Reliable Decision-Making with Imprecise Models

Sandhya Saisubramanian

RELIABLE DECISION-MAKING WITH IMPRECISE MODELS

A Dissertation Presented

by

SANDHYA SAISUBRAMANIAN

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

February 2022

College of Information and Computer Sciences

RELIABLE DECISION-MAKING WITH IMPRECISE MODELS

A Dissertation Presented

by

SANDHYA SAISUBRAMANIAN

Approved as to style and content by:

Shlomo Zilberstein, Chair

Roderic A. Grupen, Member

Philip Thomas, Member

Shannon Roberts, Member

Ece Kamar, Member

James Allan, Chair of the Faculty
College of Information and Computer Sciences

ACKNOWLEDGMENTS

With much pleasure, I take this opportunity to thank the people who have helped me through this journey.

I consider myself fortunate to have had the opportunity to work with my advisor, Shlomo Zilberstein. He has provided me with the freedom to explore and follow my research passion, helped me grow as an independent researcher, yet was always there when I needed him. I hope to reach his high standards of advising in my career as well. Thank you, Shlomo, for these great years!

I would also like to thank my other dissertation committee members. Rod Grupen's insightful suggestions have been a source of inspiration. Phil Thomas provided valuable suggestions throughout each stage of the dissertation process and during my job search. Shannon Roberts introduced me to human factors research and has been a wonderful collaborator. Her enthusiasm for helping students is inspiring. Ece Kamar has been a wonderful collaborator and mentor. Her insightful feedback has helped me better formulate my ideas. I am grateful to have worked with such a supportive and inspiring researcher.

The former and current members of the Resource-Bounded Reasoning lab have made this journey more enjoyable. Kyle Wray and Luis Pineda have been wonderful peer mentors, productive collaborators, and close friends. Our conversations have helped me grow as a researcher. Richard Freedman helped me settle down when I first moved to Amherst, and has continued to be a supportive friend. I am also thankful to my other fantastic lab members—Connor Basich, Allyson Beach, Abhinav Bhatia, Moumita Choudhury, Saaduddin Mahmud, Shuwa Miura, Minori Narita, John "Pete"

Peterson, and Justin Svegliato. Thank you for listening patiently to my evolving ideas and countless practice talks!

The CICS staff were very supportive. I would like to thank Michele Roberts, Leanne Leclerc, and Eileen Hamel for their exceptional support and helping me navigate graduate school. Michele's efficiency made conference travel smoother and easier.

Other amazing friends with whom I have shared great memories are: Anirudh, Annamalai, Sainyam, Raghav, Rowshan, Myungha, Uma, Nikita, Sandhya, Roopa, Vinitra, Lourdes, Katherine, Parul, and Sparshi. Their presence brightened my stay in Amherst and helped me stay sane during the job search in a pandemic.

This dissertation would not have been possible without the love and support of my family. My parents, Radhika and Saisubramanian, have always had faith in my abilities, encouraged me in all my endeavors, and have been a constant pillar of support. Without the love and sacrifices of my parents and grandparents, I would not have made it this far. My brother, Sankar, supported me and kept me from being overly dedicated to my research. I thank my parents-in-law for their love and support. My husband, Pavan Akula, has believed in me more than I believe in myself. This dissertation would not have been possible without his love, care, and support. Pavan, I love you and I'm excited for our next phase together!

ABSTRACT

RELIABLE DECISION-MAKING WITH IMPRECISE MODELS

FEBRUARY 2022

SANDHYA SAISUBRAMANIAN

B.Tech., PONDICHERRY UNIVERSITY

M.Comp., NATIONAL UNIVERSITY OF SINGAPORE

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Shlomo Zilberstein

The rapid growth in the deployment of autonomous systems across various sectors has generated considerable interest in how these systems can operate reliably in large, stochastic, and unstructured environments. Despite recent advances in artificial intelligence and machine learning, it is challenging to assure that autonomous systems will operate reliably in the open world. One of the causes of unreliable behavior is the impreciseness of the model used for decision-making. Due to the practical challenges in data collection and precise model specification, autonomous systems often operate based on models that do not represent all the details in the environment. Even if the system has access to a comprehensive decision-making model that accounts for all the details in the environment and all possible scenarios the agent may encounter, it may be intractable to solve this complex model optimally. Consequently, this complex, high fidelity model may be simplified to accelerate planning, introducing imprecision.

Reasoning with such imprecise models affects the reliability of autonomous systems. A system's actions may sometimes produce unexpected, undesirable consequences, which are often identified after deployment. *How can we design autonomous systems that can operate reliably in the presence of uncertainty and model imprecision?*

This dissertation presents solutions to address three classes of model imprecision in a Markov decision process, along with an analysis of the conditions under which bounded-performance can be guaranteed. First, an adaptive outcome selection approach is introduced to devise risk-aware reduced models of the environment that efficiently balance the trade-off between model simplicity and fidelity, to accelerate planning in resource-constrained settings. Second, a framework that extends stochastic shortest path framework to problems with imperfect information about the goal state during planning is introduced, along with two solution approaches to solve this problem. Finally, two complementary solution approaches are presented to minimize the negative side effects of agent actions. The techniques presented in this dissertation enable an autonomous system to detect and mitigate undesirable behavior, without redesigning the model entirely.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

A world populated with autonomous systems that simplify our lives is gradually becoming a reality. These systems, also referred to as agents, are *autonomous* in the sense that they can devise and execute a sequence of actions to achieve some given objectives or goals, without human intervention. Today, autonomous systems are deeply integrated into our daily lives through various applications such as intelligent tutoring [27], traffic patrolling [15], emergency response [94], self-driving cars [123], and clinical decision making [8]. This rapid growth in the deployment of autonomous systems has been made possible by the tremendous progress over the past two decades in advancing state-of-the-art artificial intelligence (AI) techniques, primarily focusing on standard metrics such as accuracy, run time, and scalability.

1.1 The Need for Reliable Autonomy

The increased deployment of autonomous systems and their broad societal impacts bring along new challenges, particularly ensuring that these systems operate reliably. The National Institute of Standards and Technology (NIST) defines reliability as "the ability of a system or component to function under stated conditions for a specified period of time, assuming the system specifications adequately capture customer requirements" [85]. Building on this definition, a *reliable autonomous system* can be defined as one that operates as intended, consistently in the real-world. A system may be unreliable even when it is deemed safe for operation. For example, Amazon Alexa heard the phrase "Can you get me a dollhouse" on television and or-

dered a dollhouse for Echo owners who were watching the broadcast [63]. Ordering a dollhouse based on television broadcast is unlikely to be considered as unsafe because it is not physically harmful, but it affects the reliability. Despite recent advances in AI and machine learning, assuring reliable operation of deployed autonomous systems remains a challenge.

Systems that demonstrate promising results when evaluated in structured environments or using well-known benchmarks struggle to produce a similar performance in the open world [45, 67, 69, 106, 111]. This is because current evaluation approaches focus typically on aggregate measures of accuracy, which often hide important shortcomings. In addition, many types of reliability concerns are inherently difficult to detect a priori, since details about the deployment environment may not be fully known during initial testing. As a result, many types of undesirable behavior are discovered after the system is deployed. Improving reliability requires a deeper understanding of the system's behavior, its failures, and its model limitations. Addressing this practical problem has long-term implications for the future of AI and is critical in shaping how users view, interact, collaborate, and trust AI systems.

There are many facets to reliability and many causes of unreliable behavior, which have been addressed by multiple approaches to improve reliability. Approaches to address four causes of unreliable behavior are discussed below. First, reliability is affected when the system performance does not align with user expectations. This mismatch can be addressed by improving the legibility of the system's actions so that the user has realistic expectations and accurate mental model of the system's abilities [25]. Second, reliability is affected if model or policy updates to the system introduce new errors. This can be addressed by ensuring that the system's model updates are compatible with the user's mental model [4], or by ensuring that the policy is updated only if it is provably safer than its current baseline policy [17, 30]. Third, in complex semi-autonomous systems, reliability can be improved by

operating in the right level of autonomy for a task in the given environment. By optimizing its autonomy level using human feedback, the agent can complete its task successfully, while avoiding undesirable behavior [6]. Finally, reliability can also be improved using a metareasoning component that constantly monitors the system performance and enables it to safely recover from failures [109]. In addition to addressing different types of reliability concerns, formal verification methods have been proposed to provide correctness guarantees about the system's behavior [1, 102]. These verification approaches, however, require prior knowledge about what types of undesirable behaviors may arise, and formalizing the requirements precisely for complex systems in open-world environments can be challenging.

This dissertation focuses on designing reliable autonomous systems that devise and execute a sequence of actions that result in desirable outcomes, and addresses unreliable behavior arising from three classes of model imprecision (described in detail in the following section), while minimizing the reliance on humans.

1.2 Model Imprecision

Problems that involve sequential decision-making under uncertainty are typically modeled as Markov decision processes [7] and are solved using automated planning or reinforcement learning techniques, depending on the information available to the agent a priori. This dissertation focuses on automated planning techniques. In *automated planning*, the agent computes a sequence of actions that optimize its objective, by reasoning about the effects of its actions and the events in the environment based on its model parameters. Hence, model fidelity is a key factor affecting agent behavior.

Regardless of how much effort goes into the system design and how much data is available for training and testing, it is infeasible to obtain a perfect description of open-world environments. Even if a comprehensive decision-making model is available, solving it using exact methods may be infeasible. When the computational

resources available for reasoning are limited, the model is simplified to accelerate planning and therefore does not include all the details in the environment. Operating based on such imprecise models can lead to unreliable behavior.

In the past, the AI community has focused on developing techniques that are scalable and optimal in solving complex real world problems, including problems with uncertain reward function [83] and transition function [20, 115]. With increased deployment of autonomous systems, it is critical to recognize and mitigate the undesirable consequences of operating based on imperfect models. Instead of redesigning the model entirely, as additional information becomes available, it is beneficial to leverage the fidelity of the existing model while learning to mitigate undesirable behavior. As George Box noted "All models are wrong, but some are useful" [14].

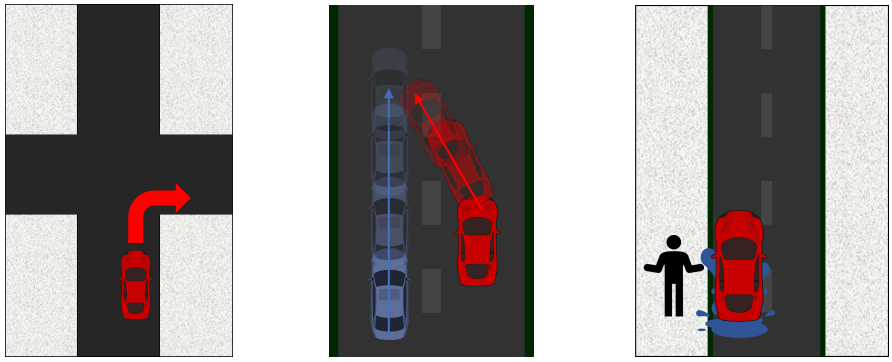
This dissertation focuses on mitigating undesirable behavior when operating under model imprecision that arises due to missing information. In general, there are three broad classes of model imprecision, arising due to:

1. limited computational resources of the agent that require it to simplify the model for planning;

2. limited information that may be available during the design phase, with awareness of missing information; or

3. unawareness of missing information that may be discovered once the agent is deployed.

These three classes of model imprecision represent challenges in decision-making with 'known knowns', 'known unknowns', and 'unknown unknowns' [21]. The first class of model imprecision listed above can be treated as 'known knowns' because the agent can plan with the high fidelity model, if its resources are unconstrained. The second class of model imprecision represents 'known unknowns' since the model indicates what aspects of the environment are uncertain or not fully described in it.



(a) Sharp turn

(b) Unsafe lane change

(c) Driving fast through puddles

Figure 1.1: Examples of unreliable behavior arising due to model incompleteness.

Finally, the third class of model imprecision represents 'unknown unknowns' since the agent does not have knowledge about the unmodeled aspects of the environment when planning with this model.

Different types of model imprecisions lead to different types of agent behaviors. Simplifying the model for planning may result in sub-optimal action selection and may compromise the safety in some states. For example, an autonomous vehicle may perform unsafe actions such as making sharp turns or unsafe lane changes, when its model does not include all the effects of its actions (Figure 1.1 (a-b)).

The effects of an agent's action can be categorized into intended effects and side effects. The intended effects of an action are directly related to the agent's assigned objective. The side effects affect other details in the environment that are typically unrelated to the agent's assigned objective. An agent's action, which is optimal for a given objective, may still be undesirable due to its negative side effects on the environment. For example, an autonomous vehicle optimizing the travel time between locations may drive fast through puddles, splashing water on nearby pedestrians as a negative side effect (Figure 1.1 (c)). While driving fast through puddles may be optimal with respect to its primary objective, the vehicle is expected to slow down so as to avoid such side effects. Human drivers naturally slow down when driving through

a puddle, even when we optimize the travel time. When details about negative side effects are missing in the model, the agent does not have the ability to minimize them. Identifying and mitigating all forms of undesirable behavior is critical for safe and reliable operation.

1.3 Overview of Main Contributions

The deployment of autonomous systems in the open world hinges on the development of risk-mitigating mechanisms to address the consequences of model incompleteness. This dissertation presents frameworks to handle three classes of model imprecision in a Markov decision process (MDP) with discrete, finite states and actions. The solution approaches presented in this dissertation enable autonomous agents operating in the open world to mitigate their undesirable behavior, by addressing the full spectrum of model imprecision. Figure 1.2 presents a visual overview of the main contributions, summarized below.

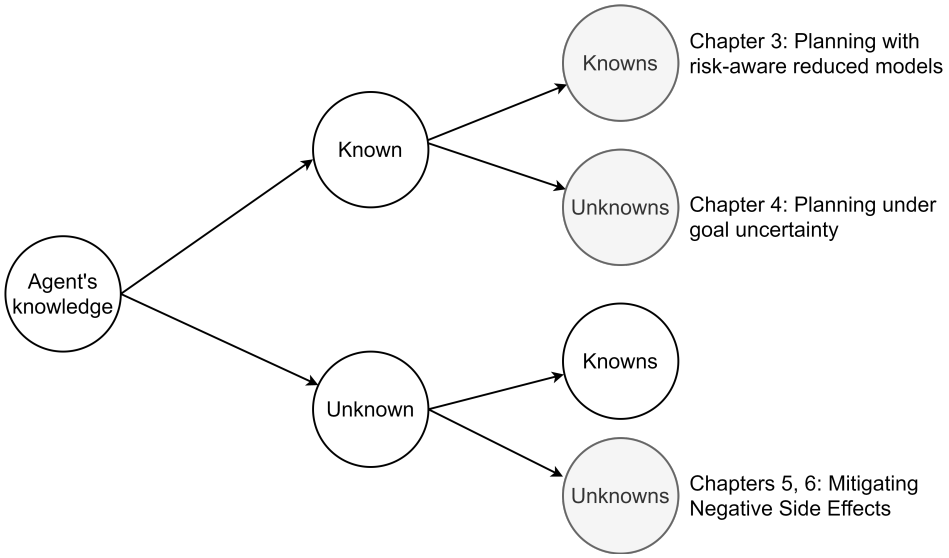


Figure 1.2: Overview of the main contributions.

Adaptive outcome selection for risk-aware reduced models When the computational resources available for reasoning are limited, the agent is required to simplify a complex, high fidelity model of the environment in order to accelerate planning. This problem arises often in field and service robots [108]. This setting, discussed in Chapter 3, is an instance of 'known knowns' since the agent can plan with the high fidelity model if its computational resources are not limited.

An adaptive outcome selection approach is introduced to devise simplified models that efficiently balance the trade-off between model simplicity and fidelity to accelerate planning in resource-constrained settings. The model is simplified or reduced for planning by considering fewer action outcomes, relative to the full model. Adaptive outcome selection approach facilitates selecting different number of outcomes and a different mechanism for selecting the specific outcomes for each state-action pair, using a portfolio of outcome selection techniques. The approach balances the trade-off by improving the model fidelity only in critical states. Theoretical analysis guaranteeing optimal action selection, under certain conditions, is presented.

Planning under goal uncertainty In many problems, model imprecision arises due to the unavailability of precise information about the goal state during system design. The agent is required to plan offline in the presence of goal uncertainty and the true goal state is discovered during execution. An example of this problem is search and rescue [107]. This class of imprecision, discussed in Chapter 4, is an instance of 'known unknowns' since the goal uncertainty is explicitly encoded in the model.

A goal uncertain stochastic shortest path framework is introduced to model problems with goal uncertainty. The imperfect goal information is modeled as a probabilistic distribution over potential goals. The agent's objective is to minimize the expected cost of reaching the true goal, in the presence of goal uncertainty. Two solution approaches are described in detail: (1) a heuristic-based approach, and (2) a determinization-based approach. Theoretical analysis of the framework and the

solution approaches are presented. Multiple tractable sub-classes of this problem are identified.

Avoiding negative side effects In many real-world problems, the uncertainty over unknown aspects of the environment are not modeled. Specifically, the focus is on settings in which a deployed agent is expected to learn about the unmodeled negative side effects of its actions and update its policy to mitigate the impact. This setting, discussed in Chapters 5 and 6, commonly occurs in many deployed systems and is an instance of 'unknown unknowns' since the agent has no prior knowledge about the side effects of its actions.

A formal definition of negative side effects of AI systems is presented. In addition, results from a user study conducted specifically to investigate how users respond to negative side effects and whether these side effects affect user trust in the system's capabilities are discussed. Further, two complementary solution approaches are presented to minimize the negative side effects of agent actions via: (1) learning from feedback, and (2) environment shaping. The solution approaches target settings with different assumptions and agent responsibilities. Algorithms to effectively learn from feedback and for environment shaping, along with a theoretical analysis of their properties are presented.

1.4 Dissertation Organization

The rest of this dissertation is organized as follows. Chapter 2 presents a general background on Markov decision processes (MDPs), stochastic shortest path problems (SSPs), and partially observable Markov decision processes (POMDPs). It also describes the popular solution methods for solving them. Related works specific to a particular chapter are discussed in the respective chapters.

Chapter 3 introduces adaptive outcome selection approach for model simplification. It also presents a sampling-based approach to estimate the reduction impact,

which is then used to guide outcome selection. Theoretical analysis guaranteeing optimal action selection, under certain conditions, is presented.

Chapter 4 presents the goal uncertain SSP framework to model settings with goal uncertainty. It also includes a determinization approach and a heuristic approach to solve the problem, along with a theoretical analysis of their properties.

Chapter 5 presents a formal definition of negative side effects, identifies key characteristics of negative side effects, discusses results from a user study conducted to understand how users respond to negative side effects of deployed systems, and highlights the challenges in mitigating negative side effects.

Chapter 6 introduces learning from feedback approach and an environment shaping approach to mitigate negative side effects. It also includes theoretical analysis guaranteeing bounded-performance of the agent, when its policy is updated to mitigate side effects.

Chapter 7 presents a summary of this work, along with a discussion of the conclusions drawn and directions for future research.

CHAPTER 2

BACKGROUND

This chapter presents formal definitions of Markov decision processes and stochastic shortest path problems, along with a discussion of their common variants and popular solution approaches to solve them.

2.1 Frameworks for Sequential Decision Making

Sequential decision making problems are those in which the agent must optimize a sequence of decisions it makes. *Automated planning* or AI planning research studies how an agent can devise a plan of action to achieve its goals [88]. The agent plans using a model of the environment. In the real-world, an agent's actions typically have probabilistic outcomes. To plan effectively, this uncertainty must be taken into effect. Probabilistic planning or planning under uncertainty research studies how an agent can maximize the expected reward or minimize the expected cost of reaching a goal, under probabilistic action outcomes.

There are two popular frameworks to model such problems: Markov decision processes (MDP) stochastic shortest path (SSP) problems. This dissertation focuses on probabilistic planning settings and the environments of interest are modeled as an MDP or an SSP.

2.1.1 Markov Decision Process (MDP)

Markov decision process is a popular framework used to model sequential decision-making problems. MDPs are an extension of Markov chains. At each decision step,

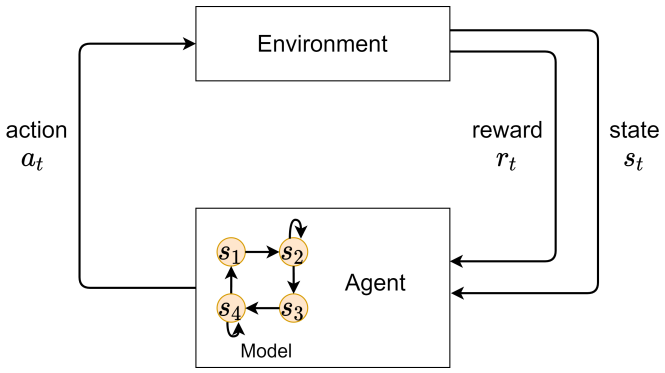


Figure 2.1: An illustration of agent interaction with the environment.

the decision-maker (an autonomous agent in this dissertation) chooses an action to perform in the state. The agent then receives a reward and the process transitions to a successor state (Figure 2.1). The transition dynamics in an MDP follows the *Markov assumption* or the *Markov property* which states that the successor state depends on the current state and the action, and is independent of the previous history of states and actions. The probability of transitioning to a successor state, given a state and an action, is determined by the transition function. The reward associated with executing an action in a state is determined by the reward function. In a stationary MDP, the transition and reward dynamics do not vary with time. MDPs assume full observability—the agent knows its current state of the environment and has no uncertainty about which state it is currently in. An MDP variant that extends to settings with partial observability is discussed later in the chapter.

This dissertation focuses on a specific sub-class of MDP with discrete time, finite states, finite actions, over an infinite horizon with a discount factor. A formal definition of an MDP is presented below. Other nuances and variants of MDPs that are specific to a particular chapter are discussed in the respective chapters.

Definition 1. *A Markov decision process [7] is defined by the tuple* $\langle S, A, T, R \rangle$ *with*

- S *denoting the set of states;*

- *A denoting the set of actions;*
- *$T : S \times A \times S \rightarrow [0, 1]$ is a transition function denoting the probability of transitioning to a state $s' \in S$, when executing action $a \in A$ in state $s \in S$, $T(s, a, s') = Pr(s'|s, a)$; and*
- *$R : S \times A \rightarrow \mathbb{R}$ is a reward function, denoted by $R(s, a)$, prescribing the reward incurred when executing an action $a \in A$ in state $s \in S$.*

At each time step, the agent executes an action a in state s and receives a reward according to $R(s, a)$. The process then transitions to a successor state probabilistically, according to $T(s, a, s')$. This process continues until the horizon h is reached. Figure 2.2 illustrates a dynamic Bayesian network description of an MDP.

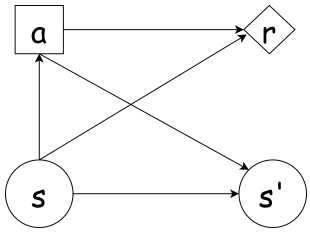


Figure 2.2: A dynamic Bayesian network describing an MDP.

The action executed by the agent in each state is determined by its *policy* π. The agent always executes the same action in a given state when following a deterministic policy $\pi : S \rightarrow A$, which is a mapping from states to actions that satisfies the given objective. When following a stochastic policy $\pi : S \rightarrow \Delta^A$, the agent selects an action to execute by drawing from a probability distribution encoded by the policy. In an infinite-horizon MDP, $h = \infty$, and the discount factor $\gamma \in [0, 1)$. For any infinite horizon discounted MDP, there exists a deterministic policy that is optimal [79].

An agent's objective is to compute a policy that maximizes its expected reward over the horizon:

$$\mathbb{E}\big[\sum_{t=0}^{\infty} \gamma^t R(s^t, \pi(s^t))|\pi\big],$$

with s^t denoting the state at time t, $\pi(s^t)$ denoting the action prescribed by the policy for the current state and $R(s^t, \pi(s^t))$ denotes the corresponding reward. A policy is evaluated using the value function of the states and the Q-values of state-action pairs, computed using the Bellman equation. The value function of states is computed as:

$$V^\pi(s) = R(s, \pi(s)) + \gamma \sum_{s' \in S} T(s, \pi(s), s') V^\pi(s'), \forall s \in S. \tag{2.1}$$

The Q-value of a given state-action pair (s, a) is computed as:

$$Q^\pi(s, a) = R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V^\pi(s). \tag{2.2}$$

An optimal policy is denoted by π^* and is calculated using the Bellman optimality equation:

$$V^*(s) = \max_{a \in A} R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V^*(s'), \forall s \in S. \tag{2.3}$$

From Equations 2.2 and 2.3, $V^*(s) = \max_{a \in A} Q^*(s, a)$. Optimal policies are not necessarily unique and all optimal policies have the same $V^*(s)$. Equation 2.3 is a contraction operator on the Banach space of value functions with Lipschitz constant γ and max norm metric. Using Banach's fixed point theorem, a unique solution can be computed. Therefore, in an infinite-horizon MDP, policies can be independent of time and do not require infinite memory for storage. Policies with this property are called *stationary*.

In a finite horizon MDP, $h \in \mathbb{N}$ with time steps $\mathcal{T} = \{1, \ldots, h\}$, and the policy may be non-stationary. The policy $\pi : S \times \mathcal{T} \to A$ maximizes the expected reward over the finite horizon:

$$\mathbb{E}\Big[\sum_{t=0}^{h} \gamma^t R(s^t, \pi(s^t, t)) | \pi\Big].$$

For the MDP models considered in this dissertation, there exists a deterministic policy $\pi : S \to A$ with $V^\pi = V^*$ [79]. In the later chapters, references to a pol-

icy indicate a deterministic policy. The complexity of solving this class of MDP is polynomial in the number of states, P-complete.

Important Terminology and Refinements

Initial state and absorbing state In this dissertation, unless otherwise specified, we consider the case where the start state or the *initial state* is known and we denote it as $s_0 \in S$. A state $s \in S$ is called an *absorbing state* if $T(s, a, s) = 1, \forall a \in A$. In many problems, the terminal state is an absorbing state.

Trial A sequence (or history) of state-action pairs visited from the initial state up to the end of horizon or to an absorbing goal state, or of a fixed length, $\{s^0, a^0, s^1, a^1, \ldots, s^h\}$, is called a *trial* in the planning literature.

Factored state representation In many problems, the state space S is described by *state factors* $F_1, \ldots, F_K$ such that $S = F_1 \times F_2 \ldots \times F_K$. Factored state-representation is a convenient form of describing the states and the transition functions. The non-factored representation is called a *flat* state representation. The models described in the following chapters frequently utilize the factored-state representation. While the complexity of solving MDPs is polynomial in the number of states, the enumerated state space of factored MDP may be exponential in the number of state factors.

Reward function notation The reward function in an MDP may depend only on the state ($\hat{R}(s)$), the state and the action ($R(s, a)$), or state, action, and the successor state ($\hat{R}(s, a, s')$). These representations are equivalent and can be rewritten as: $R(s, a) = \hat{R}(s), \forall a \in A$ and $R(s, a) = \sum_{s' \in S} T(s, a, s')\hat{R}(s, a, s')$. Unless otherwise specified, the rest of this dissertation utilizes the standard $R(s, a)$ form to denote the reward function. The reward may also be described in terms of *costs*, which is often negated rewards.

2.1.2 Stochastic Shortest Path Problem (SSP)

The stochastic shortest path (SSP) problem is an MDP with no discounting, an indefinite horizon, and a cost function for actions. In an indefinite horizon problem, the horizon is finite but unknown a priori.

Definition 2. *A stochastic shortest path problem [10] is defined by* $\langle S, A, T, C, s_0, s_G \rangle$ *with*

- S *denoting a finite set of states;*
- A *denoting a finite set of actions;*
- $T : S \times A \times S \to [0, 1]$ *is the transition function denoting the probability of reaching a successor state* $s' \in S$ *when action* $a \in A$ *is executed in state* $s \in S$*,* $T(s, a, s') = Pr(s'|s, a)$*;*
- $C : S \times A \to \mathbb{R}^+ \cup \{0\}$ *is a non-negative cost function specifying the cost of executing an action* $a \in A$ *in state* $s \in S$*, denoted by* $C(s, a)$*;*
- $s_0 \in S$ *is the initial state; and*
- $s_G \in S$ *is an absorbing goal state with* $C(s_G, \cdot) = 0$ *and* $T(s_G, \cdot, s_G) = 1$*.*

The policy in an SSP is deterministic, $\pi : S \to A$, provided it is stationary. A stationary policy is said to be a *proper* policy if the policy is *guaranteed* to reach the goal state, from all states, in a finite number of steps. Otherwise, the policy is called *improper*. States from which the goal state is not reachable are called *dead end* states or dead ends. In practice, SSPs are commonly assumed to have: (1) the existence of a proper policy and (2) all improper policies have infinite cost of reaching the goal.

The objective in an SSP is to minimize the expected cost of reaching a goal:

$$\mathbb{E}\big[\sum_{t} C(s^t, \pi(s^t))|\pi, s_0\big] \tag{2.4}$$

with s^t denoting the current state at time t. The values of the states are estimated using the Bellman equation:

$$V^{\pi}(s) = C(s, \pi(s)) + \sum_{s' \in S} T(s, \pi(s), s')V^{\pi}(s'). \tag{2.5}$$

The Bellman optimality equation is:

$$V^{*}(s) = \min_{a \in A} C(s, a) + \sum_{s' \in S} T(s, a, s')V^{*}(s'). \tag{2.6}$$

This operator is also a contraction, similar to MDPs, but with a weighted max norm metric [10]. SSPs generalize finite and infinite horizon MDPs with discrete time, finite states and finite actions [53]. An overview of this mapping is now described. For any MDP of this category, add an initial state and a goal state. The cost function is the negation of the rewards. The transitions are re-computed as follows. First, zero-cost uniform transitions are added from the initial state to all MDP states. Second, the MDP state transitions are multiplied by γ for the remaining state transitions in the original MDP. Finally, there is a zero-cost transition from these states to the goal with probability γ; the transition probabilities are normalized so that the remaining transitions sum to $1 - \gamma$. Note that the converse mapping does not always exist. Similar to MDPs, the complexity of solving discrete time SSPs with a flat state representation, finite states, and finite actions is $P-$complete.

Important Terminology and Refinements

Solutions for the single initial and goal state can be generalized to problems with multiple initial and goal states, using a zero-cost transitions [53]. In this case, the

existence of a proper policy implies the existence of a policy that reaches *some* goal state among the multiple goals.

Similar to factored-state representation in MDPs, the states in SSPs can also be described using state factors, $S = F_1 \times F_2 \times \ldots F_K$. The complexity of solving discrete time SSPs with finite states, finite actions, and a flat representation is polynomial in the number of states. However, the enumerated state space in the case of a factored representation may be exponential in the number of state factors.

2.1.3 Partially Observable Markov Decision Process

In some settings, the agent may not be able to accurately sense the current state of the world. The agent is expected to operate under partial observability. Such problems are modeled by the Partially Observable Markov Decision Process (POMDP) framework, which are an extension of MDP to partially observable settings. The agent executes actions based on the *observations* about the state of the world and it maintains a belief about its current state.

Definition 3. *A partially observable Markov decision process [105] is defined by* $\langle S, A, T, R, \Omega, O \rangle$ *where*

- S *is a finite set of states;*
- A *is a finite set of actions;*
- $T : S \times A \times S \to [0, 1]$ *is a transition function denoting the probability of transitioning to a state* $s' \in S$*, when executing action* $a \in A$ *in state* $s \in S$*,* $T(s, a, s') = Pr(s'|s, a)$*;*
- $R : S \times A \to \mathbb{R}$ *is a reward function, denoted by* $R(s, a)$*, prescribing the reward incurred when executing an action* $a \in A$ *in state* $s \in S$*;*
- Ω *is a finite set of observations; and*

- $O : A \times S \times \Omega \rightarrow [0, 1]$ *is the observation function that determines the probability of receiving an observation* $\omega \in \Omega$ *upon executing an action* a *and reaching state* s'*, and denoted as* $O(a, s', \omega)$*.*

Similar to the Markov property for transitions, the observation function is also assumed to follow a Markov property. The observation received at time t depends only on the state at t. The agent maintains a *belief*, denoted by b, about the true current state, which is a probability distribution over the states with $b(s) \in [0, 1]$ and $\sum_{s \in S} b(s) = 1$. The set of all reachable beliefs forms the belief space $B \subseteq \Delta^n$, where Δ^n is the standard $(n-1)$-simplex. The agent's initial belief is denoted by b^0. The agent updates the belief $b' \in B$, given the action $a \in A$, an observation $\omega \in \Omega$, and the current belief b. The belief is updated as follows:

$$b'(s'|b, a, \omega) = \eta O(a, s', \omega) b(s) \tag{2.7}$$

where $\eta = Pr(\omega|b, a)^{-1}$ is a normalization constant and $b(s)$ is the current belief. The belief state is a *sufficient statistic* for the history and therefore the belief update follows the Markov property. The objective is to compute a policy that maximizes the expected reward:

$$\mathbb{E}\big[\sum_{t=0}^{h} \gamma^t R(b^t, \pi(b^t))|\pi, b^0\big].$$

Since the true state is unknown to the agent, it computes a policy based on its belief. The optimal value can be calculated using the Bellman optimality equation as follows:

$$V^*(b) = \max_{a \in A} R(b, a) + \gamma \sum_{\omega \in \Omega} Pr(\omega|b, a) V^*(b'_{a\omega}) \tag{2.8}$$

Solving a finite horizon POMDP is PSPACE-complete in the size of the beliefs resulting from the states, actions, and observations [72], and the complexity of solving an infinite horizon POMDP optimally is undecidable [65]. Many approximate

algorithms have been proposed to solve POMDPs, such as PBVI [75], SARSOP [58], and finite state controller-based approaches [2]. Since the problems of interest in this dissertation are modeled as an MDP or an SSP, the solution approaches for solving POMDPs are not discussed in detail.

2.2 Methods for Solving MDPs and SSPs

This section discusses the fundamental algorithms and solution approaches for solving discrete time, finite and infinite horizon MDPs, and discrete time SSPs with finite states and finite actions. These algorithms compute stationary, deterministic policies.

