single parameter $K = 1$.

of number of gradient updates required for convergence and execution time, when using both sampling methods. Specifically, we considered an instance of the corridor scenario with length $L = 10$, $K = 3$ parameters, $N = 15$ random restarts. The termination condition was established as the l_∞ norm between two consecutive configurations getting below a threshold $1\mathrm{e}{-5}$, or reaching a maximum of 1000 iterations, whichever occurs first.

We observe that as the number of samples M increases, the number of gradient iterations required to reach convergence decreases, and conversely, the execution time increases. This is expected since, as predicted in (5.5) and (5.7), larger values of M allow for better approximations of the true gradients, resulting on a more stable optimization. However, larger values of M require more MDPs to be solved, thus increasing the execution time.

Interestingly, we observed that when using the naive sampling method, the algorithm returns

roughly the same solution, regardless of the number of samples M used. This seems to suggest that despite the instability in the gradient updates, it may be worth it to adopt a smaller M. Based on this observation, for the remainder experiments on this scenario using naive sampling we consider $M = 100$, as it seems to establish a good tradeoff between optimization stability and execution time. In the importance sampling method, however, we observed that larger number of samples M do tend to return better solutions. This difference between the naive sampling and importance sampling methods is likely due to the samples used in the gradient estimates—recall that in the former uses samples directly generated from f, whereas in the latter we use samples previously generated from a uniform distribution. As such, for the remainder experiments on this scenario using importance sampling we consider $M = 1500$.

Figure 5.6 summarizes the execution time performance of the algorithm in the CORRIDOR scenario for increasing number of parameters K and corridor length L. Figures 5.6(a) and 5.6(b) plot the execution time of the algorithm as the number of parameters K increases, and assuming a naive sampling strategy. Figures 5.6(d) and 5.6(e) plot the same but assuming the importance sampling strategy instead. These plots suggest that the execution time degrades gracefully with increasing K. Figures 5.6(c) and 5.6(f) plot the execution time as a function of the problem size $(\mathcal{X} \times \mathcal{A} \times \mathcal{X} \times K)$, with fixed number of parameters $K = 1$, according to the two sampling strategies. These plots depict a steeper growth of the execution time. However, this is expected since increasing the corridor length also increases the state space of the MDPs to be solved in order to compute the gradient in (5.4), and solving MDPs takes polynomial time [80].

In general, for the values of M selected, the importance sampling method was faster than the naive sampling alternative. This is line with our expectation, given the gradient iterations reported in Figure 5.5 and the fact that naive sampling method solves M MDPs per gradient update—the total number of MDPs solved throughout the gradient search using naive sampling surpasses that of the importance sampling counterpart. However, the better execution time performance provided by the importance sampling strategy may come at a cost with respect to the quality of the solutions computed.

Solutions

We now assess the solutions computed by our approach on instance $\mathcal{I}$ introduced in (5.10), and, in the process, we observe the potential benefits in considering the outcome uncertainty associated with requesting changes to the world. We start by recalling that we consider an instance $\mathcal{I}$ where there is a cost in requesting a door to be open and in controlling the outcome uncertainty.

Table 5.1 summarizes the solutions computed for corridors of increasing length L and number of parameters K. Specifically, for each corridor of length L we considered $K = L-1$ parameters, corresponding to all the doors of the corridor that can be opened.

Column $J(\theta_0)$ denotes the performance of the agent in the original world θ_0, where all the doors are closed. As expected, the performance of the agent deteriorates with the length of the corridor, as more steps are needed to reach the goal. Columns $F(\theta^*_{\mathcal{I}}, w^*_{\mathcal{I}})$ denote the performance of the agent in instance $\mathcal{I}$, under the solutions computed by Stochastic Outcomes P-Iteration with two different sampling strategies.

In corridor $L = 2$, the best solution is to not request any change to the world. That is because the cost of the external intervention necessary to open the door is more expensive than

Table 5.1: The quality of the solutions computed by STOCHASTIC OUTCOMES P-ITERATION in the CORRIDOR scenario, adopting either naive or importance sampling. For a corridor with length L, we assume $K = L - 1$ parameters. The results are rounded to 2 decimal places, and averaged over 7 runs. The dispersion reported refers to the standard error of the mean.

CORRIDOR			
		$F(\theta^*_{\mathcal{I}}, w^*_{\mathcal{I}})$	
L	$J(\theta_0)$	Naive Sampling	Importance Sampling
2	-2.71	-2.71 ± 0.00	-2.71 ± 0.00
3	-4.10	-3.55 ± 0.01	-3.68 ± 0.11
4	-5.22	-3.57 ± 0.00	-4.06 ± 0.08
5	-6.13	-3.57 ± 0.00	-4.03 ± 0.11
7	-7.46	-3.60 ± 0.00	-4.28 ± 0.04
10	-8.64	-3.71 ± 0.03	-5.46 ± 0.75

the potential benefit in performance. For all other corridors, in general, the best solution is to request the world configuration with the first door fully open. However, perhaps surprising, the best solution is not to ask the door to be opened by the most precise entity. In fact, due to the exponential nature of the precision cost $c(w)$ defined, we observe that the average precision requested by the agent is roughly $w = 0.22$. We note this depicts an example where allowing the agent to plan over the outcome uncertainty and associated costs allowed it to improve its performance. A standard Counterfactual MDP would not be able to compute this solution.

Finally, we observe that for the values of M selected, in general, the naive sampling strategy converged to better solutions. Moreover, the gap in the quality of the solutions returned between the two sampling strategies increases with the size of the problem (size of MDP and number of parameters). The first observation is likely due to the fact that the naive sampling method generates samples directly from the current distribution, whereas the importance sampling method uses samples previously generated from a different distribution. The second observation can be explained by the curse of dimensionality, since increasing the parameter space quickly renders the original importance sampling samples sparser.

Even though these results fall in line with our expectation, it may be interesting to understand in further detail how the performance of the two methods compares for different values of M, and when provided similar execution time budgets.

Further discussion on the results

We evaluated the performance of STOCHASTIC OUTCOMES P-ITERATION in the CORRIDOR scenario, under two sampling-based methods for estimating the gradient—naive sampling and importance sampling. We recall both these methods include a hyperparameter M that specifies the number of samples to be computed. This parameter can be tuned and, in theory, increasing M should lead to a more stable optimization, and consequently the discovery of better solutions. In the previous experimental evaluation, the choice of M was guided by the preliminary experiment

Table 5.2: The impact of the number of samples M in the solutions computed by STOCHASTIC OUTCOMES P-ITERATION. Assuming a CORRIDOR scenario with length L and $K = L - 1$ parameters. The results are rounded to 2 decimal places, and averaged over 5 runs. The dispersion reported refers to the standard error of the mean.

Quality of solutions ($F(\theta^*_{\mathcal{I}}, w^*_{\mathcal{I}})$) in the CORRIDOR scenario

Sampling Method	M	L				
		2	3	4	5	7
Naive	1	-2.71 ± 0.00	-3.61 ± 0.01	-3.90 ± 0.04	-4.06 ± 0.06	-4.64 ± 0.11
	5	-2.71 ± 0.00	-3.57 ± 0.01	-3.66 ± 0.02	-3.76 ± 0.05	-4.01 ± 0.09
	10	-2.71 ± 0.00	-3.57 ± 0.02	-3.61 ± 0.02	-3.72 ± 0.03	-3.94 ± 0.02
	25	-2.71 ± 0.00	-3.54 ± 0.02	-3.58 ± 0.01	-3.66 ± 0.03	-3.76 ± 0.04
	50	-2.71 ± 0.00	-3.54 ± 0.00	-3.57 ± 0.00	-3.60 ± 0.01	-3.67 ± 0.03
	100	-2.71 ± 0.00	-3.55 ± 0.01	-3.57 ± 0.00	-3.57 ± 0.01	-3.60 ± 0.03
Importance	500	-2.71 ± 0.00	-3.69 ± 0.05	-4.18 ± 0.15	-4.26 ± 0.01	-4.32 ± 0.05
	1500	-2.71 ± 0.00	-3.68 ± 0.11	-4.06 ± 0.08	-4.03 ± 0.11	-4.28 ± 0.04
	5000	-2.71 ± 0.00	-3.59 ± 0.08	-3.79 ± 0.08	-4.10 ± 0.13	-4.23 ± 0.02
	10000	-2.71 ± 0.00	-3.56 ± 0.05	-3.71 ± 0.05	-4.12 ± 0.11	-4.28 ± 0.02
	15000	-2.71 ± 0.00	-3.55 ± 0.06	-3.71 ± 0.06	-4.09 ± 0.11	-4.28 ± 0.02
	20000	-2.71 ± 0.00	-3.55 ± 0.07	-3.82 ± 0.07	-3.98 ± 0.10	-4.28 ± 0.02

depicted in Figure 5.5. In this preliminary experiment we analyzed the tradeoff between the number of gradient iterations and execution time, for a particular instance of the CORRIDOR scenario, with length $L = 10$ and $K = 3$ parameters. Based on this preliminary experiment, we empirically selected the number of samples M that seems to better optimize the aforementioned tradeoff, while still returning good solutions. This led us to select $M = 100$ and $M = 1500$ for the naive and importance sampling methods, respectively.

Given this selection of M we then evaluated the performance of the algorithm under both sampling methods. We observed that for these values of M, while the naive sampling method required larger execution times, it typically led to better solutions. However, one may wonder how the quality of the solutions computed by the two methods compare for larger/smaller values of M, and when provided similar execution time budgets.

Our expectation is that as we increase the number of samples M, both methods will take longer, but the average quality of the solutions should increase. Tables 5.2 and 5.3 confirm our expectation. These tables describe the impact of parameter M in the value of the solutions returned, and in the execution time performance of the algorithm.

As expected, the execution time deteriorates as we increase the problem size (length L and parameters K) and as we increase the number of samples M. At the same time, we observe that as we increase M, in general, the quality of the solutions returned improves. It is interesting to note, however, that this improvement is more noticeable in the naive sampling method. We observe that the solutions computed by the importance sampling method were only competitive up to the instance with length $L = 3$ and $K = 2$ parameters. Furthermore, in the rightmost corridor, with length $L = 7$ and $K = 6$ parameters, we stop observing a clear improvement in the quality of the solutions for increasing values of M. Even assuming $M = 20000$ samples, the

Table 5.3: The impact of the number of samples M in the execution time performance of STOCHASTIC OUTCOMES P-ITERATION. Assuming a CORRIDOR scenario with length L and $K = L - 1$ parameters. The results are rounded to 1 decimal place, and averaged over 5 runs. The dispersion reported refers to the standard error of the mean.

Execution time performance (seconds) in the CORRIDOR scenario

Sampling Method	M	L				
		2	3	4	5	7
Naive	1	2.6 ± 0.1	3.4 ± 0.1	4.2 ± 0.1	4.9 ± 0.2	6.3 ± 0.11
	5	2.8 ± 0.1	5.4 ± 0.1	8.7 ± 0.4	11.4 ± 0.4	16.2 ± 0.09
	10	2.9 ± 0.1	6.9 ± 0.5	13.9 ± 0.4	18.2 ± 0.7	29.1 ± 0.02
	25	3.4 ± 0.1	10.7 ± 0.6	28.5 ± 0.8	41.1 ± 3.0	61.5 ± 0.04
	50	4.0 ± 0.2	17.2 ± 1.7	45.5 ± 3.8	76.2 ± 4.1	122.7 ± 0.03
Importance	500	4.9 ± 0.1	7.2 ± 0.7	13.2 ± 2.1	18.3 ± 0.8	20.1 ± 0.6
	1500	9.0 ± 0.2	18.4 ± 3.8	29.9 ± 4.9	42.8 ± 3.2	53.9 ± 1.6
	5000	22.9 ± 0.7	53.0 ± 5.7	61.3 ± 6.0	116.9 ± 8.6	178.3 ± 5.7
	10000	44.4 ± 0.7	108.3 ± 9.9	133.7 ± 14.5	244.2 ± 16.6	336.0 ± 15.7
	15000	66.8 ± 1.3	151.0 ± 17.7	198.1 ± 27.5	301.9 ± 31.8	516.9 ± 27.4
	20000	85.3 ± 1.4	213.4 ± 34.2	406.2 ± 82.5	433.0 ± 47.5	714.6 ± 26.5

importance sampling method fails to compute a solution competitive with those returned by the naive sampling method. It is plausible that by testing larger values of M we would eventually converge to better solutions. In any case, it is clear that the importance sampling method is suffering from the curse of dimensionality here.

Additionally, from the tables we observe that, under similar execution time budgets, the naive sampling method seems to perform better overall, typically returning better solutions. The solutions returned by the importance sampling method only seem to be somewhat competitive for restrictive execution time budgets (for example 7 seconds) and small problem sizes.

5.3.2 Frozen Lake

We consider the FROZEN LAKE scenario previously introduced in Section 4.4.2, and depicted in Figure 4.6. In this scenario, a robot must traverse an icy grid from an initial state to a goal state, while avoiding the holes where the ice has melted. The movement of the robot is uncertain due to the frozen ice—when moving in a given direction, the wheels of the robot may slip, causing the robot to move in a different way.

