Bayesian Reinforcement Learning for POMDP-based Dialogue Systems

ShaoWei Png

Master of Science

School of Computer Science

McGill University

Montreal,Quebec

2011-05-18

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of Master of Science

©ShaoWei Png 2011

DEDICATION

To everyone who has crossed my path in life.

ACKNOWLEDGEMENTS

Without any uncertainty, I would like to express my gratitude to my supervisor Joelle Pineau. I am especially grateful for her scholarship during my two years of studies and for giving me the opportunity to present my work at the ICASSP conference in Prague. She is the most wonderful professor I have ever known, and she is always so generous with her knowledge and advice. Her guidance has improved both my research and writing skills and made me a better researcher. It amazes me as to how someone can be so outstanding but humble, strict but still always places the interests of her students as the top priority. I have learnt many life lessons from her.

I also gratefully acknowledge support from the Natural Sciences and Engineering Council of Canada (NSERC) and the Fonds Qúebécois de la Recherche sur la Nature et les Technologies (FQRNT).

I have learnt a lot from Doina Precup's COMP 652 Machine Learning course, and I like to thank her. I also like to thank Amin Atrash and Julien Villemure for giving me guidance in the SmartWheeler project. Stephane Ross has also given me plenty of advice regarding BAPOMDPs, and I am also grateful.

I also like to thank my previous supervisors David Hsu and Lee Wee Sun. They wrote recommendation letters for my application to graduate schools, and they have strongly encouraged me to pursue a graduate degree in McGill University.

I like to thank my mother and sister for believing in me for the past 20 years of my life. They have always been around for me whenever I need them.

My lab mates in 112 are a group of people I saw almost every day for the last two years. I appreciate Mahdi for his advice on how to save money and Yuri and his wife Katy for driving me to all the different hiking and ski trips. Arthur and Keith have left 112, but I like to thank them for their advice during my first year of graduate school. I thank Guillaume, Jordan, Sylvie, Athena, Cosmin, Andre, Ivan, Phil, Rob, Bob, Bert, Julieta, Emily and Aaron for their friendship in the Reasoning and Learning's research group. I know that I will miss my graduate school life badly.

I will also miss my fellow computer science graduate students including Wen-Yang, XiaoHu, Sevan, WanRu, YanCheng, Mazen, Omar, Milena, Stephen, Varun, Xiaoxi, Annie and Zhang Yi. I am especially glad to have Wen-Yang and XiaoHu's friendship during the two (often lonely) years of graduate school. Gabriel from ECE is a great friend and study mate for Math 556 Statistics.

I appreciate the friends I met during IBM Extreme Blue's internship, especially Cate, Malcolm, Dave, Xiaoyuan, Maggie and Nikolina.

Lastly, a thank you to some other friends whom I have met during my two years in Canada: Laura, Reiko, Yuka, Elaine, Keiko, Isa, Hitomi, Hatty, Modi, Clement, Audrey, Nicollette, Jocylin, Donnub, Esther, Elena, WangZi, Angeline, Jenny, Liz, Lennard, MinNing, Regina, Kenneth, Jolene, ShiDa, Ada, XueYing, Johnathon, Guanyi, Melissa, Petrina, Yuan Yee, and Jiamin. I know I might not see some of you ever again, but I am glad that we have met at some point in my life.

ABSTRACT

Spoken dialogue systems are gaining popularity with improvements in speech recognition technologies. Dialogue systems have been modeled effectively using Partially observable Markov decision processes (POMDPs), achieving improvements in robustness. However, past research on POMDP-based dialogue systems usually assumes that the model parameters are known. This limitation can be addressed through model-based Bayesian reinforcement learning, which offers a rich framework for simultaneous learning and planning. However, due to the high complexity of the framework, a major challenge is to scale up these algorithms for complex dialogue systems. In this work, we show that by exploiting certain known components of the system, such as knowledge of symmetrical properties, and using an approximate online planning algorithm, we are able to apply Bayesian RL on several realistic spoken dialogue system domains. We consider several experimental domains. First, a small synthetic data case, where we illustrate several properties of the approach. Second, a small dialogue manager based on the SACTI1 corpus which contains 144 dialogues between 36 users and 12 experts. Third, a dialogue manager aimed at patients with dementia, to assist them with activities of daily living. Finally, we consider a large dialogue manager designed to help patients to operate a wheelchair.

ABRÉGÉ

Les systèmes de dialogues sont de plus en plus populaires depuis l'amélioration des technologies de reconnaissance vocale. Ces systèmes de dialogues peuvent être modélisés efficacement à l'aide des processus de décision markoviens partiellement observables (POMDP). Toutefois, les recherches antérieures supposent généralement une connaissance des paramètres du modèle. L'apprentissage par renforcement basé sur un modèle bayésien, qui offre un cadre riche pour l'apprentissage et la planification simultanée, peut éliminer la nécessité de cette supposition. À cause de la grande complexité du cadre, le développement de ces algorithmes pour les systèmes de dialogues complexes représente un défi majeur. Dans ce document, nous démontrons qu'en exploitant certaines propriétés connues du système, comme les symétries, et en utilisant un algorithme de planification approximatif en ligne, nous sommes capables d'appliquer les techniques d'apprentissage par renforcement bayésien dans le cadre de sur plusieurs domaines de dialogues réalistes. Nous considérons quelques domaines expérimentaux. Le premier comprend des données synthétiques qui servent à illustrer plusieurs propriétés de notre approche. Le deuxième est un gestionnaire de dialogues basé sur le corpus SACTI1 qui contient 144 dialogues entre 36 utilisateurs et 12 experts. Le troisième gestionnaire aide les patients atteints de démence à vivre au quotidien. Finalement, nous considérons un grand gestionnaire de dialogue qui assise des patients à manoeuvrer une chaise roulante automatisée.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
Introduction

Dialogue systems have been modeled effectively using Partially Observable Markov decision processes (POMDPs), but past research on POMDP-based dialogue systems usually assumes that the model parameters are known. Model-based Bayesian reinforcement learning can overcome this limitation, but it is very complex and does not scale up on large domains. The goal of the thesis is to apply bayesian reinforcement learning on several realistic spoken dialogue system domains.

In this chapter, we give a brief overview of dialogue systems, reinforcement learning, our contributions, and the outline of this thesis.

1.1 Background on Dialogue Systems

A dialogue system is a computer system intended to converse with humans. It usually has a coherent structure in order to make meaningful conversations with humans. The dialogue manager is the central component of a spoken dialogue system. It interacts with external knowledge sources, producing messages to be output to the user, and controls the dialogue flow. For communicating between the end-users and the dialogue manager, dialogue systems usually make use of various forms of communications including text, speech, graphics, and gestures. A spoken dialogue system is a dialogue system that allows users to operate with their voice. It has two components that do not exist in a text dialogue system: a speech recognizer and an optional text-to-speech module.

Spoken dialogue systems are becoming increasingly popular with progressive improvements in speech recognition technologies. They have become more ubiquitous in recent years, and such trends will continue as technological improvements increase their capabilities and opportunities for deployment. Spoken dialogue systems have been applied in various domains such as an in-car spoken dialogue system [17], an automated receptionist [28], and a robotics wheelchair [4].

Participating in a spoken conversation with a dialogue systems is, however, very challenging. First, automatic speech recognition (ASR) technology is known to be error-prone, so a computer has to assume a high level of inaccuracy regardless of how powerful the input devices are. Consequently, the dialogue system often has to make a guess, or else it has to delay responding until the humans' communcations can be clarified. Second, problems often arise because human behaviour is difficult to predict. For example, humans may change the topic in the middle of a conversation. Third, a conversation is a temporal process. The speakers often behave in non-deterministic manners, and their actions have long-term effects in addition to the immediate effects. Due to the above reasons, using a computer to understand human languages is a challenging problem.

In the past few years, it has been shown that spoken dialogue systems can be modeled as Partially Observable Markov Decision Processes (POMDPs) [39, 51], a principled framework for decision-making under uncertainty. The POMDP framework is general enough for modeling a wide variety of real-world sequential decision processes. The POMDP framework originated in the Operations Research community, but it was later introduced to the Artificial Intelligence and Automated Planning

communities. Applications include robot navigation problems, patient management processes, and planning under uncertainty in general.