2.2.1 Exact Algorithms

Value Iteration Value iteration (VI) [7] is one of the fundamental algorithms to solve MDPs and SSPs. Value iteration is a dynamic programming approach that finds the optimal value of a state by repeatedly applying the Bellman equation (Equation 2.1), until the residual error is below a pre-specified threshold. The residual error is the difference between the values of the states before and after the application of Bellman equation, $|V(s) - V'(s)|$. The algorithm computes the optimal value for each state and then extracts a policy. Synchronous VI updates the values of all states in a full sweep. Asynchronous VI performs the updates asynchronously and it converges if no state is starved [9].

Policy Iteration Policy iteration [40] is another fundamental algorithm that forms the basis of many approximate solution methods to solve large MDPs and SSPs. An arbitrary policy is first initialized. The algorithm then iteratively evaluates and improves the policy until convergence. In *policy evaluation* phase, the expected utility obtained by executing a policy is computed based on the Bellman equation (Equation 2.1). In *policy improvement* phase, a new policy is computed by selecting the

best action for each state, based on the utilities calculated in the policy evaluation phase. The algorithm terminates when there is no improvement in the policy. Since there are a finite number of policies in a discrete, finite MDP, the algorithm converges in a finite number of iterations. In problems with a large state space, policy iteration generally takes longer to converge in terms of the run time, relative to value iteration.

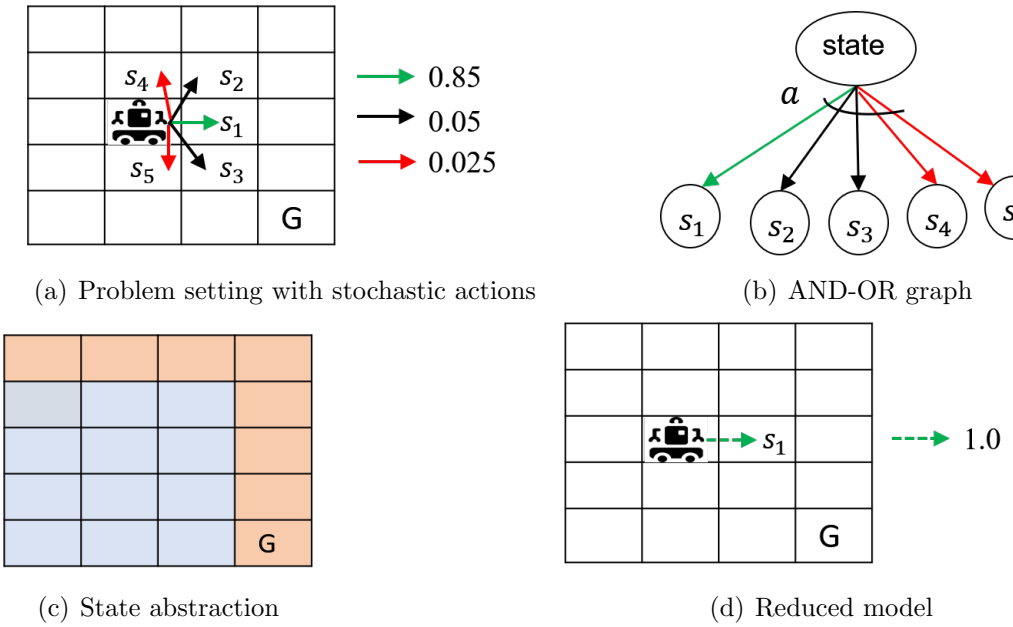
LAO* In many problems of interest, the state space may be large and not all states may be reachable from the start state. LAO* [35] is a heuristic-search algorithm that extends the A* algorithm [36] to MDPs with stochastic transitions and loops. Broadly, LAO* algorithm alternates between node (state) expansion and policy computation. The heuristic function helps accelerate the process by guiding the node expansion. The policy computation involves selecting the best action for each node, using the Bellman equation (Equation 2.1). LAO* converges to the optimal policy and terminates when the residual error is zero or below a pre-defined threshold.

LRTDP Labeled real-time dynamic programming (LRTDP) [12] is an anytime algorithm that generalizes the trial-based real-time dynamic programming (RTDP) [5] with a labeling procedure that speeds up the convergence. A state s is labeled as solved when the heuristic values, and the greedy policy defined by them, have converged (zero or low residual error) over s and all states reachable from s. LRTDP converges to an optimal policy, with fewer trials compared to RTDP.

Both LAO* and LRTDP are known to converge to the optimal policy, faster than value iteration and policy iteration.

2.2.2 Approximate Solution Methods

To solve large MDPs and SSPs, several approximate solution methods have been proposed. The literature on this topic can be classified into: (1) sparse sampling-based approaches and (2) model reductions. The sparse sampling approaches sample trials



(a) Problem setting with stochastic actions

(b) AND-OR graph

(c) State abstraction

(d) Reduced model

Figure 2.3: An example of model simplification for planning. (a) describes the problem setting with the transition probabilities of *move forward* action. (b) shows the AND-OR graph representation for the *move forward* action. (c) illustrates state abstraction where states with the same color are aggregated and have the same policy. (d) illustrates a reduced model with the probabilistic outcomes replaced by a single deterministic outcome for planning.

and rely on dynamic programming, typically using asynchronous value iteration, to compute the policy. Popular examples of sparse sampling approaches are UCT [52], PROST [47], THTS [48], and FLARES [77]. The sparse sampling approaches reduce the search efforts by performing backups on the sampled states.

Another common approach to accelerate planning in large problems is to reduce or simplify the model used for planning, thereby reducing the search space. Common model simplification approaches include state aggregation or state abstraction [66, 61, 82] and reduced models that consider fewer action outcomes [78, 118, 119]. Figure 2.3 illustrates the two model simplification approaches for a simple robot navigation setting. In planning with state abstraction, similar states are grouped together for planning and are assigned the same best action. In planning with reduced models, the policy is computed only for a subset of states. During execution, if a state is

encountered for which the policy does not prescribe an action, then replanning is performed. This dissertation focuses on reduced models that consider fewer action outcomes for planning, relative to the original model.

CHAPTER 3

PLANNING WITH RISK-AWARE REDUCED MODELS

Consider a resource-constrained agent with a high fidelity, complex model of the environment. Since the computational resources of the agent are limited, it cannot generate solutions in real-time with this model. One approach to address this issue is to simplify or reduce this model for planning by considering fewer action outcomes, relative to the original model. Planning with the resulting imprecise model may affect the solution quality and may even result in unsafe agent behavior. While improving the fidelity of the reduced model will help improve the solution quality, it may also affect the planning time and defeat the purpose of using reduced models. To quickly generate high quality solutions with a reduced model, the reduced model must be risk-aware—account for risky outcomes of actions. The key question addressed in this chapter is how to balance the trade-off between simplicity and fidelity of the reduced model, such that the resulting reduced model is risk-aware.

This chapter presents 0/1 reduced model, a paradigm to selectively improve the model fidelity in certain states, thereby improving the solution quality without substantially compromising on the run time gains of using a reduced model [96, 97, 100, 101]. The model fidelity is improved by using more informative action outcomes in certain states and adjusting the costs of actions to account for the outcomes ignored in the reduced model.

Chapter outline. The chapter begins by motivating the problem in Section 3.1. This is followed by a formal definition of reduced models and a discussion of the related works in Section 3.2. The 0/1 reduced model framework is introduced in

Section 3.3. An approach to select outcomes in 0/1 reduced model and adjusting the actions costs is described in Section 3.4. Finally, empirical results are presented in Section 3.5.

3.1 Introduction

Autonomous agents are often faced with tasks that require generating policies quickly and such problems can be modeled using the stochastic shortest path (SSP) framework [10]. Uncertainty in action outcomes, which is a characteristic of many real-world problems, increases the complexity of planning. Given the computational complexity of solving large SSPs optimally [64], there has been considerable interest in developing efficient approximations, such as *reduced models*, that trade solution quality for computational gains. The model may be simplified in many ways, such as aggregating states or actions [61, 82], or ignoring the uncertainty in transitions, thereby reducing the set of reachable states for the planner [78].

This dissertation considers reduced models in which the number of outcomes per state-action pair is reduced relative to the original model, for planning. The action outcomes considered in the reduced model determine the model fidelity and hence different outcome selection techniques result in reduced models with *varying fidelity*.

While reduced models accelerate planning by reducing the number of reachable states, they also degrade the solution quality. Solution quality is affected particularly when the "risky" states—states that significantly affect the expected cost of reaching a goal—are ignored in the reduced model. There are many ways to characterize a risky outcome. In this chapter, the notion of risk is associated with higher expected cost, since the agent can recover from the effects of a risky outcome by incurring a penalty in domains of our interest. A reduced model is said to be *risk-aware* if it fully accounts for the risky outcomes of the complete model, thus resulting in improved solution quality.

Knowledge gaps Existing reduced model techniques have focused on the reduction of planning time [118, 120, 46, 49, 119] in isolation and they do not address explicitly the risks associated with ignored outcomes. This has limited the applicability of reduced models in spite of the substantial computational savings they offer, particularly when avoiding risky outcomes is critical. Recently, reduced models that consider k primary outcomes and l exceptions were introduced, which improved the solution quality compared to traditional model reductions [78]. This shows that the solution quality can be boosted by accounting for certain outcomes. However, no principled approach to outcome selection that can balance the trade-off between model simplicity and fidelity exists. While the risk-awareness and model fidelity can always be improved by considering the full model, this defeats the purpose of using reduced models. How to formulate reduced models that balance the trade-off between model simplicity and risk awareness (Figure 3.1)?

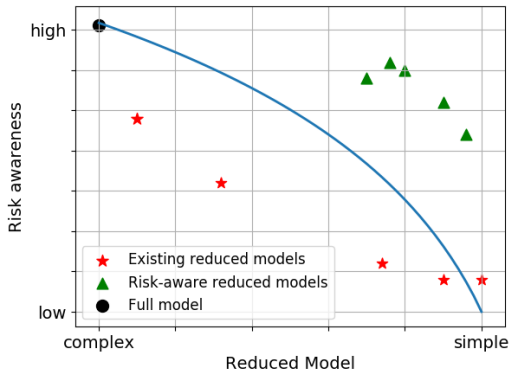


Figure 3.1: An example illustrating the trade-off between model simplicity and risk awareness. Risk awareness decreases as the model is simplified using uniform outcome selection principles (indicated by the blue trend line). The points denote reduced models formed with different outcome selection principles.

Overview of contributions Different outcome selection techniques produce reduced models of varying fidelity, but selectively improving the model fidelity in certain states requires integrating various outcome selection techniques. As Marvin Minsky noted "The power of intelligence stems from our vast diversity, not from any single,

perfect principle" [68]. This chapter presents planning using 0/1 reduced models (0/1 RM), a novel planning paradigm that enables formulating reduced models with different levels of details by switching between using a deterministic model and the full model. A factored state representation is considered and the risks are characterized in terms of the state features. When the features representing risks in the problem are not given, this information is gathered by querying an oracle (typically a human). To identify states where model fidelity is to be improved, a *reduction impact* is estimated automatically based on the features characterizing the risky states, by generating and solving sample trajectories. Intuitively, the reduction impact measures how optimistic the resulting reduced model would be, with respect to risks, thus offering a heuristic for choosing the outcome selection principles. In states where the reduction impact is high, more informative outcome selection principles are employed.

The reduction impact is also used to adjust the costs of actions in the reduced model as a means to account for the effects of the ignored outcomes. The conditions under which a cost adjusted reduced model produces optimal action selection are presented. Finally, the approach is evaluated on three domains in simulation: an electric vehicle charging problem using real world data, and two benchmark problems from the planning literature. The results demonstrate the efficiency of the approach in balancing the trade-off between risk awareness and model simplicity, without compromising the run time gains.

The primary contributions in this chapter are: (1) introducing 0/1 reduced models; (2) estimating the reduction impact and using it as a heuristic for outcome selection; (3) introducing cost adjusted reduced models and its key properties; and (4) evaluating the approach on three domains in simulation.

3.2 Planning with Reduced Models

The complexity of solving SSPs optimally [64] has led to the use of approximation techniques such as reduced models. Reduced models simplify planning by considering a subset of outcomes, which is especially useful in problems with a high branching factor for action outcomes. Let $\theta(s,a)$ denote the set of all outcomes of (s,a), $\theta(s,a) = \{s' | T(s,a,s') > 0\}$.

Definition 4. *A **reduced model** of an SSP M is represented by the tuple $M' = \langle S, A, T', C, s_0, S_G \rangle$ and characterized by an altered transition function T' such that $\forall (s,a) \in S \times A, \theta'(s,a) \subseteq \theta(s,a)$, where $\theta'(s,a) = \{s' | T'(s,a,s') > 0\}$ denotes the set of outcomes in the reduced model for action $a \in A$ in state $s \in S$.*

The probabilities of the outcomes included in the reduced model are normalized. An *outcome selection principle* (OSP) determines (1) the number of outcomes and (2) how the specific outcomes are selected. Depending on these two aspects, a spectrum of reductions exist with varying levels of probabilistic complexity that ranges from the single outcome determinization to the full model. The OSP can be a simple function such as always choosing the most-likely outcome or a more complex function.

Uniform outcome selection Traditionally, a reduced model is characterized by a single OSP—a single principle is used to determine the number of outcomes and how the outcomes are selected across the entire model. Among the different existing uniform outcome selection principles that have been explored, determinization has attracted significant interest. *Determinization* ignores the stochastic transitions and associates one deterministic outcome for each action, instead of multiple probabilistic outcomes [118, 46]. The resulting deterministic model can be efficiently solved using off-the-shelf classical planners such as A* [36]. It is complemented by online replanning when an unexpected state is encountered.

Interest in determinization increased after the success of *FF-Replan* [118], which won the 2004 International Probabilistic Planning Competition (IPPC). FF-Replan

generates a deterministic version of the problem and computes a sequential plan for solving it, using the *Fast Forward* (FF) technique [38]. FF performs forward search using a heuristic function that is automatically extracted from the domain description by ignoring the delete list. If an unexpected state—a state without an action plan—is reached during plan execution, the process repeats with the unexpected state as the initial state, until a goal state is reached. Following the success of FF-Replan, researchers have proposed various methods to improve determinization, such as *Robust FF* (RFF) [110] and *HMDPP* [49], resulting in various determinization-based planning algorithms [43, 46, 47, 54, 119, 120].

While determinization could be extremely effective, it is often hard to predict when it will work particularly well as policies produced via existing determinization techniques do not provide bounded-performance guarantees. Recently, M_l^k reductions that account for a bounded number of exceptions (k) and primary outcomes (l) have shown to significantly improve the solution quality [78]. This shows that the solution quality can be boosted by accounting for certain outcomes. However, the values of k and l are fixed for a problem and determining the best values a priori is non-trivial.

Due to the rigidity of the uniform outcome selection, the existing reduced models are incapable of selectively adapting to risks. Figure 3.2 illustrates this with a robot making a right turn in a narrow corridor (Figure 3.2(a)) and the corresponding full model (Figure 3.2(b)). The robot can turn at two speeds: low and high, each succeeding with different probabilities. If the agent crashes (hits the wall), a penalty is applied to reposition it back to the center of the corridor. While the optimal action in the original model is to turn at a low speed, the optimal action in the most-likely outcome determinization (Figure 3.2(c)) is turning at a high speed. Hence, using a uniform outcome selection for this problem results in a sub-optimal solution, which can be avoided using the portfolio approach (Figure 3.2(d)) described below.

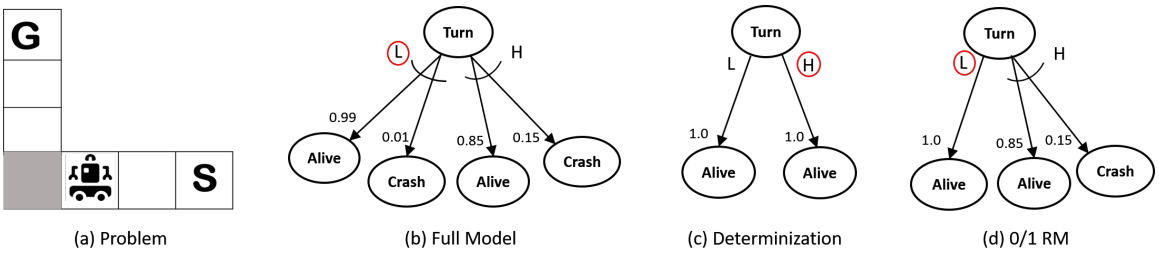


Figure 3.2: An example of reduced models with different fidelity for a robot making a turn in a narrow corridor. The shaded area in (a) denotes "turn" state, and 'S' and 'G' denote the start and goal state respectively. The robot can turn at low speed (L) or at high speed (H) with costs proportional to the time taken to turn. The robot is alive if the action succeeds or may crash when the action is unsuccessful. Optimal action in each model is circled.

3.3 0/1 Reduced Models

A **p**ortfolio of **o**utcome **s**election **p**rinciples (POSP) is a generalized approach to formulate risk-aware reduced models by switching between different OSPs. The approach is inspired by the benefits of using portfolios of algorithms to solve complex computational problems [74]. A *model selector* selects an outcome selection principle for each state-action pair.

Definition 5. *Given a portfolio of finite outcome selection principles,* $\mathrm{Z} = \{\rho_1, \rho_2, ..., \rho_k\}$, $k>1$, *a* ***model selector*** Φ *generates* T' *for a reduced model by mapping every* (s, a) *to an outcome selection principle,* $\Phi : S \times A \rightarrow \rho_i$, $\rho_i \in \mathrm{Z}$, *such that* $T'(s, a, s') = T_{\Phi(s,a)}(s, a, s')$, *where* $T_{\Phi(s,a)}(s, a, s')$ *denotes the transition probability corresponding to the outcome selection principle selected by the model selector.*

The rest of this chapter focuses on a basic instantiation of POSP — 0/1 RM — that switches between the extremes of outcome selection principles: determinization and the full model.

Definition 6. *A* ***0/1 reduced model*** *(0/1 RM) is characterized by a model selector,* $\Phi_{0/1}$, *that selects either one or all outcomes of a state-action pair to be included in the reduced model.*

In a 0/1 RM, the model selector either ignores the stochasticity completely (0) by considering only one outcome of (s, a), or fully accounts for the stochasticity (1) by considering all outcomes of the state-action pair in the reduced model. An instantiation of 0/1 RM for the robot navigation in Figure 3.2 uses the full model for the riskier action H and determinization for action L. A 0/1 RM that guarantees goal reachability with probability 1.0 can be devised, if a proper policy exists in the SSP.

Existing reduced models, such as determinization, are a special case of POSP, with a model selector that always selects the same OSP for every state-action pair. In planning using a portfolio of OSPs, however, the model selector typically utilizes the synergy of multiple OSPs. Each state-action pair may have a different number of outcomes and a different mechanism to select the specific outcomes (Figure 3.3).

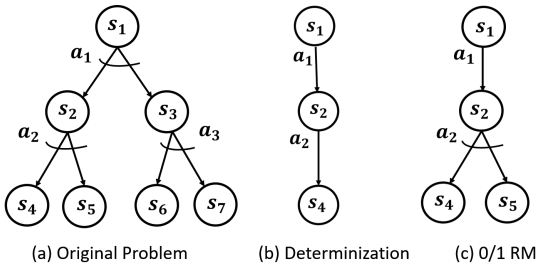


Figure 3.3: An example of reduced models formed with different techniques.

This flexibility in outcome selection can be leveraged to improve risk awareness in reduced models by using more informative outcomes in the risky states and using simple outcome selection principles otherwise. Though the model selector may use multiple OSPs to generate T' in a POSP, note that the resulting model is still an SSP. This framework offers both a unifying lens to view past work (existing OSPs), as well as a recipe for seamless integration of future improvements in OSPs. Depending on the model selector and the portfolio, a large spectrum of reduced models exists for an SSP and selecting the right one is non-trivial.

3.3.1 Model Selector (Φ)

The model selectors in existing reduced models have been devised typically to reduce planning time. An efficient Φ in a POSP optimizes the trade-off between solution quality and planning time. Devising an efficient model selector automatically can be treated as a meta-level decision problem that is computationally more complex than solving the reduced model, due to the numerous possible combinations of OSPs. Even in a 0/1 RM, devising an efficient Φ is non-trivial as it involves deciding when to use the full model and when to use determinization. This is illustrated by the following example.

Consider the navigation example in Figure 3.2, with state representation as a tuple $\langle l, w, c \rangle$ where l denotes the location, w is the width of the corridor indicating whether it is wide or narrow, and c denotes if the agent has crashed or not. The agent can move forward or backward or turn in two speeds (L and H), as in the figure. The forward and backward actions are deterministic with unit cost and the low speed and high speed turns are stochastic with costs +2 and +1 respectively. Models with different fidelities are generated by altering when the full model is used. As expected, the highest gains are observed when the full model is used in the turn state. Specifically, using the full model at start state alone had an expected cost of 21.147 ± 0.35 and using the full model in the shaded state, which represents the narrow turn state (as in Figure 3.2(d)), had an expected cost of 8.188 ± 0.10, averaged over 100 trials. Identifying the best 0/1 RM for this problem required testing all valid formulations using most likely outcome determinization, resulting in a total run time of 2651 milliseconds. Hence this exhaustive evaluation is infeasible for larger problems.

In the worst case, all OSPs in the portfolio, Z, may have to be evaluated for a problem to determine the best reduced model formulation. Let τ_{max} denote the maximum time taken for this evaluation across all states. When every action transitions to all states, the outcome selection principles in Z may be redundant in terms of

the specific outcomes set produced by them. For example, selecting the most-likely outcome and greedily selecting based on heuristic could result in the same outcome for certain (s, a) pair. Using this, we show that the worst case complexity for a model selector is independent of the size of the portfolio, which may be very a large number in the worst case.

Proposition 7. *The worst case time complexity for a model selector to generate T' for a POSP is $\mathcal{O}(|A|2^{|S|}\tau_{max})$.*

Proof. For each (s, a), at most $|\mathrm{Z}|$ outcome selection principles are to be evaluated and this takes at most τ_{max} time. Since this process is repeated for every (s, a), Φ takes $\mathcal{O}(|S||A||\mathrm{Z}|\tau_{max})$ to generate T'. In the worst case, every action may transition to all states and the outcome selection principles in Z may be redundant in terms of the specific outcomes set produced by them. Hence, the evaluation is restricted to the set of unique outcomes sets denoted by k, $|k| \leqslant |\mathcal{P}(S)|$, with $\mathcal{P}(S) = 2^{|S|}$. Then, it suffices to evaluate the $|k|$ outcome sets instead of $|\mathrm{Z}|$, reducing the complexity to $\mathcal{O}(|A|2^{|S|}\tau_{max})$. □

Proposition 8. *The worst case time complexity for $\Phi_{0/1}$ to generate T' for a 0/1 RM is $\mathcal{O}(|A||S|^2\tau_{max})$.*

Proof. This proof is along the same lines as that of Proposition 7. To formulate a 0/1 RM of an SSP, it may be necessary to evaluate every outcome selection principle, $\rho_i \in \mathrm{Z}$, that corresponds to a determinization or a full model. Hence, in the worst case, $\Phi_{0/1}$ takes $\mathcal{O}(|S||A||\mathrm{Z}|\tau_{max})$ to generate T'. The set of unique outcomes, k, for a 0/1 RM is composed of all unique deterministic outcomes, which is every state in the SSP, and the full model, $|k| \leqslant |S| + 1$. Replacing $|Z|$ with $|k|$, the complexity is reduced to $\mathcal{O}(|A||S|^2\tau_{max})$. □

Since τ_{max} could significantly reduce the run time savings of using a reduced model, these results underscore the need for faster evaluation techniques to identify relevant

OSPs. The current best approach to evaluate an OSP is to solve the corresponding reduced model and evaluate the policy in hindsight. The following section presents an approximation for outcome selection.

3.4 Solution Approach

The proposed solution approach involves two components. First, given features that characterize the risks in the setting, a reduction impact is calculated for each state-action pair, which is used as a heuristic to guide the outcome selection. Second, when using determinization, the costs of the actions can be adjusted using the reduction impact to reflect the ignored outcomes. The overall process flow is illustrated in Figure 3.4.

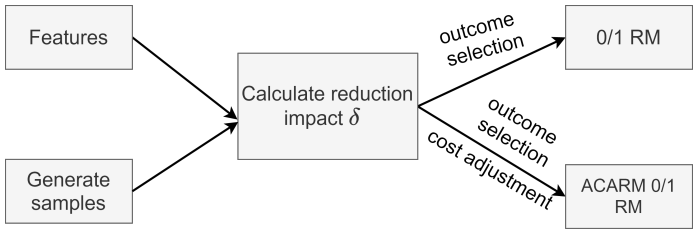


Figure 3.4: Overview of the approach.

In problems where the agent does not have information about the risky states, oracle (human) feedback is utilized. The agent may query an oracle who in turn provides features characterizing risks, denoted by $\vec{f_R}$. In our running example (Figure 3.2), crashing is an unacceptable outcome, which can be denoted by the state feature *c=true*. When the problem instances in a domain are similar or share a structure in terms of state features, actions, and goal conditions, querying once per domain may suffice.

3.4.1 Reduction Impact

One of the reasons for reduced model techniques producing poor solutions is that some outcomes are completely ignored. The reduction impact δ is a measure of the values of ignored outcomes and is calculated for each (s, a). Following π^*, the reduction impact is calculated as,

$$\delta(s,a) = Q^*(s,a) - \sum_{s' \in \theta'(s,a)} T'(s,a,s')V^*(s'), \forall (s,a) \tag{3.1}$$

where $Q^*(s,a)$ denotes the Q-value in the original model. Therefore, the reduction impact is higher if risky states are ignored in the reduced model, due to their significantly higher expected cost of reaching a goal. Since the optimal values are *unknown*, this value is estimated using samples. Sample trajectories generated by depth-limited random walk on the target problem or smaller problem instances from the domain may be used for this purpose. These samples are solved optimally and the reduction impacts corresponding to $\vec{f_R}$ are determined using these exact solutions. The reduction impact, given $\vec{f_R}$, are learned in hindsight by computing the mean values of the samples. More complex methods for aggregating the values from the samples may be considered.

Optimal Reduction Impact

For the class of problems described below, δ can be calculated optimally, without using samples or having to solve the problem.

Consider an SSP in which an action can achieve a successful outcome with probability $1-p$ or fail with probability $p>0$ (Figure 3.5). When an action fails, the state remains unchanged. Let s denote a state of the SSP for which a successful execution of action a with cost $C(s,a)$ results in outcome state s'.

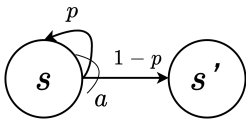


Figure 3.5: Problems with closed-form reduction impact.

For problems with this structure, it has been shown that the Q-values can be calculated optimally [49]:

$$Q^*(s,a) \;=\; \frac{C(s,a)}{1-p} + V^*(s').$$

Substituting the above equation in Equation 3.1, we get

$$\delta(s,a) = \frac{C(s,a)}{1-p}.$$

Thus, for problems with this structure, the reduction impact can be calculated optimally with a closed-form equation.

3.4.2 Outcome Selection Guided by Reduction Impact

The reduction impact can be used as a heuristic for model selection in a 0/1 RM, as it reflects the criticality of the states being ignored. In a 0/1 RM, the full model may be employed in states with approximate reduction impact above a certain threshold and determinization in other states.

By altering the δ threshold at which the full model is employed, reduced models with different levels of computational gains and sensitivity to risks can be produced, due to the differences in fraction of full model usage. To demonstrate this, solution qualities of reduced models formed with different reduction impact thresholds are compared (Figure 3.7) on four instances of the racetrack domain [5][1]. An example

[1]In racetrack domain, the objective is to move from start to goal states by applying acceleration and controlling the car correctly. If the robot (car) hits a wall, it is repositioned back to start state.

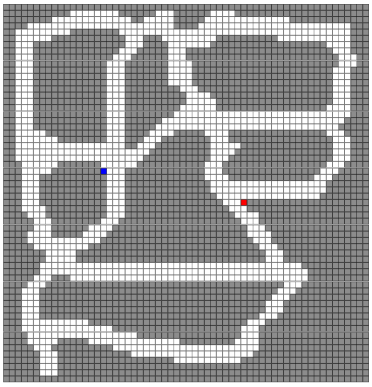
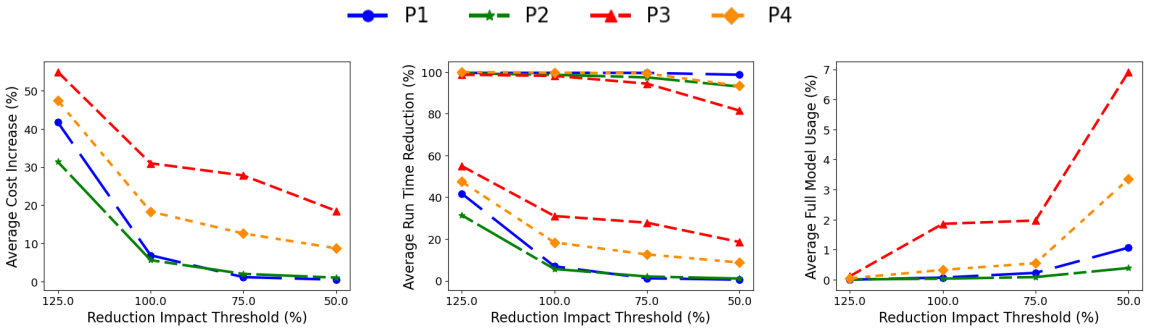


Figure 3.6: An example of the racetrack problem. Blue cell denotes start state and red denotes goal, lighter cells denote the track and darker cells denote walls (obstacles).



(a) Effect on expected cost (b) Effect on run time (c) Effect on model fidelity

Figure 3.7: Comparison of effects of different reduction impact thresholds on four instances of the racetrack domain.

racetrack problem is illustrated in Figure 3.6. The state of the car is determined by its location and its two-dimensional velocity. The car can change its speed in each of the two dimensions by at most one unit, resulting in a total of nine possible actions. The problem instances in Figure 3.7 have the same actions, goal conditions, and state representation, but differ in the size of their state space.

The threshold indicates the % difference between the estimated δ and the original costs, at which the full model is employed. In all other states, most likely outcome determinization is used. The cost increase is with respect to the lower bound (optimal expected cost obtained by solving the full model) and the run time reduction is with

respect to solving the full model. The full model usage increases as the threshold lowers, resulting in improved solution quality (lower cost) and reduced time savings.

3.4.3 Cost Adjusted Actions

In a reduced model, traditionally only the transition function is altered. This section presents a technique that accounts for the ignored outcomes by adjusting the costs of actions in the reduced model. Since the reduction impact reflects the values of the ignored outcomes, this value is used to adjust the costs of actions and the resulting reduced model is called a *cost adjusted reduced model* (CARM). The cost adjustments, in a way, reflect the long-term cost of an action in a reduced model since it accounts for the ignored outcomes of the action that may be encountered during plan execution. The cost adjustments for an optimal policy are defined below.

Definition 9. *A **cost adjusted reduced model** (CARM), M'_{ca}, of an SSP M is a reduced model in which the reduction impact is used as the cost function. That is, the cost of executing an action in a state is $C'(s,a) = \delta(s,a)$, $\forall(s,a)$ in a CARM.*

The costs are adjusted for every (s,a) in the reduced model to account for the ignored outcomes. Since the costs are adjusted based on the difference in values of states, this may lead to negative cost cycles. A necessary and sufficient condition for non-negative cost in CARM is that T' satisfies $Q^*(s,a) \geqslant \sum_{s' \in \theta'(s,a)} T'(s,a,s')V^*(s')$. This condition may be relaxed as long as there are no negative cost cycles in M'_{ca}. This can be handled in approximate estimation by using only non-negative values.

In principle, a CARM can result in optimal action selection when the optimal values are known. While this is practically infeasible, it explains the potential benefits of the technique. The optimal state values and action values in M'_{ca} are denoted by $V^*_R(s)$ and $Q^*_R(s,a)$, and its optimal policy by π^*_R. Let X^π_R and X^π denote the set of states reachable by executing a policy π in M'_{ca} and executing π in M, respectively.

Since $\theta'(s,a) \subseteq \theta(s,a)$, we get $X_R^\pi \subseteq X^\pi$. The following lemma shows that a CARM has optimal state values as the full model, when goal reachability is preserved.

Lemma 10. *For a policy π, the optimal state values in a CARM that preserves goal reachability and the full model are equal, $\forall s \in X_R^\pi : V_R^\pi(s) = V^\pi(s)$.*

Proof. We show this using proof by induction on t starting from the goal state and following policy π (assuming proper policy). The base case holds as we start from a goal. For readability, let $\theta'_{S_t,\pi} = \theta'(S_t, \pi(S_t))$, $S_{t=1} = s$ and $S_{t-1} = s'$. When $t=1 : V^\pi(s) = C(s,\pi(s))$, $\forall s \in X^\pi$ and $V_R^\pi(s) = C'(s,\pi(s))$, $\forall s \in X_R^\pi$. Using π and Definition 9, we get $Q^\pi(s,a) = V^\pi(s)$ and $C'(s,a) = V^\pi(s)$. Hence, $V_R^\pi(s) = V^\pi(s), \forall s \in X_R^\pi$. Thus, this holds true for $t=1$.

Inductive Step: Assume true for $t-1$ (induction hypothesis), must show that for t, $V_R^\pi(S_t) = V^\pi(S_t)$. Then,

$$V_R^\pi(S_t) = C'(S_t, \pi(S_t)) + \sum_{s' \in \theta'_{S_t,\pi}} T'(S_t, \pi(S_t), s') V_R^\pi(s').$$

Using Definition 9 in the above equation,

$$V_R^\pi(S_t) = Q^\pi(S_t, \pi(S_t)) + \sum_{s' \in \theta'_{S_t,\pi}} T'(S_t, \pi(S_t), s')(V_R^\pi(s') - V^\pi(s')).$$

By induction hypothesis, $V_R^\pi(s') = V^\pi(s')$, and for a fixed policy, $Q^\pi(S_t, \pi(S_t)) = V^\pi(S_t)$. Substituting these in the above equation, we get $V_R^\pi(S_t) = V^\pi(S_t)$. Thus, by induction, this holds true for all t, $V_R^\pi(s) = V^\pi(s), \forall s \in X_R^\pi$. □

This result is used to show that a CARM that preserves goal reachability produces optimal action selection for the SSP.