We assume the world can be configured in terms of the grip of the robot's wheels. The grip profile can range between NO-GRIP and FULL-GRIP. The former denotes the original configuration of the world, where the robot may slip and move in either the intended or perpendicular directions. The latter denotes a world where the robot always moves deterministically in the intended direction. We denote the transition probabilities associated with these worlds as P_{ng} and P_{g}, respectively. This leads to the following parameterization of the transition probabilities

$$P_\theta = \theta P_{\text{g}} + (1 - \theta) P_{\text{ng}},$$

Table 5.4: The quality of solutions computed by STOCHASTIC OUTCOMES P-ITERATION in the FROZEN LAKE scenario, adopting either naive or importance sampling. The results are rounded to 2 decimal places, and averaged over 15 runs. The dispersion reported refers to the standard error of the mean. The dispersion on the solutions is not shown, since it rounds to zero.

FROZEN LAKE					
		Naive Sampling		Importance Sampling	
Size	$J(\theta_0)$	$F(\theta^*_{\mathcal{I}}, w^*_{\mathcal{I}})$	$\theta^*_{\mathcal{I}}, w^*_{\mathcal{I}}$	$F(\theta^*_{\mathcal{I}}, w^*_{\mathcal{I}})$	$\theta^*_{\mathcal{I}}, w^*_{\mathcal{I}}$
4×4	-46.34	-19.91 ± 0.03	$1.00,\ 0.10$	-19.80 ± 0.03	$1.00,\ 0.10$
8×8	-58.95	-26.70 ± 0.03	$1.00,\ 0.12$	-26.64 ± 0.03	$1.00,\ 0.11$

where $\theta \in \Theta = [0, 1]$ denotes the level of grip requested—$\theta = 1$ corresponds to maximum grip, whereas $\theta = 0$ corresponds to no grip.

As before, we model the uncertainty in the outcome of an external intervention as a truncated normal $f(\theta' \mid \theta, w)$, centered in the desired world configuration θ, with standard deviation controlled by parameter w, and truncated to the interval $[0, 1]$. Moreover, we let the set of possible external interventions take the form $\mathcal{W} = [w_{\min}, w_{\max}]$, with $w_{\min} = 0.05$ and $w_{\max} = 0.25$. In this scenario, parameter w can be interpreted as the expertise of the engineering team that is supposed to configure the grip levels of the robot. A highly qualified engineering team should be able to better deliver the requested configuration, however, at the expense of a higher cost.

We assume a cost function similar to (5.9), where the costs associated with outcome uncertainty take the form

$$c(w) = \exp(-\beta w), \tag{5.11}$$

with $\beta = 20$. This assumes that the engineering effort associated with greater levels of precision grows exponentially. Moreover, we let the costs associated with requesting a grip level θ grow linearly, $c(\theta) = \theta$.

We assume an instance parameterized by

$$\mathcal{I} : K_\theta = 5.00 \qquad K_w = 25.00,$$

where the agent incurs higher costs when requesting a precise engineering external intervention, for increasing its original grip levels.

Solutions

Table 5.4 summarizes the solutions computed for the two versions of FROZEN-LAKE. All experiments assume discount factor $\gamma = 0.99$, $N = 15$ random restarts, and use the Adam gradient with learning rate $\alpha = 0.01$. Moreover, all results are averaged over 15 randomly seeded runs.

Column $J(\theta_0)$ denotes the performance of the agent in the original world NO-GRIP. Column $F(\theta^*_{\mathcal{I}}, w^*_{\mathcal{I}})$ denotes the performance of the agent on the solution computed by STOCHASTIC OUTCOMES P-ITERATION on instance $\mathcal{I}$. We report results assuming the two aforementioned

sampling strategies. We used $M = 30$ when using the naive sampling strategy. For the importance sampling strategy we considered $M = 1500$.

We start by observing that the solutions computed establish a trade-off between the benefits arising from requesting a full-grip configuration and the costs associated with a precise external intervention. The average best solution for the 4×4 scenario was $(\theta^*_{\mathcal{I}}, w^*_{\mathcal{I}}) = (1.00, 0.10)$, whereas for the 8×8 was $(\theta^*_{\mathcal{I}}, w^*_{\mathcal{I}}) = (1.00, 0.11)$. Specifically, note how the best solution is not necessarily to request the most precise external intervention possible. These solutions suggest that additional cost involved in requesting the most precise engineering team is not worth it. This depicts a clear example where planning over the outcome uncertainty and associated costs allows the agent to improve its performance. Our previous approaches for solving standard Counterfactual Markov decision processes would not compute such solutions, since none explicitly plans over the aforementioned uncertainty in the outcomes of requests for changes to the world, and associated costs.

Finally, we observe both sampling strategies typically converged to solutions of similar quality. This is likely due to the smaller number of parameters that need to be optimized and to the choice of parameters M used.

5.3.3 Water Pouring

We consider the WATER POURING scenario introduced previously in Section 4.4.2. In this scenario, a robot is trained how to pour water in cups located on a table in front (see Figure 4.7(b)).

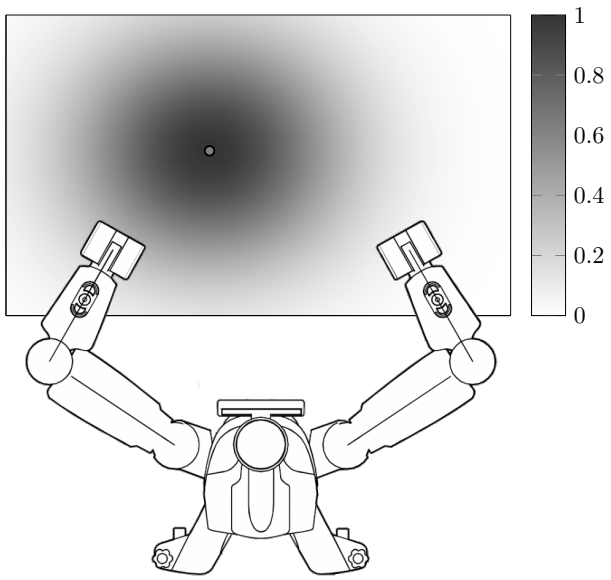
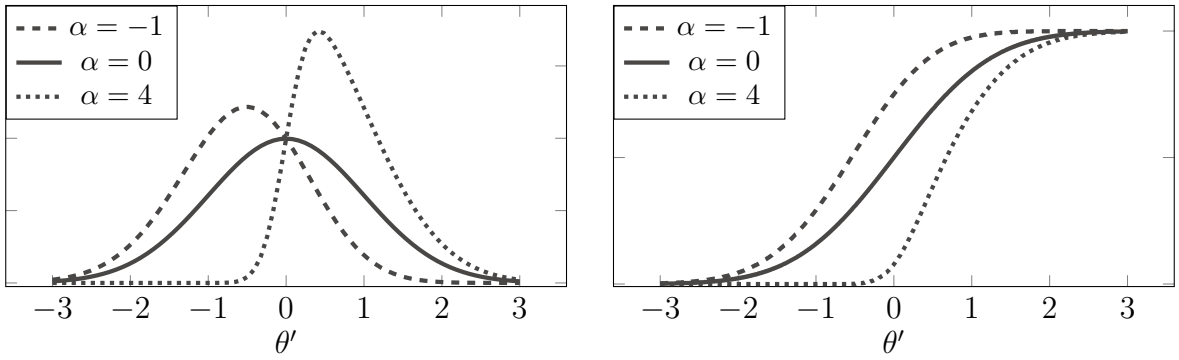


Figure 5.7: Setup of the stochastic outcomes WATER POURING scenario. The BAXTER robot has to pour water in cups located on a table in front. The execution success probability is modeled as a radial kernel, depicted by the shaded region. Darker corresponds to higher success probability in executing the pouring motion. The colorbar depicts the colors associated with different probabilities of success. Repeats Figure 4.8 for convenience.



(a) Probability density

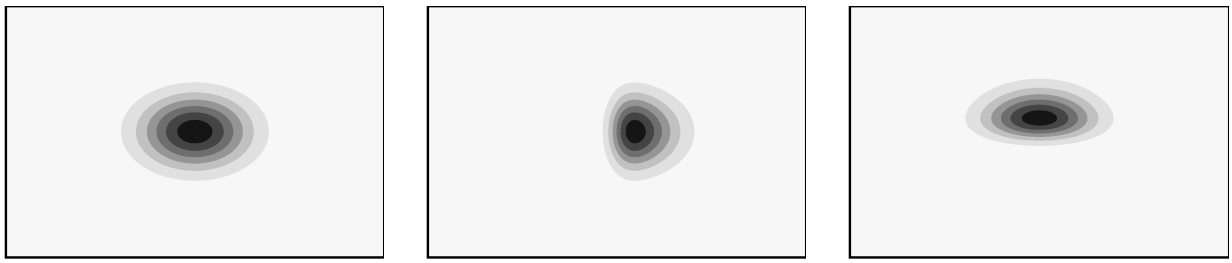
(b) Cumulative probability

Figure 5.8: Probability density function for the skew normal distribution centered at a request $\theta = 0$ and with dispersion $\sigma = 1$, assuming different skew parameters α. Illustrates the probability of outcomes θ' under different skew parameters α.

Then, the robot is tasked with pouring water on new cup locations $\boldsymbol{\theta}_0$. The robot is allowed to reason over different configurations of the world, corresponding to different locations of the cup. In Section 4.4.2 we modeled the problem as a Counterfactual MDP, and thus assumed deterministic outcomes on requests for changes to the world—*i.e.*, it was assumed the cup would be moved necessarily to the location solicited. We now tackle instead a more realistic case, where the outcomes may actually be stochastic.

As before, the robot is taught by demonstration a collection of pouring trajectories, which it is then supposed to generalize to the new cup locations (Figure 4.7(a)). The execution success of a pouring motion is assumed to follow a radial kernel centered at a point $\bar{\boldsymbol{\theta}}$—the center of the trajectories learned by demonstration. As such, the closer the target position of the cup to this center, the higher the execution success probability (Figure 5.7). We endow the agent with the ability to reason over different locations of the cup—*i.e.*, over the counterfactual *"What if the cup was in a different position?"* In fact, we allow the agent to plan over the possibility of actually operating in a world where the cup is indeed in a different location, likely closer to locations with higher success execution probability. These changes to the world are assumed to be achievable through some external intervention. However, this external intervention comes at a cost. Specifically, a cost that is proportionally with the displacement of the cup required. For a given cup position $\boldsymbol{\theta}$, this cost was defined in (4.19) as $C(\boldsymbol{\theta}) = \beta\|\boldsymbol{\theta} - \boldsymbol{\theta}_0\|_1$.

We model the uncertainty in the outcome of an external intervention that moves the cup to a different position $\boldsymbol{\theta}$ as a multivariate independent skew normal distribution f centered at position $\boldsymbol{\theta}$, with fixed dispersion σ and skewness parameter α [2]. Figure 5.8 depicts the behaviour of the skew normal distribution for different skew parameters α. This distribution can be interpreted as modeling the uncertainty associated with requesting the human user to move the cup. Using the skew normal distribution in this scenario allows us to model different skews in the uncertainty of moving a cup. In particular, we consider three cases. We start by consider a case where f is not skewed ($\alpha = [0, 0]$). Essentially, this translates in the uncertainty distribution taking the form of a standard normal distribution centered around $\boldsymbol{\theta}$ and standard deviation $\sigma = 0.05$ (see Figure 5.9(a)). Then, we consider two cases, where the movements of the cup are skewed



(a) $\alpha = [0, 0]$ (b) $\alpha = [3, 0]$ (c) $\alpha = [0, -1]$

Figure 5.9: The three skewness cases considered in the stochastic outcomes WATER POURING scenario. (a) The case where the uncertainty is not skewed, thus consisting in a standard normal distribution. (b) The case where the uncertainty is skewed on the horizontal axis, to the right. (c) The case where the uncertainty is skewed on the vertical axis.

in the horizontal and vertical directions, taking $\alpha = [3, 0]$ and $\alpha = [0, -1]$ respectively. The former represents a case where the human user typically replies to requests for a new position of the cup, by overshooting the displacement to the right (Figure 5.9(b)). Similarly, the later represents a case where the human user undershoots the displacement of the cup on the vertical axis (Figure 5.9(c)).

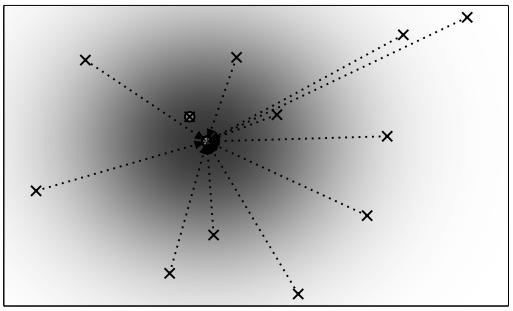
Results

We ran experiments starting from 12 initial configurations $\boldsymbol{\theta}_0$, depicted as crosses in Figure 5.10. The average value in these 12 original world configurations, $J(\boldsymbol{\theta}_0)$, is approximately -4.57. By definition of the reward function used, this suggests that the agent typically executed the "quit" action when faced with these initial locations of the cup (quitting led to a -5 reward).