POMDPs have the following three properties that naturally model spoken dialogue systems [52]. First, a POMDP assumes that we do not know with certainty the state of its world. We simply assume the presence of sensors that provide some incomplete and inaccurate knowledge of the current state. In spoken dialogue systems, the state corresponds to the intention of the user. We do not know with perfect accuracy what the user wants, but his speech gives us some indications of his intention. Second, actions in a POMDPs model have stochastic effects on the world, but the precise effects of the actions cannot be predicted exactly. Similarly, humans often change their intentions during a conversation and may change the topic halfway in the middle of a conversation. Finally, there exists a reward function in POMDPs that assigns real-valued rewards for taking certain actions (in certain states). The goal is to choose a sequence of actions to maximize the cumulative rewards obtained. Correspondingly, in spoken dialogue systems, we can give our agent a reward for taking an action that actually corresponds to the human's request, and penalize the agent otherwise. Therefore, using a POMDPs framework helps to incorporate uncertainty in the dialogue system and allows actions to be chosen based on an optimization criteria.

However, past research in POMDP-based dialogue system has usually assumed a fixed and known POMDP model [39, 52], which is unrealistic in many applications. Due to several factors, it is often difficult to characterize the noise in the speech recognition system. For example, the reliability of the voice recognition device, how

fluent the speaker is in the language, and even the accent of the speaker affect the voice recognition accuracy. Therefore, in order to use a fixed and known model, we require a training phase for the user due to speech variations between individuals. Without such a training phase, we cannot model dialogue systems as POMDPs accurately. In practice, such a training phase is a laborious and manual process. As modern domains become more complex, it is increasingly difficult to model them by hand.

It is beneficial if our agent can develop its own model of the world. If we can improve the current techniques to remove the need for such a training phase, we are able to accomodate variations and errors in the input device. The input channel in various dialogue systems varies from a text-based input, voice recognition device, graphical-user interface, to a multi-modal device. Sometimes, as the input channel is noisy, the user has to undergo a training phase before actually using the device. This is necessary since we need to know the all the input parameters in advance in order to specify the POMDP model.

In this thesis, we propose a framework for simultaneous learning and decision making on several dialogue systems domains. We present tractable algorithms for applying this framework in real-world domains. Our framework allows dialogue systems to be modeled as a POMDP even when some of the parameters are not known exactly.

1.2 Reinforcement Learning

Reinforcement Learning (RL) models have proven to be effective for learning in general, but most RL methods do not explicitly minimize the learning cost. The

cost of learning is incurred when the system selects specific actions with the intent to learn its parameters, as opposed to accomplish the dialogue task. This is particularly important in spoken dialogue systems, where acquiring data is expensive. Thus, it is important to develop methods for efficient low-cost learning.

Bayesian RL maintains a posterior distribution over all possible model parameters, and computes an action selection policy that is optimal with respect to this posterior [11, 35]. This enables us to make use of prior knowledge to learn the parameters more efficiently, and thus with a lower learning cost. Model-based Bayesian RL methods are typically computationally intensive, thus applications are still limited to small domains.

This thesis focuses on how we can obtain good decisions from the spoken dialogue model despite inaccurate initial model parameters. This is achieved by modeling the dialogue system as a Bayes-Adaptive POMDP (BAPOMDP) model [37], and exploiting certain known symmetrical properties of the system to obtain scalable solutions.

1.3 Contributions

The primary contribution of this thesis is in applying Bayesian Reinforcement Learning techniques to realistic POMDP-based dialogue systems. To achieve this, we make enhancements to an algorithm based on the existing BAPOMDP framework in three ways. First, we make use of known symmetry in the model to perform parameters tying. Second, we incorporate a branch-and-bound technique called Real Time Belief State Search (RTBSS). Last, we make an improvement to the way sampling is done in BAPOMDPs. We hypothesis that if we have prior knowledge of certain

symmetrical properties in the model(such as similarity in the speech commands) in a spoken dialogue system, and by using the RTBSS technique, we can scale up BAPOMDPs to larger domains.

Our results show that with this new improved framework, we can deploy Bayesian Reinforcement Learning techniques on the following 4 experimental domains:

- First, a small synthetic data case, where we illustrate several properties of the approach.
- Second, a small dialogue manager based on the SACTI1 corpus which contains 144 dialogues between 36 users and 12 experts.
- Third, a dialogue manager aimed at patients with dementia, to assist them with activities of daily living.
- Finally, we consider a large dialogue manager designed to help patients to operate a wheelchair.

The primary work in this thesis had been accepted for presentation at ICASSP (International Conference on Acoustics, Speech and Signal Processing) 2011. The title of the paper is *"Bayesian Reinforcement Learning for POMDP-Based dialogue Systems"* [34]. The contents of this paper includes mostly the methods of Chapter 4 and the results of Section 5.5.

1.4 Outline

We now present the overall organization structure of the thesis. In Chapter 2, we introduce the details of dialogue systems. In Chapter 3, we provide formal descriptions of the MDP and POMDP decision making frameworks. We also introduce concepts such as belief states and value functions. In Chapter 4, we describe how

BAPOMDPs can be used to model dialogue systems. We explain the concept of parameter tying, and introduce RTBSS in our online search. In Chapter 5, we present an empirical evaluation of our methods. Lastly, we conclude in Chapter 6.

CHAPTER 2
Spoken Dialogue Systems

In this chapter, we give a brief overview of spoken dialogue systems, their architecture, challenges in applying reinforcement learning to spoken dialogue systems, and example applications of spoken dialogue systems. We also introduce the SmartWheeler domain, which is one of the domains used in our empirical work.

2.1 Introduction

A spoken dialogue system is a computer system intended to converse with a human using spoken language as the medium for interaction. It often allows a human user to access useful information and services on a computer, or over the Internet.

Spoken dialogue systems are usually task-oriented. Task-oriented systems make use of dialogue to accomplish a particular task. This task can be relatively simple and well-defined, such as making a hotel booking, making a flight reservation, or directing a user's phone call to the right department. It can also be more complex, involving planning and reasoning, such as planning a family holiday, negotiating a contract, or managing a disaster scenario.

2.2 An example of a dialogue conversation

In [10], experiments were conducted on an integrated spoken man-machine dialogue system which allowed the access to a remote database using a telephone. This prototype was partially developed under the Sundial Esprit Project. The application domain consists of the Italian train time-table information. The first trial was

Table 2–1: A dialogue conversation between a human user U communicating with a dialogue system to purchase a train ticket S [10]

S1: Hello, This is Train Enquiry Service. Please speak after the tone. Which information do you need?
U1: I want to go from Torino to Milano.
S2: Do you want to go from Trento to Milano? Yes or No?
U2: No.
S3: Do you want to leave from Trento? Yes or No?
U3: No.
S4: Where do you want to leave from?
U4: I want to leave from Torino
S5: Do you want to leave from Torino? Yes or No?
U5: Yes.
S6: Do you want to go to Milano? Yes or No?
U6: Yes.
S7: At which time do you want to leave?
U7: I want to travel in the evening.
S8: Do you want to leave between 6 and 9 p.m.? Yes or No?
U8: Yes.
S9: There is a train from Torino Porta Nuova to Milano Centrale. It leaves at 7.10 p.m. and arrives at 8.55 p.m.. Do you want more detailed information about it?
U9: Is that an intercity train and how much does the ticket cost?

carried out in March 1993 and the second one in May 1993. Table 2.2 shows one typical example of a dialogue conversation. The dialogue begins with an open-ended question and continues with questions to obtain further information from the user.

2.3 Architecture of a Spoken Dialogue System

The main components of a typical Spoken Dialogue System (SDS) are illustrated in Figure 2–1 [54].

The main decision-making module is contained in the dialogue manager. Its role is to exchange information with the user. It also accesses and updates the information in the database. User interaction consists of a sequence of question-answer cycles. During each cycle, a question is frequently designed to obtain some specific information. The user's response is then processed by a speech recognizer,

and the output is converted by an interpreter into a semantic representation. Based on this new input, the dialogue manager updates its internal state and representation. The manager then plans the next action. This repeats until the user's information need is satisfied, at which point the interaction is terminated.

Some of the key issues in designing an SDS include how to specify and control the dialogue flow, how to constrain the recognizer to work within a limited domain, how to interpret the recognition output, and how to generate responses to the user that are contextually appropriate.

Design criterions for motivating the solutions to these issues are often numerous and varied, but the main top-level goal is to create a system that allows a user to complete their required tasks quickly and accurately.

2.4 Reinforcement Learning in Dialogue Systems

Reinforcement learning (RL) is introduced in Chapter 3. Here, we give an brief overview of how RL is used in dialogue systems without any formal mathematical specifications. The main goal of RL is to allow the agent to learn its behaviour based on feedback from the environment. As the agent learns further, it modifies its behaviour. A simple feedback is necessary for the agent to learn its behaviour; this is known as the reinforcement signal.