Proposition 11. *A CARM that preserves goal reachability yields optimal action selection for the SSP, if there exists a proper policy in the SSP.*

Proof. We prove this by showing that $\forall(s,a) \in M'_{ca}$, the optimal Q-values of the SSP and its CARM are equal, $Q^*_R(s,a) = Q^*(s,a)$. However, if the reduced model does not preserve goal reachability, then the Q-values will be different. Therefore, we restrict the proof to a CARM that preserves goal reachability. By definition, $\forall(s,a) \in S \times A$:

$$Q^*_R(s,a) = C'(s,a) + \sum_{s' \in \theta'(s,a)} T'(s,a,s') V^*_R(s').$$

Using Definition 4 in the above equation,

$$Q^*_R(s,a) = Q^*(s,a) + \sum_{s' \in \theta'(s,a)} T'(s,a,s') V^*_R(s') - V^*(s').$$

Using Lemma 10 in the above equation, we get $Q^*_R(s,a) = Q^*(s,a)$. □

While it is infeasible to estimate optimal cost adjustments in most problems, these results indicate that the action selection may be near-optimal when the estimated reduction impact is closer to the real value. A 0/1 RM with cost adjustments based on reduction impact calculated using samples is referred to as approximately cost adjusted 0/1 RM (ACARM-0/1 RM).

3.5 Evaluation

Experiments are conducted on three domains including an electric vehicle (EV) charging problem using real world data from a university campus, and two benchmark planning problems: the racetrack domain and the sailing domain. The expected cost of reaching the goal and planning time are used as evaluation metrics.

Baselines The performances of 0/1 RM and ACARM-0/1 RM are compared with the following reduced model techniques:

- Most-likely outcome determinization (MLOD);

- Uniformly selecting two outcomes greedily (M02);
- M_l^k reduction characterized by l primary outcomes and k exceptions, with $k=1, l=1$ [78]; and
- Reduced models, with $Z=\{\text{MLOD}, \text{M02}\}$, that alternate between MLOD and M02 (0/M02 RM).

Given the features characterizing risky states, the reduction impact is estimated using thirty samples for each domain. The samples are generated by depth-limited random walk on the problem and the reduction impact is calculated with respect to MLOD of the problem. For all states with reduction impact greater than the threshold value in Table 3.1, the model selector uses a full model in a 0/1 RM, and uses M02 reduction in 0/M02 RM. In all other states, MLOD is used. The 0/1 RM and 0/M02 RM use the reduction impact for the model selector only, while an ACARM-0/1 RM uses the reduction impact for the actions costs and model selection.

All results are averaged over 100 trials of planning and execution simulations and the average times include re-planning time. The deterministic problems are solved using the A* algorithm [36] and other problems using LAO* [35]. All algorithms were implemented with ϵ=10^{-3} convergence and using h_{min} heuristic computed using a labeled version of LRTA* [55], and tested on an Intel Xeon 3.10 GHz computer with 16GB of RAM.

3.5.1 Domains

Racetrack In this domain, the task is to move from start to goal states by applying acceleration correctly [5]. If the car hits a wall, it is repositioned back to start state. The problem is modified to increase the difficulty such that, in addition to a 0.1 probability of slipping, there is a 0.2 probability of randomly changing the intended acceleration by one unit. Six problem instances are considered. The reduction impact

uses one-step lookahead with features such as whether the successor is a wall or pothole or goal, and if the successor is moving away from the goal, estimated using the heuristic.

Sailing In this domain, the objective is to find the shortest path between two points of a grid under fluctuating wind conditions [52]. The boat cannot move in the direction opposite to that of the wind and the changes in wind direction are stochastic. Six problem instances are considered, with varying grid size (20, 40, 80) and the goal position (opposite corner (C) or middle (M) of the grid). The reduction impact is estimated using one-step lookahead and based on features such as the difference between the action's intended direction of movement and the wind's direction, and if the successor is moving away from goal, estimated using the heuristic value.

Electric Vehicle Charging In the electric vehicle (EV) charging domain, the EV can charge and discharge energy from a smart grid [100]. The objective is to minimize the long-term operational cost of the EV, given the owner's charging preferences. The problem is modified to allow for uncertainty regarding the parking duration of the EV, which is specified by a probability that certain states may become terminal states. The maximum parking duration is the horizon h. Each state is represented by $\langle l, t, d, p, e \rangle$, where l is the current charge level, $t \leqslant h$ is the time step, d and p denote the current demand and price distribution for electricity respectively, and $0 \leqslant e \leqslant h$ denotes the time steps remaining for departure. If the owner has not provided this information, the agent plans for h. The process terminates when $t = h$ or if $e = 0$.

Each t is equivalent to 30 minutes in real time. It is assumed that the owner may depart between four to six hours of parking with a probability of 0.2 that they announce their planned departure time. Outside that window, there is a lower probability of 0.05 that they announce their departure. We experimented with four reward functions (RF). The rewards and the peak hours are based on real data [26]. The

battery capacity and the charge speeds are based on Nissan Leaf configuration. We assume the charge and discharge speeds to be equal. The battery inefficiency is accounted for by adding a 15% penalty on the rewards. The reduction impact is estimated using features: time remaining for departure, if the current time is peak hour, and if there is sufficient charge for discharging, with one step-lookahead.

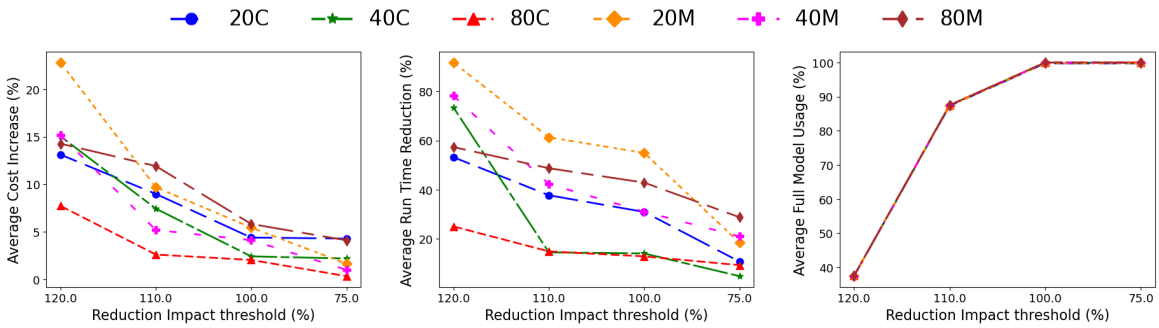
EV Dataset The data used in the experiments consist of charging schedules of electric cars over a four month duration in 2017 from an American university campus. The data is clustered based on the entry and exit charges, and 25 representative problem instances were selected across clusters for the experiments. The data is from a typical charging station, where the EV is unplugged once the desired charge level is reached. Since an extended parking scenario is considered, as in a vehicle parked at work, the parking duration (h) is set to eight hours in the experiments.

Problem	Full Model (%)	δ threshold
EV-RF-1	7.644	120%
EV-RF-2	9.956	120%
EV-RF-3	8.989	120%
EV-RF-4	9.852	120 %
Racetrack-Square-3	0.860	100%
Racetrack-Square-4	0.071	100 %
Racetrack-Square-5	0.034	100%
Racetrack-Ring-4	2.871	100%
Racetrack-Ring-5	1.859	100%
Racetrack-Ring-6	0.327	100%
Sailing-20(C)	37.414	120%
Sailing-40(C)	37.478	120 %
Sailing-80(C)	37.495	120 %
Sailing-20(M)	37.414	120%
Sailing-40(M)	37.478	120%
Sailing-80(M)	37.495	120%

Table 3.1: Percentage of full model usage in 0/1 RM using reduction impact as heuristic for model selector. δ threshold denotes the percentage difference between estimated reduction impact and the original costs at which full model is employed.

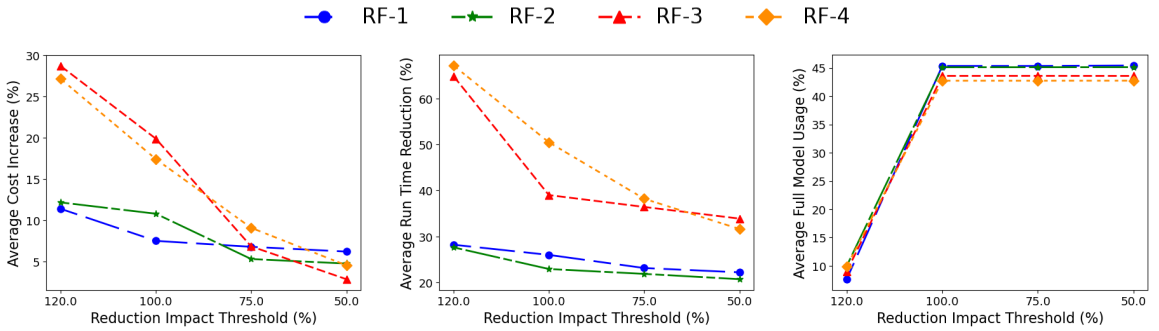
3.5.2 Results and Discussion

Table 3.1 reports the full model usage (%) for 0/1 RM in our experiments, using reduction impact as heuristic for model selector. The threshold values for the reduction impact used in the experiments are based on Figures 3.7, 3.8, and 3.9 that compare the effect of different reduction impact thresholds on the performance.



(a) Expected cost (b) Run time (c) Full model usage

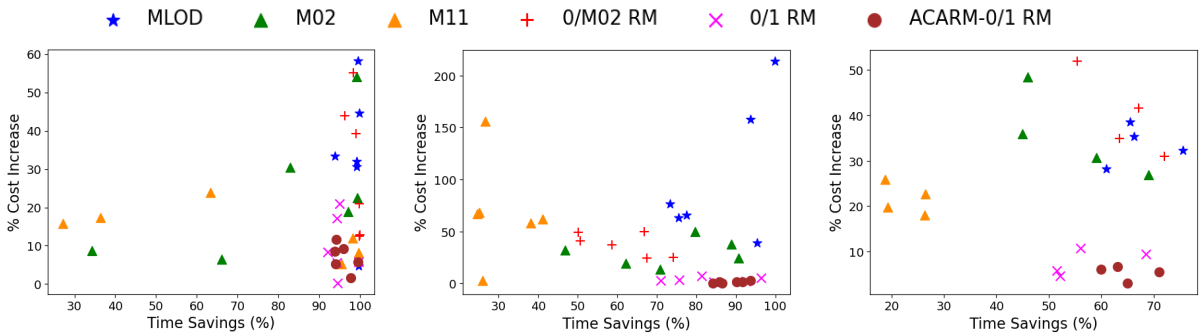
Figure 3.8: Effect of reduction impact thresholds on expected cost, run time, and full model usage in sailing domain.



(a) Expected cost (b) Run time (c) Full model usage

Figure 3.9: Effect of reduction impact thresholds on expected cost, run time, and full model usage in EV domain.

Figure 3.10 shows the average increase in cost (%) and the time savings (%), with respect to solving the original problem optimally. For the EV domain, the results are aggregated over 25 problem instances for each reward function. A low cost increase



(a) Racetrack (b) Sailing (c) EV

Figure 3.10: Comparison of trade-offs by different reduced models on three domains.

indicates that the performance of the technique is closer to optimal. A high time saving value indicates improved run time gains by using the model. Hence, the **lower right corner** of each image represents the most desired results.

These results show that ACARM-0/1 RM with a good model selector and cost estimation can achieve near-optimal performance without significantly affecting run time. ACARM-0/1 RM is at least 60% faster than solving the full model. All the reduced models are solved using an optimal algorithm, which allows us to assess the direct impact of model reduction. In practice, however, any SSP solver (optimal or not) may be used to further improve run time gains.

Results show that 0/1 RM can effectively minimize the expected costs without significantly affecting run time and by sparingly using the full model. This indicates that using δ as a heuristic works well in practice. In many problems, performance is improved further by using δ to adjust the costs. Thus, the results demonstrate the benefits of our approach in formulating reduced models that balance the trade-off between model simplicity and risk awareness.

3.6 Summary

When agents simplify a high-fidelity model to accelerate planning, reasoning with the resulting imprecise model affects the overall performance. Two general methods are introduced to devise risk-aware reduced models of large SSPs. First, planning using a portfolio of outcome selection principles that provides flexibility in outcome selection for each state-action pair is proposed. The reduction impact is measured based on the ignored outcomes and it is used as a heuristic for model selector in a 0/1 reduced model. Second, cost adjusted reduced models is introduced to account for ignored outcomes in the reduced model by modifying the actions costs. Our empirical results demonstrate the promise of this framework as cost adjustments in a basic instantiation of a POSP offer improvements—ACARM-0/1 RM improves the performance without significantly affecting the run time gains. Our results also contribute to a better understanding of how disparate reduced model techniques relate to each other and could be used together to leverage their complementary benefits.

CHAPTER 4

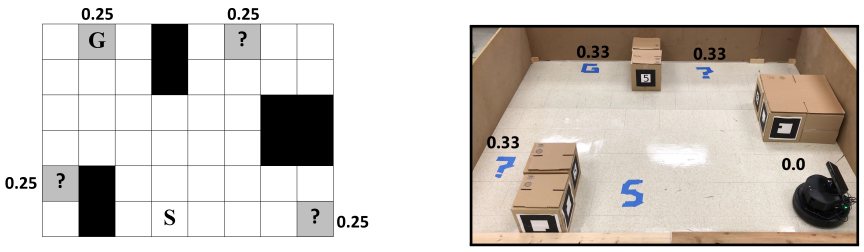
PLANNING UNDER GOAL UNCERTAINTY

This chapter addresses the model imprecision arising due to the availability of limited information during system design. Specifically, it focuses on settings in which it is impossible to accurately determine the goal states, ahead of plan execution. Existing standard frameworks to model goal-oriented problems require one or more goal states to be prescribed, which can result in misspecification when precise information is unavailable during system design. Misspecified goals affect agent behavior when deployed. To model scenarios with goal uncertainty efficiently, this chapter presents goal uncertain stochastic shortest path (GUSSP), a framework that extends the SSPs to settings with goal uncertainty [109]. The agent is aware of the goal condition but does not have knowledge of which states satisfy the goal conditions, ahead of plan execution. Given a distribution over set of states describing the probability of being a goal, the agent is expected to devise a plan that is optimal, given the goal uncertainty.

Chapter outline. The chapter begins with problem motivation in Section 4.1. Goal uncertain SSP framework is introduced and formally defined in Section 4.2, and its theoretical properties are described in Section 4.3. Approximate methods to solve GUSSPs are discussed in Section 4.4. Finally, empirical evaluation of the approach in simulation and on a real robot is presented in Section 4.5.

4.1 Introduction

Autonomous agents acting in the real world are often faced with goal-oriented tasks, which are typically modeled as a Stochastic Shortest Path (SSP) problem.



(a) Problem setting (b) Experimental setting

Figure 4.1: An illustrative example of a search and rescue problem with goal uncertainty, showing a motivating problem setting with the initial belief (left) and the corresponding experimental setting of the problem with a mobile robot and updated beliefs (right). The question marks indicate potential victim locations and values denote the robot's belief. 'S' denotes the robot's start location and 'G' is the actual victim location (goal). The robot updates its belief about the victim locations based on its observations.

The objective in an SSP is to devise a sequence of actions such that the expected cost of reaching a *known* goal state from the start state is minimized [10].

Consider a search and rescue domain (Figure 4.1), a motivating example where the agent has to devise a cost minimizing path to rescue immobile people from a building [50, 76]. While the number of victims and the map of the building may be provided to the agent, only potential victim locations may be known ahead of plan execution. The unavailability of exact goal states (victim locations) during planning time prevents the problem from being modeled as a standard SSP. While the exact goal states may be hard to identify a priori, it is relatively easier to obtain a belief distribution over possible victim locations, based on sensors or historical occupancy data of the building. The search and rescue domain is an instance of the *optimal search for stationary targets* [34, 107]—a class of problems in which the target's exact location is unknown to the agent, but it can observe its current location and determine whether the target is in the current location. The agent is given well-defined goal conditions, but has uncertainty about the states that satisfy these goal conditions.

Knowledge gaps The existing approaches [70, 71] model such problems as a Partially Observable MDP (POMDP) [44]. However, POMDPs are much harder to solve optimally [72]. The partially observable SSPs (POSSPs) extend the SSP framework to settings with partially observable states, offering a class of indefinite-horizon, undiscounted POMDPs that rely on state-based termination [73]. Other relevant POMDP variants are the mixed observable MDPs [71] that model problems with both fully observable and partially observable state factors, and the goal POMDPs [13] that are goal-oriented POMDPs with no discounting. These models are solved using POMDP solvers and are difficult to solve optimally. They also suffer from limited scalability due to their computational complexity [72]. Another related line of work is the transition-uncertain MDPs [20] which can capture the uncertainty in transitioning to the goal states. However, solving MDPs with imprecise transitions optimally is non-trivial. The objective is to efficiently solve problems with goal uncertainty by leveraging the fully observable components of the problem.

Overview of contributions This chapter presents goal uncertain SSP (GUSSP), a framework specifically designed to model problems with imperfect goal information by allowing for a probabilistic distribution over possible goals. GUSSPs fit well with many real-world settings where it is easier and more realistic to have belief over goal configurations, rather than exact knowledge about the goal states. The observation function in a GUSSP facilitates the reduction to an SSP, enabling the computation of tractable and optimal solutions. The focus is on settings where the goals do not change over time. It is assumed that there exists a unique observation that allows the agent to accurately identify a goal when it reaches one. Two solution approaches are presented to solve GUSSP: a heuristic approach and a determinization approach.

The key contributions in this chapter are: (1) a formal definition of GUSSP and its theoretical properties; (2) a domain-independent, admissible heuristic that can

accelerate probabilistic planners; (3) a determinization approach for solving GUSSP; and (4) empirical evaluation on three domains in simulation and on a mobile robot.

4.2 Goal Uncertain Stochastic Shortest Path (GUSSP)

A goal uncertain stochastic shortest path (GUSSP) problem is a generalized framework to model problems with goal uncertainty. A GUSSP is an SSP in which the agent may not know initially the exact set of goal states (which do not change over time), and instead can obtain information about the goals via observations during execution.

Definition 12. *A **goal uncertain stochastic shortest path problem** is a tuple* $\langle X, S, A, T, C, s_0, S_G, P_G, \Omega, O \rangle$ *where*

- S, A, T, C, s_0, S_G *denote an underlying SSP with discrete states and actions, and* S_G *initially unknown to the agent;*
- $P_G \subseteq S$ *is the set of potential goals such that* $S_G \subseteq P_G$*;*
- $X = S \times G$ *is the set of states in the GUSSP with* $G = 2^{P_G} \backslash \{\varnothing\}$ *denoting the set of possible goal configurations;*
- Ω *is a finite set of observations corresponding to the goal configurations,* $\Omega = G$*; and*
- $O : A \times X \times \Omega \rightarrow [0, 1]$ *is the observation function denoting the probability of receiving an observation,* $\omega \in \Omega$*, given action* $a \in A$ *led to state* x' *with probability* $O(a, x', \omega) \equiv Pr(\omega | a, x')$*.*

Each state is represented by $x = \langle s, g \rangle$, with $s \in S$ and $g \in G$. GUSSPs have mixed observable state components as s is fully observable. Each $g \in G$ represents a goal configuration (set of states), thus permitting multiple true goals in the model, $|S_G| \geqslant 1$. Every action in each state produces an observation, $\omega \in \Omega$, which is a goal

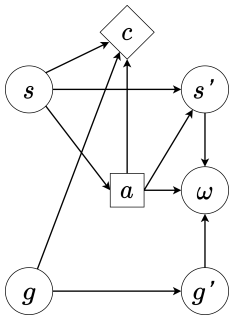


Figure 4.2: A dynamic Bayesian network describing a GUSSP.

configuration and thus provides information about the true goals. The agent's belief about its current state is denoted by $b(x)$, with $x = \langle s, g \rangle$; that is, the belief about $g = S_G$. The initial belief is denoted by $b^0\langle s_0, g\rangle \in [0, 1], \forall g \in G$, where s_0 is the start state of the SSP.

The process terminates when the agent reaches a state x with $b(x) = 1$ and $s \in g$. SSPs are therefore a special type of GUSSPs with a collapsed initial belief over the goals. Figure 4.2 shows a part of the network representation for a GUSSP. As in a (PO)SSP, the following assumptions are made in a GUSSP: (1) the existence of a proper policy with finite cost, (2) all improper policies have infinite cost, and (3) termination is perfectly recognized.

Observation Function In a GUSSP, an observation function is characterized by two properties. First, to perfectly recognize termination, all potential goals are characterized by a unique belief-collapsing (when the belief over a state is either 1 or 0) observation. That is, at potential goal states, if $s' \in g'$, then $\forall a \in A$:

$$O(a, x', \omega) = \begin{cases} 1 & \text{if } g' = \omega \\ 0 & \text{otherwise.} \end{cases} \tag{4.1}$$

Second, the observation function is *myopic*, providing information only about the current state or the potential goals in the immediate vicinity. This assumption

is motivated by real-world settings with limited range sensors, and the exploration and navigation approaches for robots that acknowledge the perceptual limitations of robots [11]. The *landmark states*, L_s, provide accurate information about certain potential goals. Each $s \in L_s$ provides observations about a subset of potential goals in the vicinity with Ω_s denoting the corresponding set of observations. Each $\omega \in \Omega_s$ provides information about the maximal set of potential goals in the vicinity. Other non-potential goal states, $s' \notin L_s$, provide no information about the true goals. Therefore, the observation at non-potential goal states is $\forall a \in A$:

$$O(a, x', \omega) = \begin{cases} 1 \text{ if } s' \in L_s \wedge \omega \subseteq g' \wedge \omega \in \Omega_{s'} \\ 0 \text{ if } s' \in L_s \wedge \omega \nsubseteq g' \wedge \omega \in \Omega_{s'} \\ \frac{1}{|\Omega|} \text{ if } s' \notin L_s \end{cases},$$

with $x = \langle s, g \rangle$ and $x' = \langle s', g' \rangle$. The potential goals along with the landmark states are called *informative states*, $\mathcal{I} = P_G \cup L_s$, since they provide information about the true goals through deterministic observations. Thus, the observation function discussed above satisfies the minimum information required for state-based termination. The next section discusses a more general setting where every state may have a noisy observation regarding the true goals.

Belief Update A belief b is a probability distribution over X, $b(x) \in [0, 1], \forall x \in X$ and $\sum_{x \in X} b(x) = 1$. The set of all reachable beliefs forms the belief space $B \subseteq \Delta^n$, where Δ^n is the standard $(n-1)$-simplex. The agent updates the belief $b' \in B$, given an action $a \in A$, an observation $\omega \in \Omega$, and the current belief b. Using the multiplication rule, the updated belief for $x' = \langle s', g' \rangle$ is calculated as:

$$
\begin{aligned}
b'(x'|b,a,\omega) &= Pr(g'|b,a,\omega,s')\,Pr(s'|b,a,\omega,s) \\
&= Pr(g'|b,a,\omega,s')\,T(s,a,s') \qquad (4.2) \\
Pr(g'|b,a,\omega,s') &= \eta Pr(\omega|b,a,s',g')Pr(g'|b,a,s') \\
&= \eta O(a,x',\omega)\sum_{g\in G} Pr(g',g|b,a,s') \\
&= \eta O(a,x',\omega)Pr(g|b,a,s') \\
&= \eta O(a,x',\omega)b(g), \qquad (4.3)
\end{aligned}
$$

where $\eta = Pr(\omega|b,a,s')^{-1}$ is a normalization constant. Substituting Equation 4.3 in 4.2:

$$
b'(x'|b,a,\omega) = \eta O(a,x',\omega)b(g)T(s,a,s'). \qquad (4.4)
$$

Policy and Value The agent's objective in a GUSSP is to minimize the expected cost of reaching a goal, $\min_{\pi\in\Pi}\mathbb{E}\Big[\sum_{t=0}^{h} C(x_t,a_t)\Big|\pi\Big]$, where x_t and a_t denote the agent's state and action at time t respectively, and $h \in \mathbb{N}$ denotes the horizon. A policy $\pi : B \rightarrow A$ is a mapping from belief $b \in B$ to an action $a \in A$. The value function for a belief, $V : B \rightarrow \mathbb{R}$ is the expected cost for a fixed policy π and a horizon h. The Bellman optimality equation for GUSSPs follows from POMDPs:

$$
V(b) = \min_{a\in A}\big[C(b,a) + \sum_{\omega\in\Omega} Pr(\omega|b,a)V(b'_{a\omega})\big],
$$

where $b'_{a\omega}$ is the updated belief following Equation (4.4), $C(b,a) = \sum_x b(x)C(x,a)$, $x = \langle s,g\rangle$, and $x' = \langle s',g'\rangle$. A proper policy, π, in a GUSSP guarantees termination in a finite expected number of steps, $V^\pi(b^0) < \infty$.

Order-k policy The number of potential goals with non-zero belief values indicate the degree of uncertainty over goals. The problem setting and the optimal policy determine when the belief values collapse to the true goals. When deploying systems in real-world settings with goal uncertainty, it is useful to understand the problem

complexity for policy execution. This is measured by the maximum number of unique visits to informative states that may be required before a true goal is discovered by the agent.

Definition 13. *A GUSSP policy π is of **order-k** if there are at most k unique visits to informative states before a true goal is reached following π.*

For state-based termination, $1 \leqslant k \leqslant |P_G|$. Unique visits are considered since no new information is obtained thereafter. For example, consider a search and rescue domain in which the agent searches for victims in a corridor with the start state on one end and followed by a series of potential goals. If the first potential goal location is a true goal, then the agent visits only one potential goal before the true goal is discovered, following the optimal policy. This property is beneficial especially in environments with landmark states that reveal the true goals, thus minimizing the need to visit the potential goals specifically to determine the true goals.

4.3 Theoretical Analysis

The observation function in a GUSSP determines the number of reachable beliefs. This section begins with an analysis of how the number of beliefs may grow in the more general (non-myopic observation) setting and then show that a GUSSP with myopic observations has finite reachable beliefs.

In a GUSSP with non-myopic observations, the nonpotential goal states, $\forall s \notin L_s \cup P_G$, provide stochastic observations about the true goals, resulting in infinitely many reachable beliefs. While this is a trivial fact, it is useful to understand the growth in complexity of the problem and it provides an important link to POMDPs via the belief-MDP. The following proposition formally proves this complexity.

Proposition 14. *For all horizon $h > 0$, the belief-MDP of a GUSSP with non-myopic observations may have $\mathcal{O}(|\Omega|^h)$ states.*

Proof. By construction, a GUSSP with non-myopic observations is mapped to a belief-MDP $\langle B, \mathcal{A}, \tau, \rho \rangle$ with a horizon h [44]. Let $\mathcal{R}(b^0)$ denote the set of reachable beliefs in the GUSSP. The set of states in the MDP is the set of reachable beliefs from b^0 in the GUSSP, $B = \mathcal{R}(b^0)$. The set of actions in the GUSSP are retained in the MDP, $\mathcal{A} = A$. The cost function $\rho(b, a) = \sum_{x \in X} b(x)C(x, a)$, where $C(x, a)$ corresponds to cost function in GUSSP. The transition function for the belief-MDP is the probability of executing action $a \in \mathcal{A}$ in belief state $b \in B$ and reaching the reaching belief b', and denoted by $\tau(b, a, b')$, is:

$$\begin{aligned}
\tau(b, a, b') &= \sum_{\omega \in \Omega} Pr(b', \omega | b, a) \\
&= \sum_{\omega \in \Omega} Pr(b' | b, a, \omega) Pr(\omega | b, a) \\
&= \sum_{\omega \in \Omega} Pr(\omega | b, a)[b' = b'_{a\omega}],
\end{aligned}$$

with Iversen bracket $[\cdot]$ and $b'_{a\omega}$ denoting the updated belief calculated using Equation (4.4), after executing action a and receiving observation ω. The probability of receiving ω is:

$$\begin{aligned}
Pr(\omega | b, a) &= \sum_{x' \in X} Pr(\omega, x' | b, a) \\
&= \sum_{x' \in X} O(a, x', \omega) \sum_{x \in X} T(s, a, s') b(g'),
\end{aligned}$$

with $x = \langle s, g \rangle$ and $x' = \langle s', g' \rangle$. Since $|S|$ in the GUSSP is finite, a finite set of reachable beliefs in the GUSSP results in a finite set of reachable states in the belief-MDP. This is a tree of depth h with internal nodes for decisions and transitions, the branching factor is $\mathcal{O}(|\Omega|)$ for each horizon, h [72]. Therefore, the total number of reachable beliefs in the GUSSP is $\mathcal{O}(|\Omega|^h)$, and thus the resulting belief-MDP may have $\mathcal{O}(|\Omega|^h)$ distinct reachable states. □

In the worst case, the observation function may be unconstrained and all the beliefs may be unique. Since there is no discounting in a GUSSP and the horizon is unknown a priori, GUSSPs may have *infinitely* many beliefs and their complexity class may be undecidable in the worst case [65]. Hence, solving GUSSPs with non-myopic observations optimally is computationally intractable.

The following proposition proves that a myopic observation function results in a finite number of reachable beliefs in a GUSSP.

Proposition 15. *A GUSSP with myopic observation function has a finite number of reachable beliefs.*

Proof. By definition, a myopic observation function produces either belief-collapsing observations or no information at all. For each case, the updated belief for the goal configurations is calculated using Equation (4.3). Therefore, $\forall x' \in X$ with $x' = \langle s', g' \rangle$:

$$b'(g') = \frac{O(a, x', \omega)\, b(g)}{\sum_{x'} O(a, x', \omega) b(g)}.$$

Case 1: Belief-collapsing observation. When $O(a, x', \omega) = 0$, the updated belief is $b'(g') = 0$. When $O(a, x', \omega) = 1$, the updated belief is $b'(g') = 1$.

Case 2: No information. When the observation provides no information, $\forall a \in A$, $O(a, x', \omega) = 1/|\Omega|$. Then,

$$b'(g') = \frac{b(g)/|\Omega|}{\sum_{x'} b(g)/|\Omega|} = b(g).$$

Thus, $\forall g \in G$, a myopic observation function produces collapsed belief or retains the same belief, resulting in a finite number of reachable beliefs for a goal configuration. Since $|S|$ is finite, the belief update following Equation (4.4) would result in finite number of reachable beliefs for a GUSSP. □

Hence, a myopic observation function weakly monotonically collapses beliefs, allowing the problem to be simplified further. The following proposition shows that a GUSSP reduces to an SSP, similar to the mapping from a POMDP to belief-MDP [44].

Proposition 16. *A GUSSP reduces to an SSP.*

Proof. A GUSSP is mapped to a belief-MDP $\langle B, \mathcal{A}, \tau, \rho \rangle$ with a horizon h [44], as in Proposition 14. By Proposition 15, a GUSSP with myopic observation function has a finite number of reachable beliefs and therefore, finite states in the belief-MDP. By construction, this belief-MDP is an SSP with the start state $\bar{s}_0 = b^0$ and the goal states, $\bar{S}_G$, are the set of states with $\bar{b}(x) = 1$ such that $\bar{b}(g) = 1$ and $s \in g$. Since there exists a proper policy in a GUSSP, the policy in this SSP is proper by construction. Thus, a GUSSP with myopic observation function reduces to an SSP. □

The reduction to an SSP facilitates solving GUSSPs using the existing rich suite of SSP algorithms. For ease of reference and clarity, the above-mentioned SSP is referred to as compiled-SSP in the rest of this chapter. The *order-k* of π^* for a GUSSP (compiled-SSP) can be calculated using a directed graph constructed using π^*. The following proposition shows that computing *order-k* is polynomial.

Proposition 17. *The worst case complexity for computing* order-k *for* π^* *is* $\mathcal{O}(|P_G|(|V|+|E|))$*, where* V *and* E *denote the vertices and edges of the corresponding directed graph.*

Proof Sketch. To calculate *order-k* for π^*, let us first construct a directed graph, Z, using π^* such that $V = \mathcal{I} \cup \{s_0\}$ and the trajectories between them are the edges, E. Let us set a potential goal to be the goal and introduce additional (artificial) edges from it to the informative states. This is followed by computing the strongly connected components, using depth first search that takes $\mathcal{O}(|V| + |E|)$, which are then condensed to form a directed acyclic graph $Z' = (V', E')$. Starting from the goal in Z' and traversing backwards, the k value of the goal state is initialized to 1

and propagated to its (unvisited) neighbors. At each vertex, k is increased to be the sum of informative states in the condensed vertex and the incoming value from the neighbor. This continues until all vertices in Z' have been visited and the start state is updated with the maximum k. This process may be repeated with every potential goal as the goal and the overall maximum k is the order of the policy. Thus, the worst case complexity is $\mathcal{O}(|P_G|(|V| + |E|))$. □

Relation to Goal-POMDPs

The Goal-POMDP [13] models a class of goal-based and shortest-path POMDPs with positive action costs and no discounting. The set of target (or goal) states, $\bar{P}$ have unique belief-collapsing observations. Hence, a Goal-POMDP is a GUSSP when the partial observability is restricted to goals, the observations set is $2^{\bar{P}} \backslash \{\varnothing\}$, and observation function is myopic.

Proposition 18. *GUSSP* $\subset$ *Goal-POMDP.*

The observations in a Goal-POMDP are not constrained and may result in infinitely many reachable beliefs (Proposition 14). This makes it computationally challenging to compute optimal policies [72], unlike GUSSPs which are more tractable can be solved optimally (Proposition 16).

GUSSP with Deterministic Transitions

A GUSSP with deterministic transitions presents an opportunity for further reduction in complexity.