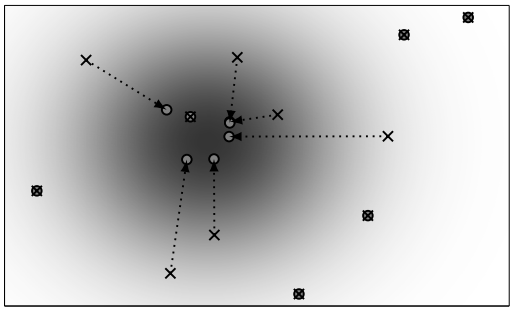
We then assessed the solutions of the Stochastic Outcomes Counterfactual MDP, assuming different values of cost parameters β and skewness parameters α. Figures 5.10(a) and 5.10(b) depict the results assuming an unskewed normal distribution, under two different costs $\beta = 0$ and $\beta = 10$.

Assuming $\beta = 0$, *i.e.*, no cost in changing the cup position, we observe that, in general, the solution returned is to have the cup located at the center of the radial kernel. Intuitively, these results are expected, because if there is no cost in changing the world, we might as well make it as easy as possible to perform the task. One interesting observation, however, is that there was one starting position for which the algorithm opted to not move the cup—the initial position closest to the center of the radial kernel. This occurs because by requesting the cup to be moved to the center of the radial kernel, the agent risks facing the outcome uncertainty underlying each request—while it is likely the cup will be placed at the center of the kernel, there is still a chance it may not. By sticking with the original location, on the other hand, the agent faces no uncertainty, and since the cup is already close to the center of the radial kernel, the value of this configuration is already high.

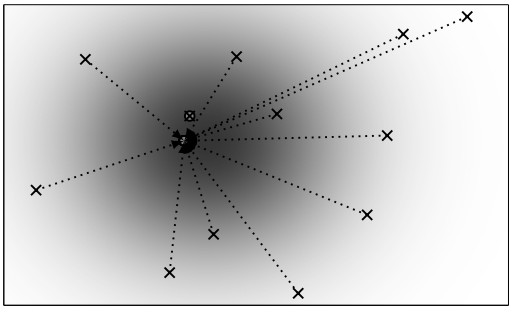
Figure 5.10(b) depicts the solutions computed for $\beta = 10$. As expected, we now observe that for initial configurations distant from the center of the radial kernel, the agent opts to not move the cup. This is because it would be too expensive to move the cup to a location where it



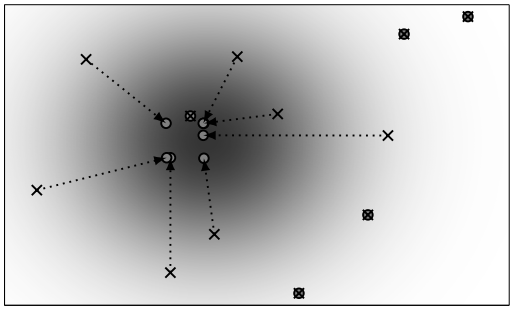
(a) Solutions assuming $\alpha = [0, 0]$ and $\beta = 0$



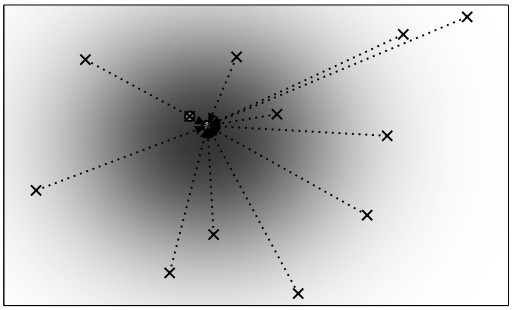
(b) Solutions assuming $\alpha = [0, 0]$ and $\beta = 10$



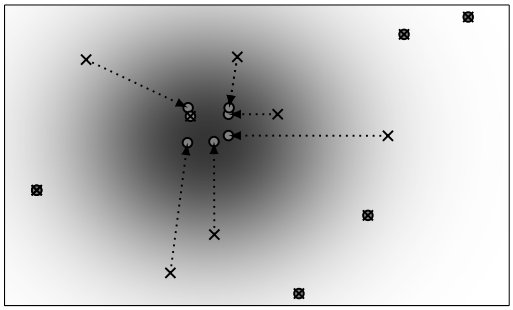
(c) Solutions assuming $\alpha = [3, 0]$ and $\beta = 0$



(d) Solutions assuming $\alpha = [3, 0]$ and $\beta = 10$



(e) Solutions assuming $\alpha = [0, -1]$ and $\beta = 0$



(f) Solutions assuming $\alpha = [0, -1]$ and $\beta = 10$

Figure 5.10: Performance in the stochastic outcomes water pouring scenario. (a) to (f) The solutions computed for different skewness parameters α and cost parameters β.

is worth it to try to execute the pouring action. More interesting is the comparison between the solutions computed with the Stochastic Outcomes Counterfactual MDP, and those returned in the deterministic setting, depicted in Figure 4.9(c). We observe that under stochastic outcomes there were more initial configurations for which the robot deemed it not worth it to move the cup. This is due to the additional uncertainty underlying each request.

The remaining figures depict the solutions assuming skewed outcome uncertainty distributions. It is interesting to observe how the skewness of the uncertainty distribution affects the solutions returned. In Figures 5.10(c) and 5.10(d) we observe that the solutions returned seem shifted to the left. This makes sense, because the uncertainty distribution is skewed to the right in the horizontal direction. To compensate for this skewness, the agent poses the requests slightly shifted to the left. Even more interesting, is to compare the solutions returned for the leftmost initial location, under a cost $\beta = 10$, assuming skewness parameters $\alpha = [0, 0]$ and $\alpha = [3, 0]$.

Table 5.5: The quality of solutions computed by STOCHASTIC OUTCOMES P-ITERATION in the WATER POURING scenario. The results are averaged over the 12 predefined targets, in 5 randomly seeded runs, and rounded to 2 decimal places.

WATER POURING						
$J(\theta_0)$	$F(\theta)$					
	$\alpha = [0, 0]$		$\alpha = [3, 0]$		$\alpha = [0, -1]$	
-4.57	$\beta = 0$	$\beta = 10$	$\beta = 0$	$\beta = 10$	$\beta = 0$	$\beta = 10$
	-1.83	-4.22	-1.63	-4.13	-1.72	-4.14

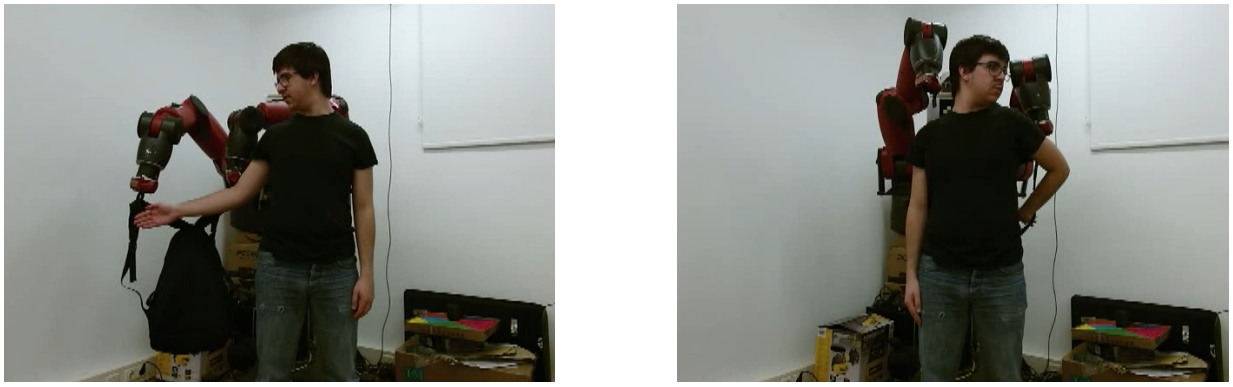
We observe that for $\alpha = [0, 0]$, the agent opted to not request the movement of the cup, as it was deemed too expensive to move it to a location where it was worth to try to execute the pouring motion. Under $\alpha = [3, 0]$, on the other hand, the agent requested the cup to be moved. Because of the skewness of the uncertainty distribution, the cost for moving the cup to an advantageous location is now smaller. In this case, the smaller cost made it worth it to move the cup closer to the center of the radial kernel. Finally, Figures 5.10(e) and 5.10(f) depict similar results, this time however, the solutions returned seem to be shifted upwards. This is due to the skewness of the distribution in the vertical direction.

Table 5.5 presents the values achieved under the different α and β values, averaged over the 12 targets. Column $J(\theta_0)$ is the average value achieved in the 12 original world configurations. We observe that, in general, the solutions returned by the Stochastic Outcomes Counterfactual MDP allow the agent to collect more reward, even assuming high costs for changing the world.

5.3.4 Robot Backpack Dressing Assistance

We consider a more complex version of the 1 STRAP BACKPACK ASSISTANCE scenario introduced in Chapter 3, where the robot now assists the human in putting both straps of the backpack.

> **Scenario** (2 STRAPS BACKPACK ASSISTANCE) *A 2-manipulator robot assists a human user to put a backpack. The robot performs the task in two steps, one strap at a time, and is free to choose which strap to start with. The placement of the first strap can be performed after the user raises his arm. The robot uses both its manipulators in putting the first strap, moving the strap through the arm of the user, up to the shoulder, and then dropping it there. The placement of the second strap can be performed after the user gives back the second arm. The robot is then to use the other manipulator to place the missing strap, following a similar procedure. The two steps are depicted in Figure 5.11. The robot is taught by demonstration how to perform the movements that place the first and second straps. The robot is provided with a total of* 9 *actions. Four of those are control actions, corresponding to the placement of first and second straps on the right or left sides, i.e.,* $\left\{\text{EXEC}^{\text{right}}_{\text{first}}, \text{EXEC}^{\text{left}}_{\text{first}}, \text{EXEC}^{\text{right}}_{\text{second}}, \text{EXEC}^{\text{left}}_{\text{second}}\right\}$. *These action trigger the execution of the corresponding movement primitive taught to the robot by demonstration. The robot is also provided* 4 *communication actions for*



(a) (b)

Figure 5.11: The robot assisting the user in dressing both straps. (a) The first step of the task. The human has his right arm raised and the robot is assisting in putting the corresponding strap. (b) The second step of the task. The human already has the first strap placed on his shoulder, and is now giving back his left arm, so the robot can assist in putting the corresponding strap. Note how the human was positioned slightly to the left of the robot.

requesting the user to raise or give each one of his arms. Finally, the robot is provided a quit action to be used when the robot is unsure about its ability to perform the task. The state space $\mathcal{X}$ consists of 18 states spanning all possible configurations of the human in terms of arms raised/given and straps placement (Figure 5.12). This includes 3 additional absorbing states that are reached when the robot quits, successfully puts both straps, or when it fails to put a strap. All communication actions have reward -1. Quitting after successfully placing one strap leads to a reward -3, whereas quitting at the beginning leads to reward -6. An unsuccessful execution leads to a reward -10. It is assumed that the user performs the request communicated by the robot 95% of the time, and otherwise stays in the same configuration.

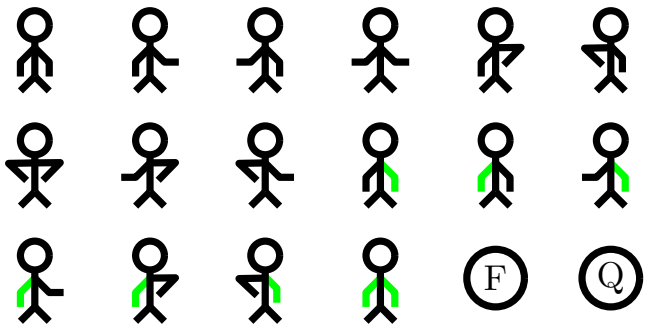


Figure 5.12: The state space in the 2 STRAPS BACKPACK ASSISTANCE scenario. Consists on all configurations of the human in terms of arms raised/given, and straps placement. Additionally, includes a failure and quit states. When the arm of the user is in green, it corresponds to a state where the corresponding strap was successfully placed.