The main objective of a dialogue system is to guess correctly what the user wants to achieve, and completes the task. The user first specifies what he wants to achieve. Then, the input recognizer of the dialogue system interprets the user's intention. This is usually not fully accurate. In the case of a spoken dialogue system, the user says what he wants to do, and the speech recognizer parses the voice input

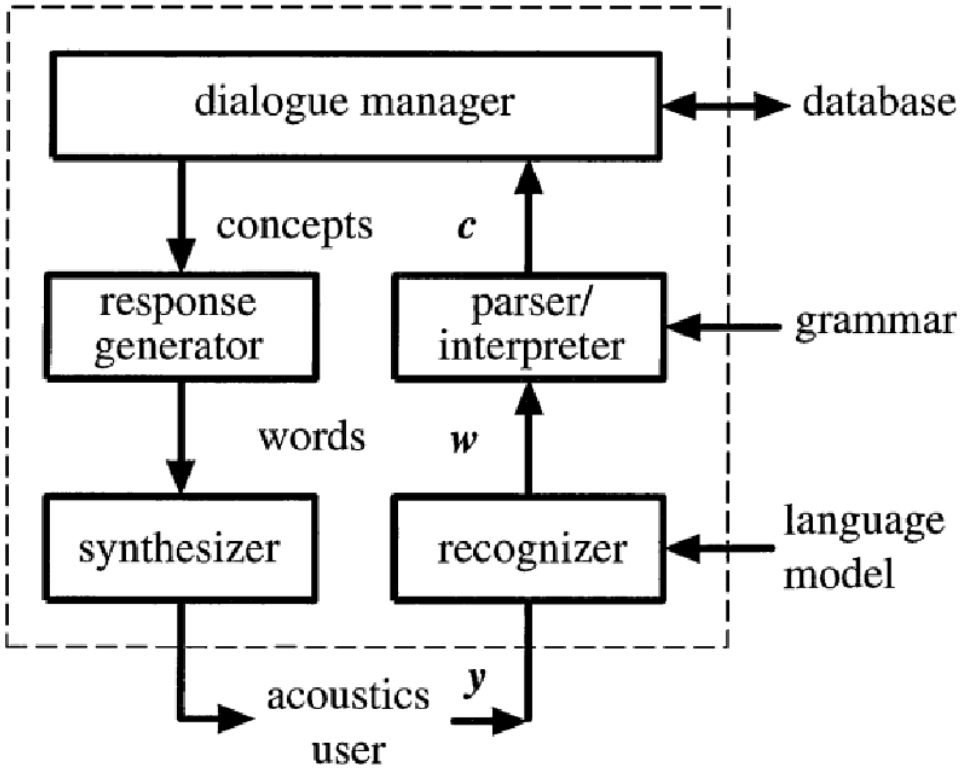


Figure 2–1: Dialogue Architecture

into sentences. Next, internal processing takes place in the dialogue manager. Lastly, the dialog manager produces an output.

There are various approaches for applying reinforcement learning in dialogue systems [40]. The simplest approach is to build a full model of the environment with the available training data. Our simulation model in the dialogue systems consists of probability distributions. We have to convert our simulation model into a model of state-to-state transition probabilities. For example, these could be the probabilities of the user changing his intention. Given some training data (including the logs

of the state transitions), we can then estimate the transition probabilities by using a simple maximum likelihood estimation that is based on the relative frequency of occurence of each transition.

In reinforcement learning, the simple feedback comes from the reward the agent receives by taking a given action in a given state. However, this reward function is only the immediate return of executing an action. What we want to achieve is a function that captures the long-term consequences, and we call this the utility function. The utility function of taking an action in some state is the sum of the immediate reward for that action and the long-term rewards over the rest of the agent's lifetime, assuming the agent adopts a good strategy.

In reinforcement learning, we adopt an iterative approach to estimate the utility function. We begin with some initial guess of the utility function. Through our experience with the dialogue system, we improve that guess with more iterations. To gather more experience, we usually use a simulated dialogue system. We collect state-action trajectories with corresponding rewards. In each step, the agent takes an action and receives a reward. We then update the estimate of the utility function with this reward. As we have more trajectories, the difference between our estimated utility function and the true utility function becomes smaller.

However, practically, it is a big technical challenge to apply RL algorithms to real dialogue systems.

1. One difficulty is that we usually do not have sufficient data. For any practical system, we need a large pool of training data. However, the available training

data is simply insufficient for us to make any accurate estimations of the transition probabilities. The state space is too large, so much more training date would be required to explore the entire state space. Therefore, representing the dialogue state by the entire dialogue is neither feasible nor conceptually useful. The estimates are unreliable, and such data collection is not scalable. Techniques such as state-aggregation and back-off schemes for parameter estimation have been proposed in [18, 12] to deal with this problem. However, these are bascially just smoothing methods and they offer no guarantee that the learned strategy is robust.

2. Since the training data is fixed, we can only learn from all the existing state-action combinations explored at the time of the training data collection. This means that it is almost impossible for us to fully explore the state space. At the same time, we could not try out completely new strategies, since no transition probabilities can be computed for the unseen state-action combinations. Furthermore, the optimal strategy is not guaranteed to be present in the available training data collection. If this happens, no matter how good our algorithms, we are unlikely to learn the optimal policy. It is also risky for us to try an unexplored state-action combination, because we do not know the consequences. For example, if a dialogue system goes live, and it is important that it stays online, any slight deviation of the user's behaviour might move the system into an unexplored state space. The system could fail and go offline.

3. Another drawback is that we have to know in advance how to represent the dialogue model. We usually do not know which features are required to represent the state space. In order for us to label the training data correctly, we have to identify the state space and action set representation before estimating the state transition probabilities. Furthermore, the state space and action set should also be fixed even before the data collection, so that we can properly record the state transitions in a log during the collection. Unfortunately, if the chosen state space representation or action set turns out to be incorrect, it is very tedious to change the dataset or its transcriptions.
4. We also do not know how to choose the reward functions even for similar dialogue systems. The reward funtions are usually designed by system designers with domain knowledge. It is not clear what is the best way to model the reward functions.

Previously, more research has been done in modeling dialogue systems as Markov decision processes (MDPs) [25, 49]. MDPs are explained later in Section 3.2. The purpose is for designing a dialogue system that takes actions so as to maximize its reward. We can thus obtain a good action-selection policy by applying the MDP framework and algorithms in reinforcement learning.

2.5 Applications of Spoken Dialogue Systems

There have been several applications of spoken dialogue systems. Elvis (EmaiL Voice Interactive System) was developed [48] to support access to email over the phone. Experiments were conducted for training an optimal dialogue strategy on a corpus of 219 dialogues in which human users interact with Elvis over the phone.

Table 2–2: An example dialogue with Elvis

A: Hi, Elvis here. You have 5 new and 0 unread messages in your inbox. Say Read, Summarize, or say Help for more options.
U: Read.
A: Select by Sender, Subject or in order of arrival?
U: Sender,
A: Which Sender?
U: Kim.
A: In the messages from Kim, there's 1 message about "Interviewing Antonio".....

Elvis does its training with a combination of Q-learning and a PARADISE evaluation framework [50]. In PARADISE, a reward function is designed based on user satisfaction studies done in the domain. User studies are necessary because in most applications, it is not obvious what is the best choice of reward function. The strategy was then tested on a corpus of 18 dialogues with 6 new users. It was demonstrated that Elvis can learn to optimize its strategy selection for agent initiative, for reading messages, and for summarizing email folders.

NJFun [43, 26] is another experimental spoken dialogue system that provides users with access to information about the fun things that one can do in New Jersey via a telephone conversation. NJFun used a practical methodology to build a dialogue system that optimizes its policy from human-computer training dialogue data. Its methodology is as follows:

1. Choose a suitable reward measure and a representation for dialogue states. Design a policy which maps every state to a set of reasonable actions.
2. Build an initial state-based training system that creates an exploratory data set. This system should have at least the basic functionality.
3. With these training data, build a MDP model on the state space.

Table 2–3: A dialogue conversation with NJFun

S1: Welcome to NJFun. How may I help you?
U1: I'd like to find um winetasting in Lambertville in the morning.
(ASR: I'd like to find out wineries in the Lambertville in the morning.)
S2: Did you say that you are interested in Lambertville?
U2: Yes.
S3: Did you say you want to go in the morning?
U3: Yes.
S4: I found a winery near Lambertville that is open in the morning. It is the Poor Richard's Winery in Lambertville. Thank you for using the system Please give me feedback by saying 'good', 'so-so', or 'bad'.
U4: Good
S5: Goodbye and have a nice day.

4. Compute the optimal dialogue policy according to this MDP with reinforcement learning.
5. Redesign the system using the newest optimal dialogue policy.

For the testing phase, 21 test subjects then performed the same 6 tasks used during training, resulting in 124 complete test dialogues.