Proposition 19. *The optimal policy for a GUSSP with myopic observations and deterministic transitions is the minimum arborescence of a weighted and directed graph* Z*.*

Proof. Consider a GUSSP with deterministic transitions and a dummy start state, r, that transitions to the actual start state with probability 1 and zero cost. This

can be represented as a directed and weighted graph, $Z = (V, E, w)$, such that $V = \{r\} \cup \{x \in X | x = \langle s, g \rangle \wedge s \in P_G\}$; that is, the start state and the potential goals are the vertices. Each edge $e \in E$ denotes a trajectory in the GUSSP between vertices. The proper policy in a GUSSP ensures that there is at least one edge between each pair of vertices. The weight of an edge connecting $x, y \in V$ is $w(e) = d(x, y)(1 - b(y))$, with $d(x, y)$ denoting the cost of the trajectory and $b(y)$ is the belief over y being a goal. The minimum arborescence (directed minimum spanning tree) of this graph, A, contains trajectories such that the total weight is minimized, $\min_{A \in \mathcal{A}} w(A)$ with $w(A) = \sum_{e \in A} w(e)$. By construction, this gives the optimal order of visiting the potential goals and hence the optimal policy for the GUSSP with $V^*(s_0) = w(A)$. □

4.4 Solving Compiled-SSPs

Two solution approaches are proposed to solve compiled-SSP: (1) an admissible heuristic for SSP solvers that accounts for the goal uncertainty and (2) a determinization-based approach for solving the compiled-SSP.

4.4.1 Admissible Heuristic

In heuristic search-based SSP solvers, the heuristic function helps avoid visiting states that are provably irrelevant. An efficient heuristic for solving the compiled-SSP guides the search by accounting for the goal uncertainty. A simple heuristic, h_{pg} for the compiled-SSP that accounts for goal uncertainty is presented below:

$$h_{pg}(x) \triangleq \min_{g \in G} \Big((1 - b(g)) \min_{i \in g} d(x, i) \Big)$$

where $d(x, i)$ denotes the cost of the shortest trajectory to the potential goal i from state x in the compiled-SSP and $b(g)$ is the agent's belief of g being a true goal. Multiplying by the probability of a state not being a goal $(1 - b(g))$ breaks ties in

favor of configurations with a higher probability of being a goal, with a lower heuristic value. The following proposition shows that the proposed heuristic is admissible.

Proposition 20. *h_{pg} is an admissible heuristic.*

Proof. To show that h_{pg} is admissible, it is important to first show that $\min_{i \in g} d(x, i)$ is an admissible estimate of the expected cost of reaching a goal configuration g from state x. Let $d^*(x, g)$ be the expected cost of reaching g from state x. Since $d(x, g)$ is the cost of the shortest trajectory to g from x, $d(x, g) \leqslant d^*(x, g)$. If all paths exist from x to all potential goal states $i \in g$, then by definition, the shortest trajectory to a goal configuration is the minimum distance to a potential goal in g. That is, $d(x, g) = \min_{i \in g} d(x, i)$ and therefore $\min_{i \in g} d(x, i) \leqslant d^*(x, g)$. Multiplying this value by the belief and using the minimum value over all possible goal configurations guarantees that h_{pg} is an admissible estimate of the expected cost reaching a true goal configuration. □

4.4.2 Determinization

Determinization is a popular approach for solving large SSPs as it simplifies the problem by replacing the probabilistic outcomes of an action with a single deterministic outcome [97, 118]. Determinization can be extended to a GUSSP by ignoring the uncertainty over the goals. A potential goal state is *determinized*—the agent treats it as the true goal and plans an optimal policy to reach it. Determinizing the potential goal reduces the problem to a standard SSP. During execution, if the determinized goal is not a true goal, the agent replans for another unvisited potential goal. This approximation scheme offers considerable speedup over solving the compiled-SSP.

Two determinization approaches are considered: (1) most-likely goal determinization (DET-MLG) and (2) closest-goal determinization (DET-CG). In the DET-MLG, the most-likely goal is determinized, based on its current belief. In DET-CG, the

agent determinizes the closest goal based on the heuristic distance to the potential goal (with non-zero belief) from its current state. Ties are resolved randomly.

4.5 Experimental Results

In this section, different approximate solution techniques for solving the compiled-SSP are compared on three domains in simulation. In addition, the model is tested on a real robot with three different initial belief settings, which shows how the initial belief distribution affects agent behavior.

4.5.1 Evaluation in Simulation

Experiments are conducted on three domains to evaluate the solution techniques in handling (1) location-based goal uncertainty (planetary rover domain, search and rescue domain) and (2) temporal goal uncertainty (electric vehicle (EV) charging problem using real-world data). The expected cost of reaching the goal and run time (in seconds) are used as evaluation metrics. A uniform initial belief is considered for all the domains in these experiments. The compiled-SSPs are solved optimally using LAO* [35], and approximately using FLARES, a domain-independent state-of-the-art algorithm for solving large SSPs using horizon=1 [77], as well as the two determinization methods. A residual error value of $\epsilon = 10^{-3}$ is used as convergence criteria for the algorithms. The h_{min} heuristic, which is a popular heuristic in planning literature is computed using a labeled version of LRTA* [12], is used as a baseline for evaluating h_{pg}. All the results are averaged over 100 trials of planning and execution simulations, and the average times include the time spent on re-planning. Standard errors are reported for expected cost. The experiments were conducted for the following domains on an Intel Xeon 3.10 GHz computer with 16GB of RAM.

4.5.1.1 Domains

Planetary Rover This domain models the rover science exploration [124, 71] that explores an environment described by a known map to collect a mineral sample. There are n sample locations, $|P_G| = n$, and the samples at each of these locations may be 'good' or 'bad'. The rover knows its own position (x, y) coordinates exactly, as well as those of the samples but does not know which samples are 'good'. The process terminates upon collecting a 'good' sample. The actions include moving in all four directions, which succeed with a probability of 0.8, and a 'sample' action which is deterministic. The 'sample' action costs +2 if the mineral is good and +10 otherwise; all other actions cost +1.

Search and Rescue In this domain, an agent explores an environment described by a known map to find victims [76]. The problem is modified such that there are m victims locations and n total victims. The agent is aware of the potential victim locations and each location may or may not have victims. The exact locations of the victims are unknown to the agent a priori. The objective is to minimize the expected cost of saving all victims. The state factors include the agent's current location and a counter to indicate the number of victims saved so far. The observations indicate the presence of victims in each state. The actions include moving in all four directions and a 'save' action that saves all the victims in a state. The move actions cost +1 and are stochastic, succeeding with 0.8 probability. The 'save' action is deterministic and costs +2.

Electric Vehicle Charging In this domain, an electric vehicle (EV), operating in a vehicle-to-grid setting [97], can charge and discharge energy from a smart grid. The objective is to devise a policy that is consistent with the owner's preferences, while minimizing the operational cost of the vehicle. The problem is modified such that parking duration of the EV is uncertain with H denoting the horizon. The potential

	LAO*		Flares-h_{min}		**Flares-h_{pg}**		**Det-MLG**		**Det-CG**	
Problem Instance	Cost	Time	Cost	Time	Cost	Time	Cost	Time	Cost	Time
rover (20,6)	28.25	14.99	35.35 ± 2.67	1.08	30.34 ± 2.37	0.17	36.71 ± 2.62	0.07	45.51 ± 3.22	0.06
rover (20,7)	42.16	30.19	43.49 ± 1.62	1.17	45.07 ± 1.77	0.83	49.69 ± 1.91	0.02	48.36 ± 1.43	0.03
rover (30,8)	36.96	190.92	38.21 ± 1.83	2.27	41.31 ± 1.97	0.16	38.54 ± 1.54	0.02	40.34 ± 1.82	0.03
rover (30,9)	34.72	832.56	38.21 ± 2.54	7.56	43.32 ± 2.54	1.73	50.27 ± 2.58	0.88	49.49 ± 1.97	0.45
search (20,4)	87.63	15.78	94.32 ± 0.58	1.45	93.32 ± 0.58	0.98	91.22 ± 0.67	1.05	90.42 ± 0.61	0.86
search (20,5)	74.61	14.42	83.83 ± 0.56	2.99	81.91 ± 0.56	1.93	78.32 ± 0.56	1.98	79.74 ± 6.37	0.98
search (20,5)	86.72	63.71	94.21 ± 0.79	6.21	91.18 ± 1.46	1.93	87.74 ± 0.65	0.66	89.98 ± 0.59	1.68
search (30,6)	90.89	267.35	94.21 ± 1.35	117.63	103.77 ± 3.42	21.07	101.67 ± 1.61	12.68	92.94 ± 0.68	19.50
ev (-,5)	2.34	8.16	3.29 ± 1.55	2.21	4.89 ± 1.36	0.92	5.15 ± 1.46	0.52	7.17 ± 1.43	0.62
ev (-,6)	3.46	10.79	4.89 ± 1.96	2.25	5.96 ± 1.96	1.14	7.15 ± 2.46	0.88	8.17 ± 1.43	0.79

Table 4.1: Comparisons of average cost, along with standard error, and planning time (seconds) of different solution approaches. **Bold** titles indicate our proposed techniques.

goals in this problem are the possible departure times. The EV can fully observe its current charge level and the time step. In our experiments, $P_G = \{H, H-1, .., H-n\}$ and $|P_G| = n$. Each t is equivalent to 30 minutes in real time. If the EV's exit charge level does not meet the owner's desired exit charge level, a penalty is incurred.

The battery capacity and the charge speeds for the EV are based on Nissan Leaf configuration and the action costs and peak hours are based on real data [26]. The charge levels and entry time data are based on charging schedules of electric cars over a four month duration in 2017 from a university campus. The data is clustered based on the entry and exit charges, and we selected 25 representative problem instances across clusters for our experiments.

4.5.1.2 Results and Discussion

Table 4.1 shows the results of the five techniques on various problem instances, in terms of cost and runtime(s) respectively. The grid size and the number of potential goals for each problem are indicated in parenthesis in the table. The results are averaged over 100 trials and standard errors are reported for the expected cost. The results for the EV domain are averaged over 25 problem instances. Experiments were

Figure 4.3: Demonstration of the path taken by the robot with three different initial beliefs for the map in Figure 4.1. The start state and the true goal state are denoted by 'S' and 'G', respectively. The other potential goals are denoted by the question mark symbol. Green, blue, and red show the path taken by the robot with 0.1, 0.25, and 0.9 as the initial belief for the true goal state and equal probability for other potential goal states.

conducted with no landmark states to demonstrate the performance in the worst case setting. In terms of expected costs, the performance of the approximate techniques are comparable. The run times for solving the problems optimally, however, scales rapidly as the number of potential goals increases. The advantage of using FLARES with h_{pg} and the determinization techniques are more evident in the runtime savings. FLARES using our heuristic h_{pg} is significantly faster than using the baseline h_{min} heuristic. Both the determinization techniques are faster than solving the problem using FLARES.

4.5.2 Evaluation on a mobile robot

The robot experiment aims to visually explain how the belief distribution alters the robot's trajectory, apart from the comparison using an abstract notion of cost. Figure 4.3 shows the results in a ROS simulation and on a Kobuki robot for a simple search and rescue problem with one agent and four potential victim locations for the map shown in Figure 4.1. Three different initial beliefs are tested: uniform, optimistic, and pessimistic. The corresponding belief of the true goal, G, in each belief setting is: 0.25, 0.9, and 0.1, with the other potential goals having equal probability. For each setting, the policy is computed offline and the robot updates its belief online.

The *order-k* of the optimal policy with respect to the true goal in each belief setting is 4, due to stochastic transitions. The *order-k* for the optimal policies of the GUSSP with deterministic transitions for this problem are: 3, 1, and 4, corresponding to the three initial beliefs.

4.6 Summary

The goal uncertain SSP (GUSSP) provides a natural model for representing the *known unknowns* about goal states in problems where it is non-trivial to identify the exact goals ahead of plan execution. While a general GUSSP could be intractable, several tractable classes of GUSSPs are identified in Section 4.3 and effective approaches for solving them are proposed. Specifically, a GUSSP with a myopic observation function reduces to an SSP, allowing us to efficiently solve it using existing SSP solvers. To solve the compiled-SSP, an admissible heuristic that accounts for goal uncertainty in its estimation is proposed. In addition, a determinization-based approach to solve GUSSP is also presented in Section 4.4. The empirical results show that GUSSPs can be solved efficiently using scalable algorithms that do not rely on POMDP solvers.

CHAPTER 5

NEGATIVE SIDE EFFECTS

The previous chapters considered settings in which the agent is aware of the model imprecision, either because it simplified the model to accelerate planning or because the model included information to indicate uncertainty over missing details. In many scenarios, however, the agent has no prior knowledge about which aspects of the environment are unmodeled.

This chapter focuses on the potential undesirable impacts of planning with imprecise models in settings where the agent is *unaware* of the model imprecision. Specifically, the agent's model is assumed to include all the details in the environment required to complete its assigned task (or goal) but other details in the environment are ignored. As a result, the agent's actions may have *negative side effects*, which are undesirable, unmodeled effects of agent actions on the environment [93, 99]. Negative side effects are inherently challenging to identify at design time, and may affect the reliability, usability and safety of the system.

Chapter outline. The chapter begins by motivating the problem in Section 5.1. Section 5.2 formally defines negative side effects of agent actions. Section 5.3 identifies key characteristics of negative side effects. Section 5.4 presents results from our user study to understand how users respond to negative side effects of AI systems. Finally, Section 5.5 highlights the challenges in avoiding negative side effects.

5.1 Introduction

Deploying AI systems requires complex design choices to support safe operation in the open world. During the design and initial testing, the system designer typically ensures that the agent's model includes all the necessary details relevant to its assigned task. Inevitably, some details in the environment that are unrelated to the agent's primary objective may be ignored. The incompleteness of any given model—handcrafted or machine acquired—is inevitable due to practical limitations in model specification (the ramification and qualification problems) [21], and limited information that may be available during the design phase [95], including unanticipated domain characteristics and cultural differences among the target user and development team [99]. Due to the limited fidelity of its model, an agent's actions may produce *negative side effects* (NSE) [3, 37, 92, 98, 99]. Negative side effects are undesired, unmodeled effects of an agent's actions that occur in addition to the agent's intended effects when operating in the open world (Figure 5.1).

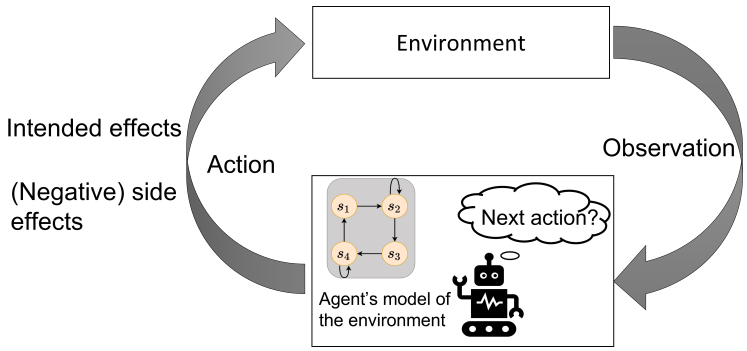


Figure 5.1: Negative side effects of an agent's behavior.

For example, consider an autonomous vehicle (AV) that was carefully designed and tested for safety aspects such as yielding to pedestrians and conforming to traffic rules. When deployed, the AV may not slow down when driving through puddles and splash water on nearby pedestrians. Another documented example of undesirable behavior in AVs is the vehicle swerving left and right multiple times to localize itself

for active lane-keeping. During this process, the vehicle rarely prompted the driver to take control [42]. This behavior, especially on curvy and hilly roads, can startle the driver or cause panic.

Undesirable behaviors may occur even when performing relatively simple tasks. For example, robot vacuum cleaners are becoming increasingly popular and they have a simple task—to remove dirt from the floor. A robot vacuum cleaner in Florida ran over animal feces in the house and continued its cleaning cycle, smearing the mess around the house [106]. In an extreme case in South Korea, a robot vacuum cleaner locked into the hair of a woman who was sleeping on the floor, mistaking her hair for dust [67].

In these examples, the agent was performing its task, perhaps optimally with respect to the information provided to it, but there were serious negative side effects to the agent's actions. Design decisions that may be innocuous during initial testing may have a significant impact when a system is widely deployed. For example, the issue of a Roomba locking into the hair of a person lying on the floor emerged only after the system was deployed in Asia. In practice, it is not feasible to anticipate all possible negative side effects and accurately encode them in the model at design time. As a result, side effects are often discovered after the system is deployed.

Negative side effects occur because the agent's model and objective function focus on some aspects of the environment but its operation could impact additional aspects of the environment. The value alignment problem studies the unsafe behavior of an agent when its objective does not align with human values [33, 86, 87]. Misaligned systems are more likely to produce negative side effects. However, the occurrence of negative side effects does not necessarily indicate that there is a value alignment problem. Negative side effects can occur even in settings where the agent optimizes legitimate objectives that align with the user's goals, due to incomplete knowledge and distributional shift. For example, while driving in Boston, AVs that are programmed

to not run into obstacles were stopped by the local breed of unflappable seagulls standing on the street [18]. Not running into obstacles is well-aligned with the users' intentions and objectives, but there are side effects because the agent lacks knowledge that it can edge to startle the birds and then continue driving. In fact, such knowledge was later added to the system to resolve the problem. In addition, some systems may cause unavoidable negative side effects that cannot be mitigated. While the side effects may be undesirable, the user may accept the system as is, once they learn about it and recognize that the side effects are unavoidable. In such cases, we cannot say that there is a value alignment problem, even though the negative side effects may occur.

Knowledge gaps Identifying and overcoming the negative side effects of agent actions is a nascent, but fast emerging field that is attracting increased attention within the AI community [3, 32, 37, 56, 86, 92, 104, 113, 122]. However, there exists no formal definition of negative side effects of agent actions and its characteristics. Further, there are no published reports on how users respond to NSE, their willingness to tolerate NSE, and how side effects affect their trust in the AI system. These factors are critical in evaluating existing solutions and developing new approaches that are realistic and deployable in the real-world.

Overview of contributions The primary contributions in this chapter are: (1) a formal definition of negative side effects of agent actions; (2) identifying the key characteristics of negative side effects; (3) presenting results from our user study conducted specifically to understand user attitudes towards negative side effects; and (4) a discussion of the inherent challenges in avoiding negative side effects.

5.2 Definition of Negative Side Effects

Consider an environment E in which an agent reasons using its acquired model—a Markov decision process (MDP) $\tilde{M} = \langle \tilde{S}, \tilde{A}, \tilde{T}, \tilde{R} \rangle$. The agent's state space is denoted by $\tilde{S}$, its action space by $\tilde{A}$, transition function by $\tilde{T}$, and its reward function by $\tilde{R}$. The agent's model $\tilde{M}$ includes a single objective, which is its primary task, and all the components necessary to complete this primary task. A factored state representation is considered. A *primary policy* is an optimal policy for $\tilde{M}$, optimizing the agent's primary objective defined by $\tilde{R}$. Executing a primary policy may produce *negative side effects* (NSE) in some states since it only optimizes the primary task. It is assumed that there exists a function that determines the probability of NSE occurrence associated with an agent trajectory in an environment.

Definition 21. *Let $\zeta = \{(s_1, a_1), \ldots, (s_n, a_n)\}$, $n \geqslant 1$, be a finite trajectory of an agent operating in an environment $E \in \mathcal{E}$, with $\mathcal{E}$ denoting a finite set of environment configurations. Let Z denote the set of possible trajectories in $\mathcal{E}$. A* ***negative side effect*** *is an event that occurs when the agent executes certain trajectories in an environment, with probability determined by a function $N : Z \times \mathcal{E} \to [0, 1]$ where $N(\zeta, E)$ denotes the probability of negative side effects occurring when the agent follows a trajectory $\zeta \in Z$ in environment $E \in \mathcal{E}$.*

When $|\zeta| = 1$, such as $\zeta = \{(s_1, a_1)\}$, $N(\zeta, E)$ denotes the *immediate*, Markovian negative side effects of an agent's action. States in which an agent's actions have immediate NSE are called *susceptible states*. The severity of NSE may range from tolerable events to severe impacts, depending on the agent's primary task and the environment in which it is situated. In general, a user can estimate N by observing the agent behavior and its consequences. Tolerance to NSE and the associated penalty are user-specific. Let $\theta \in \Theta$ denote user tolerance parameter for NSE in a given environment, where Θ denotes the discrete set of values θ can take.

Definition 22. *The **penalty** for negative side effects is defined by* $R_{N,\theta} : Z \times \mathcal{E} \rightarrow \mathbb{R}$*, with* θ *denoting user tolerance parameter and* $R_{N,\theta}(\zeta, E)$ *denoting the penalty associated with trajectory* $\zeta \in Z$ *in environment* $E \in \mathcal{E}$*.*

Since NSE occurrence depends on the environment and the penalty depends on user tolerance, it is challenging to accurately specify details about NSE during system design. Further, the designer may inadvertently omit details that are unrelated to the primary task. As a result, the agent often does not have prior knowledge about the side effects of its actions and therefore does not have the ability to minimize NSE. When the agent learns the penalty for NSE from a single user and optimizes its performance based on their preferences and tolerance, the parameter θ is dropped and the penalty is denoted simply as R_N. The side effects may be avoidable or unavoidable by the agent, when it is performing its assigned task.

Definition 23. *Negative side effects are **avoidable** in an environment* E *if there exists a policy* $\tilde{\pi}$ *to complete agent's assigned task such that* $N(\zeta, E) = 0, \forall \zeta \sim \tilde{\pi}$*, and unavoidable otherwise.*

5.3 Taxonomy of Negative Side Effects

This section introduces a taxonomy of negative side effects, outlined in Table 5.1. Understanding the characteristics of negative side effects helps design better solution approaches to detect and mitigate their impacts in deployed systems.

Severity The severity of negative side effects ranges from mild side effects that can be largely ignored to severe impacts that require suspension of the system deployment. Safety-critical side effects, such as an AV failing to detect a construction worker's hand gestures [19], are typically addressed by redesigning the model and hence require extensive evaluation before redeployment. I conjecture that many negative side effects lie in the middle with significant impacts that require attention, but not sufficiently

Property	Property Values
Severity	Ranges from mild to safety-critical
Reversibility	Reversible or irreversible
Avoidability	Avoidable or unavoidable
Frequency	Common or rare
Stochasticity	Deterministic or probabilistic
Observability	Full, partial, or unobserved
Exclusivity	Prevent task completion or not

Table 5.1: Taxonomy of negative side effects.

critical to suspend the service. An autonomous vehicle that does not slow down when going through shallow puddles can cause significant impacts, but those are unlikely to be considered sufficiently critical to roll-back its deployment, particularly if mechanisms are provided to mitigate the negative impacts.

Reversibility Side effects are reversible if the impact can be reversed or negated, either by the agent causing it or via external intervention. For example, breaking a vase is an irreversible side effect, regardless of the agent's skills [3]. Side effects such as leaving marks on a wall can be fixed by repainting it, but the agent may require external assistance for it.

Avoidability In some problems, it may be impossible to avoid the negative side effects during the course of the agent's operation to complete its assigned task. This introduces a trade-off between performing agent's assigned task and avoiding the side effects. For example, the side effects of driving through puddles are unavoidable if all roads leading to the destination have puddles. Addressing unavoidable NSE requires a principled approach to balance the trade-off between avoiding side effects and optimizing the completion of the assigned task.

Frequency The frequency of occurrence of negative side effects depends on the environmental conditions and the action plan. Certain NSE may occur rarely, considering all use cases, but may occur frequently for a small subset of cases. A robot pushing a box over a rug may dirty it as a negative side effect. This is an example of a frequently occurring negative side effect when the domain of operation is largely covered with a rug. The frequency of occurrence impacts the choice of solution approach to identify negative side effects and the corresponding mitigation approach.

Stochasticity The occurrence of negative side effects may be deterministic or probabilistic. Deterministic NSE always occur when some action preconditions arise in the open world. Side effects are probabilistic when their occurrence is not certain even when the right preconditions arise. For example, there may be a small probability that a robot may accidentally slide and scratch the wall while pushing a box, but that undesired effect may happen only 10% of the times the robot slips. Solution approaches designed to handle deterministic side effects can often be extended to stochastic settings, and vice versa, with minor modifications.

Observability The agent's observability of the actual NSE or the conditions that trigger them are generally determined by the agent's state representation and sensory input. Side effects may be fully observable, partially observable, or even unobserved by the agent. Observing a side effect is different from *identifying or recognizing* the impact as a side effect. For example, the agent may observe the scratch it made on the wall but may not be aware that it is undesirable, and as a result may not try to avoid it. Observability is a critical factor when learning to avoid NSE. When an external authority provides feedback to the agent, it may be sufficient for the agent to observe the conditions that trigger the negative side effect. However, when an agent may need to identify NSE on its own, it needs more complex general knowledge about its environment.

Exclusivity Negative side effects may prevent the agent from completing its assigned task. This category is relatively easier to identify. Often, however, the side effects negatively impact the environment without preventing the agent from completing its assigned task. Such side effects are more common in the real-world and are difficult to identify at design time.

5.4 Understanding User Attitudes Towards NSE

Despite the interest in addressing the negative side effects of AI systems [56, 57, 92, 98, 99, 113, 122], there has been no prior efforts to understand user attitudes towards NSE.

Inconsistent and unpredictable system behavior, some of which may be unsafe, affects user trust in the system's capabilities and operation. In fact, studies show that users may stop trusting a system after witnessing a mistake, even if the system outperforms humans in the task [22]. Recently, researchers investigated the effect of accuracy on user expectations and trust in machine learning models [51, 117]. These results show that user trust in the system diminishes when the observed accuracy is lower, regardless of its stated accuracy. User surveys conducted to understand how users interact with specific systems, such as self-driving cars [41, 59] and autonomous vacuum cleaners [28], highlight the concerns and promise of these technologies, and how they are perceived by users from different backgrounds. While these studies provide a broad overview of user expectations and trust in AI systems, they do not provide specific insights about the side effects problem.

User tolerance of NSE depends on many factors such as their individual preferences and the severity of the side effect. When the NSE are safety-critical, it is clear that users will not tolerate them and system's operation needs to be suspended to address the NSE and reevaluate the system performance. In many deployed systems, however, the impacts of NSE are significant but not catastrophic. How do users respond to

such side effects? Since this is an emerging topic, a survey conducted specifically to identify general user attitudes towards NSE is critical to develop effective solutions to this practical problem. Our user study focused on answering the following questions.

1. To what extent are users willing to tolerate negative side effects that are not safety-critical?

2. How do negative side effects affect the user's trust in the system?

3. Are users willing to assist the system in mitigating the impacts of the side effects—by providing feedback, applying minor changes to the environment, or specifying regions where the system can operate?

4. Are users willing to tolerate a sub-optimal behavior of the AI system (such as taking a longer route) in order to avoid negative side effects?

Answering these questions will deepen our understanding of the side effects problem, and shape future research directions on this topic.

5.4.1 Survey Design

Domains Two IRB-approved surveys were conducted, focusing on NSE in two domains: an autonomous vacuum cleaner (Roomba) and an autonomous vehicle (AV). The survey considered NSE such as the Roomba spraying water on the wall when cleaning the floor, the AV driving fast through potholes which results in a bumpy ride for the users, and the AV slamming the brakes to halt at stop signs which results in sudden jerks for the passengers.

The Roomba domain represents a setting where the NSE is relatively mild, the users do not directly experience the NSE, and the system does not require constant supervision when it is performing its task. The AV domain represents a setting in which the NSE have moderate impact, the users experience the NSE directly

(bumpiness or sudden jerk), and users generally supervise the AV performance and can take control when issues related to safety arise.

Participants 500 participants were recruited on Amazon mechanical turk to complete a pre-survey questionnaire to assess their familiarity with AI systems and fluency in English. This questionnaire has six questions and takes less than 30 seconds to complete. All participants were informed about the purpose of the study. Based on the pre-survey responses, 300 participants aged above 30 were invited to complete each survey (Roomba and AV). The age criteria was included for selection because study shows that participants aged above 30 are less likely to game the survey conducted on the Mturk platform [24]. The surveys generally take less than ten minutes to complete. Responses that were incomplete or with a survey completion time of less than one minute were discarded. A total of 204 valid responses were received for the Roomba domain and 183 valid responses were received for the AV domain. To facilitate a direct comparison between the responses in both the domains, 183 responses were randomly sampled for the Roomba domain.

5.4.1.1 Survey Questionnaire

The survey questionnaires contained similar questions for the two domains, with ten questions for the Roomba domain and eleven questions for the AV domain. The questions included a description of NSE and participants were required to select an option that best describes their attitude. To understand the effect of severity of NSE on user tolerance, two forms of NSE are studied in the AV domain: bumpiness and sudden jerks. All other survey questions for the AV domain focused *only* on the bumpy ride side effect. The survey questionnaires are included in Appendix A.

User Tolerance For each domain, our survey participants were required to indicate their level of tolerance of NSE: *low*—indicating their unwillingness to use the AI

system due to its NSE; *medium*—indicating the system will be used less frequently due to its NSE; and *high*—indicating their willingness to continue using the system, despite the occurrence of NSE.

Trust To determine if NSE affected user trust in the system's capabilities, participants were required to select an option that best describes their trust level: *low*—do not trust the system to be capable of completing its task; *medium*—trust is affected if the system does not learn to avoid NSE over time; and *high*—trust is unaffected by NSE. This simple categorization allows us to understand how NSE may affect the system usability.

Slack Preferences In many instances, NSE can be avoided if the system is allowed to act sub-optimally with respect to its assigned task. For example, the bumpy ride—which occurs when the AV drives fast through potholes as a result of optimizing travel time—may be avoidable if the AV takes a longer route or navigates at a lower speed. A fixed slack of 25% for the AV domain allows the AV to drive slow or pick an alternate route that takes up to 25% longer to reach the destination. Similarly, the slack for the Roomba domain allows it to skip cleaning the area within five inches from the wall. Slack for the Roomba domain can be considered as allowing the system to not complete its task fully, while the slack for the AV allows it to take longer to complete the task. Participants were required to select yes or no, to indicate their willingness of allowing for a slack.

Willingness to Assist Humans can leverage their skills and knowledge about the environment to assist the agent in avoiding negative side effects. For instance, AI systems often operate in environments that are configurable, which can be leveraged to mitigate NSE. Participants were surveyed to determine their willingness to reconfigure the environment in order to mitigate the impacts of NSE. Reconfigurations for the Roomba domain involved installing a protective sheet on the surface to overcome

the NSE of spraying water on the walls. Reconfiguration in the AV domain involved installing a pothole-detection sensor that detects potholes and limits the velocity of the vehicle. Participants were required to select an option that best describes their attitude: purchase and install the sheet or sensor, install the sheet or sensor if it is provided by the manufacturer, and not willing to reconfigure.

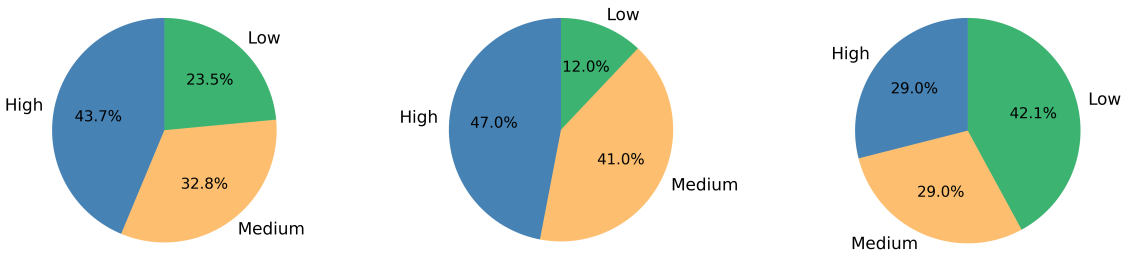
In addition, the questionnaires focused on eliciting user preferences over providing feedback and how often they are willing to provide feedback by pressing a button when they notice NSE. This is to validate a common approach in AI research which indicates that feedback, particularly from users, can be used to improve the performance of the AI system [37, 80, 92].

Further, the survey gathered information about what type of tools will encourage them to continue using the system. The participants were asked to select *all* the tools they would be willing to utilize to mitigate NSE. They were presented with three tools: providing feedback by pressing a button every time the system produces NSE; tools to reconfigure the environment; and specifying areas where the system is not allowed to operate due to NSE.

5.4.2 Results

This section presents the results of the user study. A detailed discussion of the results is presented in the next section.

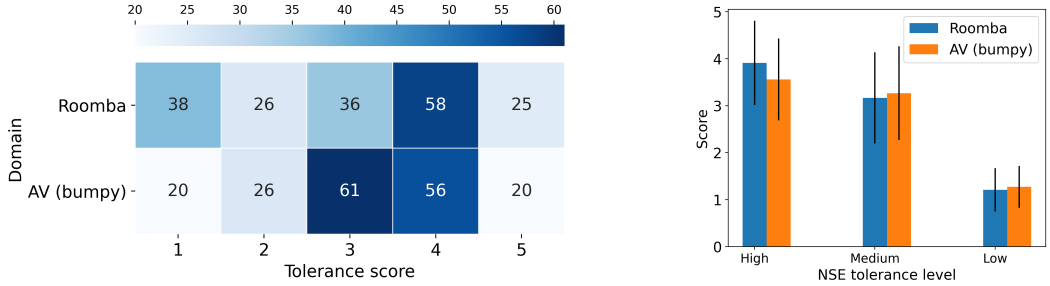
User Tolerance of Negative Side Effects Responses for the Roomba setting show that 76.50% of the participants are willing to tolerate the negative side effects. For the driving domain, 87.97% of the respondents are willing to tolerate milder NSE such as bumpiness when the AV drives fast through potholes and 57.92% are willing to tolerate relatively severe NSE such as hard braking at a stop sign. These results are plotted in Figure 5.2.



(a) Roomba (b) AV mild (bumpy) (c) AV moderate (jerks)

Figure 5.2: User tolerance of negative side effects.

Participants were also required to enter a tolerance score on a scale of 1 to 5, with 5 indicating the highest level of tolerance. Figure 5.3(a) shows the score distribution and Figure 5.3(b) shows the average score in each NSE tolerance category, along with the standard deviation.



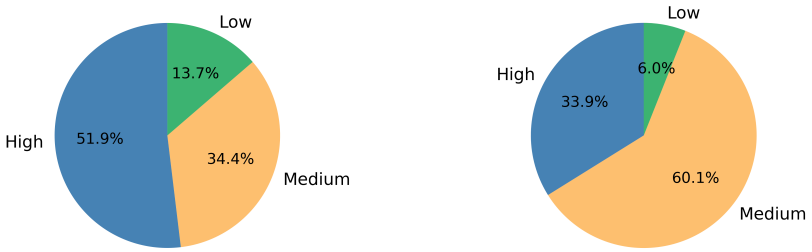
(a) Score distribution

(b) Average score and standard deviation per tolerance level

Figure 5.3: Tolerance score.