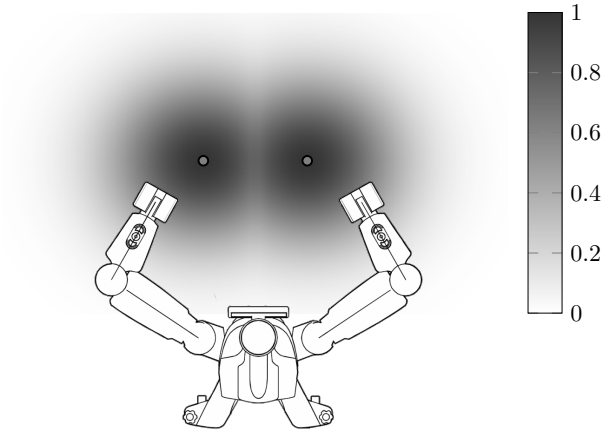


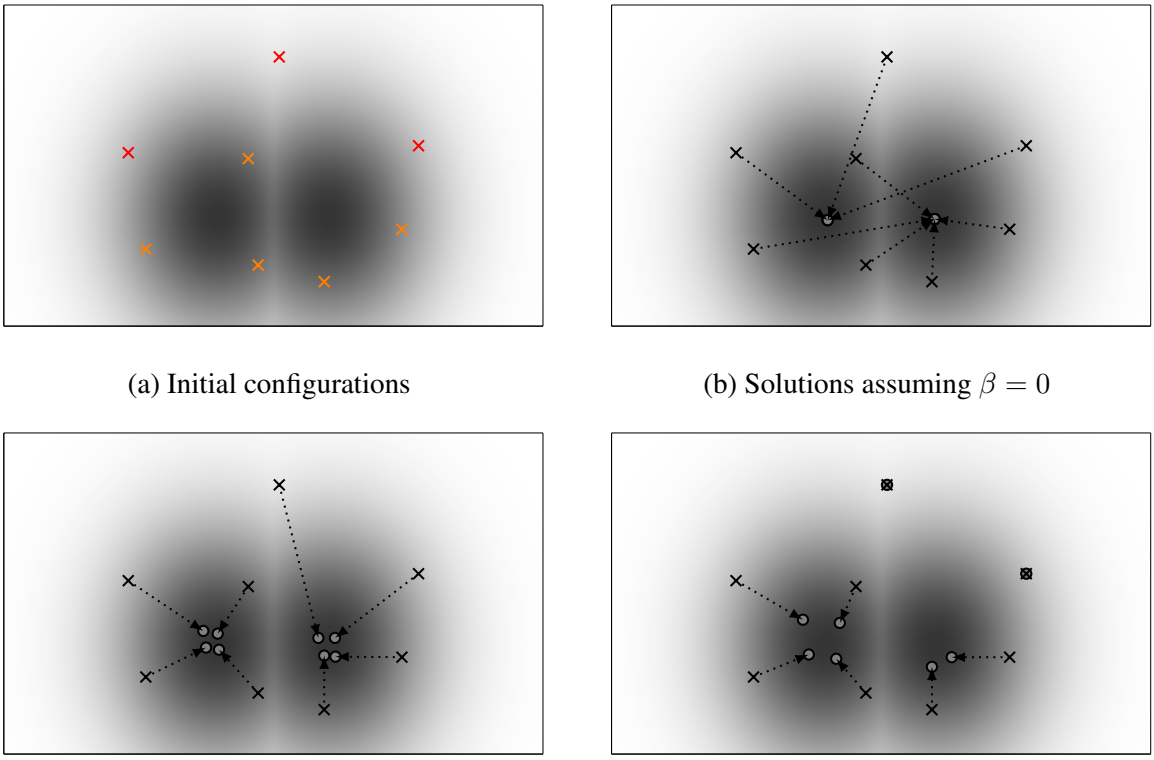
Figure 5.13: The setup of the 2 STRAPS BACKPACK ASSISTANCE scenario, with the two radial kernels that model the execution success probability as the shaded region. Darker corresponds to higher success probability. The kernel on the right models the execution success probability of executing the movements primitives associated with first putting the left strap, and then putting the right strap, as a function of the user's position. This is contrast with the kernel on the left, which regards the symmetric movements primitives pair where first the right strap is placed, and then the left strap.

The aforementioned movement primitives were built by exploiting the symmetry of the task. Specifically, the robot was only taught how to first place the right strap, and then the left strap. This involved collecting 10 trajectories on a real BAXTER robot, alongside the corresponding (x, y) position of the user with respect to the robot. Then, exploiting the symmetry of the task it was possible to generalize these movements primitives for the symmetric case—first placing the left strap and then the right strap [23].

We assume the execution success of these movement primitives depends on the (x, y) position of the user. There is a larger probability of success when the user adopts a position similar to those observed during training. In practice, we assume the execution success probability follows a radial kernel centered at a position $\bar{\boldsymbol{\theta}}$—the average position the user adopted on the trajectories used to train the robot. Due to the symmetry of the task we actually have two radial kernels. One radial kernel models the execution success probability associated with the movement primitives that place the right strap first, and the left strap second. The second radial kernel models the symmetric case, and is associated with the movement primitives that place the left strap first, and the right strap second. Figure 5.13 depicts the space in front of the robot, with the two aforementioned radial kernels.

In this task, we endow the agent with the ability to reason over the impact of the initial location $\boldsymbol{\theta}_0$ of the user. The robot is then allowed to plan over the possibility of having the user moving to a location where there is a higher execution success probability. However, requesting the user to move can be cumbersome and as such, these requests are penalized proportionally with the displacement requested, using a cost function $C(\theta) = \beta\|\boldsymbol{\theta} - \boldsymbol{\theta}_0\|_1$.

We assume that the uncertainty underlying the outcome of a request for the human user



(a) Initial configurations

(b) Solutions assuming $\beta = 0$

(c) Solutions assuming $\beta = 5$

(d) Solutions assuming $\beta = 10$

Figure 5.14: Solutions computed on the stochastic outcomes 2 STRAPS BACKPACK ASSISTANCE scenario. (a) The 8 initial positions of the user considered. Red markers correspond to positions where the robot immediately quit, whereas orange markers correspond to positions where the robot quit after placing one strap. (b) to (d) The solutions computed for increasing values of β.

to move follows a distribution f that takes the form of a multivariate independent Gaussian distribution. Specifically, a Gaussian distribution centered at the requested position $\boldsymbol{\theta}$ and with fixed dispersion $\sigma = 0.05$.

Results

We ran experiments assuming 8 random initial positions of the user across the space in front of the robot. These positions are depicted in Figure 5.14(a) as crosses. Red crosses correspond to initial positions of the human where the optimal policy for the robot is to quit immediately, whereas in orange crosses the robot quits after placing one strap. We observe that, in these initial positions, the robot never tries to assist in the dressing of both straps. This is also supported by Table 5.6, which shows that the average value of the agent when the user adopts these initial positions is -5.59.

We then assessed the solutions of the Stochastic Outcomes Counterfactual MDP assuming different values of cost parameter β. Figure 5.14(b) depicts the solutions for $\beta = 0$, *i.e.*, when there is no cost for requesting the user to move. The results are as expected, with the solutions

Table 5.6: The quality of solutions computed by STOCHASTIC OUTCOMES P-ITERATION in the 2 STRAPS BACKPACK DRESSING ASSISTANCE scenario. The results are averaged over the 8 predefined targets, in 5 randomly seeded runs, and rounded to 2 decimal places. The dispersion reported refers to the standard error of the mean.

2 STRAPS BACKPACK ASSISTANCE			
$J(\theta_0)$	$F(\theta)$		
	$\beta = 0$	$\beta = 5$	$\beta = 10$
-5.59	-3.00 ± 0.00	-4.16 ± 0.16	-5.06 ± 0.26

returned suggesting the robot should request the user to move to the optimal location for executing the movement primitives—the average location of the trajectories. Additionally, we observe that since there is no cost in bothering the user, the solutions returned disregard the displacement requested—that is why the agent does not necessarily request the user to move to the closest optimal position. Figures 5.14(c) and 5.14(d) depict the solutions returned assuming $\beta = 5$ and $\beta = 10$, respectively. We observe that as β increases, the displacements requested to the user become smaller. Assuming $\beta = 10$ we also observe two initial configurations where the robot opts to not request the user to move. This occurs since it too expensive to request the user to move to a position where it was worth it to try to execute the task. From Table 5.6 we observe that even under high cost parameters β the solutions computed are still, on average, more valuable than the 8 initial configurations. For instance, all the solutions returned under $\beta = 5$ lead to optimal policies where the robot tries to place both straps.

5.4 Summary of the Chapter

This chapter tackled the problem of reasoning over the uncertainty underlying possible changes to the world, in the context of Counterfactual MDPs. This problem was motivated by the observation that, in real-life scenarios, even though the agent may be able to imagine a better alternative world configuration, it is possible that the resulting configuration may not necessarily yield the expected one. This is particularly common in human-robot interaction scenarios, where the external intervention for changing the world can be interpreted as assistance requests posed to a human—the result of this assistance will, in general, be stochastic. The resulting model was thus dubbed *Stochastic Outcomes* Counterfactual MDPs.

This model assumes that the uncertainty in the outcome of an external intervention follows some distribution f, with parameters that the agent can reason over/control. For example, these parameters may include the mean or the dispersion of the distribution. The formulation of Stochastic Outcomes Counterfactual MDPs follows that of the standard model, and takes the form of an optimization problem (5.1). The key difference lies in the objective function, as it now takes the form of an expectation over the value of possible world configurations sampled according to the uncertainty distribution f.

In formulating the new model, we concluded that standard Counterfactual MDPs can be reduced to Stochastic Outcomes Counterfactual MDPs. Specifically, by adopting an uncertainty

distribution that, in the limit, takes the form of a Dirac function. This allows us to conclude that, in its general form, Stochastic Outcomes Counterfactual MDPs are also hard to solve. As before, this conclusion provided an important insight that motivated the choice of adopting a gradient-based approach for solving the underlying optimization problem.

The choice of adopting a gradient method unveiled an interesting insight, in that in this new model the exact solution of the gradient can be computed by solving another expected value (5.4) over world configurations sampled according to f. A second interesting insight is that this takes the form of an expected value of the product between the value of a world configuration and the log gradient of the uncertainty distribution f. That is, the gradient computation in Stochastic Outcomes does not require computing the gradient of the optimization problem with respect to the transition probabilities. From a practical perspective, this is interesting as those computations can be expensive in some parameterizations.

We proceeded by proposing two approaches for approximating the gradient via sampling methods. This allowed us to extend the P-ITERATION algorithm to the new class of problems described by Stochastic Outcomes Counterfactual MDPs. We observe a tradeoff regarding the performance of the gradient computations. On one hand, these computations become simpler as the derivatives of the optimization problem with respect to the transition probabilities are no longer necessary to be computed. On the other hand, the estimation of the gradient via sampling methods now requires solving multiple MDPs.

The chapter concluded with an evaluation of the applicability of Stochastic Outcomes Counterfactual MDPs in multiple planning problems, and of the performance of the algorithms proposed. These results showed the significant improvements that can be achieved on the performance of the agent by reasoning over the uncertainty underlying the outcomes of requests for changes to the world. We showed, in particular, examples where the best solution found does not regard the most precise external intervention possible. The standard Counterfactual MDP model would not be able to compute these solutions, as it does not reason over the uncertainty in the changes to the world.

Chapter 6

Related Work

The work in this thesis builds upon two key ideas. The first idea is that, by analyzing how different transition probabilities impact the optimal policy in an MDP, an agent may be able to identify those associated with more rewarding optimal policies. A second idea is that, through some external intervention, the agent may be able to actually operate in a modified version of the world, governed by different transition probabilities.

As such, we now provide an overview of relevant literature structured around these two lines of work. On one hand, we review work that considers the impact of different transition dynamics in the optimal policies in the context of MDPs. On the other hand, we review work that contemplates the situation in which an agent/robot reaches out to external entities (namely humans), enlisting their assistance at different stages of the agent loop: *perception*, *decision*, *execution*, *learning*.

6.1 Reasoning Over Transition Dynamics in MDPs

We now consider classes of MDPs that reason over the impact of transition dynamics on the optimal policies for different purposes. We consider three major classes of problems: Markov decision processes with imprecise probabilities, linearly solvable MDPs, and configurable environments.

6.1.1 Markov Decision Processes with Imprecise Probabilities

Markov decision processes with imprecise probabilities (MDPIPs) are a subclass of MDPs, which allow the representation of uncertainty in the transition probabilities of MDPs. When solving MDPIPs, this uncertainty allows for different objective criteria. For example, it allows the computation of optimal policies under either optimistic or pessimistic assumptions over the true distribution of the transition probabilities. Typical approaches model this uncertainty by constraining the transition probabilities to lie on a prespecified polytope described by a finite number of linear inequalities [60, 78, 79]. Bounded Markov Decision Processes (BMDPs), on the other hand, are a specialization of MDPIPs where the representation of uncertainty is limited to uncertainty intervals, specified independently at different entries of the probability transition

function [17]. This particular uncertainty representation allows for optimal methods that solve BMDPs efficiently under both optimistic and pessimistic assumptions of the transition probabilities.

The optimal policy under pessimistic assumptions is typically called a *robust* policy. The class of MDPs that concerns the computation of robust policies is known as robust Markov decision processes (RMDPs). RMDPs allow the computation of optimal worst-case policies in scenarios where the model of the world is not fully known, but we may have an idea on the magnitude and structure of the underlying uncertainty—this uncertainty is common, for example, in scenarios where the model of the world is learned from data. In their general form RMDPs are NP-Hard, as they can be reduced to the 3-SAT-CNF problem [3]. However, RMDPs can be solved efficiently when the uncertainty set is *s,a*-rectangular [24, 50] or *s*-rectangular [21]. That is, the uncertainty sets are constrained by l_1 norms, and defined independently for each state s and action a, or for each state s, respectively.