2.6 SmartWheeler Autonomous Wheelchair

Finally, we present an overview of the SmartWheeler autonomous wheelchair project. The goal of the SmartWheeler project project is to increase the autonomy and safety of individuals with severe mobility impairments by developing a robotic wheelchair that is adapted to their needs.

Controlling a robot such as this requires a unique interface, in this case an intelligent interaction interface. This interface acts as the layer between the user and the lower-level robot control, taking sensory information, and determining an appropriate action.

Figure 2–2: SmartWheeler

This interface provides a challenging environment for demonstrating the dialogue learning framework on a deployed system. We have to make a decision given that the input is noisy, and we do not know the user's true intention. Models for such a complex tasks may vary from user to user, and would be difficult to specify manually.

2.6.1 Dialogue Conversations in the SmartWheeler domain

This section presents two examples of an interaction betweeen the user and the intelligent interaction interface. The examples present the flow and format of information during the interaction. The transcribed audio can be seen in lines labelled "AllPhrases". This is how the system interprets the user's input, and may contain errors. The ground truth transcription as determined by the researchers is labelled as "Actual", and it describes what the user really wants. This information is not available to the system but is presented here for clarification. Phrases that are successfully parsed by the semantic grammar have SentenceRecognized set to true. The action as selected by the interaction manager is presented as "Selected Action". Table 2–4 demonstrates a case where the agent requests additional information via a specific query. In this case, the agent determines that "move backward" does not uniquely identify an intention, and thus requests move information. The user repeats "backward" which helps to isolate the command. It is not clear at the first glance that clarification is required. However, there are actually several commands which are simple variations of "move backward". Based on the sensitivity of the system and the preferences of the user, the agent may or may not request the clarification. These are characteristics controlled by the parameters of the model representation. In this instance, the system is more cautious and tends to query more often.

In this second example found in Table 2–5, the speech recognition system has significant error in the recognition. However, the agent is still able to determine the correct action. This is likely due to the word "wall" which, in our current system, can only be found in the action phrase"Align to wall".

Table 2–4: Example of a dialogue conversation involving a query action in SmartWheeler

Actual: “Move backward”
<SentenceRecognized value=“true” / >
<Phrase value=“move backward” / >
<AllPhrases value=“[move backward]” / >
<Action value=“move” type=“move-action” / >
<SpeechAct value=“command” / >
<Direction value=“backward” type=“vector-direction” / >

Selected Action: Move Query

Actual: “Backward”
<SentenceRecognized value=“false” / >
<Phrase value=“backward” / >
<AllPhrases value=“[backward]” / >

Selected Action: Move Backward

Table 2–5: Example of a dialogue conversation with misrecognition in SmartWheeler

Actual: “Let’s move towards the wall and end up parallel to the wall”
<SentenceRecognized value=“false” / >
<Phrase value=“” / >
<AllPhrases value=“[the move to wall to wall and and and upveer overcome wall on]” / >
Selected Action: Align to Wall

In Section 5.5, we introduce a POMDP model that has 25 actions such as "align to wall" and 4 "query" actions. This POMDP problem models the Interaction Manager in SmartWheeler. Each wheelchair function such as "move forward one meter", "turn the controller on", or "turn right ninety degrees" represents a user's intention and defines the state space. Further details about the SmartWheeler can be found in [4].

2.7 Discussion

With progressive improvements in speech recognition technologies, spoken dialogue systems have become more ubiquitous in recent years. Such trends will continue because technological improvements increase their capabilities and opportunities for deployment. Spoken dialogue systems have been applied in various domains such as an in-car spoken dialogue system [17], an automated receptionist [28], and a robotics wheelchair [4].

However, the applications of spoken dialogue systems still face many challenges. First, automatic speech recognition (ASR) technology is known to be error-prone, so a computer has to assume a high level of inaccuracy regardless of how powerful the input devices are. Consequently, the dialogue system often has to make a guess, or else it has to delay responding until the humans' communcations can be clarified. Second, problems often arise because human behaviour is difficult to predict. For example, humans may change their speech topic in the middle of a conversation. Third, a conversation is a temporal process. The speakers often behave in non-deterministic manners, and their actions have long-term effects in addition to the

immediate effects. Due to the above reasons, trying to understand human languages is a challenging problem.

Reinforcment learning has been applied to spoken dialogue systems, but there are many difficulties as well. First, we often do not have sufficient data to fully explore the state space. Second, the optimal strategy is not guaranteed to be present in the available training data collection. Third, we to know in advance how to represent the dialogue model before learning. Last, choosing the reward functions even for similar dialogue systems is not easy.

CHAPTER 3
Technical Background

In this chapter, we present the specifications of two useful frameworks in the decision-making community: Markov Decision Processes (MDPs) and Partially Observable Markov Decision Processes (POMDPs). We explain the algorithms for decision-making using these models, including approximate methods for determining a policy. We also discuss how spoken dialogue systems are typically modeled with POMDPs. Finally, we present bayesian reinforcement learning (RL).

3.1 Introduction

Markov Decision Processes (MDPs) are the basis for solving probabilistic models in decision making [5, 36]. MDPs provide a mathematical framework for modeling decision-making in situations where outcomes are partly random, and partly under the control of a decision maker. According to this framework, agents move stochastically between states by executing actions and then receiving a reward. The agent can perform robust decision-making in light of uncertainy from the transitions. Markov decision processes are an extension of Markov chains; the difference is the addition of actions (allowing choice) and rewards (giving motivation). Conversely, if only one action exists for each state and all rewards are zero, a Markov decision process reduces to a Markov chain.

The MDP framework assumes that the state is known when an action is to be taken. When this assumption is not true, the problem is called Partially Observable Markov Decision Process (POMDP). POMDPs extend MDPs by assuming the world is partially observable, whereas MDPs assume the state of the world is always known. In an MDP, the resulting state is observed after a stochastic action is executed. POMDPs reduce this constraint and assume the underlying state must be inferred using observations. While POMDPs provide a much more powerful model, the increase in model complexity results in more difficult decision-making.

3.2 MDPs

A Markov decision process can be described as a tuple $< S, A, T, R >$ where

- S is a finite set of states of the world,
- A is a finite set of actions,
- The state transition function is

$$T(s, a, s') := Pr(s_t = s' | s_{t-1} = s, a_{t-1} = a), \tag{3.1}$$

  giving for each world state and agent action, a probability distribution over world states. This is the probability of ending in state s, given that the agent starts in state s and takes action a

- The reward function is

$$R(s, a) : S * A \rightarrow \mathbb{R}, \tag{3.2}$$

  giving the expected immediate reward gained by the agent for taking each action in each state. This is the the expected reward for taking action a in state s.

In an MDP, the next state and the expected reward depend only on the previous state and the action taken. Even if we were to condition on additional previous states, the transition probabilities and the expected rewards would remain the same. This is known as the Markov property: the state and reward at time $t+1$ depends only on the state at time t and the action at time t. A process is said to be *Markovian* if the future states are conditionally dependant only on the current state.

We would like our agent to act in such a way as to maximize the long-run reward received given an infinite lifetime. The most straightforward approach to achieve is to sum rewards over the infinite lifetime of the agent, but discount them geometrically using a discount factor $0 < \gamma < 1$; the agent should act so as to optimize

$$\sum_{t=0}^{\infty} \gamma^t r_t. \tag{3.3}$$

In this model, rewards received earlier in its lifetime have more value to the agent. Even though the infinite lifetime is considered, the discount factor γ ensures that the sum is finite. This sum is also the expected amount of reward received if a decision to terminate the run is made on each step with probability 1. Future rewards have more effects on current decision-making if the discount factor is larger (closer to 1).

A policy is a description of the behavior of an agent. There are two kinds of policies: stationary and nonstationary. We can consider a stationary policy. A stationary policy π is any mapping

$$\pi : S \to A, \tag{3.4}$$

that specifies, for each state, an action to be taken. The choice of action depends only on the state and is independent of the time step. In the infinite-horizon discounted model, the agent always has a constant expected amount of time remaining, so there is no reason to change action strategies. In other words, there is a stationary optimal policy.

The value function of a policy π is a map $V^{\pi} : S \rightarrow R$, so that $V_{\pi}(s)$ gives the expected discounted sum of rewards for executing π starting from state s.

In the infinite-horizon discounted case, for any initial state s, we want to execute the policy π that maximizes $V_{\pi}(s)$. Howard [20] showed that there exists a stationary policy, π^*, that is optimal for every starting state. The value function for this policy, V_{π}^*, also as written V^* , is defined by the set of equations

$$V^*(s) = max_{a \in A}[R(s,a) + \gamma \sum_{s' \in S} T(s,a,s')V^*(s')], \tag{3.5}$$

which has a unique solution. This is known as the Bellman equation and acts as the basis for many MDP solution techniques.