For the Roomba domain, 65.02% voted a score of 3 or more. Similarly for the AV (bumpy) domain, 74.86% voted a score of 3 or more. The mean tolerance score, along with the 95% confidence interval, is 3.03 ± 0.20 for the Roomba domain and 3.18 ± 0.16 for the AV domain. The relation between tolerance level and score cross-validates the responses on user tolerance, as users with a higher tolerance of NSE consistently assigned a higher tolerance score.

Effect on Trust For the Roomba domain, 51.91% respondents selected high trust and 34.43% selected medium trust. Similarly for the AV domain, 34.43% selected high trust and 60.10% selected medium trust. The remaining participants indicated that they do not trust the system to be capable of completing its assigned task, when it produces NSE. These results are shown in Figure 5.4.



(a) Roomba (b) AV

Figure 5.4: Effect of negative side effects on trust.

Domain	NSE tolerance	Trust Level (% responses)		
		Low	Medium	High
Roomba	High	0%	21.25%	78.75%
	Medium	3.33%	50.00%	46.67%
	Low	53.49%	37.21%	9.30%
AV	High	0%	47.67%	52.32%
	Medium	5.33%	76.00%	18.67%
	Low	31.82%	54.54%	13.64%

Table 5.2: User trust in the system, corresponding to NSE tolerance level.

Table 5.2 shows the relationship between user trust and tolerance of NSE. The correlation coefficient between user tolerance of NSE and their trust in the system's capabilities when it produces NSE is 0.65 for the Roomba domain and 0.47 for the AV (bumpy) domain.

Slack Preferences Among the 183 responses, 66.12% are willing to allow for a slack to avoid NSE of the AV. Similarly, 45.36% are willing to allow the Roomba to

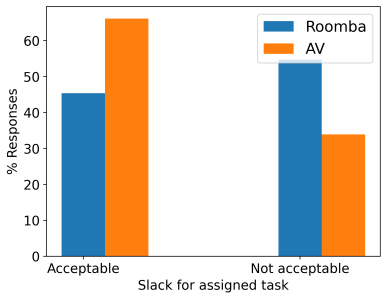


Figure 5.5: Slack preferences to mitigate NSE

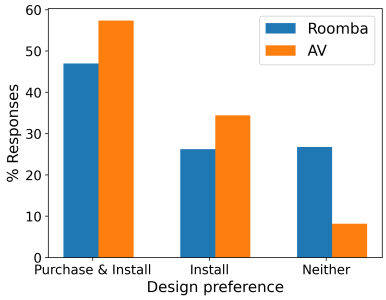


Figure 5.6: Willingness to reconfigure the environment

skip cleaning areas near the wall so as to avoid the negative side effects. These results are plotted in Figure 5.5.

Table 5.3 reports the relationship between user tolerance of NSE and their slack preferences. The results suggest that participants with high NSE tolerance are generally willing to allow for a slack. The correlation coefficient between user tolerance of NSE and their slack preferences is 0.4 for the Roomba domain and 0.07 for the AV (bumpy) domain.

Domain	NSE tolerance	Slack preference (% responses)	
		Acceptable	Not Acceptable
Roomba	High	66.25%	38.33%
	Medium	38.33%	61.66%
	Low	16.28%	83.72%
AV (bumpy)	High	66.28%	33.72%
	Medium	70.67%	29.33%
	Low	50.00%	50.00%

Table 5.3: Slack preferences of users, corresponding to NSE tolerance.

Willingness to Assist the System Results on the Roomba domain show that 73.22% respondents are willing to install the sheet to mitigate NSE. If the sheet is not provided by the manufacturer, 64.18% are willing to purchase the sheet ($10). In the AV domain, 91.80% are willing to install the sensor. If the sensor is not provided by

the manufacturer, 57.38% are willing to purchase the sensor ($50). These results are reported in Figure 5.6. Table 5.4 reports user willingness to apply minor modifications to the environment, corresponding to their NSE tolerance. Users with low tolerance of NSE are less willing to perform reconfigurations.

Domain	NSE tolerance	User Preference (% responses)		
		Purchase & Install	Install	Neither
Roomba	High	71.25%	20.00%	8.75%
	Medium	40.00%	40.00%	20.00%
	Low	11.63%	18.60%	69.77%
AV	High	66.28%	30.23%	3.49%
	Medium	52.00%	38.67%	9.33%
	Low	40.91%	36.36%	22.73%

Table 5.4: Willingness to reconfigure the environment, corresponding to NSE tolerance

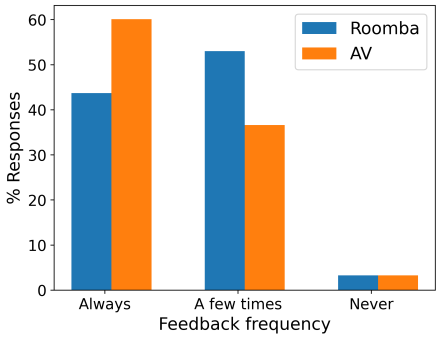


Figure 5.7: User willingness to provide feedback to the system

Figure 5.7 plots user willingness to provide feedback. In the Roomba domain, 43.71% participants are willing to provide feedback until the Roomba learns to overcome its undesirable behavior, 53.00% are willing to provide feedback a few times and when they are around the system to supervise it, and 3.29% are not interested in providing feedback. The AV (bumpy) domain has a similar trend—60.11% are willing to provide feedback until the AV learns to overcome the NSE, 36.61% are willing to provide feedback a few times, and 3.29% are not willing to provide feedback. Table 5.5

reports user interests in utilizing the available tools to mitigate the impacts of NSE. As participants could select more than one tool, number of responses corresponding to each tool are reported.

Tool	AV (bumpy)	Roomba
Feedback	30	154
Reconfigure environment	25	5
Specify operation regions	22	18
Feedback + Reconfigure environment	18	4
Feedback + Specify operation regions	45	0
Reconfigure environment + Specify operation regions	9	0
Feedback + Reconfigure environment + Specify operation regions	65	0

Table 5.5: Number of responses corresponding to tools that encourage users to continue using the system, when NSE occur.

5.4.3 Discussion

In both domains, higher NSE tolerance correlates with higher trust. Respondents with lower NSE tolerance have low to medium trust in the system. These results indicate that (1) individual preferences and tolerance of NSE varies and depends on the severity of NSE; (2) users are generally willing to tolerate NSE that are not severe or safety-critical, but prefer to reduce them as much as possible; and (3) users are generally willing to give the AI systems some time to learn to avoid the side effects and their trust is affected when the system does not adapt. The results also highlight the importance of developing techniques to mitigate NSE in order to design trustworthy AI systems. In addition, the results support the design of customizable systems to improve user satisfaction.

The results on slack tolerance (Figure 5.5 and Table 5.3) indicate that respondents are more willing to allow for a slack in the AV (bumpy) domain. In fact, at least 50% of the respondents are willing to allow for a slack, independent of their NSE tolerance. This is likely because the behavior with a slack—selecting a longer route

or driving slowly to avoid a bumpy ride—is similar to how human drivers try to avoid the NSE. Since many respondents expressed willingness to allow for a slack in the AV, independent of their tolerance of NSE, the correlation coefficient for the AV domain has a lower value than the Roomba domain. Overall, the results indicate that users are generally willing to accept sub-optimal behavior with respect to the system's assigned task in order to mitigate NSE, as long as the system completes its assigned task.

Our results also show that respondents prefer the direct feedback method the most (Table 5.5). This is likely due to the simplicity of the interaction with the system, as they are required to only press a button every time they observe an undesirable behavior. Furthermore, the results in Figure 5.7 show that a higher fraction of respondents are willing to provide feedback to an AV *until* it learns to avoid the NSE. This is likely because the users of an AV are usually in the vehicle when it operates, making it is easier to provide feedback when they observe the NSE. The results suggesting user willingness to provide feedback, often until the system learns to avoid NSE, backs an important assumption in current AI research.

The results in Figure 5.6 and Table 5.4 indicate that users are generally willing to engage in environment reconfiguration to mitigate the impacts of NSE. In fact, many respondents expressed willingness to pay for procuring the items for reconfiguration. Further, our respondents are more willing to utilize all the tools available to mitigate the NSE in the AV domain, compared to the Roomba setting. This interest may be due to the direct implications of the NSE on the user's ride experience. These results are in accordance with a recent study that shows that users are generally willing to tolerate an imperfect AI system if they are able to make minor modifications to its performance and outcomes [23]. Since people prefer different tools to mitigate NSE, depending on the severity and their preferences, it is important to recognize that no one solution approach will work well for all settings.

Different trends in the results may be observed when the assumptions made in our study are relaxed. Our study focused on a setting where the negative side effects are (1) *known* to the user—we fully describe the side effects to the participants; (2) *deterministic*—the same type of negative side effects always occur when the system executes a certain action, such as the sudden jerk to the passengers when the AV halts suddenly; and (3) *transparent*—the users can observe the occurrence of these side effects. When a new user interacts with a system, they may not know what types of NSE to anticipate and whether their occurrence is stochastic. The results of this study and the trends observed may change when users are uncertain about when and why the NSE occurs or what the NSE may be. This study focused on NSE that are undesirable but not safety-critical. User tolerance, trust, slack preferences, and the preferred tools will likely change when NSE are severe or safety-critical. Furthermore, this survey considered participants aged above 30 since they are less likely to game the Mturk platform. The results trend may be different with a younger population who may be more willing to tolerate certain types of negative side effects in the interest of adopting new technologies early.

Overall, this user study encourages the development of effective mechanisms to identify and mitigate negative side effects of deployed AI systems as a way to increase their usability, trustworthiness, and cost effectiveness. The following section discusses some of the inherent challenges in identifying and avoiding negative side effects.

5.5 Challenges in Avoiding Negative Side Effects

The challenges in avoiding negative side effects broadly stem from the difficulty in obtaining knowledge about NSE a priori, gathering user preferences to understand their tolerance for side effects, and balancing the potential trade-off between completing the task and avoiding the side effects.

Prior knowledge about NSE There are three key reasons why the agent may not have prior knowledge about NSE. First, predicting or identifying NSE a priori is inherently challenging, since it depends on the environmental conditions. As a result, this information is often missing in the agent's model. Second, many AI systems are deployed in a variety of settings, which may be different from the environment used for agent training and testing. This distributional shift may cause NSE and is difficult to assess during the design process. Third, it is generally difficult to precisely learn or encode individual user preferences about NSE, or account for cultural differences between the target users and the development team.

Feedback collection An agent that is unaware of the side effects of its actions can gather this information through various feedback mechanisms from users or through autonomous exploration. Though learning from feedback produces good results in many problems [60, 81, 92, 121, 122], there are three potential challenges in employing this approach in real-world systems. First, the learning process may require feedback in a certain format to be sample efficient, such as correcting the agent policy by providing alternate actions for execution. Feedback collection in general is an expensive process, particularly when the feedback format requires constant human oversight or imposes significant cognitive overload on the user. Second, the feedback may be biased or delayed or both, which in turn affects the agent's learning process. Finally, it is generally assumed that the agent uses human-interpretable representations for querying and feedback collection, but there may be mismatches between the models of the agent and human. There are some recent efforts towards addressing the problem of sample efficiency in learning [16, 114] and investigating the impact of bias in feedback for agent learning [80, 92]. Identifying and evaluating human-interpretable state-action representations for querying humans is largely an open problem.

Managing side-effect tradeoffs When negative side effects are unavoidable and interfere with the performance of the agent's assigned task, there is a trade-off between completing the task efficiently and avoiding the NSE. To what extent should an agent deviate from its optimal policy in order to minimize the impacts of negative side effects? Balancing this trade-off requires user feedback since it depends on their tolerance for negative side effects. This can be challenging when the agent's objective and the side effects are measured in different units.

Due to the factors discussed above, avoiding NSE is a non-trivial problem. The next chapter discusses two solution approaches to detect and mitigate negative side effects, when the agent does not have any prior knowledge about them.

CHAPTER 6

AVOIDING NEGATIVE SIDE EFFECTS

This chapter presents two complementary solution approaches to mitigate the immediate negative side effects via: (1) learning from feedback [91, 92], and (2) environment shaping [98]. The solution approaches target settings with different assumptions and agent responsibilities, focusing on negative side effects that are (1) undesirable, but not safety-critical; (2) deterministic—always occur when the agent executes certain actions in some states; and (3) not preventing the agent from completing its assigned task. In addition, the agent has no prior knowledge about the NSE of its actions and the side effects are experienced immediately after action execution.

Chapter outline. The chapter begins by motivating the problem in Section 6.1 and a discussion of the related work, contrasting their benefits and limitations, in Section 6.2. Section 6.3 formalizes the problem of mitigating negative side effects as a multi-objective MDP with slack and presents results from empirical evaluation of the approach on two domains. Section 6.4 discusses the challenges in learning from feedback. Section 6.5 presents environment shaping as a method to mitigate negative side effects and results from empirical evaluation of the approach on two domains. Section 6.6 discusses the challenges and limitations in mitigating side effects via shaping.

6.1 Introduction

When an agent operates based on an incomplete model of the world, its actions may create *negative side effects* (NSE) [3, 37, 92, 98, 99]. Different types of model

incompleteness lead to different types of negative side effects, whose severity ranges from mild events that are tolerable to severe impacts. Mitigating NSE is critical to improve the safety and reliability of deployed AI systems. However, it is practically impossible to identify all the NSE at design time since agents are deployed in varied settings. Deployed agents often do not have any prior knowledge about NSE, and therefore they do not have the ability to minimize them.

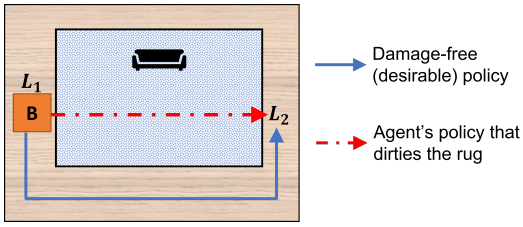


Figure 6.1: Illustration of negative side effects in the boxpushing domain: a policy based on the agent's incomplete model, which overlooks the impact of pushing the box over the rug, dirties the rug.

This chapter focuses on NSE that are undesirable, but not catastrophic or prohibitive. That is, the NSE do not affect the agent's ability to complete its assigned task. For example, consider an agent that is required to push a box B as quickly as possible from location L_1 to location L_2 (Figure 6.1). The agent's model includes all the details essential to optimize the time taken to push the box, including the reward for pushing the box and the associated transition dynamics. However, details such as the impact of pushing the box over a rug may not be included in the model if the issue is overlooked during system design. Consequently, when operating based on this model, the agent may push the box over the rug, dirtying the rug as a side effect. *How can we mitigate the negative side effects, when the agents are unaware of the side effects and the associated penalties?*

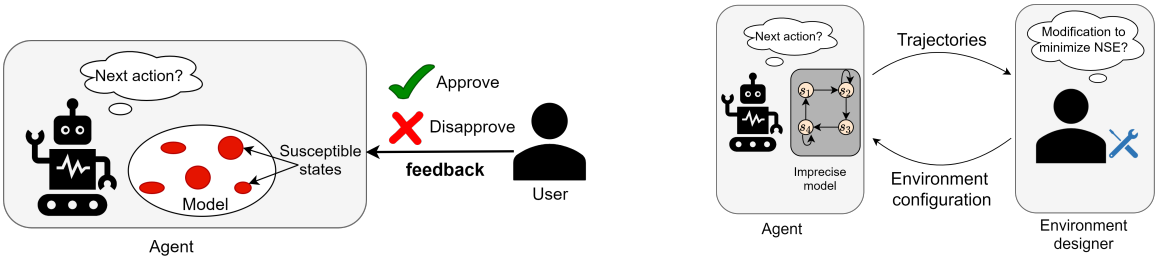
Knowledge gaps Prior works mitigate NSE by redesigning the reward function for the agent's primary objective [32], constraining the agent's actions [122, 121], minimizing deviations from a baseline state [56], or maximizing the attainable util-

ity of auxiliary objectives and completion of future tasks [57, 112, 113]. All these approaches target settings with avoidable NSE and assume that the agent has some prior knowledge about the cause and occurrence of NSE. However, these assumptions are often violated when deploying AI systems in the open world. While some negative side effects could be anticipated or detected during system development, many types of negative side effects are discovered after the system is deployed. This is due to a variety of factors such as unanticipated domain characteristics, unanticipated consequences of system or software upgrade, or cultural differences among the target user and development team. Furthermore, in situations where the primary objective is prioritized, the user may be willing to trade off some NSE for solution quality. For example, if the agent takes ten times longer to push the box so as to avoid the rug area, the user may be willing to tolerate some dirt on the rug instead. Prior works, however, do not offer a principled approach to balance this trade-off and do not provide bounded performance guarantees. Table 6.1 summarizes the characteristics of the existing approaches.

Approach	Multi-objective formulation	Bounded-guarantees wrt. agent's task	Generalize learned NSE across states	Distinguish and handle different magnitudes of NSE
[32]	✗	✗	✓	✗
[122]	✗	✗	✗	✗
[104]	✓	✗	✗	✗
[56]	✓	✗	✗	✓
[121]	✗	✗	✗	✗
[112]	✗	✗	✗	✗
[113]	✗	✗	✗	✗
Our approaches	✓	✓	✓	✓

Table 6.1: An overview of the characteristics of different approaches in mitigating NSE.

Overview of contributions To mitigate NSE in settings where the agent does not have prior knowledge about the side effects of its actions, two complementary approaches (Figure 6.2) that have different properties and underlying assumptions are proposed: (1) learning from feedback [92], and (2) environment shaping [98]. Our user study results [93] discussed in the previous chapter show that users are willing to provide feedback and perform reconfigurations to mitigate NSE.



(a) Learning from user feedback

(b) Environment shaping

Figure 6.2: An illustration of our proposed complementary solution approaches with varying nature of autonomy to mitigate NSE.

In learning from feedback approach, the agent's model is augmented with a secondary reward function that represents the penalty for NSE of its actions. Separating the reward functions of the agent's primary objective and NSE allows us to ensure that the model aspects corresponding to agent's assigned task are not corrupted during model update. The efficiency of different feedback mechanisms, including human feedback and autonomous exploration, to learn about NSE are investigated. The problem is formulated as a multi-objective Markov decision process with lexicographic preferences over reward functions, such that optimizing the completion of the agent's assigned task is prioritized over mitigating NSE. Using multi-objective models makes it easier for users to understand and control the trade-off between the agent's ability to mitigate NSE and optimize the completion of its assigned task. This proposed approach exploits the reliability of the existing model with respect to the primary objective, while allowing a deployed agent to learn to avoid NSE as much as possible.

The second solution approach examines how a human can assist an agent, beyond providing feedback, and exploits their broader scope of knowledge to mitigate the impacts of NSE. *Environment shaping* is the process of applying simple modifications to the current environment to make it more agent-friendly and minimize the occurrence of NSE. The problem is formulated as a human-agent collaboration with decoupled objectives. The agent optimizes the completion of its assigned task and may produce NSE during its operation. The human assists the agent by performing modest reconfigurations of the environment so as to mitigate the impacts of NSE, without affecting the agent's ability to complete its assigned task. An algorithm is presented for shaping and its properties are analyzed.

The proposed solution approaches are complementary, with different agent and human responsibilities. In learning from feedback, the agent is directly involved in mitigating NSE as it learns about them and updates its model, while the human facilitates this by providing feedback. In environment shaping, the human has more control over the environment, by definition, and the agent has an indirect involvement in mitigating NSE since it only reacts to the reconfiguration by revising its policy. Table 6.2 summarizes the characteristics and assumptions of the two approaches. Empirical evaluations demonstrate the trade-offs in the performance of different approaches in mitigating NSE across settings.

Assumptions/ Requirements	**Learning from Feedback**	**Environment Shaping**
Agent state representation	sufficiently rich to learn about NSE	–
Model update requirements	augment with NSE penalty function	no model update
Type of human assistance	provide immediate feedback	reconfigure environment
Factors affecting performance	sampling biases in feedback collection	designer's ability to accurately predict actor policy from trajectories
Environment configuration	–	user-configurable & described by a fie
Unavoidable NSE	cannot be eliminated	can be eliminated

Table 6.2: Assumptions and requirements of the two proposed solution approaches. "–" indicates that the approach is indifferent to the corresponding characteristic.

The primary contributions in this chapter are: (1) formalizing the problem of mitigating NSE as a multi-objective MDP with slack; (2) presenting a solution approach to update the agent's policy by learning about NSE as a secondary reward function and estimating the minimum slack required to avoid NSE; (3) studying various types of feedback mechanisms to learn the penalty associated with NSE; (4) evaluating the performance and analyzing the bias associated with each feedback mechanism; (5) presenting environment shaping as a method to mitigate NSE and an algorithm for shaping; and (6) empirical evaluation of environment shaping on two domains.

6.2 Overview of Existing Approaches to Mitigate NSE

This section reviews the emerging approaches to mitigate the impacts of negative side effects. Table 6.3 summarizes the characteristics of side effects handled by each one of the methods described below.

A natural approach to overcome NSE caused by reward misspecification is to update the agent's model by learning the intended reward function. In [32], the specified reward is considered as a proxy for the intended reward function. By treating the proxy reward as a set of demonstrations and using approximate solutions for inference, the agent can learn the intended reward function. As acknowledged by the authors, this algorithm is not scalable to large, complex settings. Further, modifying the reward function of a deployed system entirely requires extensive evaluation before redeployment to ensure no new risks are introduced by the update. This approach is therefore better suited for handling safety-critical NSE that justify the suspension of the deployed system.

Another form of NSE arises when an agent alters features in the environment that the user does not expect or desire to be changed. This can be addressed by constraining the features that can be altered by the agent during its operation. In [122], the authors consider a setting in which the uncertainty over the desirability of altering a

	Severity	Reversibility	Avoidability	Frequency	Stochasticity	Observability	Exclusivity
[32]	-	irreversible	-	frequent	deterministic	-	-
[122]	-	irreversible	avoidable	-	deterministic	observable	non-interfering
[56]	-	-	-	-	-	observable	non-interfering
[104]	-	irreversible	-	frequent	deterministic	observable	non-interfering
[121]	-	irreversible	-	-	deterministic	observable	-
[113]	-	irreversible	avoidable	frequent	deterministic	-	non-interfering
[92]	not safety-critical	irreversible	-	frequent	deterministic	-	non-interfering
[112]	-	-	-	frequent	deterministic	-	-
[57]	not safety-critical	-	-	-	-	observable	-
[98]	-	-	-	frequent	deterministic	-	non-interfering

Table 6.3: Summary of the characteristics of the existing approaches to mitigate negative side effects. "-" indicates the approach is indifferent to the values of that property. Although some existing works do not explicitly refer to the severity of the side effects they can effectively handle, in general these approaches target side effects that are undesirable and significant, but not safety-critical.

feature is included in the agent's model. The agent first computes a policy assuming all the uncertain features are "locked" for alteration. If a policy exists, then the agent executes it. If no policy exists, the agent queries the human to determine which features can be altered and recomputes a policy. A regret minimization approach is used to select the top-k features for querying. Recently, the authors extended this approach to identify if the NSE are unavoidable by casting it as a set-cover problem [121]. If the side effects are unavoidable, the agent ceases operation. Therefore, these approaches are not suitable for settings where the agent is expected to alleviate (unavoidable) NSE to the extent possible, while completing its assigned task.

Another class of solution methods defines a penalty function for NSE as a measure of deviation from a baseline state, based on the features altered. In [56], the authors present a multi-objective formulation with scalarization, with the deviation from a baseline state measured using reachability-based metrics. The agent's sensitivity to NSE can be adjusted by tuning the scalarization parameters. This approach may penalize all side effects (even positive side effects) since it does not account for human preferences. To overcome this drawback, [104] presents an approach to infer human preferences from the initial state. They assume that an environment is typically optimized for human preferences and the agent can mitigate NSE by inferring human preferences before it starts acting.

Attainable utility [112, 113] measures the impact of side effects as the shift in the agent's ability to optimize auxiliary objectives, generalizing the relative reachability measure presented earlier [56]. All these approaches employ impact regularizers to mitigate NSE. Designing the right impact regularizer is challenging since the agent's behavior is sensitive to the choice of baseline state, the metric used to calculate the deviation, and requires knowledge about the dynamics of the environment [62]. Certain NSE may also impact the agent's ability to complete tasks in the future.

In [57], the authors present an approach that provides the agent with an auxiliary reward for preserving its ability to perform future tasks in the environment.

This chapter targets settings in which the agent is has no prior knowledge about NSE, and presents two complementary solution approaches to mitigate NSE, without entirely redesigning the model and while providing bounded-performance guarantees.

6.3 Mitigating NSE using Feedback

An agent that is unaware of the negative side effects of its actions can gather this information through feedback signals from an oracle or through autonomous exploration. This problem is formulated as a multi-objective planning problem with lexicographic ordering over the reward functions. The agent learns a secondary reward function corresponding to NSE penalty. A lexicographic approach is adopted because (1) we target settings in which optimizing the reward associated with agent's assigned task is prioritized over mitigating NSE, (2) the reward for the primary objective and the penalty for NSE may have different units, such as time taken to push a box and the cost of cleaning a rug, and (3) alternative scalarization methods require non-trivial parameter tuning and may not offer the performance guarantees with respect to the primary objective [84].

6.3.1 Background on Lexicographic MDP

This section provides a brief background on lexicographic Markov decision process (LMDP) [116], which is a multi-objective MDP with lexicographic preferences over reward functions. LMDPs are particularly useful and convenient to model multi-objective MDPs with competing objectives and with an inherent lexicographic ordering over them.

An LMDP is defined by the tuple $M = \langle S, A, T, \mathbf{R}, \mathbf{\Delta}, \boldsymbol{o} \rangle$ [116] with S denoting the finite set of states; A denoting the finite set of actions; $T : S \times A \times S \rightarrow [0, 1]$ is the

transition function indicating the probability of reaching state $s' \in S$ when executing action $a \in A$ in state $s \in S$; $\mathbf{R} = [R_1, ..., R_k]^T$ is a vector of reward functions, with $R_i : S \times A \rightarrow \mathbb{R}$ denoting the reward associated with executing action $a \in A$ in state $s \in S$ corresponding to i^{th} objective; $\mathbf{\Delta} = [\delta_1, ..., \delta_{k-1}]^T$ is a vector of *slack* variables with $\delta_i \geqslant 0$; $\boldsymbol{o} = [o_1, \ldots, o_k]^T$ denotes the k objectives with strict preference ordering over them such that $o_1 \succ o_2 \succ \ldots o_{k-1} \succ o_k$. The lexicographic preference operator $\succ$ denotes that the left term has a higher preference ordering and is always optimized prior to the right term.

The slack δ_i is an additive value denoting the acceptable deviation from the optimal expected reward for objective o_i so as to improve the lower priority objectives. The set of value functions is denoted by $\boldsymbol{V} = [V_1, ..., V_k]^T$, with V_i denoting the value function corresponding to o_i, and calculated as

$$\boldsymbol{V}^{\pi}(s) = \mathbf{R}(s, \pi(s)) + \gamma \sum_{s' \in S} T(s, \pi(s), s') \boldsymbol{V}^{\pi}(s'), \forall s \in S.$$

LMDP sequentially processes each objective in the lexicographic order and therefore the policy of the current objective o_i and the slack δ_i determine the actions available for optimizing the next objective o_{i+1}. The set of restricted actions for o_{i+1} is:

$$A_{i+1}(s) = \{a \in A | \max_{a' \in A_i} Q_i(s, a') - Q_i(s, a) \leqslant \eta_i\}$$

where $\eta_i = (1 - \gamma)\delta_i$, with a discount factor $\gamma \in [0, 1)$. LMDPs are typically solved using modified Value Iteration or LAO* [116]. We refer readers to [116] for a detailed background on LMDP.

6.3.2 Problem Formulation

In this section, the problem of mitigating the immediate NSE of agent actions is formulated using the LMDP framework. The agent's model is augmented with a secondary reward function that represents the penalty for NSE of its actions. The

efficiency of different forms of feedback to learn about NSE is investigated, including oracle feedback and agent exploration. The agent may not be able to observe the NSE except for the penalty, which is proportional to the severity of the NSE, provided by the feedback mechanism. Given $\tilde{M}$ and feedback data regarding NSE, the agent is expected to compute a policy that optimizes its assigned task, while avoiding NSE as much as possible, subject to a slack.

Definition 24. *The **augmented MDP** of a given model $\tilde{M}$ is a lexicographic MDP, denoted $M{=}\langle S, A, T, \boldsymbol{R}, o, \delta\rangle$ such that:*

- $S = \tilde{S}$ *denotes the state space;*
- $A = \tilde{A}$ *denotes the set of actions;*
- $T = \tilde{T}$ *denotes the transition function;*
- $\boldsymbol{R}{=}[\mathrm{R}_1, \mathrm{R}_2]^T$ *with* $\mathrm{R}_1{=}\tilde{R}$ *denotes the reward associated with primary objective of the agent and* $\mathrm{R}_2{=}R_N$ *denotes the reward associated with NSE of the actions;*
- $\boldsymbol{o} = [o_1, o_2]^T$ *denotes the objectives where o_1 is the primary objective denoting the agent's assigned task and o_2 is minimizing NSE with $o_1 \succ o_2$; and*
- $\delta \geqslant 0$ *is the slack denoting the maximum deviation from optimal expected reward for o_1 in order to minimize NSE.*

Since the agent cannot predict the NSE a priori, it must learn R_N using feedback mechanisms (discussed in Section 6.3.4). Our overall framework for minimizing the NSE involves the following three steps (Figure 6.3):

1. The agent collects data about NSE through various types of oracle feedback or by exploring the environment;
2. A predictive model of NSE is trained using the gathered data to generalize the agent's observations to unseen situations, represented as a reward function R_N;

3. The agent computes a policy π by solving the augmented MDP optimally with the given δ and learned R_N.

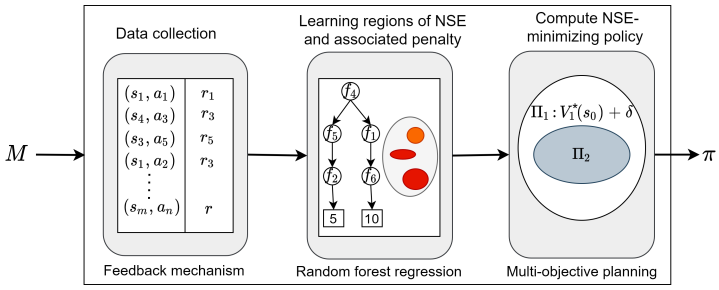
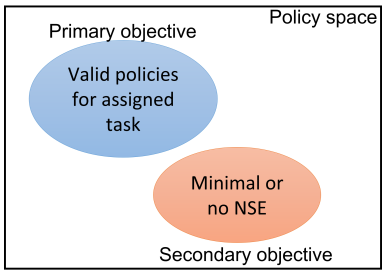


Figure 6.3: Overview of our approach for mitigating NSE using feedback.

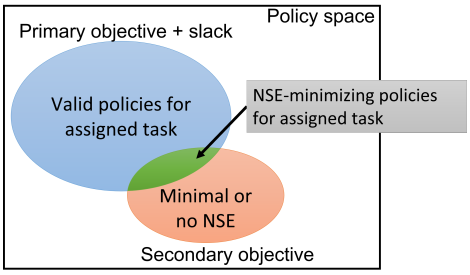
6.3.3 Slack Estimation

The slack δ denotes the maximum allowed loss in the expected reward of the agent's primary objective in order to minimize NSE. But why is slack necessary? There may be limited opportunity for avoiding NSE when optimizing the agent's assigned task, denoted by o_1. In many problems, the set of policies that are valid (optimal) for o_1 and the set of policies that avoid NSE may be distinct (Figure 6.4(a)). Allowing for a slack on the primary objective indicates willingness to tolerate a certain degree of sub-optimal behavior, with respect to o_1. This expands the set of policies that are valid for its primary objective, thereby enabling the agent to execute a NSE-minimizing policy for the assigned task (Figure 6.4(b)).

A smaller slack value limits the scope for minimizing NSE. A very high slack can allow the agent to not fulfill o_1 in an attempt to minimize the NSE. Therefore, the slack determines the overall performance of the agent with respect to both its objectives. Typically the slack value is based on user preferences and the general tolerance towards NSE. An approach to determine the minimum slack required to avoid NSE altogether, when feasible (once knowledge about the NSE is obtained), is discussed below.



(a) Disjoint policy sets



(b) Slack-induced availability of NSE-minimizing policies for assigned task

Figure 6.4: An illustration of the benefits of slack.

Algorithm 1 `Slack Estimation` $(\tilde{M}, N, E)$

1: $\delta \leftarrow \infty$
2: $\tilde{V}_1^*(s_o) \leftarrow$ Solve $\tilde{M}$ optimally with respect to o_1
3: Compute NSE-free transition $(\hat{T})$ by disabling all actions with NSE, $\forall(\tilde{s}, \tilde{a}, \tilde{s}')$:
4: $\hat{T}(\tilde{s}, \tilde{a}, \tilde{s}') \leftarrow \begin{cases} \tilde{T}(\tilde{s}, \tilde{a}, \tilde{s}'), & \text{if } N((\tilde{s}, \tilde{a}), E) = 0 \\ 0, & \text{otherwise} \end{cases}$
5: **if** solution exists for $\langle \tilde{S}, \tilde{A}, \hat{T}, \tilde{R} \rangle$ with respect to o_1 **then**
6: $\quad \hat{V}_1^*(s_o) \leftarrow$ Solve $\langle \tilde{S}, \tilde{A}, \hat{T}, \tilde{R} \rangle$ optimally for o_1
7: $\quad \delta \leftarrow |\tilde{V}_1^*(s_o) - \hat{V}_1^*(s_o)|$
8: **return** δ

Algorithm 1 determines the slack as the difference between the expected reward of the model before and after disabling all the actions that lead to NSE. The optimal expected reward for completing the task from the single start state s_0 and using $\tilde{M}$ is denoted by $\tilde{V}_1^*(s_0)$. The optimal expected reward after disabling all the actions with NSE, denoted by $\hat{V}_1^*(s_0)$, is the maximum reward that can be achieved without causing any NSE, when possible. Thus the difference in values indicates the minimum slack required to avoid NSE, when feasible, and ensures that the slack is in the same unit as the primary objective. The absolute difference is calculated to allow for negative rewards. A solution may no longer exist after the actions are disabled (Lines 3-4), in which case $\delta = \infty$ is returned, indicating that it is impossible to completely avoid NSE and still accomplish the task.