The line of work in this thesis, however, is closer to the computation of optimal policies under optimistic assumptions of the transition probabilities. Assuming there is no cost involved in modifying the original world, one could potentially model the set of possible world configurations as uncertainty and, in practice, the best realization of the transition probabilities could be seen as the most rewarding world configuration. Moreover, if the possible changes to the world could be modeled as independent uncertainty intervals, BMDPs could then be used for efficiently computing the optimal policy. Our problem formulation, however, assumes general specifications of possible changes to the world, and a cost associated with such changes.

6.1.2 Linearly Solvable Markov Decision Processes

Another line of related work are *linearly solvable MDPs* (LMDPs) [74]. LMDPs are a class of MDPs with no explicit actions, and where the solver works by modifying some predefined transition probabilities $\bar{\mathbf{P}}$ of an uncontrolled Markov chain, at a cost measured by the KL-divergence function. Typically, the predefined transition probabilities $\bar{\mathbf{P}}$ are defined such that, for each state, the transitions follow a uniform distribution over possible next states—for example, if the agent can move from state x to either state y or z, the default transitions follow $\bar{P}(y \mid x) = \bar{P}(z \mid x) = 0.5$. When solving an LMDP, the solver can modify all default transitions with positive probability, returning a solution P—the transition probabilities that maximize the expected reward of the associated Markov Chain. However, this comes at a cost proportional to the KL-divergence between the default transitions $\bar{\mathbf{P}}$ and the new transitions P. As such, the LMDP solver seeks to optimize the tradeoff between the expected rewards in P, with the aforementioned KL-divergence cost.

The formulation of LMDPs leads to a linear Bellman equation and, as a result, LMDPs can be solved in linear time. In a sense, LMDPs transform a discrete optimization over actions (a regular MDP) to a continuous optimization problem over transition probabilities. Besides being fast to solve, LMDPs are also composable in the sense that, under some assumptions, the solution to an LMDP can be computed as a linear combination of known solutions to other LMDPs [75]. Hierarchical extensions for LMDPs have also been considered [27].

Despite similarities with some of the methods proposed in this thesis, the problem addressed by LMDPs is fundamentally different. In LMDPs, the agent is not actively planning over dif-

ferent world configurations and reasoning over the benefit/cost tradeoff associated with them. The optimization over the transition dynamics is the end result of a problem formulation that emerged from the goal of deriving efficient methods for solving MDPs. Moreover, in LMDPs the controller is free to modify the transition dynamics without any constraints. There are, however, extensions of the LMDP model that allow the specification of some limited constraints, at the expense of less efficient solution methods [9].

6.1.3 Configurable Environments

In recent years, the idea of planning over changes to the environment where an agent operates has received some attention from the planning community. Early examples explored the idea of finding changes to the environment, which encourage a certain behavior by the agent, without directly mandating it [82]. More recent research explored the concept of *goal recognition design*, which aims at computing modifications to the environment that allow for a faster recognition of acting agents' goals [31]. Extensive work has been proposed to consider different instances of the problem, considering non-optimal agents [32], partial observability [33], stochastic actions [76], or different types of modifications to the environment [35].

Other research follows more closely the work developed in this thesis, focusing on the problem of actively planning over the changes to the world with the goal of maximizing the agent's performance, namely, in problems modeled as MDPs. While developed independently from our work, these other approaches bear some similarities with ours. One such approach resembles the alternative formulation we discussed in Section 3.4, and models the requests for changes to the world as a planning problem [34]. Similar to our alternative formulation, this other approach incorporates the different possible transition probabilities as part of the planning state space, and then adds actions that allow the selection of these transition probabilities. The main difference is that our alternative approach models the problem as an (extended) MDP, whereas theirs follows a more classical planning flavour, modeling the problem with a general planning language—PDDL [16]. Modeling the problem as PDDL effectively allows the resulting planning problem to be solved with a generic, off-the-shelf, planner, taking advantage of general heuristics and optimization techniques, such as pruning [36]. One disadvantage of this approach is that it can only consider modifications over a finite set of world configurations. This is in contrast with our formulation of Counterfactual MDPs, which can account for continuous space of possible changes to the world.

A second line work proposed recently that resembles our approach is that of *Configurable* Markov decision processes [45, 46]. Configurable MDPs, while developed independently from our work, bear some similarities in formulating the problem as a joint optimization of the model of the world and optimal policy. Their approach, however, assumes necessarily a global parameterization, with possible world configurations being limited to the convex combination of a finite set of possible worlds given a priori. Then, it optimizes over the convex hull of that set of world configurations, looking for the configuration that maximizes the expected rewards. Our formulation of the problem is more general, in that it allows the specification of a cost function that penalizes changes to the original world configuration, and models the space of possible worlds through a generic parameterization of the transition probabilities. As a result, we have that Configurable MDPs are a particular instance of Counterfactual MDPs. Even if we disregard the cost

component, Counterfactual MDPs do not reduce in polynomial time to Configurable MDPs. As previously discussed in Section 3.3, the transformation of a local parameterization to a global parameterization is exponential in the number of parameters. We conclude the complexity analysis performed for Counterfactual MDPs does not directly apply to Configurable MDPs, as our reduction from 3 SAT CNF makes use of a local parameterization built in polynomial time. This seems to suggest that Configurable MDPs may be an easier problem to solve.

6.2 Agents that request assistance

There are several works, particularly in the robotics community, that have contemplated the situation in which the agent/robot reaches out to human users, enlisting their assistance. This assistance has been considered at different stages of the agent loop: *perception*, *decision*, *execution*. Additionally, it has also been explored in the context of *learning*.

6.2.1 Perception

Agents rely on sensors in order to perceive the world and its state. In many problems, however, the perception of the world is limited. This can be the case, for example due to noisy sensors, or more generally, to the agent being provided with a limited set of sensors.

A common approach for tackling these problems is to model them as *Partially Observable MDPs* (POMDPs), which plan considering a space of *beliefs* over the true state of the world [69], and have been successfully applied to multiple different problems [66, 68]. *Oracular* POMDPs (OPOMDPs) extend the POMDP formalism, allowing the agent to query an always-available *oracle* for full-state knowledge [1]. In OPOMDPs the agent can make these observation requests from any state at a fixed cost. Recent work pointed that these assumptions become unrealistic in scenarios where the help is provided by humans, since there are states where it is unlikely for a human to be available, and there may be different costs associated to asking different humans [54]. *Human Observation Provider POMDPs* (HOP-POMDPs) take this into account, and extend OPOMDPs to plan according to the availability and costs of asking different humans for help [54].

The idea of providing agents the ability of requesting additional perception information has also been considered in multiagent scenarios, for example, in multiagent scenarios that do not assume joint-observability. *Decentralized Sparse Interaction MDPs* (Dec-SIMDPs) augment the action space of each agent with a *coordination* pseudo-action which, when executed, provides access to the state/action information of the other agent [44]. The execution of this coordination action, however, comes at a cost. As a result, the agents learn a trade-off between the benefits arising from the coordination at the expense of executing the coordination action [43].

6.2.2 Planning

A relevant line of work concerns the case when planners are not able to find solutions. In the symbolic planning setting, excuse-making approaches have been proposed for explaining why a

solution for a planning problem could not be found, while at the same time, proposing different initial state conditions so that it becomes possible to generate one [18]. Similar ideas have been explored in the context of temporal problems with the goal of solving over-constrained problems—temporal problems for which no execution strategy can be found that meets all constraints [13]. Solving over-constrained problems requires identifying and weakening some constraints, and a typical approach is to resort to partial constraint satisfaction [5, 47]. However, more recent approaches focus on proposing semantically similar alternatives [81]. Similarly, in motion planning, there are approaches that concern the problem of removing [20] or displacing [19] physical constraints such that there exists a feasible path from start to goal position. In logic specification problems there is also work that looks into revising inconsistent specifications, while minimizing a cost associated with such revisions [29].

6.2.3 Execution

The robotics community has considered the situation in which the robot reaches out to human users, during execution time, enlisting their assistance. A common reason for requesting such assistance is for fault recovery purposes, for example, when the expected state of the world after the execution of an action is different than the state actually perceived. Some works consider these failures, and propose an approach for detecting them, and to generate natural language requests for help, to be posed to human users [39, 72].

The idea of requesting assistance during execution has also been considered in scenarios where multiple robots are deployed in a complex environment, and may request the assistance of remote human operators to deal with unexpected real-world irregularities. In *Human Help Provider* MDPs (HHP-MDPs) [12], a human operator receives an assistance request when an agent ends up in an absorbing state (a state that, once visited, cannot be left) other than the goal, or when the agent fails to make progress for a predefined amount of time. The agents execute in decentralized fashion, and their decision process is modeled with an MDP. However, the system relies on a centralized controller for assigning assistance requests to human operators, considering the assistance type and expected resolution time.

Another line of work explores the idea of using human assistance as a way of augmenting the abilities of the robot. The concept of *symbiotic autonomy* provides robots the ability of explicitly considering their own limitations and the existence of humans in the environment, and to create plans that enroll human assistance in the execution of the task [57]. These ideas have been successfully applied to the scenario of a mobile service robot that navigates on a building, and voices human assistance requests in order to use the elevator [55, 57]. More recently, the same concept has been applied to a dressing task, where the robot is provided the ability of requesting the human to adopt more convenient poses [38, 62].

The approaches discussed so far focused on the questions of which assistance to request, and when to request it. However, there is also work that focuses on how to request human assistance. This problem has been considered in the context of mobile service robots that operate on an office setting [56, 58], or a shopping mall [59], or that need directions from passers-by [77]. There is also research on how the human users react to robots that request assistance, for example, in the context of care robots [40], or in collaborative game scenarios [11].

6.2.4 Learning

The idea of agents that make requests has also been explored in the context of learning. For example, *active* learning is a technique where the learning system (an agent) interactively queries a user for labels, while considering some form of expected information gain [10, 61]. Among others, this idea of active learning has been considered in POMDPs. The MEDUSA algorithm, in particular, considers a class of POMDPs with model uncertainty represented as a Dirichlet distribution [25]. In MEDUSA, the agent keeps a number of models sampled from the Dirichlet distribution, along with the optimal policies for those models. At each timestep the agent selects one of the models and acts according to the corresponding policy. After executing the action, the agent may query an oracle for the true identity of the current state, and this information is then used for updating the parameters of the Dirichlet distribution. Extensions have been considered for non-stationary POMDPs [26].

Similar ideas have been explored in the context of imitation learning. In *inverse reinforcement learning* (IRL)—the problem of recovering the reward function describing a task, given demonstrations of how to perform such task [49]—there are approaches that allow the agent to query an expert for the best action to take at a particular state, thus reducing the amount of policy samples required from the expert [41]. Other approaches adopted similar principles while focusing on directly learning the policy instead [28].

Chapter 7

Conclusions and Future Work

In this chapter we summarize the contributions of this thesis and discuss possible directions for future work.

7.1 Contributions

The main contributions of this thesis are the following.

7.1.1 Counterfactual MDPs

The first contribution of this thesis is the formulation of Counterfactual MDPs—a novel class of MDPs that allows the agent to reason over alternative versions of the world, and to plan over the possibility of actually operating in such worlds when beneficial. In other words, Counterfactual MDPs allow the agent to reason, plan, and act, over the counterfactual *"What if the world were different?"* Our formulation of the model supports the specification of constraints that allow the agent to only reason over possible (or realistic) configurations of the world. In fact, we show that allowing the agent to freely reason over the aforementioned counterfactual may result in the agent expecting unfeasible, yet creative, changes to the world—we saw an example where the agent wishes the laws of physics to be broken, and to be granted the power of "teleporting". Additionally, our formulation of the problem considers that changes to the world may come at a cost. As such, solving a Counterfactual MDP corresponds to maximizing a benefit/cost trade-off over possible changes to the environment.

In this thesis, we focused on planning problems modeled as Markov decision processes, where the dynamics of the world are described in terms of transition probabilities. In this setting, Counterfactual MDPs extend the original MDP model in two ways. First, by allowing the agent to reason over a (constrained) set of possible transition probabilities, and the corresponding impact on the rewards collected by following the associated optimal policy. Secondly, by allowing the agent to plan over the possibility of actually operating in a world configuration governed by such transition probabilities. We formulate the resulting problem as a joint optimization over the transition probabilities and optimal policy of the MDP.

Our second contribution is a complexity analysis of the model proposed. We show that, in general, solving Counterfactual MDPs is hard both in practice and in theory. In fact, from a computational perspective, we formally prove that the problem is NP-Hard, which suggests we should not expect an efficient algorithm for computing the exact solution of a Counterfactual MDP in its general form. This observation motivated the development of the gradient-based approaches proposed in Chapter 4.