An optimal policy, π^* is just a greedy policy with respect to V^*

$$\pi^*(s) = argmax_{a \in A}[R(s,a) + \gamma \sum_{s' \in S} T(s,a,s')V^*(s')]. \tag{3.6}$$

3.3 POMDPs

MDPs assume that the environment is fully observable. Even though the agent does not know the outcome of executing an action beforehand, the agent has knowledge of the resulting state after execution. While this assumption may be valid in

certain domains, many agents only have limited or incomplete information about the environment. In this case, we call it a partially observable domain

The POMDP provide a principled mathematical framework for modeling non deterministic, sequential decision making problems [46, 22]. It models taking a sequence of actions under uncertainty to maximize its total reward. Formally, a discrete POMDP is specified as a tuple $(S, A, Z, T, O, R, \gamma)$, where S is a set of states, A is a set of actions, and Z is a set of observations.

At each time step, the agent lies in some state $s \in S$. It takes an action $a \in A$ and moves from s to a new state s'. Due to the uncertainty in action effect, the end state s' is modeled as a conditional probability function $T(s, a, s') = p(s'|s, a)$ for $s \in S$ and $a \in A$,

$$T(s, a, s') := Pr(s_t = s'|s_{t-1} = s, a_{t-1} = a). \tag{3.7}$$

The agent then makes an observation to gather information on its end state s'. Again, due to uncertainty in the observation process, the observation result $z \in Z$ is modeled as a conditional probability function $O(s', a, z) = p(z|s', a)$ for $s' \in S$ and $a \in A$,

$$O(s', a, z) := Pr(z_t = z|s_t = s', a_t = a). \tag{3.8}$$

In order to control the desirable agent's behaviour, a reward function $R(s, a)$ is defined. At each time step, the agent receives a reward $R(s, a)$ if it takes action a in state s. The agent's goal is to maximize its expected total reward by choosing a suitable sequence of actions. For infinite-horizon POMDPs, the sequence of actions has infinite length. We specify a discount factor, $\gamma \in [0, 1)$ so that the total reward

is finite, and the problem is well defined. The expected discounted total reward is

$$E[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)], \tag{3.9}$$

where s_t and a_t denote the agent's state and action at time t, respectively.

3.3.1 Belief State Tracking

Since the system is partially observable, the agent does not generally observe the true state of the world. Instead, a history maintains a trace of the agent's actions and observations over time. The history at time t is defined as:

$$h_t = a_0, z_1, a_1, z_2..., a_{t-1}, z_t. \tag{3.10}$$

Maintaining an extensive history quickly becomes intractable. Instead, a distribution, known as a belief state, is maintained to summarize its previous experience. The choice for belief state is the probability distributions over the states of the world, S. These distributions encode the agent's subjective probability about the current state of the world, and they provide a basis for acting with incomplete information. Furthermore, they comprise a *sufficient statistic* for the past history and initial belief state of the agent. This means that given the agent's current belief state, no additional data about its past actions or observations would give any more information about the current state of the world [2, 44]. Also, the process over belief states is $Markovian$, meaning that no additional data about the past would help to increase the agent's expected reward.

We let $b(s)$ denote the probability assigned to world state s by belief state b. Solving a POMDP consists of two steps executed iteratively. The first step is action

selection. If the agent's current belief is b, it finds the action a that maximizes the future reward. The second step is belief estimation. After the agent takes an action a and receives an observation o, its new belief b' is given by

$$\begin{aligned} b'(s') &= Pr(s'|z,a,b) && (3.11)\\ &= \frac{Pr(z|s',a,b)Pr(s'|a,b)}{Pr(z|a,b)} && (3.12)\\ &= \frac{Pr(z|s',a)\sum_{s\in S} Pr(s'|a,b,s)Pr(s|a,b)}{Pr(z|a,b)} && (3.13)\\ &= \frac{O(s',a,z)\sum_{s\in S} T(s,a,s')b(s)}{Pr(z|a,b)}. && (3.14) \end{aligned}$$

The denominator, $Pr(z|a,b)$, can be treated as a normalizing factor, independent of s, that causes b to sum to 1. The process then repeats.

We can also write the equation as

$$b'(s) = \eta O(s',a,z)\sum_{s\in S} T(s,a,s')b(s), \tag{3.15}$$

where η is a normalizing constant.

3.3.2 Solving POMDPs

The POMDP model has to go through a planning phase. During this phase, it finds a policy that describes a mapping of action to belief for all possible beliefs. A POMDP policy,

$$\pi : b \rightarrow A, \tag{3.16}$$

maps a belief $b \in B$, which is a probability distribution over S, to an action $a \in A$.

The optimal policy for a POMDP is one that chooses an action that maximizes the expected future discounted cumulative reward:

$$\pi^*(b_{t_o}) = argmax_\pi E_\pi[\sum_{t=0}^{T} \gamma^t r_t | b_{t_o}]. \tag{3.17}$$

The value function of a policy is the expected discounted sum of rewards for executing π starting from belief b_o,

$$V^*(b) = E_\pi[\sum_{t=0}^{T} \gamma^t r_t | b_o], \tag{3.18}$$

where r_t is the reward received at timestep t, γ is the discount factor.

Computing the optimal policy is difficult. There are two main obstacles. The first obstacle is the *curse of dimensonality*. The belief space, which allows us to reason about the agent's uncertainty, is actually the source of this difficulty. In a problem with n phyiscal states, the corresponding belief space then has dimensionality equal to the number of states. An agent with 1000 states results in a belief space of 1000 dimensions. The second obstacle is the *curse of history*. Many planning tasks actually require the agent to take many actions before it can reaches its goal, resulting in a long time horizon for planning. The complexity of planning grows exponentially with the time horizon.

In the infinite-horizon discounted case, for any initial belief b, we want to execute the policy π that maximizes V^*, where it is defined by the set of equations:

$$V^*(b) = max_a[\sum_{s \in S} b(s)R(s,a) + \gamma \sum_{z \in Z} Pr(z|b,a)V^*(\tau(b,a,z))], \tag{3.19}$$

which has a unique solution. An optimal policy, π^*, is just a greedy policy with respect to $V^*(b)$:

$$\pi^*(b) = max_a[\sum_{s \in S} b(s)R(s,a) + \gamma \sum_{z \in Z} Pr(z|b,a)V^*(\tau(b,a,z))], \tag{3.20}$$

where $\tau(b,a,z)$ is the new belief found after taking action a in belief b and observing z. More details on the theory of POMDP and the exact solution to POMDP can be found in [22, 44].

3.3.3 Offline Planning in POMDPs

Finding an optimal policy exactly for non-trivial POMDPs problems is computationally intractable. A near-optimal policy can be computed significantly faster than an exact one. Some of the most popular offline approaches are the point-based algorithms [33, 45, 47, 24, 29, 14].

It is known that V^*, the value function for an optimal policy π^*, can be approximated arbitrarily closely by a piecewise-linear, convex function

$$V(b) = \max_{\alpha \in \gamma}(\alpha \cdot b), \tag{3.21}$$

where b is a discrete vector representation of a belief and γ is a finite set of vectors called α-vectors. Each α-vector is associated with an action, and the policy can be executed by selecting the action corresponding to the best α-vector at the current belief b. So a policy can be represented by a set Γ of α-vectors. Policy computation, which, in this case, involves the construction of γ, is usually performed offline.

Point-based algorithms have been highly successful in computing approximate solutions to POMDPs with a large number of states. The details of point-based

Algorithm 1 Point-based POMDP planning [14]

1: Insert the initial belief point b_0 as the root of the tree $T_{\mathcal{R}}$.
2: Initialize the set Γ of α-vectors.
3: **repeat**
4: Sample new belief points and insert them into $T_{\mathcal{R}}$ by repeatedly calling SAMPLE($T_{\mathcal{R}}$, Γ).
5: Choose a subset of nodes from $T_{\mathcal{R}}$. For each chosen node b, BACKUP($T_{\mathcal{R}}, \Gamma, b$).
6: PRUNE($T_{\mathcal{R}}$, Γ).
7: **until** termination conditions are satisfied.
8: **return** Γ.

SAMPLE($T_{\mathcal{R}}$, Γ)

10: Pick a node b from $T_{\mathcal{R}}$, $a \in A$, and $z \in Z$.
11: $b' \leftarrow \tau(b, a, z)$.
12: Insert b' into $T_{\mathcal{R}}$ as a child of b.