Proposition 25. *Given N, Algorithm 1 calculates the minimum slack required to avoid NSE, while still accomplishing the agent's primary objective, when NSE are avoidable.*

Proof. Proof by contradiction. Let δ denote the slack returned by Algorithm 1 and δ^* denote the minimum slack required to avoid NSE in a setting with avoidable NSE. Assume $\delta^* > \delta$. This indicates that Algorithm 1 produced a lower value of slack than required, resulting in NSE during policy execution. This is possible when the model is not solved optimally or if all the actions that lead to NSE have not been disabled. By Lines 2-6 in Algorithm 1, all unacceptable actions are disabled based on Ω and the model is solved optimally. Therefore, $\delta^* \leqslant \delta$. Let $\delta^* < \delta$. By Lines 2-6 in Algorithm 1, all actions with NSE are disabled and the model is solved optimally. Hence, $\delta^* \nleqslant \delta$. Therefore, $\delta^* = \delta$. □

Algorithm 1 breaks ties in favor of the primary objective. That is, if there are multiple policies, within the slack, that result in the same expected penalty for NSE, the algorithm will output a policy that optimizes the primary objective. Note that Algorithm 1 will fully utilize the slack value to minimize NSE, even if the improvement in NSE is minimal. For example, consider an agent operating with $\delta = 0.5\tilde{V}_1^*(s_0)$ to avoid NSE. However, if the agent operates with $\delta = 0.25\tilde{V}_1^*(s_0)$, the expected penalty for NSE is 3. For a 25% increase in the slack value, the resulting improvement in NSE is three units. In many problems, such a trade-off may be undesirable. Algorithm 1 calculates the minimum slack required to avoid the NSE, and does not optimize this trade-off in the expected rewards.

The slack is specified by the user when NSE are unavoidable or when δ estimated using Algorithm 1 is beyond the user tolerance. Picking the "right" δ for the problem can be challenging for the user if they do not have a deep understanding about the trade-offs. If the models are solved approximately, without solution guarantees,

Algorithm 1 is not guaranteed to return the minimum slack but our approach still produces a policy that minimizes NSE, given the slack.

6.3.4 Learning from Feedback

To learn about immediate NSE, two forms of feedback that correlate with features in $\tilde{S}$ are considered: feedback acquired from an oracle, and feedback the agent acquires by exploring the environment. The oracle feedback typically represents human feedback and provides signals about undesirable actions, with respect to NSE. Alternatively, the agent may explore the environment to gather reward signals with respect to NSE.

6.3.4.1 Learning from Human Feedback

Four different forms of human feedback are discussed below, each providing information about NSE directly or indirectly, along with their limitations.

Random Queries This feedback type represents an ideal setting in which the agent randomly selects an (s, a) pair for querying an oracle, given a budget and without replacement, and receives the exact, corresponding penalty for NSE.

Despite the benefits offered by this approach, it is often unrealistic to expect exact penalty specification for randomly selected (s, a) pair. Hence other feedback mechanisms where the human *approves* the agent's actions, *corrects* the agent's policy, or *demonstrates* an acceptable trajectory are also considered. Since such feedback types do not provide the exact penalty for NSE, feedback indicating acceptable actions are mapped to a zero penalty and others are mapped to a fixed, non-zero penalty denoted by k, which in turn contributes to R_N. In our experiments, k is set to the maximum penalty incurred for NSE in the problem.

Approval (HA) The agent randomly selects (s, a) pairs, without replacement, to query the human, who in turn either approves or disapproves the action in that state.

The agent learns by mapping the approved actions to zero penalty, $R_N(s,a)=0$, and the disapproved actions are mapped to a non-zero penalty $R_N(s,a)=k$. Two types of human approval are considered: *strict* (HA-S) and *lenient* (HA-L). Strict feedback disapproves all actions that result in NSE. Lenient approval only disapproves actions with severe NSE. The severity threshold for HA-L is a tunable parameter that is problem-specific. Thus with HA-L, the agent will not learn about NSE with low severity and with HA-S, the agent cannot distinguish between actions with different severities of NSE in a state. Despite the simplicity of this approach, it may be difficult for the human to approve or disapprove actions in isolation, since they do not convey the agent's overall behavior.

Corrections (C) In this form of feedback, the agent performs a trajectory of its primary policy, with human supervision. If the human observes an unacceptable action at any state, they stop the agent and specify an acceptable action to execute in that state. If all actions in a state lead to NSE, then the user specifies an action with the least NSE. The agent proceeds until the goal is reached or until interrupted again. When interrupted, the agent assumes that all actions, except the correction, are unacceptable in that state. If not interrupted, actions are considered to be acceptable. Acceptable actions are mapped to zero penalty, $R_N(s,a)=0$, and unacceptable actions are mapped to a non-zero penalty, $R_N(s,a)=k$. This approach requires constant human oversight, which is practically infeasible in many settings.

Demo-action mismatch (D-AM) In demo-action mismatch, the human provides limited number of demonstrations of how to perform the task safely. Each demonstration is a complete trajectory of performing the task, from start to the goal. The agent collects these trajectories and compares them with its primary policy. For all states in which there is an action mismatch, the agent assumes its policy leads to NSE and assigns $R_N(s,a)=k$, otherwise assigns $R_N(s,a)=0$. The agent does not mimic

human behavior. Instead, it only uses the demonstrations to gather knowledge about NSE. A limitation of this approach is that the agent does not receive any information about NSE in states unvisited by the human.

6.3.4.2 Learning from Exploration

When human feedback is expensive to collect or unavailable, it may be easier to allow the agent to identify susceptible states through limited exploration. Learning from exploration uses ϵ-greedy action selection. That is, the agent exploits the action prescribed by its primary policy or explores a random action to learn about NSE. The agent executes an action and observes the corresponding NSE penalty, $R_N(s, a)$. Note that the agent explores only to learn R_N and the final policy is computed by solving the augmented model.

Three exploration strategies are considered: *conservative*–where the agent explores an action with probability 0.1 or follows its primary policy, *moderate*–where the agent either explores an action with probability 0.5 or follows its primary policy, and *radical*–where the agent predominantly explores with probability 0.9, allowing the agent to possibly identify more NSE than the other exploration strategies. Learning by exploring the real-world may be risky in some contexts, such as autonomous vehicles. In such settings, exploration may be performed in a simulator that is designed to train the agent to avoid the negative side effects, or the agent can learn from human feedback.

6.3.4.3 Model Learning

The agent's observations are generalized to unseen situations by training a random forest regression (RF) model to predict the penalty R_N. The RF model is used to handle the continuous nature of the penalty and any regression technique may be used in practice. The hyperparameters for the training are determined by a randomized search in the space of RF parameters. For each hyperparameter setting, a three-fold

cross validation is performed and the mean squared error is calculated. Parameters with the least mean squared error are selected for training, which is then used to predict the penalty R_N. The augmented MDP is then updated with this learned R_N and the agent computes an NSE-minimizing policy for execution by solving the augmented MDP.

6.3.5 Experimental Setup

This section discusses the experimental setup to evaluate the different feedback mechanisms for mitigating avoidable and unavoidable NSE.

Baselines The performance of our approach is compared with three baselines. First is the simulated *Oracle* agent that avoids NSE while satisfying the primary objective. The *Oracle* has a perfect NSE model and its policy is computed by solving the model after avoiding all the actions with NSE. In problems with unavoidable NSE, it selects the action with the least NSE since that is the best possible performance that can be achieved while satisfying the primary objective. The performance of the *Oracle* provides a lower bound on the penalty for NSE incurred by the agent. The second is the *No queries* case in which the agent does not query or learn about NSE. Instead, it naively executes its primary policy. This provides an upper bound on the penalty for NSE incurred by the agent. Third is the relative reachability-based approach (RR), which is a multi-objective approach with scalarization [56]. This approach optimizes $r(s_t, a_t) - \beta d(s_t, b_t)$, where $r(s_t, a_t)$ is the reward corresponding to o_1 and $d(s_t, b_t)$ is the measure of deviation from the baseline state b_t, denoting the NSE. A direct comparison with this approach is not feasible since it is based on assumptions that do not hold in our setting. Therefore, the RR approach is modified to make it applicable in our setting, by calculating the deviation based on a model of NSE learned with Random Query approach as it does not introduce any bias. The deviation is computed from inaction baseline, which measures the NSE of the agent's action with respect

to no action execution in that state [56]. The side effects considered are irreversible by an agent, once occurred, making the baseline state unreachable. We tested with $\beta \in [0.1, 0.9]$ since o_1 is prioritized in our formulation and report results with $\beta = 0.8$ as it achieved the best trade-off in training.

Side Effects In the interest of clarity, two types of NSE severity are considered: mild and severe. Each action can either result in a mild NSE, severe NSE, or no NSE. The strict human approval (HA-S) feedback disapproves all actions that result in NSE. The lenient human approval (HA-L) only disapproves actions with severe NSE. For learning NSE by exploration, the agent is assumed to learn by exploring in a simulator where the reward signals indicate NSE penalty.

Our experiments optimize costs, which are negations of the rewards. Random forest regression from `sklearn` Python package is used for model learning. The augmented MDP is solved using a lexicographic variant of LAO* [35]. The slack is computed using Algorithm 1 and $\gamma = 0.95$. Values averaged over 100 trials of planning and execution, along with their standard errors, are reported for the following domains.

Boxpushing A modified boxpushing domain [103] is considered, in which the agent is expected to minimize the expected time taken to push a box to the goal (Figure 6.1). Each action takes +1 unit of time. Each state is represented as $\langle x, y, b, c \rangle$ where x, y denote the agent's location, b indicates if the agent is pushing the box, c indicates the current cell's surface type. Pushing the box on a surface type $c = 1$ results in severe NSE with a penalty of 10, pushing the box on a surface $c = 2$ results in mild NSE and a penalty of 5, and no NSE otherwise. The state features used for training are $\langle b, c \rangle$. Experiments are conducted on five instances with grid size 15×15 and with varying initial location of the box and NSE regions.

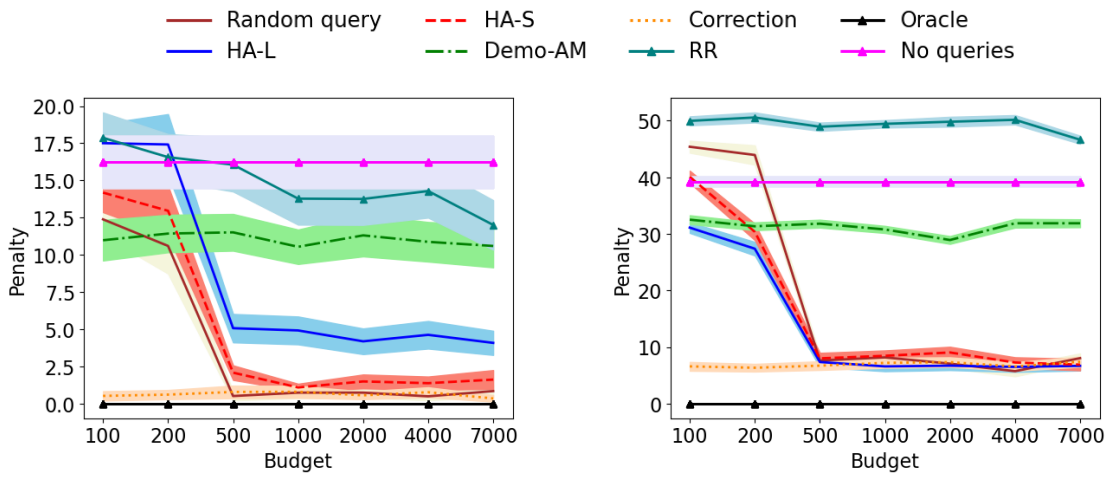
Driving Our second domain is based on simulated autonomous driving [91, 116] in which the agent's primary objective is to minimize the expected cost of navigation from start to a goal, during which it may encounter some puddles. The agent can move in all four directions at low and high speeds. The cost of navigating at a low speed is +2 and that of high speed is +1. When the agent navigates over a puddle at high speed, it spatters water which is undesirable. Some puddles may have pedestrians in the vicinity and splashing water on them results in severe NSE with a penalty of 10 and a mild NSE otherwise with a penalty of 5. Each state is represented by $\langle x, y, l, p, h \rangle$ where x, y denote the agent's location, l is the speed, p, h indicate the presence of a puddle and a pedestrian in the vicinity. Features used for training are $\langle l, p, h \rangle$. Five test instances are generated with grid size 15×15 and by varying the start, goal, puddle, and pedestrian locations.

6.3.6 Results and Discussion

This section discusses the effectiveness of learning from feedback, and the effect of slack values on the performance.

Effectiveness of learning from feedback The effectiveness of various feedback approaches in learning about NSE is first discussed in settings with avoidable NSE, followed by results on problems with unavoidable NSE.

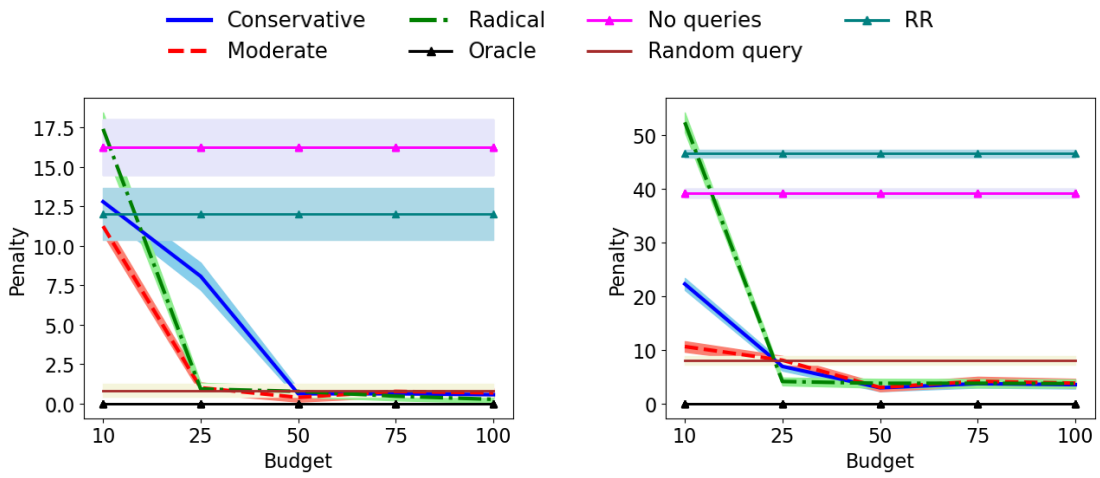
Figures 6.5 and 6.6 show the average penalty incurred for NSE in settings with avoidable NSE, as the feedback budget is increased. The Corrections feedback is efficient in terms of samples required since the human corrects the agent by prescribing an acceptable action in each state. Random querying and HA-S, which rely on random samples of states, achieve significant reduction in NSE with 500 samples. Although HA-L also relies on random sampling of states, it does not provide information about mild NSE, which affects its performance. Training with Demo-AM feedback does not always minimize NSE even as the budget is increased, since the agent does not



(a) Boxpushing (b) Driving

Figure 6.5: Effect of learning from human feedback methods on problems with avoidable NSE.

receive sufficient negative samples and has no information about the NSE in states unvisited in the demonstrations. Additionally, it is unable to distinguish between the different levels of severity of the NSE of its actions. The performance with Demo-AM is also affected when the acceptable actions demonstrated by the oracle violate the local slack constraints for the agent and therefore, the agent may not be able to minimize NSE. However, Demo-AM is still better than *No queries*. *RR* approach, with the model of NSE learned using Random Query, performs poorly irrespective of the budget. Apart from the reported results, *RR* with a perfect model of NSE was also tested. With a perfect model of NSE, *RR* performance was significantly better and sometimes comparable to *Oracle*'s performance. However, obtaining a perfect model of NSE is non-trivial in practice. In Figure 6.6, Random querying with the maximum budget (7000) and *RR* with this learned model are plotted in to compare the performance of exploration strategies and to understand how the correlated samples affect the performance. Irrespective of the exploration type, the agent gathers knowledge about NSE with a reasonable number of trials.

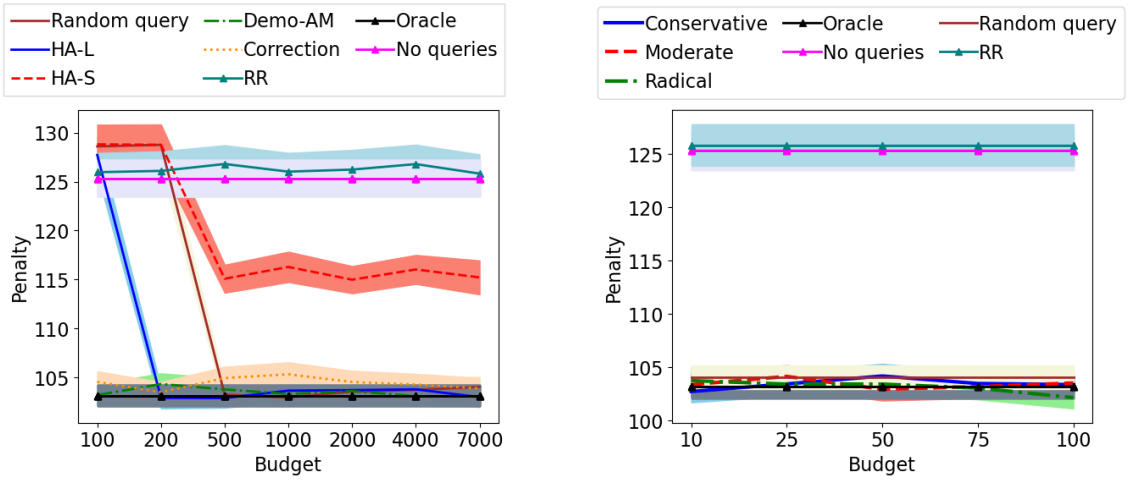


(a) Boxpushing (b) Driving

Figure 6.6: Effect of learning with exploration strategies on problems with avoidable NSE.

When NSE are unavoidable, Algorithm 1 does not produce a finite slack. Therefore, 15% of the expected cost of o_1 is used as the slack value for experiments. This is based on the observation that the slack values returned by the algorithm for problems with avoidable NSE are typically within 15% of the expected cost for o_1. Figure 6.7 plots the results for boxpushing problems with unavoidable NSE. Problem instances with unavoidable NSE case were generated by making sure there is no path to the goal over surface $c = 3$. Results on the driving domain are not reported because there are no unavoidable NSE in the problem setting we consider—the agent can avoid NSE by navigating at a low speed.

The performance of human feedback techniques are similar to that of avoidable NSE setting, except for HA-S and Demo-AM. Since HA-S cannot distinguish between different severities, its performance is affected when NSE are unavoidable. Demo-AM matches the performance of other techniques when NSE are unavoidable. Similar to Figure 6.6, learning by exploration in settings with unavoidable NSE matches the



(a) Learning from human feedback

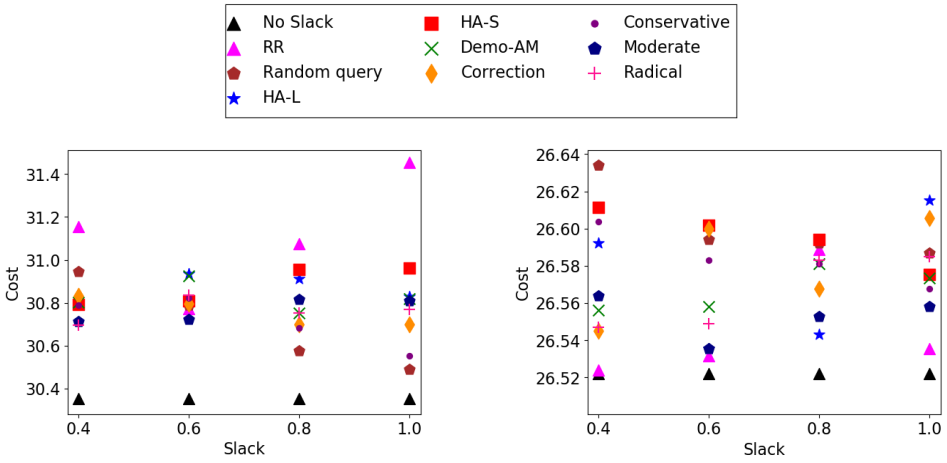
(b) Learning with exploration

Figure 6.7: Performance on the boxpushing problems with unavoidable NSE.

performance of Random query. Overall, the results indicate that our framework can effectively learn and mitigate the impacts of both avoidable and unavoidable NSE.

Effect of slack The effect of slack is studied on the expected cost of o_1 and on NSE (o_2), by varying the slack from 40% to 100% of the value returned by Algorithm 1 on problems with avoidable NSE. Figures 6.8 and 6.9 show the results with 7000 queries for human feedback and 100 exploration trials.

The values show the average cost in 100 trials of executing the updated policy with each slack value. The baseline approaches which are unaffected by the variation in slack are not reported. Results with no slack bound the performance of other techniques. As the slack is increased, the average penalty incurred for NSE tends to decrease. The minimal differences in the average costs for o_1 shows that the slack values returned by Algorithm 1 are reasonable and affect o_1 only by a small margin. The performance of some feedback approaches are unaffected by the variation in slack, which shows their limitations in minimizing NSE.



(a) Boxpushing (b) Driving

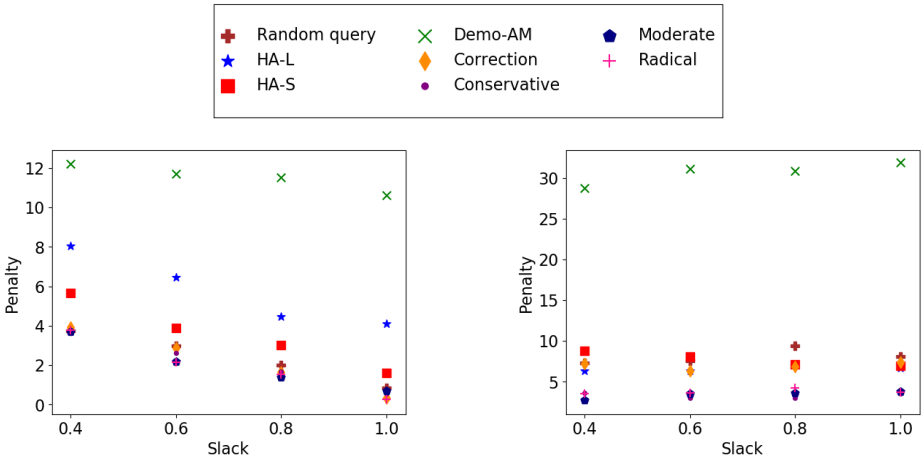
Figure 6.8: Effect of slack on o_1.

6.4 Limitations of Learning from Feedback

While our results so far show that learning from feedback can successfully mitigate NSE, four key factors affect agent learning and its ability to mitigate NSE: (1) challenges in collecting accurate feedback; (2) sampling biases in feedback; (3) limited fidelity of agent state representation; and (4) unavoidability of certain NSE.

6.4.1 Challenges in Accurate Feedback Collection

This section discusses the challenges and drawbacks of collecting feedback using the approaches presented in Section 6.3.4.1. In random query feedback, the human is expected to provide exact penalty for NSE, which is non-trivial and can result in misspecification in practice. While the approval (HA) approach overcomes this drawback, the agent cannot differentiate between NSE with different severities, and it is often infeasible for the human to provide feedback on randomly selected state-action pairs. The corrections approach overcomes these limitations, but it requires constant human oversight, similar to action approval or random query approach. The Demo-AM approach alleviates the need for constant human supervision and it is relatively easier for the human to provide demonstrations than correcting or supervising



(a) Boxpushing (b) Driving

Figure 6.9: Effect of slack on NSE.

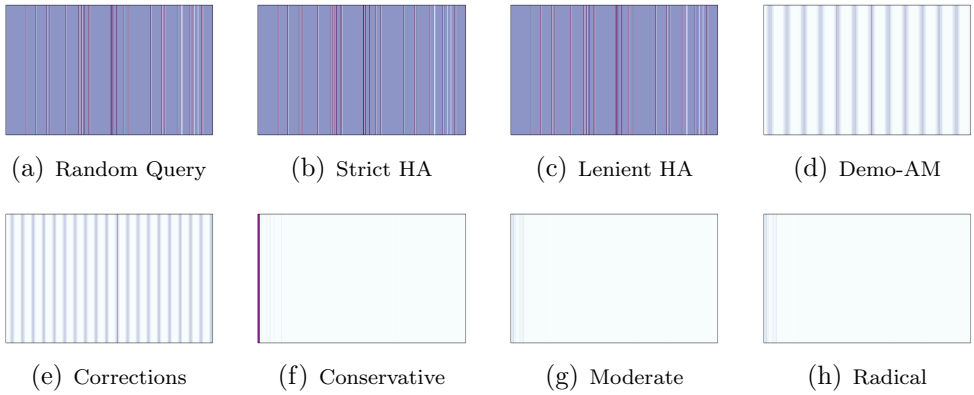
the agent. However, the agent does not receive information about NSE in the unvisited states, and cannot differentiate between NSE with different severities. The autonomous exploration approach avoids the need for human supervision altogether, but it still requires the designer to specify the penalty for agent training in the simulator and may lead to misspecification. Further, autonomous exploration may be unsafe in certain settings.

These approaches, except random query and approval, also introduce sampling bias since the samples are not *i.i.d.*, as the visited states are correlated. Furthermore, all actions other than the corrections or those demonstrated are assumed to be unacceptable. Since there may be multiple acceptable actions in each state, this introduces additional bias.

6.4.2 Bias in Various Feedback Mechanisms

The sampling biases in a feedback approach affects agent performance. Different types of agent behaviors emerge when learning from different forms of feedback, as shown in Figures 6.5 and 6.6, due to the sampling biases. Figure 6.10 plots the frequency of querying in different regions of the state space with different feedback

mechanisms. The x-axis denotes the state space and darker shades indicate regions that were frequently queried. The results are plotted for driving domain with avoidable NSE and using 7000 queries for human feedback and 100 trials for exploration.



(a) Random Query (b) Strict HA (c) Lenient HA (d) Demo-AM

(e) Corrections (f) Conservative (g) Moderate (h) Radical

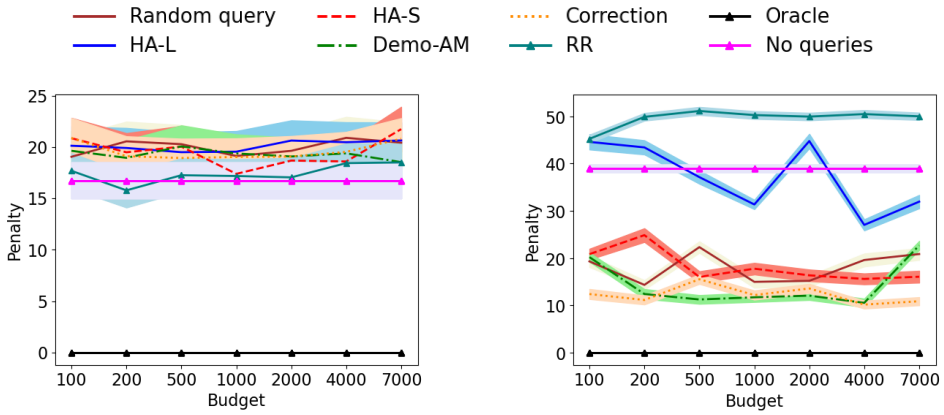
Figure 6.10: Frequency of querying across the state space with different feedback approaches.

Feedback techniques that are based on random sampling of states have a higher coverage of the state space, contributing to a better performance. The exploration techniques are the most restricted, due to which their performances are largely similar. Since an ϵ-greedy approach is followed for exploration, these techniques likely cover only the region surrounding that of the primary policy. As states in this region are often critical for satisfying the agent's primary objective, learning about NSE in this restricted region is often sufficient to effectively mitigate the penalty for NSE. These results show that different feedback types focus on different regions of the state space, which in turn affects the NSE model learning.

6.4.3 Fidelity of State Representation

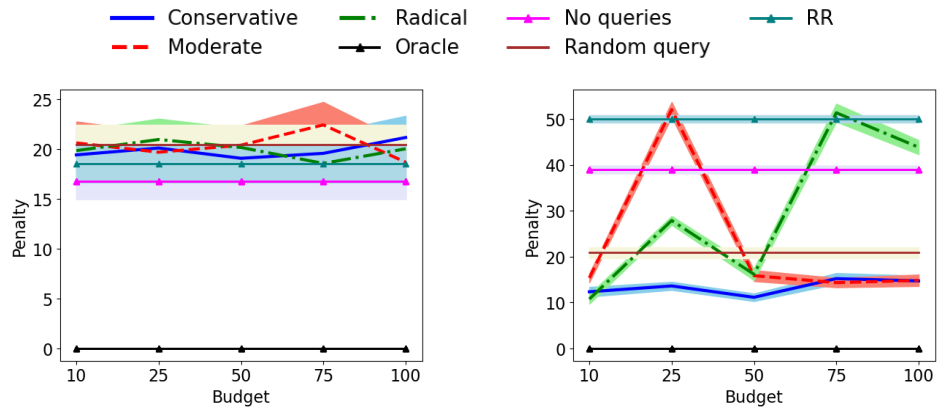
Learning from feedback requires the agent state representation to have all the necessary features in order to effectively learn about NSE. However, in many scenarios, the agent's state representation may include only the features related to the agent's task. Limited fidelity of the state representation may affect the agent's learning

and adaptation, since the NSE could potentially be non-Markovian and the feedback signals received for a state-action pair may be conflicting.



(a) Boxpushing (b) Driving

Figure 6.11: Effect of learning from human feedback when the agent state representation has low fidelity.



(a) Boxpushing (b) Driving

Figure 6.12: Effect of learning with autonomous exploration when the agent state representation has low fidelity.

Figures 6.11 and 6.12 plot the effect of learning from various feedback approaches when the agent's state representation has limited fidelity. For the boxpushing domain, the agent's state representation does not consider the surface type in the representation, since it is unrelated to its task. The state features used for training in this

case are $\langle b \rangle$, where b indicates if the agent is currently pushing the box. However, the occurrence of NSE correlates with surface type. Hence, the agent's learning and its ability to mitigate NSE are affected. Similarly for the driving domain, we do not include the presence of puddles and nearby pedestrians in the state representation, since they are not relevant to o_1. The state features used for training in this case are $\langle l \rangle$, where l denotes the current speed. Since the presence of puddles and nearby pedestrians are important features for learning a model of NSE, the agent's performance is affected when the state representation does not include these features.

6.4.4 Unavoidable NSE and Non-Immediate NSE

Updating the agent's policy may not considerably mitigate the impacts of NSE when they are unavoidable, with respect to the agent's task, as shown in Figure 6.7. For example, in the boxpushing domain, NSE are unavoidable if the entire floor is covered with a rug. Alternate approaches that do not rely on agent learning and adaptation are required to avoid NSE in such settings. It is also non-trivial to extend the proposed learning from feedback approach to mitigate NSE associated with agent trajectories that are not decomposable into additive penalties associated with state-action pairs. For example, R_N does not represent the penalties associated with a sequence of actions. The following section presents an approach to avoid NSE in such settings.

6.5 Mitigating NSE via Environment Shaping

This section presents another approach for mitigating NSE when agents operate in environments that are configurable. In many settings, the human operating the system has a broader scope of knowledge and more control over the environment in which the agent is situated. These insights can be leveraged to mitigate the impacts of NSE when the agent has no prior knowledge about the side effects of its actions. This

problem is formulated as a human-agent team with decoupled objectives. The agent optimizes its assigned task, during which its actions may produce NSE. The human shapes the environment through minor reconfiguration actions so as to mitigate the impacts of agent's side effects, without affecting the agent's ability to complete its assigned task. This decoupled approach does not make any assumptions regarding the agent's model and its learning capabilities. Further, by controlling the space of available modifications, the user can prioritize which side effects should be avoided.

6.5.1 Problem Formulation

The formulation consists of an actor agent and an environment designer agent. The actor agent operates based on an MDP $\tilde{M}$ in an environment that is configurable and described by a configuration file E, such as a map of the environment. Describing the environment using configuration files, such as 2D maps, is a common practice in robotics. A factored state representation is assumed. The actor optimizes the reward for its assigned task, which is its primary objective o_P, and its model includes the necessary details relevant to o_P. Executing the policy π computed using $\tilde{M}$ may lead to NSE, unknown to the actor. The environment designer measures the impact of NSE associated with the actor's π and shapes the environment, if necessary. While the environment designer can be any autonomous agent, the formulation is inspired by having a human in the role. The actor and the environment designer share the configuration file of the environment, which is updated by the environment designer to reflect the modifications. Optimizing o_P is prioritized over avoiding NSE. Hence, shaping is performed *in response* to the actor's policy. In the rest of this chapter, the environment designer is simply referred to as designer—not to be confused with the designer of the agent itself. The terms 'actor' and 'agent' are used interchangeably to denote the actor agent.

Each modification is a sequence of design actions. An example of a modification is $\{move(table, l_1, l_2), remove(rug)\}$, which moves the table from location l_1 to l_2 and removes the rug in the current setting, resulting in a new environment configuration. The set of modifications is assumed to be finite since the environment is generally optimized for the human and the agent's primary task [104], and the human may not be willing to drastically modify it. Additionally, the set of modifications for an environment is included in the problem specification since it is typically controlled by the user and is rooted in the NSE they want to mitigate.

The following assumptions are made about the nature of modifications: (1) the start and goal conditions of the actor are fixed and cannot be altered, so that the modifications do not alter the agent's task or its capabilities; and (2) modifications are applied tentatively for evaluation purposes and the environment is reset if the reconfiguration affects the actor's ability to complete its task or the actor's policy in the modified setting does not minimize the side effects.