We also make contributions on a more practical level, setting the foundation for a wider applicability of Counterfactual MDPs to different planning problems. We propose different methods for specifying the set of possible world configurations through a parameterization of the transition probabilities. In this setting, we introduce two alternative parameterizations, and discuss in detail their advantages and disadvantages from modeling and algorithmic perspectives. These parameterizations are showed in practice in Section 4.4 on multiple planning scenarios. In the process, this section also shows the clear benefits that arise from letting the agent reason over additional possible world configurations. Specifically, we demonstrate how Counterfactual MDPs allow the agent to find more valuable world configurations and optimal policies, on multiple different planning scenarios.

The aforementioned contributions resulted in multiple peer-reviewed scientific publications. Our preliminary approaches to what ultimately resulted in Counterfactual MDPs started with a one-shot decision-theoretic approach [62], and with the action-based alternative formulation discussed in Section 3.4 [63]. We then improved on these approaches introducing the Counterfactual MDPs model [64], and later contributed its complexity analysis, and the detailed discussion on parameterizations [65].

7.1.2 Algorithmic Approaches for Solving Counterfactual MDPs

Counterfactual MDPs are formulated as an optimization problem over the transition probabilities and optimal policy of an MDP. This thesis contributes a gradient-based approach for solving this optimization problem. A key step of this approach regards the computation of the gradient of the joint optimization with respect to the transition probabilities. We derive and propose two methods for computing these gradients. The first method derives directly from the Karush-Kuhn-Tucker conditions of the joint optimization problem. The second method exploits the linear programming structure of MDPs, leading to a more efficient approach. Additionally, we show that the parameterization of the transition probabilities used also impacts the gradient computation—following the chain rule we derive the gradient of the transition probabilities with respect to the local and global parameterizations proposed. The approaches proposed for computing the gradient culminate in the P-ITERATION algorithm—an iterative gradient ascent algorithm that iteratively builds a sequence of world configurations that converge to a local maximum of the optimization problem. In order to tackle the non-convexity, the algorithm performs random restarts starting from different possible initial world configurations.

As a second contribution, we discuss mechanisms for speeding up the methods proposed, so they can be applied in larger problems. We started by observing that one of the main bottlenecks of our approach is that it requires solving an MDP at every gradient step. As such, we explored mechanisms for alleviating the computational effort in solving these MDPs. We started by exploring a factored representation of MDPs typically used for solving large state and

action spaces scenarios. Specifically, we adopted a representation based on Algebraic Decision Diagrams, which allows the compression of the transition dynamics and reward model of the MDP, by factoring similar states. Extending our methods to accommodate this representation required redesigning some steps of the gradient computation that involve matrix inverses and other operations that cannot be efficiently implemented with ADDs. The final approach ends up establishing a trade-off between the improvements in performance when solving the MDPs, and a decrease in performance when computing the gradients. We also discuss a second mechanism that seeds the MDP solving process by reusing information computed in the previous iteration. Our experimental evaluation shows that these techniques improve the overall performance of the algorithm.

The aforementioned contributions also span different publications. The P-ITERATION algorithm was originally introduced with an informal derivation of the fixed policy differentiation method for computing the gradients [64]. Additionally, this publication also included most of the experimental evaluation in Section 4.4. The formal derivation of the gradient methods was introduced in later work [65].

7.1.3 Stochastic Outcomes Counterfactual MDPs

Counterfactual MDPs implicitly assume that the world configuration envisioned by the agent is necessarily materialized. In many real-world scenarios, however, this is not the case. The outcome for a change in the world may not always yield the expected configuration—there exists an underlying uncertainty in the process of modifying the world that depends on the external intervention required. This external intervention may be more/less precise, at the expense of different costs.

We extend the Counterfactual MDPs model to account for these underlying costs and uncertainty, resulting in a model that we dubbed Stochastic Outcomes Counterfactual MDPs. The uncertainty in the outcome of an external intervention to change the world is assumed to follow a distribution. This distribution is parameterized by the desired world configuration, and (possibly) by some additional parameters the agent can tune to control the dispersion. These additional dispersion-controlling parameters can be interpreted as the type of external intervention required. The cost function is also extended to account for these additional parameters. Our formulation of Stochastic Outcomes Counterfactual MDPs follows that from standard Counterfactual MDPs, and takes the form of an optimization problem. The main difference lies in the objective function, where it now takes the form of an expectation over the values of the possible world configurations that may result from a given request.

We show that Counterfactual MDPs can be reduced to Stochastic Outcomes Counterfactual MDP, by adopting an uncertainty distribution that, in the limit, approaches a Dirac. As such, the complexity analysis performed for Counterfactual MDPs extends to the stochastic outcomes variant, allowing us to conclude that, in general, Stochastic Outcomes Counterfactual MDPs are also hard to solve. Again, this motivated the choice of adopting a gradient-based method. We show that in this new model the exact solution of the gradient can be computed by solving an expected value. As such, we propose an approach that approximates this gradient through sampling mechanisms, which allowed us to extend the P-ITERATION algorithm to this new class of problems with stochastic outcomes. An interesting insight was that in our new approach the

computation of the gradient no longer required the derivatives of the MDP optimization problem with respect to the transition probabilities. The disadvantage, however, is that estimating the gradient via sampling requires the solution of multiple MDPs.

Finally, we demonstrate the applicability of Stochastic Outcomes Counterfactual MDPs in different scenarios. We show the significant improvements that can be achieved on the performance of the agent by accounting for the uncertainty underlying the outcomes of requests for changes to the world. Specifically, we show examples where the best solution is not to request changes to the world to the most precise external entity. The standard Counterfactual MDP model would be able to compute these solutions.

7.2 Future Work

This thesis contributed Counterfactual MDPs—a novel planning approach that endows the agent with the ability to reason over alternative configurations of the world, and to plan over the possibility of operating in such worlds when beneficial, thanks to some form of external intervention. We focused in planning problems modeled as MDPs where the dynamics of the world are described in terms of the transition probabilities. In this setting, our model extends MDPs to allow the agent to plan over possible transition probabilities, and we formulate this problem as a joint optimization of the transition probabilities and optimal policy. We constrain this optimization to a set of possible worlds through a parameterization of the transition probabilities. We propose algorithms for solving this optimization and show, in the experimental evaluation, the applicability of the models and performance of the algorithms proposed.

While we envision multiple possible directions for future work, we discuss three of them in more detail. The first direction of future work regards an automatic generation of the set of possible world configurations used in constraining the optimization problem. The second, consists in exploring links between Counterfactual MDPs and linearly solvable MDPs, in order to develop more efficient (approximate) algorithms. Finally, the third direction of future work consists in the implementation of some of our models on real robots.

7.2.1 Generation of Space of Possible World Configurations

Our formulation of Counterfactual MDPs constrains the optimization over a space of possible configurations of the world, in order to prevent unfeasible solutions from being considered. In practice, this space is defined in terms of a parameterization of the transition probabilities of the underlying MDP. In this thesis we discuss different classes of parameterizations, and show the applicability of those parameterizations across multiple planning scenarios. It would be interesting, however, to explore automatic mechanisms for determining possible changes to the world.

We envision a possible method that somewhat "reverses" the approach followed in this thesis. Instead of defining a parameterization that spans the set of possible/feasible worlds, one could specify a base set of rules describing the state transitions the agent should be able to do, in an ideal world with no obstacles or restrictions—*i.e.*, the base dynamics that govern the presence of the agent in the world. Then, by letting the agent explore the environment, it may be possible

to reason over/infer existing obstacles or restrictions, as these essentially block or hinder the expected base dynamics of the world.

Let us consider, one last time, the CORRIDOR scenario. In this case, the base dynamics could be set to only allow deterministic movements to adjacent cells, and no "teleporting". Then, having the agent exploring the world would allow it to realize that some transitions between the upper and bottom rows are not as expected, and thus an obstacle may be present. While these ideas come intuitively within the context of the CORRIDOR scenario, there exist challenges in extending them to more complex problems.

One such challenge regards handling non-deterministic blocks of the base dynamics. This would occur, for example, in the FROZEN LAKE scenario. Similarly to the CORRIDOR scenario, we could specify a set of base rules that only allow deterministic movements to adjacent cells. The ice on the floor, however, would prevent the expected transitions in stochastic fashion—sometimes the agent would move in the expected direction, other times the tires of the agent may lose grip, with the agent slipping and moving in a different direction. Detecting the "slipperiness" of the icy world is challenging as it requires significant exploration by the agent—it would be necessary to test multiple times the different actions at the different states. Additionally, it does not seem trivial how to infer that the "slipperiness" that is common to all states can be controlled by a single variable—the grip level of the tires of the agent.

We envision other challenges, for example, when it comes to the definition of the costs associated to different changes to the world. It may be complex to imprint human intuition to what each of request would cost. Finally, we note that the exploration method proposed seems to bear some similarities with DOORMAX, an algorithm proposed for learning and solving deterministic Object Oriented MDPs (OO-MDPs) [15].

7.2.2 Explore Connections with Linearly Solvable MDPs

In Section 6.1.2 we discussed how linearly solvable MDPs relate to the models proposed in this thesis. We concluded that the problems are different, since in LMDPs the agent does not actively plan over world configurations, nor reasons over the benefits/tradeoffs associated with the changes to the world. Instead, the LMDP solver performs an optimization over the transition probabilities of a Markov Chain, as a mechanism for (approximately) solving standard MDPs in an efficient way.

Despite these differences, there seems to exist interesting links between Counterfactual and linearly solvable MDPs, which could potentially allow for an efficient approximated method for solving our models. Allowing the LMDP solver to (freely) reason over additional transitions seems straightforward. Following the discussion in Section 6.1.2, we only need to modify the predefined transition probabilities $\bar{\mathbf{P}}$ to include some probability for the new transitions. It is then up to the LMDP solver to figure out whether that probability should be increased or decreased in the final solution. We envision, however, two challenges.

The first challenge lies on how to imprint a specific cost to the changes to the world requested. The general LMDP model assumes a cost measured in terms of a KL-divergence. As a result, all changes to the predefined transition probabilities cost the same—those related with the new changes to the world to be proposed, and those related with the optimal policy of the underlying MDP. Possible workarounds may exist by exploiting the compositional properties of LMDPs,

and through a clever introduction of extra intermediate states, transitions to/from those states, and costs associated with reaching those states. However, our preliminary experiments seem to suggest that we will need a mechanism for constraining transition probabilities. While there is work on extending the LMDP model to allow the specification of constrained transitions, this usually comes at the expense of less efficient methods [9].

The second challenge we foresee regards the modeling of more complex changes to the world that affect multiple transition probabilities at once—for example, in the FROZEN LAKE scenario the grip of the robot's tires affects all transitions jointly. Since the LMDP solver can freely modify the default transitions probabilities, it is not clear how to constrain some of the resulting transitions to take the exact same value. Again, it may be possible to devise a workaround by introducing extra states and joint transitions. However, it seems that line of solutions would end up being very task-specific and hard to generalize as an approach applicable to multiple different scenarios.

7.2.3 Experiments with Real Robots

In Chapter 3 we showed a concrete application to robotics of an alternative formulation of Counterfactual MDPs. In particular, we addressed a robotic assisted backpack dressing task. We look forward to testing our main formulation of Counterfactual MDPs in real-robots, ranging from service robots operating in office settings, to manipulators interacting in close proximity with humans. We believe the Stochastic Outcomes Counterfactual MDPs formulation, in particular, can be useful in modeling real-world scenarios where the outcomes of a request for a change in the world are not necessarily deterministic. This observation is supported by the interesting results we introduced in Chapter 5 in different simulated HRI scenarios.

◇ ◇ ◇

This thesis focuses on endowing agents with the ability to reason, plan, and act, over the counterfactual *"What if the world were different?"* We propose a planning paradigm that allows the agent to reason over alternative configurations of the world where more rewarding optimal policies may be possible, and to plan over the possibility of actually operating in such configurations. We envision this planning paradigm can ultimately empower agents through the discovery of configurations of the world where the agents' limitations are mitigated, and their overall performance is improved.

Appendix A

Scenarios

We list here for convenience all the scenarios considered throughout the thesis. We started with three navigation scenarios of practical interest. In particular, the CORRIDOR scenario is inspired by research regarding the deployment of real service robots in an office setting, where the robots are allowed to reach out to nearby humans in order to extend their own abilities—for example, the robot may request the assistance of a nearby user for calling the elevator.