BACKUP($T_{\mathcal{R}}$, Γ, b)

13: For all $a \in A$, $z \in Z$, $\alpha_{a,z} \leftarrow \text{argmax}_{\alpha \in \Gamma}(\alpha \cdot \tau(b, a, z))$.
14: For all $a \in A, s \in S, \alpha_a(s) \leftarrow R(s, a) +$
$\gamma \sum_{z \in Z, s' \in S} T(s, a, s') O(s', a, z) \alpha_{a,z}(s')$.
15: $\alpha' \leftarrow \text{argmax}_{a \in A}(\alpha_a \cdot b)$
16: Insert α' into Γ.

algorithms are shown in Algorithm 1. They are based on value iteration, which is an iterative approach for computing the optimal value function V^*. Value iteration starts with an initial approximation to V^*, represented as a set Γ of α-vectors. It exploits the fact that V^* must satisfy the Bellman equation and performs backup operations on the approximation by iterating on the Bellman equation, until the iterations converge. The key idea behind the success of point-based algorithms is to sample a set of points from the belief space B and use it as an approximate representation of B, rather than represent B exactly. The various existing point-based algorithms differ mainly in how they sample B.

3.3.4 Online Planning in POMDPs

With offline approaches, the algorithm returns a policy defining which action to execute in every possible belief state. As this policy construction step takes significant time, such an approach is only applicable to small to medium-size domains. In large POMDPs, using offline approximate algorithms such as the blind policy [19], QMDP [27], and FIB [19] give a very rough value function approximation and tends to give weak performance in the resulting approximate policy. Even recently, offline point-based methods produced solutions that are not of very high quality in very large domains [31].

In large POMDPs, a potentially better alternative is to use an online approach, which only tries to find a good local policy for the current belief state of the agent. Online approaches reduce the complexity of the problem by planning online for only the current information state [38, 41, 6]. They consider only a small horizon of possible scenarios. The advantage of such approaches is that they only need to consider belief states that are reachable from the current belief state, which focuses computation on a small set of beliefs. Furthermore, since online planning is done at every step, it is sufficient to calculate only the maximal value for the current belief state, and not the full optimal α-vector.

The policy construction steps and the execution steps are interleaved with one another as shown in Figure 3–1. In some cases, online approaches may require a few extra execution steps (and online planning), since the policy is locally constructed and therefore not always optimal. However the policy construction time is often substantially shorter. Consequently, the overall time for the policy construction and

execution is usually less for online approaches [23]. In practice, a potential limitation of online planning is when we need to meet short real-time constraints. In such cases, the time available to construct the plan is very small compared to offline algorithms, and so the policy may be substantially worse.

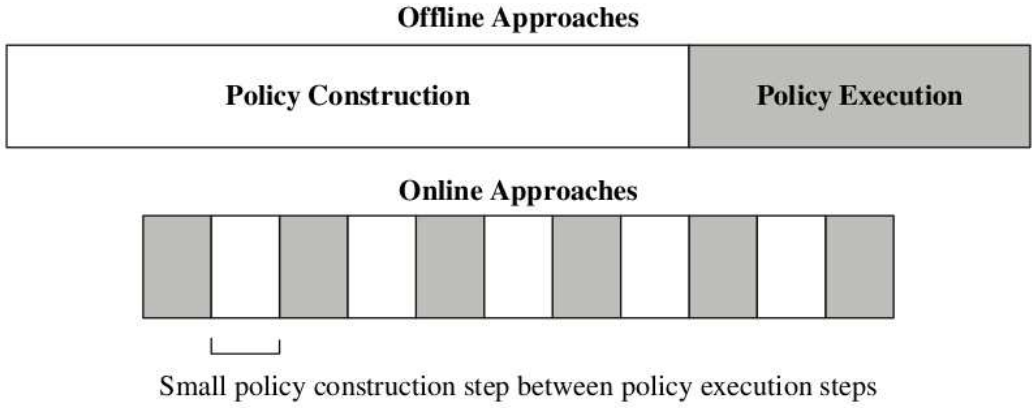


Figure 3–1: Comparison between offline and online approaches [32]

Online POMDP planners typically use forward search, from the current history or belief state, to form a local approximation to the optimal value function. The majority of online planners are based on point-based value iteration [33, 38]. These algorithms need to know the exact probability distributions. They construct a search tree of belief states. Each value in the search tree is updated by a full-width computation that takes account of all possible actions, observations and next states. An offline planning method can also be introduced to help obtain better results [38]. In this case, upper or lower bounds on the value function can be computed offline, and these bounds are propagated up the tree during search.

Recently, a Monte Carlo approach has been introduced in solving POMDP online [42]. Unlike most approaches that require explicit knowledge of the POMDP probability distributions, this approach requires only a black box simulator that provides a sample of a successor state, observation and reward, given a state and action. The algorithm works by combining a Monte-Carlo update of the agent's belief state with a Monte-Carlo tree search from the current belief state.

In the next section, we introduce Real Time Belief State Search (RTBSS), which is used in our improved BAPOMDPs framework for spoken dialogue systems.

3.3.5 RTBSS

In many practical systems, an exact online planning algorithm is not tractable. We can approximate the solution using Real Time Belief State Search (RTBSS) [32, 30]. RTBSS is a forward branch and bound search in the belief space. For practical systems with large states, the number of trees in the search space grows exponentially. To overcome this problem, RTBSS cuts down some redundant subtrees by pruning away some actions, thus reducing the time for the online search.

RTBSS does this as follows: We initially sort the actions according to those that are most likely to return higher values. This enables us to have pruning much earlier in the search. A simple way to perform this sort is by comparing the immediate reward of the action. Then, in any belief state, we check if it is possible to improve the maximum returned value. As we are only interested in the maximum value for all the possible actions, we can simply prune actions that are unlikely to improve the maximum returned value. Intuitively, if an action has a higher upper bound than the other actions, then it cannot be pruned by the other actions since their lower

bound will never exceed their upper bound. Another advantage of expanding actions in descending order of their upper bound is that as soon as we find an action that can be pruned, then we also know that all remaining actions can be pruned, since their upper bounds are necessarily lower.

The algorithm is described in Algorithm 2. In line 9, the upper bound is computed as the sum of the current reward and the heuristic value. The current reward is defined by:

$$R_B(b, a) = \sum_{s \in S} b(s) R(s, a). \tag{3.22}$$

This heuristic value has to be an upper bound. At line 10, if the upper bound value is less than the maximum value, the action is then pruned and the subtree is not expanded, reducing the time taken. Ideally, the heuristic value has to be the maximal value that any algorithm can find if it searches the tree to the maximal depth d. This ensures that we do not prune the wrong actions. When the depth is 0, we simply compute the immediate return of the belief state. This is defined by :

$$V_{immediate}(b) = max_{a \in A} R_B(b, a). \tag{3.23}$$

Further details about RTBSS can be found in [32, 30].

3.4 Bayesian Reinforcement Learning

The aim of Bayesian reinforcement learning (BRL) is to maintain a posterior distribution over possible models parameters and to compute an action selection policy which is optimal with respect to this posterior [15]. A general formulation for Bayesian RL is shown in Algorithm 3. Initially, distributions are initialized over the unknown parameters in the models. Using the current information about those

Algorithm 2 The RTBSS algorithm

```
1: Function RTBSS(b,d) returns the estimated value of b
   Inputs
   b: the current belief state
   d: the current depth
   Statics:
   D: the maximal search depth
   action: the best action
2: if d = 0 then
3:    return V_immediate(b)
4: end if
5: actionList ← Sort(b,A)
6: max ← -∞
7: for all a ∈ actionList do
8:    curReward ← R_B(b, a)
9:    uBound ← curReward + Heuristic(b,a,d)
10:   if uBound > max then
11:      for all z ∈ Z do
12:         curReward ← curReward + γPr(z|b,a)RTBSS(τ(b,a,z),d-1)
13:      end for
14:      if curReward > max then
15:         max ← curReward
16:         if d = D then
17:            action ← a
18:         end if
19:      end if
20:   end if
21: end for
22: return max
```

parameters, an action is selected. After the action is executed, the agent takes the resulting information about the environment and updates the distributions over the parameters.

Algorithm 3 A general framework describing Bayesian reinforcement learning. Various approaches address the challenges in each step through different techniques.

```
Initialize distributions over unknown parameters
Loop
    Select action based on distributions
    Execute action
    Observe resulting reward and observation
    Update posterior counts of unknown parameters based on observed information
end Loop
```

The current literature on model-based BRL mostly focuses on observables domains (MDPs). This is a considerable limitation as in many real-world problems, because the state of the system is only partially observable.

For Bayesian Reinforcement learning in finite MDPS [11, 15, 35], the key idea is to maintain counts $\phi^a_{ss'}$ of the number of times the transition $s \xrightarrow{a} s'$ is observed for each action a, starting from prior ϕ_0. These counts define the Dirichlet prior over transition T. Planning is then an MDP problem that has to take ϕ into account. The next state, s', becomes a combination of the physical state, $s \in S$, with the information state, ϕ. T' then describes a probability of update from $s, \phi \xrightarrow{a} (s', \phi')$.