Definition 26. *An actor-designer framework to mitigate negative side effects (***AD-NSE***) is defined by* $\langle E_0, \mathcal{E}, \tilde{M}, M_d, \delta_A, \delta_D \rangle$ *with:*

- E_0 *denoting the initial environment configuration of the actor;*
- $\mathcal{E}$ *denoting a finite set of possible reconfigurations of* E_0*;*
- $\tilde{M}$ *is the actor's MDP;*
- $M_d = \langle \Omega, \Psi, C, R_N \rangle$ *is the model of the designer with*
  - Ω *denoting a finite set of valid modifications that are available for* E_0*, including* $\varnothing$ *to indicate that no changes are made;*
  - $\Psi : \mathcal{E} \times \Omega \to \mathcal{E}$ *determines the resulting environment configuration after applying a modification* $\omega \in \Omega$ *to the current configuration,* $E \in \mathcal{E}$ *and is denoted by* $\Psi(E, \omega)$*;*

- $C : \mathcal{E} \times \Omega \to \mathbb{R}$ *is a cost function that specifies the cost of applying a modification to an environment, denoted by* $C(E, \omega)$*, with* $C(E, \varnothing) = 0, \forall E \in \mathcal{E}$*; and*
- R_N *is a model specifying the penalty for negative side effects in environment* $E \in \mathcal{E}$ *for the actor's policy* π*.*

- $\delta_A \geqslant 0$ *is the actor's slack, indicating the maximum allowed deviation from the initial optimal policy for* o_P *when recomputing its policy in a modified environment; and*

- $\delta_D \geqslant 0$ *indicates the designer's tolerance threshold for NSE.*

Decoupled Objectives The actor agent's objective is to compute a policy π that maximizes its expected reward from start state $\tilde{s}_0$ in the current environment configuration E, with respect to o_P:

$$\max_{\pi \in \Pi} V_P^{\pi}(\tilde{s}_0|E),$$
$$V_P^{\pi}(\tilde{s}) = R(\tilde{s}, \pi(\tilde{s})) + \sum_{\tilde{s}' \in \tilde{S}} T(\tilde{s}, \pi(\tilde{s}), \tilde{s}') V_P^{\pi}(\tilde{s}'), \forall \tilde{s} \in \tilde{S}.$$

When the environment is modified, the actor agent recomputes its policy and may end up with a longer path to its goal. The slack δ_A denotes the maximum allowed deviation from the optimal expected reward in E_0, $V_P^*(\tilde{s}_0|E_0)$, to facilitate minimizing the NSE via shaping. A policy π' in a modified environment E' satisfies the slack δ_A if the following condition holds:

$$V_P^*(\tilde{s}_0|E_0) - V_P^{\pi'}(\tilde{s}_0|E') \leqslant \delta_A,$$

where $V_P^*(\tilde{s}_0|E_0)$ denotes the optimal expected value of o_P in the original, unmodified environment E_0 and $V_P^{\pi'}(\tilde{s}_0|E')$ denotes the expected value in the modified environment E' with policy π'.

The designer first estimates the penalty for NSE associated with agent behavior, denoted by $R_N(\pi, E)$. If the NSE exceeds the designer's tolerance threshold, δ_D, then the environment is reconfigured, assuming agent's policy π is fixed. Given π and a set of modifications Ω, the designer selects a modification that maximizes the utility:

$$\max_{\omega \in \Omega} U_\pi(\omega)$$
$$U_\pi(\omega) = \underbrace{\big(R_N(\pi, E) - R_N(\pi, \Psi(E, \omega))\big)}_{\text{reduction in NSE penalty}} - C(E, \omega). \tag{6.1}$$

Typically the environment designer is the user of the system and their preferences and tolerance for NSE is captured by the utility of a modification. The designer's objective is to balance the trade-off between minimizing the NSE and the cost of applying the modification. It is assumed that the cost of the modification is measured in the same units as the NSE penalty. The cost of a modification may be amortized over episodes of the actor performing the task in the environment.

The following properties are used to guarantee bounded-performance of the actor when shaping the environment to minimize NSE.

Definition 27. *Shaping is **admissible** if it results in an environment configuration where (1) the actor can complete its assigned task, given δ_A, and (2) the NSE does not increase, relative to E_0.*

Definition 28. *Shaping is **proper** if (1) it is admissible, and (2) reduces the actor's NSE to be within δ_D.*

Definition 29. *Shaping is **robust** if it results in an environment configuration E where* all *valid policies of the actor for o_P, given δ_A, produce NSE within δ_D. That is, $R_N(\pi, E) \leqslant \delta_D$, $\forall \pi : V_P^*(\tilde{s}_0|E_0) - V_P^\pi(\tilde{s}_0|E) \leqslant \delta_A$.*

Shared Knowledge The actor and the designer do not have details about each other's model. Since the objectives are decoupled, knowledge about the exact model

parameters of the other agent is not required. Instead, the two key details that are necessary to achieve collaboration are shared: (1) configuration file describing the environment and (2) the actor's policy. The shared configuration file, such as the map of the environment, allows the designer to effectively communicate the modifications to the actor. The actor's policy is required for the designer to shape the environment. However, compact representation of the actor's policy is a practical challenge in large problems [31]. Therefore, instead of sharing the complete policy, the actor provides a finite number of demonstrations $\mathcal{D} = \{\tau_1, \tau_2, \ldots, \tau_n\}$ of its optimal policy for o_P in the current environment configuration. Each demonstration τ is a trajectory from start to goal, following π. Deterministic policies for the actor are considered.

Using $\mathcal{D}$, the designer can extract the actor's policy and measure its NSE. Policy is extracted by associating the observed actions with states in the demonstrations. The designer is aware of the general capabilities of the agent and its objectives as they are assigning the task, which makes it easier for them to construe the agent's trajectories. Naturally, increasing the number and diversity of sample trajectories that cover the actor's reachable states helps the designer improve the accuracy of estimating the actor's NSE and select an appropriate modification. If $\mathcal{D}$ does not starve any reachable state, following π, then the designer can extract the actor's complete policy.

6.5.2 Algorithm for Environment Shaping

Our solution approach for solving AD-NSE, described in Algorithm 2, proceeds in two phases: a planning phase and a shaping phase. In the planning phase (Line 4), the actor computes its policy π for o_P in the current environment configuration and generates a finite number of sample trajectories $\mathcal{D}$, following π. The planning phase ends with disclosing $\mathcal{D}$ to the designer. The shaping phase (Lines 7-14) begins with the designer associating states with actions observed in $\mathcal{D}$ to extract a policy $\hat{\pi}$, and

estimating the corresponding NSE penalty, denoted by $R_N(\hat{\pi}, E)$. If $R_N(\hat{\pi}, E) > \delta_D$, the designer applies a utility-maximizing modification and updates the configuration file. The planning and shaping phases alternate until the NSE is within δ_D or until all possible modifications have been tested.

Algorithm 2 Environment shaping to mitigate NSE

Require: $\langle E_0, \mathcal{E}, \tilde{M}, M_d, \delta_A, \delta_D \rangle$: AD-NSE
Require: d: Number of sample trajectories
Require: b: Budget for evaluating modifications
$E^* \leftarrow E_0$ ▷ Initialize best configuration
$E \leftarrow E_0$
$n \leftarrow \infty$
$\mathcal{D} \leftarrow$ Solve $\tilde{M}$ for E_0 and sample d trajectories
$\bar{\Omega} \leftarrow$ Diverse_modifications(b, Ω, M_d, E_0)
while $|\bar{\Omega}| > 0$ **do**
    $\hat{\pi} \leftarrow$ Extract policy from $\mathcal{D}$
    if $R_N(\hat{\pi}, E) < n$ **then**
        $n \leftarrow R_N(\hat{\pi}, E)$
        $E^* \leftarrow E$
    if $R_N(\hat{\pi}, E) \leqslant \delta_D$ **then** break
    $\omega^* \leftarrow \text{argmax}_{\omega \in \bar{\Omega}} U_{\hat{\pi}}(\omega)$
    if $\omega^* = \varnothing$ **then** break
    $\bar{\Omega} \leftarrow \bar{\Omega} \backslash \omega^*$
    $\mathcal{D}' \leftarrow$ Solve $\tilde{M}$ for $\Psi(E_0, \omega^*)$ and sample d trajectories
    if $\mathcal{D}' \neq \{\}$ **then**
        $\mathcal{D} \leftarrow \mathcal{D}'$
        $E \leftarrow \Psi(E_0, \omega^*)$
return E^*

Shaping phase

The actor returns $\mathcal{D} = \{\}$ when the modification affects its ability to reach the goal, given δ_A. Modifications are applied tentatively for evaluation and the environment is reset if the actor returns $\mathcal{D} = \{\}$ or if the reconfiguration does not minimize the NSE. Therefore, all modifications are applied to E_0 and it suffices to test each ω without replacement as the actor always calculates the corresponding optimal policy. When $\tilde{M}$ is solved approximately, without bounded-guarantees, it is non-trivial to verify if δ_A is violated but the designer selects a utility-maximizing ω corresponding

to this policy. Algorithm 2 terminates when at least one of the following conditions is satisfied: (1) the NSE is within δ_D; (2) all $\omega \in \Omega$ have been tested; or (3) the utility-maximizing option does not make any modification at all, $\omega^* = \varnothing$. The first condition is straightforward. The second criterion describes the worst case scenario when no modification reduces the NSE to be within δ_D. In such cases, our algorithm identifies the ω that results in an environment configuration with the least NSE possible. When $\omega^* = \varnothing$, it indicates that cost of modification exceeds the reduction in NSE or no modification can reduce the NSE further.

Though the designer calculates the utility for all modifications, using R_N, there is an implicit pruning in the shaping phase since only utility-maximizing modifications are evaluated (Line 15). However, when multiple modifications have the same cost and produce similar environment configurations, the algorithm will alternate between planning and shaping multiple times to evaluate all these modifications, which is $|\Omega|$ in the worst case. To minimize the number of evaluations in settings with large Ω, consisting of multiple similar modifications, a greedy approach to identify and evaluate diverse modifications is presented below.

6.5.2.1 Selecting diverse modifications

Let $b > 0$ denote the maximum number of modifications the designer is willing to evaluate. When the budget $b < |\Omega|$, it is beneficial to evaluate b diverse modifications. Algorithm 3 presents a greedy approach to select b diverse modifications. If two modifications result in similar environment configurations, the algorithm prunes the modification with the higher cost. The similarity threshold is controlled by ϵ. This process is repeated until b modifications are identified. Measures such as the Jaccard distance or embeddings may be used to estimate the similarity between two configurations.

Algorithm 3 Diverse_modifications(b, Ω, M_d, E_0)

1: $\bar{\Omega} \leftarrow \Omega$
2: **if** $b \geqslant |\Omega|$ **then return** Ω
3: **for all** $\omega_1, \omega_2 \in \bar{\Omega}$ **do**
4: **if** similarity$(\omega_1, \omega_2) \leqslant \epsilon$ **then**
5: $\bar{\Omega} = \bar{\Omega} \setminus \text{argmax}_{\omega_1,\omega_2}(C(E_0, \omega_1), C(E_0, \omega_2))$
6: **if** $|\bar{\Omega}| = b$ **then return** $\bar{\Omega}$

Remark 30. *Shaping with Algorithm 2 is guaranteed to be admissible but may not be proper.*

Algorithm 2 ensures that a modification does not negatively impact the actor's ability to complete its task, given δ_A (Lines 16-19), and stores the configuration with least NSE (Line 10). Therefore, Algorithm 2 is guaranteed to return an E^* with NSE equal to or lower than that of the initial configuration E_0, thereby guaranteeing admissible shaping.

Shaping with Algorithm 2 is not guaranteed to be proper because (1) no modifications in Ω may be able to reduce the NSE penalty below δ_D; or (2) the cost of such a modification may exceed the corresponding reduction in NSE.

6.5.3 Theoretical Properties

This section discusses the conditions under which the actor's policy is unaffected by shaping, and how the proposed approach for shaping can be viewed as an extensive form game.

6.5.3.1 Policy-Preserving Modification

With each reconfiguration, the actor is required to recompute its policy. This section describes a class of problems for which the actor's policy is unaffected by shaping.

Definition 31. *A modification is* ***policy-preserving*** *if the actor's initial policy is unaltered by environment shaping.*

This property induces policy invariance before and after shaping and hence the policy is backward-compatible. Additionally, the actor is guaranteed to complete its task with $\delta_A = 0$. Figure 6.13 illustrates the dynamic Bayesian network for this class of problems, where r_P denotes the reward associated with o_P and r_N denotes the penalty for NSE.

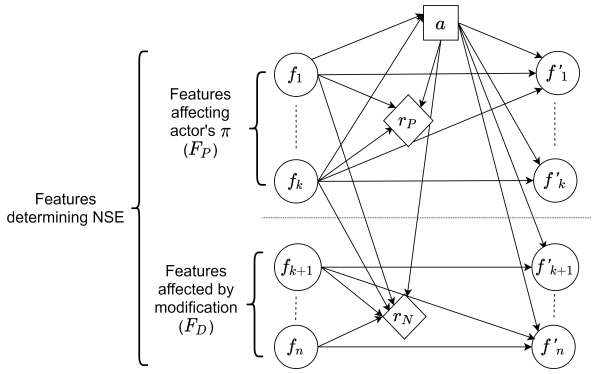


Figure 6.13: A dynamic Bayesian network description of a policy-preserving modification.

Let F denote the set of features in the environment, with $F_P \subset F$ affecting the actor's policy and $F_D \subset F$ altered by the designer's modifications. Let $\vec{f}$ and $\vec{f'}$ be the feature values before and after environment shaping. Given the actor's π, a policy-preserving ω follows $F_D = F \backslash F_P$, ensuring $\vec{f_P} = \vec{f'_P}$. When $\vec{f} = \vec{f_P} \cup \vec{f_D}$, a policy-preserving ω mitigates NSE by enforcing $\Pr(F_D = \vec{f_D} | \vec{f_P}, \pi) = 0, \forall \vec{f}$. For example in the boxpushing domain, regions in the agent's shortest path to its goal cannot be covered by a rug.

Proposition 32. *Given an environment configuration E and actor's policy π, a modification $\omega \in \Omega$ is guaranteed to be* ***policy-preserving*** *if $F_D = F \backslash F_P$.*

Proof. Proof by contradiction. Let $\omega \in \Omega$ be a modification consistent with $F_D \cap F_P = \varnothing$ and $F = F_P \cup F_D$. Let π and π' denote the actor's policy before and after shaping the environment with ω such that $\pi \neq \pi'$. Since the actor always computes an optimal policy for o_P (assuming fixed strategy for tie-breaking), a difference in the

policy indicates that at least one feature in F_P is affected by ω, $\vec{f_P} \neq \vec{f'_P}$. This is a contradiction since $F_D \cap F_P = \varnothing$. Therefore $\vec{f_P} = \vec{f'_P}$ and $\pi = \pi'$. Thus, $\omega \in \Omega$ is policy-preserving when $F_D = F \backslash F_P$. □

Corollary 33. *A policy-preserving modification identified by Algorithm 2 guarantees* admissible *shaping.*

Furthermore, a policy-preserving modification results in robust shaping when $\Pr(F_D = \vec{f_D}) = 0$, $\forall \vec{f} = \vec{f_P} \cup \vec{f_D}$ with NSE greater than δ_D. The policy-preserving property is sensitive to the actor's state representation. However, many real-world problems exhibit this feature independence. In the boxpushing example (Figure 6.1), the actor's state representation may not include details about the rug since it is not relevant to the task. Moving the rug to a different part of the room that is not on the actor's path to the goal is an example of a policy-preserving and admissible shaping. Modifications such as removing the rug or covering it with a protective sheet are policy-preserving and robust shaping since there is no direct contact between the box and the rug, for all policies of the actor.

6.5.3.2 Shaping as an Extensive Form Game

Our solution approach with decoupled objectives can be viewed as a game between the actor and the designer, which is useful to understand the link between two often distinct fields of decision theory and game theory. This also opens the possibility for future work on game-theoretic approaches for mitigating NSE.

The action profile or the set of strategies for the actor is the policy space Π, with respect to o_P and $\mathcal{E}$, with payoffs defined by its reward function R. The action profiles for the designer is the set of all modifications Ω with payoffs defined by its utility function U. In each round of the game, the designer selects a modification that is a best response to the actor's policy π:

$$b_D(\pi) = \{\omega \in \Omega | \forall \omega' \in \Omega, U_\pi(\omega) \geqslant U_\pi(\omega')\}. \tag{6.2}$$

The actor's best response is its optimal policy for o_P in the current environment configuration, given δ_A:

$$b_A(E) = \{\pi \in \Pi | \forall \pi' \in \Pi, V_P^\pi(\tilde{s}_0) \geqslant V_P^{\pi'}(\tilde{s}_0) \ \wedge V_P^{\pi_0}(\tilde{s}_0) - V_P^\pi(\tilde{s}_0) \leqslant \delta_A\}, \tag{6.3}$$

where π_0 denotes the optimal policy in E_0 before initiating environment shaping. These best responses are pure strategies since there exists a deterministic optimal policy for a discrete MDP with finite horizon or infinite horizon with discounting [79], and a modification is selected deterministically.

The policy constraints induced by Equations 6.2 and 6.3 induce an equivalent extensive form game with strategy profiles Ω for the designer and Π for the actor, given start state $\tilde{s}_0$. However, the agents have incomplete information: each agent selects a best response based on the information available to it and its payoff matrix, and is unaware of the strategies and payoffs of the other agent. The designer is unaware of how its modifications may impact the actor's policy until the actor recomputes its policy and the actor is unaware of the NSE of its actions. Hence this is an extensive form game with incomplete information.

6.5.4 Experimental Setup

The effectiveness of shaping is investigated in settings with avoidable and unavoidable NSE.

Baselines The performance of shaping with budget is compared with the following baselines. First is the *Initial* approach in which the actor's policy is naively executed and does not involve shaping or any form of learning to mitigate NSE, similar to the *No queries* approach considered in Section 6.3.5. Second is *shaping* with exhaustive

search to select a modification, $b = |\Omega|$. Third is the *Feedback* approach in which the agent performs a trajectory of its optimal policy for o_P and the human approves or disapproves the observed actions based on the NSE occurrence. The agent then disables all the disapproved actions and recomputes a policy for execution. If the updated policy violates the slack, the actor ignores the feedback and executes its initial policy since o_P is prioritized. Using human feedback as training data, the agent learns a predictive model of NSE.

A random forest classifier from the `sklearn` Python package is used for learning a predictive model. Jaccard distance is used to measure the similarity between environment configurations. The actor's MDP is solved using value iteration. The algorithms were implemented in Python and tested on a computer with 16GB of RAM. Action costs are optimized, which are negation of rewards. Values averaged over 100 trials of planning and execution, along with the standard errors, are reported for the following domains.

Boxpushing In this domain, the actor is required to minimize the expected time taken to push a box to the goal location. Each action takes +1 unit of time. The actions succeed with probability 0.9 and may slide right with probability 0.1. Each state is represented as $\langle x, y, b \rangle$ where x, y denote the agent's location and b is a Boolean variable indicating if the agent is pushing the box. Pushing the box over the rug or knocking over a vase on its way results in NSE, incurring a penalty of +5. The designers are the system users. Experiments are conducted on problems with grid size 15×15. Shaping in this domain considers 24 modifications, such as adding a protective sheet over the rug, moving the vase to a corner of the room, removing the rug, block access to the rug and vase area, among others. Removing the rug costs 0.4/unit area covered by the rug, moving the vase costs +1, and all other modifications cost 0.2/unit. Except for blocking access to the rug and vase area, all the other modifications are policy-preserving for the representation considered.

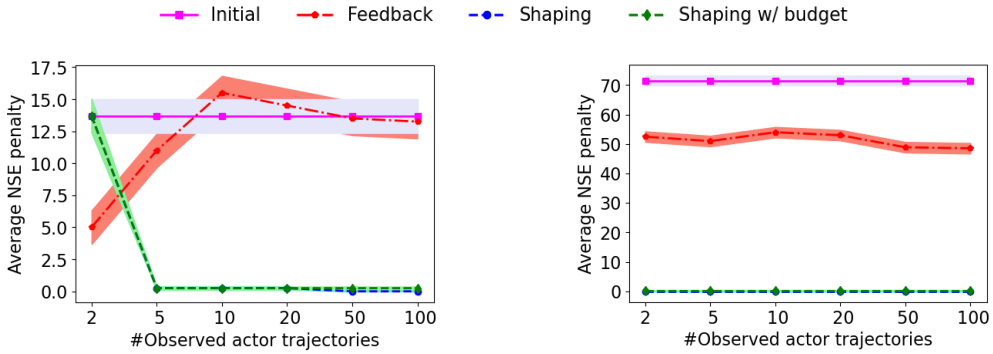
Driving Our second domain is based on simulated autonomous driving, in which the actor's objective is to minimize the expected cost of navigation from start to a goal location, during which it may encounter some potholes. Each state is the agent's location represented by $\langle x, y \rangle$. A grid of size 25×15 is considered. The actor can move in all four directions at low and high speeds, with costs $+2$ and $+1$ respectively. Driving fast through shallow potholes results in a bumpy ride for the user, which is a mild NSE with a penalty of $+2$. Driving fast through a deep pothole may damage the car in addition to the unpleasant experience for the rider and therefore it is a severe NSE with a penalty of $+5$. The cost of modifications are amortized over multiple episodes of agent operation in the environment. The modifications considered are: disabling 'move fast' action for the actor in zone i, $1 \leqslant i \leqslant 4$, which denotes reduced speed limit,; disabling 'move fast' action always; disabling 'move fast' action in zones with shallow potholes; fill all potholes; fill deep potholes; and fill deep potholes in zone i, $1 \leqslant i \leqslant 4$. Disabling 'move fast' costs 0.5 units per pothole in that zone. Filling a pothole costs $+2$ units, and it is a policy-preserving modification.

6.5.5 Results and Discussion

This section discusses the effectiveness of shaping and the effect of slack on the performance.

Effectiveness of shaping The effectiveness of shaping is evaluated in terms of the average NSE penalty incurred and the expected value of o_P after shaping. Figure 6.14 plots the results for the boxpushing domain with $\delta_A = 0$, $\delta_D = 0$, and $b = 3$, as the number of observed actor trajectories is increased.

The results for o_P are not reported since a slack value $\delta_A = 0$ is considered and therefore the agent always optimizes o_P. Feedback budget is 500. With at most five trajectories, the designer is able to select a *policy-preserving* modification that avoids NSE. Shaping w/ budget $b = 3$ performs similar to shaping by evaluating all



(a) Avoidable NSE

(b) Unavoidable NSE

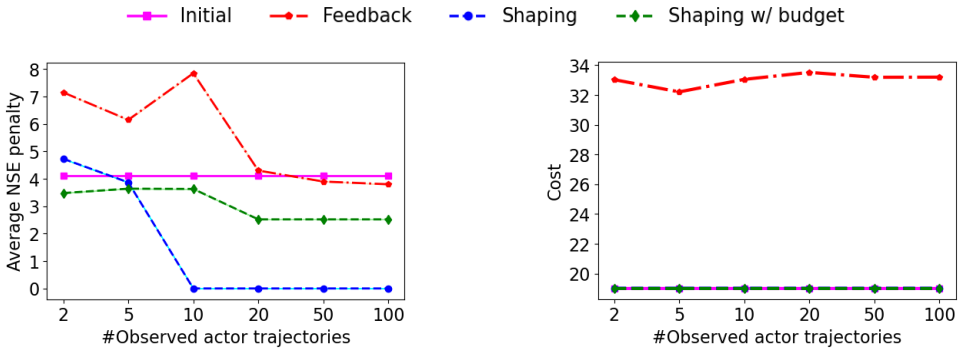
#Trajectories	Shaping	Shaping w/ budget
2	1	1
5	2	1
10	2	1
20	2	1
50	6	3
100	6	3

(c) Number of modifications evaluated when NSE are avoidable.

Figure 6.14: Results on the boxpushing domain.

modifications, and reduces the number of shaping evaluations by 50%. The feedback approach is not able to mitigate NSE since it violates the actor's slack and the agent state representation has limited fidelity, similar to the results in Section 6.4.

Figure 6.15 plots the results on the driving domain with avoidable NSE, $\delta_A = 25\%$, $\delta_D = 0$, and $b = 4$. The driving domain does not have unavoidable NSE since the agent can always avoid NSE by slowing down over potholes. Shaping with budget reduces NSE as more trajectories are observed but is unable to avoid NSE. However, shaping by evaluating all modifications avoids NSE by observing ten actor trajectories. The feedback approach is unable to mitigate NSE due to the limited state representation of the agent. In both domains, it is observed that shaping avoids NSE, after observing few actor trajectories, and shaping w/ budget mitigates NSE considerably.



(a) Average NSE penalty. (b) Expected cost of o_P.

Figure 6.15: Results on driving domain with avoidable NSE.

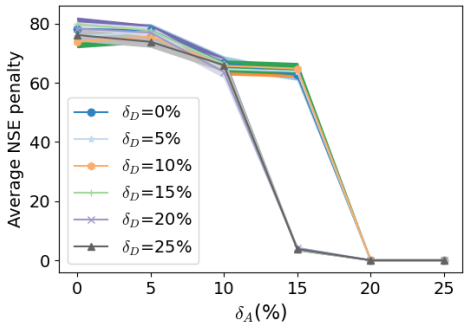


Figure 6.16: Effect of δ_A and δ_D on the average NSE penalty.

Effect of slack The values of δ_A and δ_D are varied and the resulting NSE penalty are plotted in Figure 6.16. Results on the driving domain are reported for shaping based on 100 actor trajectories. The value of δ_A is varied between 0-25% of $V^*_P(\tilde{s}_0|E_0)$ and δ_D is varied between 0-25% of the NSE penalty of the actor's policy in E_0. Our results show that increasing the slack helps reduce the NSE, as expected. In particular, when $\delta_D \geqslant 15\%$, the NSE penalty is considerably reduced with $\delta_A = 15\%$. Overall, increasing δ_A is most effective in reducing the NSE. The effect of slack was also tested on the cost for o_P. The results showed that the cost was predominantly affected by δ_A. A similar performance was observed for all values of δ_D for a fixed δ_A and therefore do not include that plot. This is an expected behavior since o_P is prioritized over minimizing NSE in our setting.

6.6 Limitations of Environment Shaping

While our results show that environment shaping can successfully mitigate both avoidable and unavoidable NSE, the following factors may affect the performance.

6.6.1 Challenges in Performing Shaping

Environment shaping requires the designer, typically the user, to supervise the agent and have control over the environment to perform shaping. In general, certain reconfigurations may require specific skills. Our approach, however, does not model the designer's competency and instead assumes that the set of available modifications is optimized for the designer's skills. Access to a model of the designer's competency is required to automatically identify useful modifications that the designer can implement. In addition, our formulation assumes that the designer can predict the actor's policy from trajectories to perform shaping effectively. In practice, the designer requires knowledge about agent behavior to be able to predict the agent policy with high confidence.

While our algorithm can guide the user in selecting an appropriate modification, the designer may be required to evaluate multiple reconfigurations to effectively avoid NSE when there is a large set of diverse modifications. This can be time-consuming for the user. Though shaping w/ budget helps alleviate this concern, it introduces trade-off between number of modifications to evaluate and the final NSE. A lower budget reduces the number of modifications to evaluate, but it is not guaranteed to find an optimal, utility-maximizing reconfiguration. Another approach to identify the right modifications quickly is to generalize the observed performance across modifications. However, it is non-trivial to generalize the effect of shaping on agent performance and the resulting NSE, since we do not assume any knowledge about the agent's model.

Our framework requires the agent to recompute its policy every time a modification is being evaluated, which can be computationally expensive if the agent operates

based on a large, complex model. Though policy-preserving modifications overcome this limitation, it is challenging to determine a priori whether a given modification is policy-preserving since it depends on the agent's model.

6.6.2 Interference with Future Tasks and Other Agents

Reconfigurations can be temporary or permanent. Permanent modifications may affect the agent's ability to perform future tasks in the environment or even introduce new NSE when the agent performs different types of tasks. This issue did not arise in our settings because the focus was on scenarios where the agent repeatedly performs the same task. It is not straightforward to extend our algorithm to handle situations where an agent performs different types of task, since this requires knowledge about the agent's policies corresponding to different tasks and estimating slack for all the tasks performed by the agent. Further, there may be multiple agents, with different tasks and capabilities, operating simultaneously in an environment. In such settings, even temporary modifications may affect the performance of other agents or introduce NSE of their actions. However, our approach currently does not account for these types of interferences when identifying an appropriate reconfiguration to mitigate NSE.

6.7 Summary

This chapter formalizes the problem of NSE and presents two solution approaches that are complementary in their assumptions and agent responsibilities. First, we investigate the effectiveness of various forms of feedback in learning a model of NSE. This problem is formulated as a multi-objective problem with slack and an algorithm to determine the minimum slack required to avoid these side effects is proposed. Second, an environment shaping approach is presented. This approach leverages human assistance, beyond providing feedback, by reconfiguring the environment. The

problem is formulated as a human-agent collaboration with decoupled objectives and present an algorithm for shaping. Empirical evaluations show that our approaches are effective in mitigating avoidable and unavoidable NSE.

CHAPTER 7

CONCLUSION

Reliable decision-making is critical for long-term deployment of autonomous systems in the open world. Model imprecision can cause unreliable agent behavior. The goal of this dissertation is to develop techniques to identify and mitigate the undesirable behavior of an agent operating under uncertainty and model imprecision. Towards this goal, solutions are presented to address three classes of model imprecision, each with different assumptions on what information is known to the agent a priori. This chapter summarizes the main contributions of this dissertation and suggests directions for future research on reliable AI systems.

7.1 Summary of Contributions

This dissertation presents frameworks to avoid undesirable behavior arising due to three classes of model imprecision in a Markov decision process.

Chapter 3 addresses the model imprecision arising from limited computational resources that requires the agent to simplify its model for planning. The model is simplified for planning by considering fewer action outcomes, relative to the original model, which may lead to unsafe behavior. Building on the observation that the solution quality can be boosted by accounting for certain outcomes, two general methods are introduced to devise risk-aware reduced models that balance the trade-off between model simplicity to fidelity. First, planning using a portfolio of outcome selection principles is presented, to selectively improve the model fidelity and risk awareness in certain states. The reduction impact is used as a heuristic to determine

when all outcomes of an action should be considered. Second, a technique that accounts for the ignored outcomes in the reduced model by modifying the actions costs is introduced. The expected values of the ignored outcomes are added to the actions costs, reflecting the long-term costs of actions in a reduced model. Our results contribute to a better understanding of how disparate reduced model techniques relate to each other and could be used together to achieve optimal action selection when planning with a reduced model.

Chapter 4 relaxes the assumption that precise information about the agent's target environment is available during system design. It targets scenarios in which it is impossible to accurately determine the goal states for planning, ahead of execution, but the uncertainty over the goal states can be modeled via a probability distribution. In such scenarios, using standard SSP models, which require specifying a goal state, can lead to misspecified goal state. The chapter introduces goal uncertain SSP (GUSSP) and presents two solution approaches: a heuristic-based approach and a determinization-based approach. Our results show that by leveraging the fully observable components of the problem, it is possible to solve it optimally without relying on POMDP solvers.

Chapters 5 and 6 relax the assumption that the agent's model encodes uncertainty over the missing details. The chapters target settings in which the agent has no prior information about the missing details, which may lead to negative side effects. Chapter 5 formally defines the negative side effects of agent actions and discusses the key characteristics. Results of a user study show that negative side effects can affect user trust, and that users are willing to assist the system in detecting and mitigating the negative side effects. The user study results also highlight the need for developing effective solution approaches to mitigate negative side effects, and support the design of customizable systems to improve user satisfaction.

Chapter 6 presents two solution methods to mitigate negative side effects: learning from feedback approach, and environment shaping. The two approaches are complementary in their assumptions and agent responsibilities. In learning from feedback, the agent gathers information about side effects and updates its policy, using feedback. In environment shaping, the user reconfigures the environment so as to mitigate the negative side effects of the agent. The agent updates its policy in response to the reconfiguration, and the approach does not involve agent learning. The problem of learning from feedback is formulated as a multi-objective problem with lexicographic reward preferences, and environment shaping as a human-agent collaboration approach. In both the approaches, the reward function for the agent's primary assigned task and the penalty for negative side effects are decoupled. This ensures that the reliability of the agent's model, with respect to its assigned task, is not affected when updating agent policy to mitigate the impacts of negative side effects.

Throughout this dissertation, the proposed approaches are evaluated on simple example problems that capture the real problem without other confounding factors, which allows us to understand the potential benefits and limitations of the techniques. There are many more problems that are yet to be studied and many promising avenues exist for future research.

7.2 Future Work

This section identifies key research directions that can further deepen our understanding of what factors affect reliability and how to design autonomous systems that are cognizant of their limitations, without excessively relying on human assistance.

Our solutions to mitigate negative side effects considered a single agent setting. But agents rarely operate in isolation and their actions impact other agents in the environment. Extending our approaches to avoid negative side effects in a multi-agent setting is an interesting direction for future research. In addition, our approaches

performed a single update of the agent's policy, after gathering data about side effects. But as the system operates for extended periods of time, additional side effects may be discovered. It is therefore important to update the policy, as and when a certain number of new instances of negative side effects are discovered.

There are many factors that affect the reliability of a deployed system, beyond model fidelity. Reliability can be affected by components that inform the reasoning module of the agent. For instance, an agent behavior can be unreliable due to sensor or perception failure, despite using a perfect model for planning. Developing techniques to autonomously recognize such failures and quickly recover from them is an important step for improving the reliability of complex systems such as autonomous vehicles.

Further, perception of reliability by the user is affected when there is a mismatch between user expectations and system performance. Interpretable models and solutions help users understand the model limitations and identify errors in the system. Sometimes, a sub-optimal solution may sometimes be more interpretable than an optimal solution [25, 90]. Further, interpretability depends on the background knowledge of the user. Approaches to improve interpretability must be able to quickly infer user preferences and their background knowledge, customize solutions to maximize interpretability and user satisfaction [89], and efficiently balance the trade-off between solution quality and interpretability.