Scenario (CORRIDOR) *Consider the scenario depicted in Figure 3.1(a). A mobile service robot docked at its charging station (located in the top left state) receives a request from a user to pick up a package from the user's office (located in the bottom left state). The robot operates in a corridor located in an office setting, and is able to move in four directions (*UP, DOWN, LEFT, RIGHT*) or stay in place (*STAY*). The robot is rewarded for traversing the corridor—navigating from the top left to the bottom left—in the most efficient way. The obstacles between the top and bottom rows of the corridor actually correspond to doors, which are usually to remain shut. The problem is described as an MDP* $(\mathcal{X}, \mathcal{A}, P_0, r, \gamma)$*, where* $\mathcal{X} = \{A, B, C, D, E, F\}$ *and* $\mathcal{A} = \{\text{UP}, \text{DOWN}, \text{LEFT}, \text{RIGHT}, \text{STAY}\}$*. Each moving action moves the agent deterministically to the adjacent cell in the corresponding direction, except if an obstacle is found. Action* STAY *makes the agent stay in the same cell, and not move. The reward is* -1 *for all state-action pairs except* (G, STAY)*, where it is* 0*.*

Scenario (MAZE) *Consider the environment depicted in Figures 4.3(a) to 4.3(c). The* MAZE *scenario extends the* CORRIDOR *to* $L \times L$ *grids in which every pair of adjacent rows is separated by obstacles except in one of the ends (see Figure 4.3(c) for* $L = 4$*). The goal of the agent is to reach the goal position* G*. The state space* $\mathcal{X}$ *includes all the locations on the grid, and the action space includes the actions* UP, DOWN, LEFT, RIGHT, STAY*. Each moving action moves the agent deterministically to the adjacent cell in the corresponding direction, except if an obstacle is found. The reward is* -1 *for all state-action pairs except* (G, STAY)*, where it is* 0*.*

Scenario (TAXI) *Consider the environment depicted in Figure 4.3(d). An agent must navigate a* 5×5 *maze-like environment, picking up and dropping off a passenger from/to a set of pre-designated locations (marked as* R*,* G*,* Y *and* B *in the figure). The state space* $\mathcal{X}$ *includes the possible positions of the agent, the location of the passenger (in one of the marked sites or inside the taxi) and its destination (one of the marked sites), yielding up to* 500 *states. The action space* $\mathcal{A}$ *includes four moving actions (*UP*,* DOWN*,* LEFT*,* RIGHT*), as well as actions for picking up and dropping off the passenger—*PICK UP *and* DROP OFF*, respectively. The navigation actions move the agent deterministically in the corresponding direction, except if in the presence of the boundaries of the grid, or one of the five gates—Figure 4.3(d) depicts the five gates with thicker lines. The reward function penalizes the agent with* -1 *for every navigation action. Additionally, the reward for a successful drop-off is* 20*, while an unsuccessful drop-off (in the wrong location) penalizes the agent with* -10*.*

$\diamond$

We also considered the FROZEN-LAKE scenario, a popular benchmark problem from the reinforcement learning literature. This scenario is particularly interesting in depicting the applicability of Counterfactual MDPs to engineering related problems, where it may be possible to estimate/model the value, and engineering costs, of changes to the configuration of the world.

Scenario (FROZEN LAKE) *Consider the scenarios depicted in Figures 4.6(a) and 4.6(b). A robot must traverse an icy grid, from an initial state* S *to a goal state* G*, while avoiding the holes* H *where the ice has melted. The movement of the robot is uncertain due to the frozen ice—when moving in a given direction, the wheels of the robot may slip, causing the robot to move in a different way. The state space* $\mathcal{X}$ *includes all the cells of the grid, and the action space is* $\mathcal{A} = \{\text{UP}, \text{DOWN}, \text{LEFT}, \text{RIGHT}, \text{STAY}\}$*. All moving actions are uncertain. For example, when moving left or right, there is a probability the robot moves up or down instead. Similarly, when moving up or down, the robot may end up actually moving left or right. As a result, the transition probabilities associated with a given moving action are sampled uniformly between the three possible directions (barring obstacles). Upon falling on a hole, the robot will stay in that state forever. Action* STAY *is deterministic and keeps the robot in the same state. The reward function takes value* -1 *in all state-action pairs, except* (G, STAY) *where it is* 0*.*

$\diamond$

Finally, we considered three domains that show the applicability of Counterfactual MDPs to human-robot interaction scenarios. In these scenarios the agent reasons over possible changes to the world, where the external intervention required for achieving such changes can be interpreted as taking the form of assistance requests to be posed to a human user.

Scenario (WATER POURING) *A robot is trained to pour water in cups located on top of a table, as depicted in Figure 4.7(a). Specifically, the robot is trained by demonstration how to pour water around a specific position* $\bar{\boldsymbol{\theta}}$*. After training, the robot is then tasked with pouring water in (possibly) different cup locations* $\boldsymbol{\theta}_0$*, as depicted in*

Figure 4.7(b). The robot is provided two actions $\mathcal{A} = \{x, q\}$, including the execution of the trained pouring motion, or "quit", signaling that the robot is not confident in executing the task for the current configuration of the world $\boldsymbol{\theta}_0$. The state space $\mathcal{X} = \{I, S, F, A\}$ includes initial, success, failure *and* absorbing *states. The reward function penalizes the agent with -5 for executing q in state I. In that case, the agent also transitions immediately to state A. An execution x is penalized with reward -1 and has two possible outcomes. The agent transitions to S if it succeeds the task, and to F otherwise. In both outcomes, the agent ultimately transitions to state A, but when transitioning from F receives a penalty of -9.*

Scenario (1 STRAP BACKPACK ASSISTANCE) *A 2-arm manipulator assists a human to put a backpack, as depicted in Figure 3.10. The robot is trained to move a strap of the backpack through the arm of the human user up to the shoulder, and drop it there. The robot is provided with a unique control action $\mathcal{A} = \{\text{EXECUTE}\}$, which triggers the execution of a movement primitive previously taught to the robot by demonstration. The state space $\mathcal{X} = \{I, S, F\}$ includes initial, success and failure states. An execution has two possible outcomes. The agent transitions to S if it succeeds in the task, and to F otherwise. The agent receives a reward 1 when transitioning to state S, or a reward -5 when transitioning to state F.*

Scenario (2 STRAPS BACKPACK ASSISTANCE) *A 2-manipulator robot assists a human user to put a backpack. The robot performs the task in two steps, one strap at a time, and is free to choose which strap to start with. The placement of the first strap can be performed after the user raises his arm. The robot uses both its manipulators in putting the first strap, moving the strap through the arm of the user, up to the shoulder, and then dropping it there. The placement of the second strap can be performed after the user gives back the second arm. The robot is then to use the other manipulator to place the missing strap, following a similar procedure. The two steps are depicted in Figure 5.11. The robot is taught by demonstration how to perform the movements that place the first and second straps. The robot is provided with a total of 9 actions. Four of those are control actions, corresponding to the placement of first and second straps on the right or left sides, i.e., $\left\{\text{EXEC}^{\text{right}}_{\text{first}}, \text{EXEC}^{\text{left}}_{\text{first}}, \text{EXEC}^{\text{right}}_{\text{second}}, \text{EXEC}^{\text{left}}_{\text{second}}\right\}$. These action trigger the execution of the corresponding movement primitive taught to the robot by demonstration. The robot is also provided 4 communication actions for requesting the user to raise or give each one of his arms. Finally, the robot is provided a quit action to be used when the robot is unsure about its ability to perform the task. The state space $\mathcal{X}$ consists of 18 states spanning all possible configurations of the human in terms of arms raised/given and straps placement (Figure 5.12). This includes 3 additional absorbing states that are reached when the robot quits, successfully puts both straps, or when it fails to put a strap. All communication actions have reward -1. Quitting after successfully placing one strap leads to a reward -3, whereas quitting at the beginning leads to reward -6. An unsuccessful execution leads to a reward -10. It is assumed that the user performs the request communicated by the robot 95% of the time, and otherwise stays in the same configuration.*

◇

We conclude with some final remarks and observations on the scenarios considered. First, we recall that our problem formulation is suited for scenarios characterized by continuous sets of possible configurations of the world. All domains considered in this thesis fit into this class of scenarios: in the WATER POURING scenario, the different configurations of the world correspond to the different positions of the cup on top of the table in front of the robot; in the BACKPACK ASSISTANCE scenarios, correspond to the different positions of the human; in FROZEN LAKE, correspond to the different grip profile configurations of the robot; in the navigation scenarios correspond to the openness of the doors through which the robot can move.

One may argue that this last scenario has an artificial flavour, specially when instantiated to the case where the door is assumed to be open by a nearby human user—it may be hard for the human user to carry out a request for opening the door by some percentage value. A possibly more intuitive version of this scenario would only entail two configurations of the world, where the door is either fully open or fully closed. While our formulation of Counterfactual MDPs is not suited to such scenarios with discrete sets of possible worlds, it can still solve approximated versions. Specifically, we can approximate such scenarios through the specification of a cost function that "guides" the optimization to a discrete set of possible solutions. We showed an example of this in practice, in the experimental evaluation of the CORRIDOR scenario, where we introduced the smooth *step function* in (4.17) as the cost function. This effectively allows Counterfactual MDPs to model a broad class of domains.

On that note, it is worth highlighting that Counterfactual MDPs can even model human-robot interaction scenarios, as suggested by our experimental evaluation in the WATER POURING and BACKPACK ASSISTANCE domains. However, we note that in our models the agent reasons over different configurations of the world at planning time, and these worlds are then assumed to be available at execution time. Even though Stochastic Outcomes Counterfactual MDPs extend the original model to account for the uncertainty underlying the possible changes to the world, it still assumes that the reasoning over different worlds is performed at planning time.

In some interaction scenarios, however, it may be interesting to provide the agent with the ability to reason over the different configurations of the world at execution time. Our models could potentially be used in such domains through some form of *replanning*, where at execution time the agent would solve new Counterfactual MDPs as needed. However, this replanning approach is computationally expensive. As such, for the class of scenarios where the agent requests changes to the world at execution time, the alternative action-based formulation discussed in Section 3.4 is more suitable.

Bibliography

[1] Nicholas Armstrong-Crews and Manuela Veloso. Oracular partially observable markov decision processes: A very special case. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, pages 2477–2482, 2007. 6.2.1

[2] Adelchi Azzalini and A. Dalla Valle. The multivariate skew-normal distribution. *Biometrika*, 83(4):715–726, 1996. 5.3.3

[3] J Andrew Bagnell, Andrew Y Ng, and Jeff G Schneider. Solving uncertain Markov decision processes. Technical Report CMU-RI-TR-01-25, Carnegie Mellon University, Robotics Institute, 2001. 6.1.1

[4] R Iris Bahar, Erica A Frohm, Charles M Gaona, Gary D Hachtel, Enrico Macii, Abelardo Pardo, and Fabio Somenzi. Algebric decision diagrams and their applications. *Formal methods in system design*, 10(2-3):171–206, 1997. 4.3.1

[5] Matthew Beaumont, Abdul Sattar, Michael Maher, and John Thornton. Solving overconstrained temporal reasoning problems. In *Australian Joint Conference on Artificial Intelligence*, pages 37–49, 2001. 6.2.2

[6] Dimitri P Bertsekas. *Nonlinear programming*. Athena Scientific, 1999. 4.1.2

[7] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge University Press, 2004. 4.1.1

[8] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016. 4.4.2

[9] Ana Bušić and Sean Meyn. Action-Constrained Markov Decision Processes With Kullback-Leibler Cost. In *Proceedings of the Thirty-first Conference On Learning Theory*, pages 1431–1444, 2018. 6.1.2, 7.2.2

[10] David A Cohn, Zoubin Ghahramani, and Michael I Jordan. Active learning with statistical models. *Journal of artificial intelligence research*, 4:129–145, 1996. 6.2.4

[11] Filipa Correia, Carla Guerra, Samuel Mascarenhas, Francisco S. Melo, and Ana Paiva. Exploring the impact of fault justification in human-robot trust. In *Proceedings of the Seventeenth International Conference on Autonomous Agents and Multiagent Systems*, pages 507–513, 2018. 6.2.3

[12] Nicolas Côté, Arnaud Canu, Maroua Bouzid, and Abdel-Iillah Mouaddib. Humans-robots sliding collaboration control in complex environments with adjustable autonomy. In *Proceedings of the 2012 IEEE/WIC/ACM International Conferences on Web Intelligence and*

Intelligent Agent Technology, volume 2, pages 146–153, 2012. 6.2.3

[13] Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraint networks. *Artificial intelligence*, 49(1-3):61–95, 1991. 6.2.2

[14] Thomas G Dietterich. Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition. *Journal of Artificial Intelligent Research*, 13(1):227–303, 2000. 4.4.1

[15] Carlos Diuk, Andre Cohen, and Michael L. Littman. An object-oriented representation for efficient reinforcement learning. In *Proceedings of the 25th International Conference on Machine Learning*, page 240–247, 2008. 7.2.1

[16] Malik Ghallab, Adele Howe, Craig Knoblock, Drew Mcdermott, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. PDDL—The Planning Domain Definition Language, 1998. 6.1.3

[17] Robert Givan, Sonia Leach, and Thomas Dean. Bounded-parameter Markov decision processes. *Artificial Intelligence*, 122(1-2):71–109, 2000. 6.1.1

[18] Moritz Göbelbecker, Thomas Keller, Patrick Eyerich, Michael Brenner, and Bernhard Nebel. Coming Up with Good Excuses: What to Do when No Plan Can Be Found. In *Proceedings of the Twentieth International Conference on Automated Planning and Scheduling*, pages 81–88, 2010. 6.2.2