In a finite POMDP, besides having the counts $\phi^a_{ss'}$, we also have ψ^a_{sz}, which are the counts of seeing z at s after doing action a. One of the main challenges in solving such a system is how to update the Dirichlet count parameters when the state is a hidden variable. In an infinite-state MDP, we have a known model, and the state is defined over (s, ϕ). At each time step, s is observable, and ϕ is updated. However,

in an infinite-state POMDP, we cannot observe s. The state is defined over (s, ϕ, ψ). At each time step, s is not observable, and neither are ϕ and ψ. One important challenge is to update the counters ϕ and ψ when we do not observe s.

The MEDUSA algorithm [21] was proposed for learning POMDP parameters in this framework. Just like other Bayesian Reinforcement Learning approaches for MDPs, a Dirichlet distribution is defined for each unknown transition probability. Similarly, distributions are also defined for each unknown observation probability. To address the issue of not observing s and being unable to update the counters ϕ and ψ, MEDUSA took advantage of the concept of an oracle. This approach is based on ideas from the active learning research area [1]. During execution, the agent can choose to query an oracle to know the true underlying state of the world, and then the oracle reveals (s, s'). Knowing this information, the Dirichlet parameters were updated just like the other MDP-based Bayesian RL approaches. In MEDUSA, the agent also samples a number of POMDP models according to the current Dirichlet distribution. The agent must decide when to query the oracle for the true identity of the hidden state, Following a query, it updates the Dirichlet parameters according to the result of the query. This process is repeated until the distribution over models is sufficiently well-known.

A bayesian reinforcement method for learning and adapting probabilistic models with the help of a human operator was proposed for the SmartWheeler project [4]. The goal is to develop a method to learn the user's preferences during execution, by observing the target behavior. It mixes both learning and execution, and it takes advantage of prior information about the world. It also assumes the presence

of an oracle, in this case, a human operator that returns an optimal action. This approach focuses on learning a reward model that results in the same behavior as that the target policy. Even though the reward functions are not necessarily the same, the end policy should be identical to the optimal policy. The assumption is that by observing a target action provided by an oracle, we can learn a class of utility functions which are consistent with the optimal behavior.

An approach for active learning in POMDPs was proposed in [13]. This approach makes use of two new techniques. First, an approximation is used which minimizes the immediate Bayes risk for choosing actions when transition, observation, and reward models are unknown. Second, to gather information about the model without assuming state observability, meta-queries are introduced. The meta-queries request policy information, not state information. For instance, if we have a robot which is unclear of the human's instructions, and it wants to clarify the instructions with the human, it is more natural for the robot to ask the human, "I think you want me to go to the coffee machine. Should I go there?" than "Please enter the position coordinates." The former models the real world more naturally. This approach is shown to learn good policies.

In the next section, we describe BAPOMDPs, which is our approach for applying bayesian reinforcement learning on spoken dialogue systems.

3.5 BAPOMDPs

The Bayes-Adaptive POMDP (BAPOMDP) framework allows learning and planning in POMDPs under parameter uncertainty [37]. Here, we assume that the state, action, observation spaces are finite and known. In this chapter, we focus on the case

where only $p(z|s,a)$ is unknown, as this is most relevant for practical dialogue systems. This is explained in Section 4. The other model parameters $p(s'|s,a), R(s,a)$ are known. In general, the BAPOMDP can solve problems when $p(s'|s,a), R(s,a)$ are also unknown [37].

To account for the uncertainty, the BAPOMDP framework uses Dirichlet distributions, which are probability distributions over the parameters of multinomial distributions. Given ψ_i, the frequency event e_i occurs over n trials, the probabilities p_i of each event has a Dirichlet distribution, i.e. $(p_1, p_2, ..., p_k) \sim Dir(\psi_1, \psi_2, ...\psi_k)$. If the counts $(\psi_1, \psi_2, ...\psi_k)$ are observed over n trials ($n = \sum_{i=1}^{k} \psi_i$), the distribution represents the probability of a discrete random variable according to the probability distribution $(p_1, p_2, ..., p_k)$. The probability density function is $f(p, \psi) = \frac{1}{B(\psi)} \prod_{i=1}^{k} p_i^{\psi_i - 1}$, where B is the multinomial beta function. The expected value of p_i is $E(pi) = \frac{\psi_i}{\sum_{j=1}^{k} \psi_j}$

An BAPOMDP is built on top of an POMDP, but it has additional unknown parameters. Formally, a discrete POMDP is specified as a tuple $(S, A, Z, T, O, R, \gamma)$, where S is a set of states, A is a set of actions, and Z is a set of observations. The uncertainty on the distribution $O^{s'a\cdot}$ is represented by the counts $\psi^a_{s'z} \forall z$ of the number of times observation z was made in state s' after doing action a, with ψ the vector of all the observation counts. The expected transition probability for $O^{s'az}$ is $O^{s'az}_{\psi} = \frac{\psi^a_{s'z}}{\sum_{z' \in Z} \psi^a_{s'z'}}$.

The objective is to learn an optimal policy, such that actions are chosen to maximize reward with respect to the posterior captured by the Dirichlet distribution. The state space S' of the BAPOMDP is defined as $S' = S * \mathcal{O}$ where $\mathcal{O} = \{\psi \in N^{|S||A||Z|} | \forall (s,a), \sum_{z \in Z} \psi^a_{sz} > 0\}$ represents the space in which ψ lies.

In domains with a very large state space, offline POMDP algorithms are not practical. With offline approaches, the algorithm returns a policy defining which action to execute in every possible belief state. As this policy construction step takes significant time, such an approach is only applicable to small to medium-size domains. Solving a BAPOMDP exactly for all belief states is impossible due to the dimensionnality of the state space The count vectors can grow unbounded. Therefore, we use online lookahead search algorithms in solving a BAPOMDP model.

It has been shown that the BAPOMDP is an instance of POMDP [37]. As such we need to track the belief state b. The belief state of the BAPOMDP represents a distribution over both states and count values. It is not the same belief state as in a POMDP. The model is learned by simply maintaining this belief state, as the distribution will concentrate over most likely models, given the prior and experience so far. If b_0 is the initial belief state of the unknown POMDP, and the count vector ϕ_0 represents the prior knowledge on this POMDP, then the initial belief of the BAPOMDP is: $b_0^{'}(s, \phi_0)$=$b_0(s)$, if $(\phi) = (\phi_0)$; 0,otherwise.

To do this in a tractable way, we consider an approximation whereby we do the exact belief update (Eqn. 3.15) at a given time step, but only keep the K most probable belief states in the new belief b', and we renormalise b' accordingly.

The algorithm is described in Algorithm 4. At each step, the agent computes $V(b, D, K)$ for its current belief b and then executes the best action found in Line 18. In BAPOMDP, $R_B(b, a)$ is now defined as:

$$R_B(b, a) = \sum_{(s,\phi)\in S_b^{'}} b(s, \phi) R(s, a). \tag{3.24}$$

ParticleFilterKProbable(b, a, z, k) does the belief update and returns an updated belief with k particles. For more details about BAPOMDPs, please refer to [37].

Algorithm 4 Online Planning Algorithm for BAPOMDP

```
1: function V(b,d,k) Inputs
   b: the current belief state
   d: the current depth
   Statics:
   k: the value for k most probable update
   action: the best action
2: if d = 0 then
3:    return V_immediate(b)
4: end if
5: maxQ ← -∞
6: for all a ∈ A do
7:    q ← R_B(b,a)
8:    for all z ∈ Z do
9:       b' ← ParticleFilterKProbable(b,a,z,k)
10:      q ← q + γPr(z|b,a)V(b',d-1,k)
11:   end for
12:   if q > maxQ then
13:      maxQ ← q
14:      maxA ← a
15:   end if
16: end for
17: if d = D then
18:   action ← maxA
19: end if
20: return maxQ
```

CHAPTER 4
BAPOMDPs for Dialogue Systems

Dialogue systems have been successfully modeled as POMDPs [39, 52]. By casting the dialogue problem in a statistical framework, it is well-suited to coping with the uncertainty in human spoken dialogue. However, most offline and online solvers for POMDPs assume that the model parameters are known, which is uncommon in many problems

Much of the recent literature on model-based Bayesian Reinforcement Learning have mostly focused on observables domains (MDPs), where it is easier to update the posterior distribution. In using dialogue systems, there are speech variations between individuals that are hard to determine such as the accent of the speaker. Furthermore, it is not possible to know exactly how noisy the speech recognition is or to determine the reliability of the voice recognition device.