The approaches presented in this dissertation for identifying undesirable behavior relied on some form of human assistance, and further assumed that the assistance is readily available and accurate. However, human feedback is often noisy and delayed, when available. Designing solution approaches that are robust to errors in human assistance is an important future research direction. No one form of human assistance may work well for all settings or for all the tasks the agent is required to perform. To leverage the broad background knowledge of humans, we should aim to develop techniques to automatically switch between different forms of human assis-

tance, depending on the type of undesirable behavior and the human's preferred form of assistance. To enable long-term autonomy, the system must be able to operate reliably while minimizing the reliance on humans. Further investigations must be conducted to determine how to design metareasoning approaches to monitor system performance, learn to recognize new risks, and automatically identify error sources.

This dissertation focused on the consequences of operating based on an imprecise model, from a safety and control perspective. The system may produce other forms of undesirable behavior as well. The system's decisions may be considered unfair in different ways to different groups, when optimizing an incompletely specified objective [29, 39]. A key challenge in fair sequential decision-making is precise specification of the fairness metric. Developing general techniques for fair sequential decision-making is an important direction for future research.

7.3 Final Thoughts

AI systems will continue to have broad societal impacts in the years to come. Researchers today have a unique opportunity to shape this impact and a massive responsibility to ensure these systems operate reliably and benefit society. This thesis is merely a step towards enabling reliable operation of autonomous systems in the open world. It is our hope that techniques introduced in this dissertation will inspire others to propose solutions, improving upon our techniques, to enable widespread deployment of autonomous systems that are reliable and which benefit society.

APPENDIX

NEGATIVE SIDE EFFECTS USER STUDY SURVEY QUESTIONNAIRE

This chapter presents the survey questionnaires used for our human subjects study to understand user attitudes towards negative side effects. The results of this study are discussed in detail in Chapter 5.

Pre-Survey Questionnaire

The pre-survey questionnaire was used to assess participants' familiarity with AI systems and their fluency in English.

Unless otherwise specified, select an option that describes your attitude for the following questions.

1. Are you over 30 years of age?

   - Yes
   - No

2. Are you fluent in English?

   - Yes
   - No

3. "Autocorrect" is a common tool in smartphones that automatically corrects misspelled words in our text. On most occasions, the autocorrect tool corrects the spelling

of the word we misspell. Sometimes, it may guess the word incorrectly and modify the intended word. From the following options, select a response that best summarizes your experience.

- I rarely experience this problem, I continue to use autocorrect
- I often experience this problem but I continue to use autocorrect
- I experienced this problem and disabled autocorrect
- I have not experienced this issue

4. Imagine you are riding in a self-driving car. The car is trained to follow the standard traffic rules but does not slow down when driving through low visibility areas. Will you continue to use the car in self-driving mode?

- I will continue to use the car in the self-driving mode
- I will not use the self-driving mode in low visibility areas
- I will not use the self-driving mode; I do not trust it

5. How comfortable are you/would you be using products with Artificial Intelligence algorithms? (Examples of such products: robotic vacuum cleaner, Google translate)

- Uncomfortable, I will not use such products
- Somewhat uncomfortable, I prefer not to use such products
- Neutral, I do not mind using such products
- Somewhat comfortable, I like to use products powered by Artificial Intelligence algorithms
- Comfortable, I often use products powered by Artificial Intelligence algorithms

6. Which of the following technologies have you used in the past year? Select all that apply.

- Virtual assistants like Apple Siri, Amazon Alexa or Google Nest
- Tools for navigation such as Waze, Google maps, Apple maps, Uber, Lyft
- Check deposits using mobile banking applications
- None

Roomba Questionnaire

This section presents the questionnaire for the Roomba domain.

Scenario: Consider a robotic vacuum cleaner that is assigned the task of mopping the wooden floor of your living room everyday. The robot takes 10 minutes to complete this task. However, when cleaning the areas closer to the wall, it sprays some water on the wall.

1. From the following options, select one that best describes your tolerance towards this behavior.

- I am willing to tolerate the robot spraying water on the wall
- I am willing to tolerate the robot spraying water on the wall but will use the robot less frequently
- I do not tolerate this behavior and I will not use this robot in its current condition

2. On a scale of 1 to 5, how acceptable is it to spray water on walls? [1 is unacceptable, 5 is acceptable] _____

3. You can personalize the robot and help it learn to avoid spraying water on the wall by pressing a button every time the robot makes this mistake. This will help the robot avoid spraying water on walls in any room in the house. How often are you willing to provide feedback to the robot by pressing the button?

- Till the robot learns to avoid it
- A few times
- Whenever I have time to closely monitor it
- Never

4. By avoiding cleaning the area near the walls (∼5 inches from the wall), the robot can prevent spraying water on the walls. Is this an acceptable behavior?

- Yes, I do not mind the robot skipping the areas closer to the wall
- No, I want the entire floor to be cleaned

5. You can add a protective sheet on areas of the wall closer to the floor when the robot is cleaning. Assume this protective sheet costs $10. This will prevent the robot from spraying water on the wall while cleaning the floor entirely. Are you willing to purchase and install the protective sheet?

- I am willing to purchase the sheet and install it
- I am not willing to purchase the sheet but I am willing to install it, if it is provided by the manufacturer for no additional cost
- I am neither willing to purchase nor install the sheet

6. You can either provide feedback to the robot by pressing a button every time it makes mistakes or install the protective sheet (assuming the sheet is provided by the manufacturer) once. Select the option that best describes your preference.

- Providing feedback by pressing the button as many times till the robot learns to avoid spraying water on the walls
- Install protective sheet right away
- Install protective sheet if the robot does not learn after providing feedback 50-100 times
- Neither, I can tolerate water spray on the walls

7. Imagine a setting where the dust only accumulates in the corners. There is no way for the robot to remove the dust without spraying water on the wall. Which of the following best describes your tolerance towards this situation?

- The robot can clean the floor; I can tolerate some water sprayed on the walls
- I want the robot to minimize spraying water on the walls as much as possible; I can tolerate some water sprayed on some areas
- I do not tolerate spraying water on the walls, I prefer the robot to skip cleaning the areas near the wall
- I do not tolerate this behavior and I don't want to use this robot in its current condition

8. Imagine a setting where the dust only accumulates in the corners. There is no way for the robot to remove the dust without spraying water on the wall. But you can help by adding a protective sheet on the wall when the robot is cleaning. Are you willing to purchase and install the protective sheet?

- I am willing to purchase the sheet and install it
- I am not willing to purchase the sheet but I am willing to install it, if it is provided by the manufacturer for no additional cost

- I am neither willing to purchase nor install the sheet

9. How does the robot spraying water on the walls when mopping the floor affect your trust on the product?

- The robot's actions are sometimes inconvenient but I trust it to complete its cleaning task
- I trust the robot less when it does not learn to avoid its mistakes over time
- I do not trust the robot to be capable of completing its task and I will not use it

10. Which of the following features will encourage you to continue using the robotic vacuum cleaner instead of returning it to the manufacturer? Select all that apply.

- Ability to press a button and provide feedback to the robot when its actions are not desirable, so it can learn to avoid its mistakes
- Ability to select areas where the robot is not allowed to operate, such as areas near the wall
- Tools such as protective sheet for sockets

AV Questionnaire

This section presents the questionnaire for the autonomous vehicle (AV) domain.

Scenario: Imagine you are riding in a self-driving car from home to work every day. The car has been tested for conforming to standard traffic rules. The car takes 20 minutes to reach your destination from your start location. However, the ride is bumpy since the car drives fast through potholes.

1. From the following options, select one that best describes your tolerance towards this scenario.

- It is sometimes inconvenient but I am willing to tolerate the bumpy ride
- I am willing to tolerate the bumpy ride but will use the car less frequently
- I do not tolerate bumpy rides

2. Imagine the car can recognize the stop signs only when it is closer to it and therefore slams the brake *every time* a stop sign is encountered. From the following options, select one that best describes your tolerance towards this consequence.

- It is sometimes inconvenient but I am willing to tolerate the jerk
- I am willing to tolerate the jerk but will use the car less frequently
- I do not tolerate this behavior

3. On a scale of 1 to 5, with 1 being the lowest and 5 being the highest, how acceptable is it to you if the ride is bumpy? [1 is unacceptable and 5 is acceptable] _____

4. You can personalize the car and help it learn to avoid driving fast through potholes by pressing a button every time the car makes this mistake. How often are you willing to provide feedback to the car by pressing the button?

- Till the car learns to avoid it
- A few times
- Never

5. The bumpy ride can be avoided by taking a longer route with no potholes. This takes 25 minutes to reach the destination, instead of 20 minutes. Is this an acceptable behavior?

- Yes, I am willing to travel a longer route
- No, I want the car to drive slowly through potholes and travel in the current route

- No, I want to reach the destination in 20 minutes

6. You can add a pothole-detection sensor to the car. This sensor, costing $50, will detect potholes and prevent the car from going fast through potholes. Are you willing to purchase and install this sensor?

- Yes, I am willing to purchase the sensor and install it
- No, I am not willing to purchase the sensor but willing to install the sensor, if it is provided by the manufacturer for no additional cost
- No, I am neither willing to purchase nor install the sensor

7. You can either provide feedback to the car by pressing a button every time it drives fast through a pothole or install the pothole-detection sensor once (assuming the sensor is provided by the manufacturer). Select the option that best describes your preference.

- Providing feedback by pressing the button as many times till the car learns to avoid driving fast through potholes
- Install pothole-detection sensor right away
- Install pothole-detection sensor if the car does not learn after providing feedback 50-100 times
- Neither, I can tolerate the bumpy ride

8. Imagine a setting where all the roads to your destination have equal number of potholes. There is no way to avoid roads with potholes to reach your destination. Which of the following best describes your tolerance towards this situation?

- I can tolerate the bumpy ride

- I want the car to slow down on potholes as much as possible; I can tolerate some bumpiness
- I do not tolerate bumpy rides

9. Imagine a setting where all the roads to your destination have equal number of potholes. There is no way to avoid roads with potholes to reach your destination. You can add a sensor, which costs $50, to the car that will detect potholes and prevent the car from going fast through potholes. Are you willing to purchase and install this sensor?

- Yes, I am willing to purchase the sensor and install it
- No, I am not willing to purchase the sensor but willing to install the sensor, if it is provided by the manufacturer for no additional cost
- No, I am neither willing to purchase nor install the sensor

10. How does the car's performance when driving on potholes affect your attitude towards it?

- It is sometimes inconvenient but I can tolerate it
- I trust the car less when it does not learn to avoid its mistakes over time
- I will not use the self-driving mode on roads with potholes, I prefer to use the manual driving mode on roads with potholes
- I do not trust the car to be capable of driving safely

11. Which of the following features will encourage you to continue using the self-driving car instead of returning it to the manufacturer? Select all that apply.

- Ability to press a button and provide feedback to the car when its actions are not desirable, so it can learn to avoid its mistakes

- Ability to select areas where self-driving can be disabled and manual driving is enabled
- Tools such as sensors to detect potholes and reduce speed automatically

BIBLIOGRAPHY

[1] Alves, Gleifer Vaz, Dennis, Louise, Fernandes, Lucas, and Fisher, Michael. Reliable decision-making in autonomous vehicles. *Validation and Verification of Automated Systems* (2020), 105–117.

[2] Amato, Christopher, Bonet, Blai, and Zilberstein, Shlomo. Finite-state controllers based on mealy machines for centralized and decentralized POMDPs. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI)* (2010).

[3] Amodei, Dario, Olah, Chris, Steinhardt, Jacob, Christiano, Paul, Schulman, John, and Mané, Dan. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565* (2016).

[4] Bansal, Gagan, Nushi, Besmira, Kamar, Ece, Lasecki, Walter S., Weld, Daniel S., and Horvitz, Eric. Beyond accuracy: The role of mental models in human-AI team performance. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing (HCOMP)* (2019).

[5] Barto, Andrew G., Bradtke, Steven J., and Singh, Satinder P. Learning to act using real-time dynamic programming. *Artificial Intelligence 72* (1995), 81–138.

[6] Basich, Connor, Svegliato, Justin, Wray, Kyle Hollins, Witwicki, Stefan J., Biswas, Joydeep, and Zilberstein, Shlomo. Learning to optimize autonomy in competence-aware systems. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)* (2020).

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

[8] Bennett, Casey C., and Hauser, Kris. Artificial intelligence framework for simulating clinical decision-making: A markov decision process approach. *Artificial intelligence in medicine 57* (2013), 9–19.

[9] Bertsekas, Dimitri P., and Tsitsiklis, John N. *Parallel and Distributed Computation: Numerical Methods*, vol. 23. Prentice hall Englewood Cliffs, 1989.

[10] Bertsekas, Dimitri P., and Tsitsiklis, John N. An analysis of stochastic shortest path problems. *Mathematics of Operations Research 16* (1991), 580–595.

[11] Biswas, Joydeep, and Veloso, Manuela. Multi-sensor mobile robot localization for diverse environments. In *Robot Soccer World Cup* (2013), pp. 468–479.

[12] Bonet, Blai, and Geffner, Hector. Labeled RTDP: Improving the convergence of real-time dynamic programming. In *Proceedings of the 13th International Conference on Automated Planning and Scheduling (ICAPS)* (2003).

[13] Bonet, Blai, and Geffner, Héctor. Solving POMDPs: RTDP-bel vs. point-based algorithms. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)* (2009).

[14] Box, George E.P. Robustness in the strategy of scientific model building. In *Robustness in statistics*. 1979, pp. 201–236.

[15] Brown, Matthew, Saisubramanian, Sandhya, Varakantham, Pradeep Reddy, and Tambe, Milind. STREETS: Game-theoretic traffic patrolling with exploration and exploitation. In *Proceedings of the 26th Innovative Applications of Artificial Intelligence (IAAI)* (2014).

[16] Buckman, Jacob, Hafner, Danijar, Tucker, George, Brevdo, Eugene, and Lee, Honglak. Sample-efficient reinforcement learning with stochastic ensemble value expansion. In *Advances in Neural Information Processing Systems (NeurIPS)* (2018).

[17] Chandak, Yash, Jordan, Scott, Theocharous, Georgios, White, Martha, and Thomas, Philip S. Towards safe policy improvement for non-stationary MDPs. In *Advances in Neural Information Processing Systems (NeurIPS)* (2020).

[18] Coren, Michael J. All the things that still baffle self-driving cars, starting with seagulls. Quartz, September 23, 2018. "https://qz.com/1397504/all-the-things-that-still-baffle-self-driving-cars-starting-with-seagulls/".

[19] Crane, Daniel A., Logue, Kyle D., and Pilz, Bryce C. A survey of legal issues arising from the deployment of autonomous and connected vehicles. *Michigan Telecommunications and Technology Law Review 23* (2017), 191–320.

[20] Delgado, Karina Valdivia, Sanner, Scott, and De Barros, Leliane Nunes. Efficient solutions to factored MDPs with imprecise transition probabilities. *Artificial Intelligence 175*, 9-10 (2011), 1498–1527.

[21] Dietterich, Thomas G. Steps toward robust artificial intelligence. *AI Magazine* (2017).

[22] Dietvorst, Berkeley J., Simmons, Joseph P., and Massey, Cade. Algorithm aversion: People erroneously avoid algorithms after seeing them err. *Journal of Experimental Psychology: General 144*, 1 (2015), 114.

[23] Dietvorst, Berkeley J., Simmons, Joseph P., and Massey, Cade. Overcoming algorithm aversion: People will use imperfect algorithms if they can (even slightly) modify them. *Management Science 64*, 3 (2018), 1155–1170.

[24] Downs, Julie S., Holbrook, Mandy B., Sheng, Steve, and Cranor, Lorrie Faith. Are your participants gaming the system? screening mechanical turk workers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)* (2010), pp. 2399–2402.

[25] Dragan, Anca D., Lee, Kenton, and Srinivasa, Siddhartha S. Legibility and predictability of robot motion. In *Proceedings of the 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI)* (2013).

[26] Eversource. Eversource energy - time of use rates, 2017. "https://www.eversource.com/clp/vpp/vpp.aspx".

[27] Folsom-Kovarik, Jeremiah T., Sukthankar, Gita, and Schatz, Sae. Tractable POMDP representations for intelligent tutoring systems. *ACM Transactions on Intelligent Systems and Technology (TIST) 4*, 2 (2013), 29.

[28] Forlizzi, Jodi, and DiSalvo, Carl. Service robots in the domestic environment: A study of the roomba vacuum in the home. In *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-Robot Interaction (HRI)* (2006).

[29] Galhotra, Sainyam, Saisubramanian, Sandhya, and Zilberstein, Shlomo. Learning to generate fair clusters from demonstrations. In *Proceedings of the AAAI/ACM Conference on Artificial Intelligence, Ethics and Society (AIES)* (2021).

[30] Ghavamzadeh, Mohammad, Petrik, Marek, and Chow, Yinlam. Safe policy improvement by minimizing robust baseline regret. In *Advances in Neural Information Processing Systems (NeurIPS)* (2016).

[31] Guestrin, Carlos, Koller, Daphne, and Parr, Ronald. Max-norm projections for factored MDPs. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI)* (2001).

[32] Hadfield-Menell, Dylan, Milli, Smitha, Abbeel, Pieter, Russell, Stuart J., and Dragan, Anca. Inverse reward design. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)* (2017).

[33] Hadfield-Menell, Dylan, Russell, Stuart J., Abbeel, Pieter, and Dragan, Anca. Cooperative inverse reinforcement learning. vol. 29, pp. 3909–3917.

[34] Hansen, Eric A. Indefinite-horizon POMDPs with action-based termination. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence (AAAI)* (2007).

[35] Hansen, Eric A., and Zilberstein, Shlomo. LAO*: A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence 129* (2001), 35–62.

[36] Hart, Peter E., Nilsson, Nils J., and Raphael, Bertram. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics 4* (1968), 100–107.

[37] Hibbard, Bill. Avoiding unintended AI behaviors. In *Proceedings of the 5th International Conference on Artificial General Intelligence (AGI)* (2012).

[38] Hoffmann, Jörg. FF: The fast-forward planning system. *AI Magazine 22* (2001), 57.

[39] Howard, Ayanna, and Borenstein, Jason. Hacking the human bias in robotics. *ACM Transactions on Human-Robot Interaction 7* (2018).

[40] Howard, Ronald A. *Dynamic Programming and Markov Processes*. John Wiley, 1960.

[41] Hulse, Lynn M., Xie, Hui, and Galea, Edwin R. Perceptions of autonomous vehicles: Relationships with road users, risk, gender and age. *Safety Science 102* (2018), 1–13.

[42] Insurance Institute for Highway Safety. Reality check: Research, deadly crashes show need for caution on road to full autonomy. *Status Report Newsletter 53*, 4 (2018), 1–12.

[43] Issakkimuthu, Murugeswari, Fern, Alan, Khardon, Roni, Tadepalli, Prasad, and Xue, Shan. Hindsight optimization for probabilistic planning with factored actions. In *Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS)* (2015).

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

[45] Kalra, Nidhi, and Paddock, Susan M. Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? *Transportation Research Part A: Policy and Practice 94* (2016), 182–193.

[46] Keller, Thomas, and Eyerich, Patrick. A polynomial all outcome determinization for probabilistic planning. In *Proceedings of the 21st International Conference on Automated Planning and Scheduling (ICAPS)* (2011).

[47] Keller, Thomas, and Eyerich, Patrick. PROST: Probabilistic planning based on UCT. In *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS)* (2012).

[48] Keller, Thomas, and Helmert, Malte. Trial-based heuristic tree search for finite horizon MDPs. In *Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS)* (2013).

[49] Keyder, Emil, and Geffner, Hector. The HMDP planner for planning with probabilities. In *6th International Planning Competition* (2008).

[50] Kitano, Hiroaki, Tadokoro, Satoshi, Noda, Itsuki, Matsubara, Hitoshi, Takahashi, Tomoichi, Shinjou, Atsuhi, and Shimada, Susumu. RoboCup rescue: Search and rescue in large-scale disasters as a domain for autonomous agents research. In *IEEE Conference on Systems, Man, and Cybernetics* (1999).

[51] Kocielnik, Rafal, Amershi, Saleema, and Bennett, Paul N. Will you accept an imperfect AI? exploring designs for adjusting end-user expectations of AI systems. In *Proceedings of the CHI Conference on Human Factors in Computing Systems (CHI)* (2019).

[52] Kocsis, Levente, and Szepesvári, Csaba. Bandit based Monte-Carlo planning. In *Proceedings of the 17th European Conference on Machine Learning (ECML)* (2006).

[53] Kolobov, Andrey, and Mausam. Planning with Markov decision processes: An AI perspective. *Synthesis Lectures on Artificial Intelligence and Machine Learning 6* (2012), 1–210.

[54] Kolobov, Andrey, Mausam, and Weld, Daniel S. ReTrASE: Integrating paradigms for approximate probabilistic planning. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)* (2009).

[55] Korf, Richard E. Real-time heuristic search. *Artificial Intelligence 42* (1990), 189–211.

[56] Krakovna, Victoria, Orseau, Laurent, Martic, Miljan, and Legg, Shane. Penalizing side effects using stepwise relative reachability. In *AI Safety Workshop, IJCAI* (2019).

[57] Krakovna, Victoria, Orseau, Laurent, Ngo, Richard, Martic, Miljan, and Legg, Shane. Avoiding side effects by considering future tasks. In *Advances in Neural Information Processing Systems (NeurIPS)* (2020).

[58] Kurniawati, Hanna, Hsu, David, and Lee, Wee Sun. Sarsop: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Robotics: Science and systems (RSS)* (2008).

[59] Kyriakidis, Miltos, Happee, Riender, and de Winter, Joost C.F. Public opinion on automated driving: Results of an international questionnaire among 5000 respondents. *Transportation research part F: traffic psychology and behaviour 32* (2015), 127–140.

[60] Lakkaraju, Himabindu, Kamar, Ece, Caruana, Rich, and Horvitz, Eric. Identifying unknown unknowns in the open world: Representations and policies for guided exploration. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI)* (2017).

[61] Li, Lihong, Walsh, Thomas J., and Littman, Michael L. Towards a unified theory of state abstraction for MDPs. In *Proceedings of the International Symposium on Artificial Intelligence and Mathematics* (2006).

[62] Lindner, David, Matoba, Kyle, and Meulemans, Alexander. Challenges for using impact regularizers to avoid negative side effects. *CoRR abs/2101.12509* (2021).

[63] Liptak, Andrew. Amazon's alexa started ordering people dollhouses after hearing its name on TV. The Verge, January 7, 2017. "https://www.theverge.com/2017/1/7/14200210/amazon-alexa-tech-news-anchor-order-dollhouse".

[64] Littman, Michael L. Probabilistic propositional planning: Representations and complexity. In *Proceedings of the 14th AAAI International Conference on Artificial Intelligence (AAAI)* (1997).

[65] Madani, Omid, Hanks, Steve, and Condon, Anne. On the undecidability of probabilistic planning and infinite-horizon partially observable Markov decision problems. In *Proceedings of the 16th AAAI Conference on Artificial Intelligence (AAAI)* (1999).

[66] McCallum, R. Andrew. Overcoming incomplete perception with utile distinction memory. In *Proceedings of the 10th International Conference on Machine Learning (ICML)* (1993).

[67] McCurry, Justin. South korean woman's hair 'eaten' by robot vacuum cleaner as she slept. The Guardian, February 8, 2015. "https://www.theguardian.com/world/2015/feb/09/south-korean-womans-hair-eaten-by-robot-vacuum-cleaner-as-she-slept".

[68] Minsky, Marvin. *Society of mind*. Simon and Schuster, 1988.

[69] Murphy, Mike. Alphabet's Eric Schmidt: The design of AI should "avoid undesirable outcomes". Quartz, December 22, 2015. "https://qz.com/579741/alphabets-eric-schmidt-the-design-of-ai-should-avoid-undesirable-outcomes/".

[70] Nie, Xinkun, Wong, Lawson L.S., and Kaelbling, Leslie Pack. Searching for physical objects in partially known environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)* (2016).

[71] Ong, Sylvie, Png, Shao Wei, Hsu, David, and Lee, Wee Sun. Planning under uncertainty for robotic tasks with mixed observability. *International Journal of Robotics Research 29* (2010), 1053–1068.

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

[73] Patek, Stephen D. On partially observed stochastic shortest path problems. In *Proceedings of the 40th IEEE Conference on Decision and Control (CDC)* (2001).

[74] Petrik, Marek, and Zilberstein, Shlomo. Learning parallel portfolios of algorithms. *Annals of Mathematics and Artificial Intelligence 48* (2006), 85–106.

[75] Pineau, Joelle, Gordon, Geoff, and Thrun, Sebastian. Point-based value iteration: An anytime algorithm for POMDPs. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI)* (2003).

[76] Pineda, Luis, Takahashi, Takeshi, Jung, Hee-Tae, Zilberstein, Shlomo, and Grupen, Roderic. Continual planning for search and rescue robots. In *Proceedings of the 15th IEEE Conference on Humanoid Robots* (2015).

[77] Pineda, Luis, Wray, Kyle, and Zilberstein, Shlomo. Fast SSP solvers using short-sighted labeling. In *Proceedings of the 31st International Conference on Artificial Intelligence (AAAI)* (2017).

[78] Pineda, Luis, and Zilberstein, Shlomo. Planning under uncertainty using reduced models: Revisiting determinization. In *Proceedings of the 24th International Conference on Automated Planning and Scheduling (ICAPS)* (2014).

[79] Puterman, Martin L. Markov decision processes. *Handbooks in operations research and management science 2* (1990), 331–434.

[80] Ramakrishnan, Ramya, Kamar, Ece, Dey, Debadeepta, Shah, Julie, and Horvitz, Eric. Discovering blind spots in reinforcement learning. In *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)* (2018).

[81] Ramakrishnan, Ramya, Kamar, Ece, Nushi, Besmira, Dey, Debadeepta, Shah, Julie, and Horvitz, Eric. Overcoming blind spots in the real world: Leveraging complementary abilities for joint execution. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI)* (2019).

[82] Ravindran, Balaraman. *An algebraic approach to abstraction in reinforcement learning*. PhD thesis, University of Massachusetts at Amherst, 2004.

[83] Regan, Kevin, and Boutilier, Craig. Robust policy computation in reward-uncertain MDPs using nondominated policies. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI)* (2010).

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

[85] Ross, Ron, Pillitteri, Victoria, Graubart, Richard, Bodeau, Deborah, and McQuaid, Rosalie. Developing cyber resilient systems: A systems security engineering approach. Tech. rep., National Institute of Standards and Technology, 2019.

[86] Russell, Stuart. Provably beneficial artificial intelligence. *Exponential Life, The Next Step* (2017).

[87] Russell, Stuart. *Human compatible: Artificial Intelligence and the Problem of Control*. Penguin, 2019.

[88] Russell, Stuart, and Norvig, Peter. *Artificial Intelligence: A modern approach*. Prentice Hall, 1995.

[89] Saisubramanian, Sandhya, Basich, Connor, Zilberstein, Shlomo, and Goldman, Claudia V. Satisfying social preferences in ridesharing services. In *Proceedings of the IEEE Intelligent Transportation Systems Conference (ITSC)* (2019).

[90] Saisubramanian, Sandhya, Galhotra, Sainyam, and Zilberstein, Shlomo. Balancing the tradeoff between clustering value and interpretability. In *Proceedings of the AAAI/ACM Conference on Artificial Intelligence, Ethics and Society (AIES)* (2020).

[91] Saisubramanian, Sandhya, Kamar, Ece, and Zilberstein, Shlomo. Mitigating the negative side effects of reasoning with imperfect models: A multi-objective approach. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)* (2020).

[92] Saisubramanian, Sandhya, Kamar, Ece, and Zilberstein, Shlomo. A multi-objective approach to mitigate negative side effects. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI)* (2020).

[93] Saisubramanian, Sandhya, Roberts, Shannon C., and Zilberstein, Shlomo. Understanding user attitudes towards negative side effects of AI systems. In *Proceedings of the CHI Conference on Human Factors in Computing Systems, Late Breaking Work Track (CHI-LBW)* (2021).

[94] Saisubramanian, Sandhya, Varakantham, Pradeep, and Lau, Hoong Chuin. Risk based optimization for improving emergency medical systems. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI)* (2015).

[95] Saisubramanian, Sandhya, Wray, Kyle, Pineda, Luis, and Zilberstein, Shlomo. Planning in stochastic environments with goal uncertainty. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2019).

[96] Saisubramanian, Sandhya, and Zilberstein, Shlomo. Safe reduced models for probabilistic planning. In *ICML/IJCAI/AAMAS Workshop on Planning and Learning (PAL)* (2018).

[97] Saisubramanian, Sandhya, and Zilberstein, Shlomo. Adaptive outcome selection for planning with reduced models. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2019).

[98] Saisubramanian, Sandhya, and Zilberstein, Shlomo. Mitigating negative side effects via environment shaping. In *Proceedings of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)* (2021).

[99] Saisubramanian, Sandhya, Zilberstein, Shlomo, and Kamar, Ece. Avoiding negative side effects due to incomplete knowledge of AI systems. *CoRR abs/2008.12146* (2020).

[100] Saisubramanian, Sandhya, Zilberstein, Shlomo, and Shenoy, Prashant. Optimizing electric vehicle charging through determinization. In *ICAPS Workshop on Scheduling and Planning Applications* (2017).

[101] Saisubramanian, Sandhya, Zilberstein, Shlomo, and Shenoy, Prashant. Planning using a portfolio of reduced models. In *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)* (2018).

[102] Seshia, Sanjit A., Sadigh, Dorsa, and Sastry, S. Shankar. Formal methods for semi-autonomous driving. In *Proceedings of the Design Automation Conference (DAC)* (2015), pp. 1–5.

[103] Seuken, Sven, and Zilberstein, Shlomo. Improved memory-bounded dynamic programming for decentralized POMDPs. In *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence (UAI)* (2007).

[104] Shah, Rohin, Krasheninnikov, Dmitrii, Alexander, Jordan, Abbeel, Pieter, and Dragan, Anca. Preferences implicit in the state of the world. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)* (2019).

[105] Smallwood, Richard D., and Sondik, Edward J. The optimal control of partially observable markov processes over a finite horizon. *Operations research 21*, 5 (1973), 1071–1088.

[106] Solon, Olivia. Roomba creator responds to reports of 'poopocalypse': 'we see this a lot'. The Guardian, August 15, 2016. "https://www.theguardian.com/technology/2016/aug/15/roomba-robot-vacuum-poopocalypse-facebook-post".

[107] Stone, Lawrence D., Royset, Johannes O., and Washburn, Alan R. Search for a stationary target. In *Optimal Search for Moving Targets*. Springer, 2016, pp. 9–48.

[108] Styler, Breelyn, and Simmons, Reid. Plan-time multi-model switching for motion planning. In *Proceedings of the 27th International Conference on Automated Planning and Scheduling* (2017).

[109] Svegliato, Justin, Wray, Kyle Hollins, Witwicki, Stefan J., Biswas, Joydeep, and Zilberstein, Shlomo. Belief space metareasoning for exception recovery. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2019).

[110] Teichteil-Königsbuch, Florent, Kuter, Ugur, and Infantes, Guillaume. Incremental plan aggregation for generating policies in MDPs. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)* (2010).

[111] Thomas, Philip S., da Silva, Bruno Castro, Barto, Andrew G., Giguere, Stephen, Brun, Yuriy, and Brunskill, Emma. Preventing undesirable behavior of intelligent machines. *Science 366*, 6468 (2019), 999–1004.

[112] Turner, Alex, Ratzlaff, Neale, and Tadepalli, Prasad. Avoiding side effects in complex environments. In *Advances in Neural Information Processing Systems (NeurIPS)* (2020).

[113] Turner, Alexander Matt, Hadfield-Menell, Dylan, and Tadepalli, Prasad. Conservative agency via attainable utility preservation. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society (AIES)* (2020).

[114] Wang, Ziyu, Bapst, Victor, Heess, Nicolas, Mnih, Volodymyr, Munos, Remi, Kavukcuoglu, Koray, and de Freitas, Nando. Sample efficient actor-critic with experience replay. *CoRR abs/1611.01224* (2016).

[115] Witwicki, Stefan, Melo, Francisco, Capitán, Jesús, and Spaan, Matthijs. A flexible approach to modeling unpredictable events in MDPs. In *Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS)* (2013).

[116] Wray, Kyle Hollins, Zilberstein, Shlomo, and Mouaddib, Abdel-Illah. Multi-objective MDPs with conditional lexicographic reward preferences. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI)* (2015).

[117] Yin, Ming, Wortman Vaughan, Jennifer, and Wallach, Hanna. Understanding the effect of accuracy on trust in machine learning models. In *Proceedings of the CHI Conference on Human Factors in Computing Systems (CHI)* (2019).

[118] Yoon, Sungwook, Fern, Alan, and Givan, Robert. FF-Replan: A baseline for probabilistic planning. In *Proceedings of the 17th International Conference on Automated Planning and Scheduling (ICAPS)* (2007).

[119] Yoon, Sungwook, Fern, Alan, Givan, Robert, and Kambhampati, Subbarao. Probabilistic planning via determinization in hindsight. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI)* (2008).

[120] Yoon, Sungwook, Ruml, Wheeler, Benton, J., and Do, Minh B. Improving determinization in hindsight for on-line probabilistic planning. In *Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS)* (2010).

[121] Zhang, Shun, Durfee, Edmund H., and Singh, Satinder. Querying to find a safe policy under uncertain safety constraints in markov decision processes. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI)* (2020).

[122] Zhang, Shun, Durfee, Edmund H., and Singh, Satinder P. Minimax-regret querying on side effects for safe optimality in factored Markov decision processes. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)* (2018), pp. 4867–4873.

[123] Zilberstein, Shlomo. Building strong semi-autonomous systems. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI)* (2015).

[124] Zilberstein, Shlomo, Washington, Richard, Bernstein, Daniel S., and Mouaddib, Abdel-Illah. Decision-theoretic control of planetary rovers. In *Revised Papers from the International Seminar on Advances in Plan-Based Control of Robotic Agents* (2002), Springer-Verlag, pp. 270–289.