[19] Kris K Hauser. Minimum Constraint Displacement Motion Planning. In *Robotics: Science and Systems*, 2013. 6.2.2

[20] Kris K Hauser. The minimum constraint removal problem with three robotics applications. *International Journal of Robotics Research*, 33(1):5–17, 2014. 6.2.2

[21] Chin P Ho, Marek Petrik, and Wolfram Wiesemann. Fast Bellman updates for robust MDPs. In *Proceedings of the Thirty-fifth International Conference on Machine Learning*, pages 1984–1993, 2018. 6.1.1

[22] Jesse Hoey, Robert St-Aubin, Alan Hu, and Craig Boutilier. Spudd: Stochastic planning using decision diagrams. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 279–288, 1999. 4.3.1, 4.3.1, 4.4.3

[23] Pedro Ildefonso, Pedro Remédios, Rui Silva, Miguel Vasco, Francisco S. Melo, Ana Paiva, and Manuela Veloso. Robot assisted dressing with symmetry-based approach in collaborative human-robot interaction. In preparation. 5.3.4

[24] Garud Iyengar. Robust dynamic programming. *Mathematics of Operations Research*, 30 (2):257–280, 2005. 6.1.1

[25] Robin Jaulmes, Joelle Pineau, and Doina Precup. Active learning in partially observable markov decision processes. In *Proceedings of the Sixteenth European Conference on Machine Learning*, pages 601–608, 2005. 6.2.4

[26] Robin Jaulmes, Joelle Pineau, and Doina Precup. Learning in non-stationary partially observable markov decision processes. In *ECML Workshop on Reinforcement Learning in non-stationary environments*, volume 25, pages 26–32, 2005. 6.2.4

[27] Anders Jonsson and Vicenç Gómez. Hierarchical linearly-solvable Markov decision prob-

lems. In *Proceedings of the Twenty-sixth International Conference on Automated Planning and Scheduling*, pages 193–201, 2016. 6.1.2

[28] Kshitij Judah, Alan Paul Fern, and Thomas Glenn Dietterich. Active imitation learning via reduction to iid active learning. In *2012 AAAI Fall Symposium Series*, 2012. 6.2.4

[29] Kim Kangjin, Georgios Fainekos, and Sriram Sankaranarayanan. On the minimal revision problem of specification automata. *International Journal of Robotics Research*, 34(12): 1515–1535, 2015. 6.2.2

[30] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972. 3.2.1

[31] Sarah Keren, Avigdor Gal, and Erez Karpas. Goal recognition design. In *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling*, pages 154–162, 2014. 6.1.3

[32] Sarah Keren, Avigdor Gal, and Erez Karpas. Goal recognition design for non-optimal agents. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 3298–3304, 2015. 6.1.3

[33] Sarah Keren, Avigdor Gal, and Erez Karpas. Goal recognition design with non-observable actions. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 3152–3158, 2016. 6.1.3

[34] Sarah Keren, Luis Pineda, Avigdor Gal, Erez Karpas, and Shlomo Zilberstein. Equi-reward utility maximizing design in stochastic environments. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 4353–4360, 2017. 6.1.3

[35] Sarah Keren, Avigdor Gal, and Erez Karpas. Goal recognition design in deterministic environments. *Journal of Artificial Intelligence Research*, 65:209–269, 2019. 6.1.3

[36] Sarah Keren, Luis Pineda, Avigdor Gal, Erez Karpas, and Shlomo Zilberstein. Efficient heuristic search for optimal environment redesign. In *Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling*, pages 246–254, 2019. 6.1.3

[37] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint:1412.6980*, 2014. 4.4.1, 5.2.2, 5.3.1

[38] Steven D Klee, Beatriz Quintino Ferreira, Rui Silva, Joao Paulo Costeira, Francisco S Melo, and Manuela Veloso. Personalized assistance for dressing users. In *International Conference on Social Robotics*, pages 359–369, 2015. 3.4.1, 6.2.3

[39] Ross A Knepper, Stefanie Tellex, Adrian Li, Nicholas Roy, and Daniela Rus. Recovering from failure by asking for help. *Autonomous Robots*, 39(3):347–362, 2015. 6.2.3

[40] Lara Lammer, Andreas Huber, Astrid Weiss, and Markus Vincze. Mutual care: How older adults react when they should help their care robot. In *Proceedings of the Third International Symposium on New Frontiers in Human–Robot Interaction*, 2014. 6.2.3

[41] Manuel Lopes, Francisco Melo, and Luis Montesano. Active learning for reward estimation in inverse reinforcement learning. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 31–46, 2009. 6.2.4

[42] Moisés Martínez, Fernando Fernández, and Daniel Borrajo. Planning and execution through variable resolution planning. *Robotics and Autonomous Systems*, 83:214–230, 2016. 4.5

[43] Francisco S Melo and Manuela Veloso. Learning of coordination: Exploiting sparse interactions in multiagent systems. In *Proceedings of the Eigth International Conference on Autonomous Agents and Multiagent Systems*, pages 773–780, 2009. 6.2.1

[44] Francisco S Melo and Manuela Veloso. Decentralized MDPs with sparse interactions. *Artificial Intelligence*, 175(11):1757–1789, 2011. 6.2.1

[45] Alberto M Metelli, Mirco Mutti, and Marcello Restelli. Configurable Markov decision processes. In *Proceedings of the Thirthy-fifth International Conference on Machine Learning*, pages 3488–3497, 2018. 6.1.3

[46] Alberto Maria Metelli, Emanuele Ghelfi, and Marcello Restelli. Reinforcement learning in configurable continuous environments. In *Proceedings of the Thirty-sixth International Conference on Machine Learning*, pages 4546–4555, 2019. 6.1.3

[47] Michael D Moffitt and Martha E Pollack. Partial Constraint Satisfaction of Disjunctive Temporal Problems. In *Proceedings of the Eighteenth International Florida Artificial Intelligence Research Society Conference*, pages 715–720, 2005. 6.2.2

[48] Gergely Neu and Csaba Szepesvári. Apprenticeship learning using inverse reinforcement learning and gradient methods. In *Proceedings of the Twenty-third Conference on Uncertainty in Artificial Intelligence*, pages 295–302, 2007. 1, 4.1.2

[49] Andrew Y Ng and Stuart J Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 663–670, 2000. 6.2.4

[50] Arnab Nilim and Laurent El Ghaoui. Robust control of Markov decision processes with uncertain transition matrices. *Operations Research*, 53(5):780–798, 2005. 6.1.1

[51] Alexandros Paraschos, Christian Daniel, Jan Peters, and Gerhard Neumann. Probabilistic movement primitives. In *Advances in neural information processing systems*, pages 2616–2624, 2013. 4.4.2

[52] Joelle Pineau, Geoff Gordon, and Sebastian Thrun. Point-Based Value Iteration: An Anytime Algorithm for POMDPs. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pages 1025–1030, 2003. 1

[53] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014. 2.1, 2.2, 2.2

[54] Stephanie Rosenthal and Manuela Veloso. Modeling humans as observation providers using POMDPs. In *Proceedings of the Twentieth IEEE International Symposium on Robot and Human Interactive Communication*, pages 53–58, 2011. 6.2.1

[55] Stephanie Rosenthal and Manuela Veloso. Mobile Robot Planning to Seek Help with Spatially-Situated Tasks. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, pages 2067–2073, 2012. 6.2.3

[56] Stephanie Rosenthal, Anind K Dey, and Manuela Veloso. How robots' questions affect the

accuracy of the human responses. In *Proceedings of the Eighteenth IEEE International Symposium on Robot and Human Interactive Communication*, pages 1137–1142, 2009. 6.2.3

[57] Stephanie Rosenthal, Joydeep Biswas, and Manuela Veloso. An effective personal mobile robot agent through symbiotic human-robot interaction. In *Proceedings of the Ninth International Conference on Autonomous Agents and Multiagent Systems*, pages 915–922, 2010. 4.4.1, 6.2.3

[58] Stephanie Rosenthal, Manuela Veloso, and Anind K Dey. Learning accuracy and availability of humans who help mobile robots. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011. 6.2.3

[59] Satoru Satake, Takayuki Kanda, Dylan F Glas, Michita Imai, Hiroshi Ishiguro, and Norihiro Hagita. How to approach humans?: strategies for social robots to initiate interaction. In *Proceedings of the Fourth ACM/IEEE International Conference on Human Robot Interaction*, pages 109–116, 2009. 6.2.3

[60] Jay K Satia and Roy E Lave Jr. Markovian decision processes with uncertain transition probabilities. *Operations Research*, 21(3):728–740, 1973. 6.1.1

[61] Burr Settles. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114, 2012. 6.2.4

[62] Rui Silva, Francisco S Melo, and Manuela Veloso. Adaptive symbiotic collaboration for targeted complex manipulation tasks. In *Proceedings of the Twenty-second European Conference on Artificial Intelligence*, pages 863–870, 2016. 3.4.1, 6.2.3, 7.1.1

[63] Rui Silva, Miguel Faria, Francisco S Melo, and Manuela Veloso. Adaptive Indirect Control through Communication in Collaborative Human-Robot Interaction. In *Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017. 3.4.1, 7.1.1

[64] Rui Silva, Francisco S Melo, and Manuela Veloso. What if the world were different? gradient-based exploration for new optimal policies. In *Proceedings of the Fourth Global Conference on Artificial Intelligence*, pages 229–242, 2018. 7.1.1, 7.1.2

[65] Rui Silva, Gabriele Farina, Francisco S Melo, and Manuela Veloso. A theoretical and algorithmic analysis of configurable MDPs. In *Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling*, pages 455–463, 2019. 7.1.1, 7.1.2

[66] Reid Simmons and Sven Koenig. Probabilistic robot navigation in partially observable environments. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1080–1087, 1995. 1, 6.2.1

[67] Fabio Somenzi. Cudd: Cu decision diagram package release 3.0.0, 2015. *URL: http://vlsi. colorado. edu/˜ fabio/CUDD*. 4.4.3

[68] Matthijs T. Spaan, Tiago S. Veiga, and Pedro U. Lima. Decision-theoretic planning under uncertainty with information rewards for active cooperative perception. *Autonomous Agents and Multi-Agent Systems*, 29(6):1157–1185, 2015. 6.2.1

[69] Matthijs TJ Spaan. Partially observable Markov decision processes. In *Reinforcement*

Learning, pages 387–414. Springer, 2012. 6.2.1

[70] Breelyn Styler and Reid Simmons. Plan-time multi-model switching for motion planning. In *Proceedings of the Twenty Seventh International Conference on Automated Planning and Scheduling*, pages 558–566, 2017. 4.5

[71] Richard S Sutton and Andrew G Barto. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998. 3.4.1

[72] Stefanie Tellex, Ross A Knepper, Adrian Li, Daniela Rus, and Nicholas Roy. Asking for help using inverse semantics. In *Robotics: Science and systems*, volume 2, 2014. 6.2.3

[73] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005. ISBN 0262201623. 1

[74] Emanuel Todorov. Linearly-solvable Markov decision problems. In *Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems*, pages 1369–1376, 2006. 6.1.2

[75] Emanuel Todorov. Compositionality of optimal control laws. In *Advances in Neural Information Processing Systems*, pages 1856–1864, 2009. 6.1.2

[76] Christabel Wayllace, Ping Hou, and William Yeoh. New metrics and algorithms for stochastic goal recognition design problems. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 4455–4462, 2017. 6.1.3

[77] Astrid Weiss, Judith Igelsböck, Manfred Tscheligi, Andrea Bauer, Kolja Kühnlenz, Dirk Wollherr, and Martin Buss. Robots asking for directions—The willingness of passers-by to support robots. In *Proceedings of the Fifth ACM/IEEE International Conference on Human-Robot Interaction*, pages 23–30, 2010. 6.2.3

[78] Chelsea C White III and Hany K El-Deib. Parameter imprecision in finite state, finite action dynamic programs. *Operations Research*, 34(1):120–129, 1986. 6.1.1

[79] Chelsea C White III and Hany K Eldeib. Markov decision processes with imprecise transition probabilities. *Operations Research*, 42(4):739–749, 1994. 6.1.1

[80] Yinyu Ye. The simplex and policy-iteration methods are strongly polynomial for the Markov decision problem with a fixed discount rate. *Mathematics of Operations Research*, 36(4):593–603, 2011. 5.3.1

[81] Peng Yu, Jiaying Shen, Peter Z Yeh, and Brian Charles Williams. Resolving over-constrained conditional temporal problems using semantically similar alternatives. In *Proceedings of the Twenty-fifth International Conference on Artificial Intelligence*, pages 3300–3307, 2016. 6.2.2

[82] Haoqi Zhang, Yiling Chen, and David C Parkes. A general approach to environment design with one agent. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence*, 2009. 6.1.3

[83] Stefan Zickler and Manuela Veloso. Variable level-of-detail motion planning in environments with poorly predictable bodies. In *Proceedings of the Nineteenth European Conference on Artificial Intelligence*, pages 189–194, 2010. 4.5