Bayes-Adaptive POMDPs provide a solution to the above problem [37]. This framework is based on the Bayes-Adaptive MDP framework [15]. As mentioned in Section 3.4, one issue is to update the counters ϕ and ψ without observing state s. BAPOMPs solve this problem by including the Dirichlet parameters in the state space, and maintaining a belief state over these parameters. BAPOMPs also bound the space of Dirichlet parameters to a finite subspace so that we can approximate the infinite dimensional belief space to perform the belief update.

In this chapter, we show how spoken dialogue systems can be modeled with BAPOMDPs. We also discuss the advantages and the limitations of our approach.

4.1 BAPOMDPs for Dialogue Systems

The state of the BAPOMDP model captures the user's intent and the dialogue state. This state represents what the user wants to achieve. It represents the user's objective of using the dialogue system. In the SmartWheeler domain in Section 5.5, for example, the user might want its wheelchair to move one meter forward. In the HandWashing domain in Section 5.4, the user might want to put soap on his hand.

The user has an unknown intent, and the agent has to execute an action based on its guess of the user's intent. The actions define the set of possible responses of the dialogue system. In the SmartWheeler domain, the dialogue manager might start moving the wheelchair forward, or ask the user to repeat his request. In the HandWashing domain, the dialogue manager might prompt the user to put soap on his hand.

Throughout this thesis, we assume that we do not know some or all the observation parameters, but we assume that we know the state transitions and the rewards. In the SmartWheeler domain, we do not know exactly how noisy the voice recognition device is. In the HandWashing domain, we have no idea how our accurate we can judge where the user's hand is. We model the unknown observation parameters using Dirichlet counts.

In implementing BAPOMPs for dialogue systems, we enhance the algorithm based on the existing BAPOMDP framework in two ways. First, we adopt Real Time Belief State Search (RTBSS), which is a branch-and-bound technique. Second,

we make use of known symmetry in the model to perform parameters tying because We also have some prior knowledge of similiarities in the observation parameters. Our goal is to come up with a policy that gives us a reasonable action for all beliefs despite not knowing the parameters at the onset. We hypothesis that if we have prior knowledge of certain symmetrical properties in the model, for example, the speech commands in a spoken dialogue system, and by using the RTBSS technique, we can scale up BAPOMDPs to rich real-world domains.

4.1.1 Symmetry in the model

In a typical dialogue system, many parameters are likely to be similar. The system designers of the dialogue systems can provide us with domain knowledge of the system. For instance, in the SmartWheeler domain, the phrase *"drive slowly backward"* is similar to *"drive slowly forward"* and *"drive slowly one meter backward"*, but is very different from *"avoid obstacle"*. In the HandWashing problem, we know that the observation we obtain depends on only one state variable: the position of the user's hand. We also know that the observation parameters should be identical regardless of the action we take. Using the knowledge that certain observation parameters have similar values, we can learn the parameters in a faster manner.

This is how we make use of symmetry to update the Dirichlet parameters upon receiving a new observation. Since certain observation parameters have approximately the same values, whenever we have an observation for a particular observation parameter, besides updating the corresponding Dirichlet parameter, we also update the Dirichlet parameters corresponding to other "similar" observation parameters.

Such parameter tying is used so that all observations that have nearly identical distributions are tied to the same counts. This dramatically reduces the number of parameters to learn. Parameter tying allows data to be used across several instances. When more than one parameter are tied, the value of these parameters is shared among them. This provides a speedup in learning as each example has an influence on several parameters in the model.

4.1.2 Online Planning with RTBSS

With BAPOMDPs, one of the main issues is that the count vectors can grow unbounded. Consequently, the search tree increases exponentially, especially if the branching factor is high, slowing down our online search for the optimal action. To overcome this problem, we use RTBSS (described in Section 3.3.5) to cut down redundant sub-trees by pruning away some actions, thus reducing the time for the online search.

In RTBSS, the heuristic value described in Line 9 of Algorithm 2 has to be an upper bound. Ideally, the heuristic value has to be the maximal value that any algorithm can find if it searches the tree to the maximal depth d. This ensures that we do not prune the wrong actions. In modeling dialogue systems with BAPOMDPs, we use RTBSS with a heuristic$(b, a, d) = \sum_{i=1}^{d} \gamma^i R_{max}$, where d is the depth of the search and γ is the discount factor. R_{max} is the maximal reward for all states and all actions of the underlying MDP.

4.1.3 Belief Tracking

We consider an approximation whereby we do the exact belief update (Eqn. 3.15) at a given time step, but only keep the K most probable belief states in the new belief b', and we renormalise b' accordingly.

4.1.4 Minor modification to the original BAPOMDP framework

In the original BAPOMDP framework [37], the mean (expectation) of the Dirichlet distribution was used to estimate the unknown transition and observation probability. We modify this framework by sampling from the Dirichlet distribution instead of using the mean (expectation) of the Dirichlet distribution. This allows us to have a more unbiased estimation at the beginning, and leads to a more exploratory process.

4.2 Steps in applying BAPOMDPs to Dialogue Systems

We summarize the steps in applying BAPOMDPs to dialogue systems as follows:

1. Represent the dialogue system as a POMDP.
   (a) Represent the state space S as a set of the user intentions and dialogue information.
   (b) Represent the action space A as a set of the possible responses by the dialogue system.
   (c) Represent the observation space Z as a set of the possible information the dialogue system can perceive upon executing a command.
   (d) Define the transition probabilities $T : p(s'|s, a)$ based on domain knowledge and previously collected data.
   (e) Define the observation probabilities $O : p(o|s, a)$ based on domain knowledge and previously collected data. Label the observation probabilities

that are unknown. Note the actions that are associated with these unknown observation probabilities.

(f) Define the reward function $R(s, a)$ based on domain knowledge and previously collected data.

2. Represent the dialogue system as a BAPOMDP.
   (a) For the unknown observation parameters, label them with Dirichlet counts with the prior knowledge.
   (b) If there is no prior knowledge for any of the unknown observation parameters, label them with uniform initial priors.
3. Using domain knowledge, look for symmetry in the observation parameters that are unknown.
   (a) Label the similar parameters.
   (b) Write algorithms to perform parameter tying for the similar parameters. Whenever a parameter count is updated, the corresponding parameters (tied to it) should also be updated.
4. Solve the BAPOMDP and learn the unknown observation parameters.
   (a) Whenever an action associated with the unknown observation parameters is picked, increment the Dirichlet counts for the observation parameters accordingly.
   (b) Perform belief tracking by choosing the k most probable belief states and normalizing them.
   (c) Pick the best action to perform based on the new posterior, and repeat step 4.

4.3 Discussion

Our framework is mathematically sound, and we will show in the experiments in the next Section that the algorithms are tractable on large domains. However, we assume that we can obtain useful domain knowledge from the system designer. It is not obvious whether we can find similar parameters for parameter tying if we do not have domain knowledge Also, in the BAPOMDP framework, we only learn the probabilities associated with an action as we preform the action. It is likely that an optimal policy may pick this action infrequently, resulting in a slower learning of the parameters. Noisy initial priors might possibly bias the policy, and the action is not performed at all. In such a case, there is no learning of the observation parameters in each episode.

In our approach, we adopt RTBSS, an online planning algorithm, in BAPOMDPs to solve dialogue systems. It may be possible to also make use of informative lower and upper bound computed offline in speeding our online search. However, we have not found an efficient way of computing such bounds on the BAPOMDP's value function. This is an interesting area for future research. Another area for future research is to generalize the current framework to the case where the number of states is unknown. Since the states are not observed, knowning the number of states is a strong assumption in practice.

CHAPTER 5
Empirical Evaluation

Our goals are to show that by exploiting certain known components of the dialogue system, such as knowledge of symmetrical properties, and by using an approximate online planning algorithm (RTBSS) as mentioned in Section 4.1.2, we are able to apply Bayesian RL on several realistic spoken dialogue system domains. We consider four different experimental domains. First, a small synthetic data case, where we illustrate several properties of the approach. Second, a small dialogue manager based on the SACTI1 corpus which contains 144 dialogues between 36 users and 12 experts. Third, a dialogue manager designed to help patients to operate a wheelchair. Finally, we consider a large dialogue manager aimed at patients with dementia, to assist them with activities of daily living. We then present results obtained and the issues faced with using these techniques.

5.1 Empirical Methodology

We run our experiments using two different ways of updating the counts for estimating the observation probability parameters. The first approach is the usual way of updating each parameter independently. The second approach makes use of symmetry with the parameters. Each BAPOMDP simulation consists of 100 episodes. Each episode is a short dialogue sequence trial, which may be terminated by a terminating action or when a prefixed number of actions is taken. At this

point, the POMDP state (the user's intent) is reset, but the distribution over the observation count vector is carried over to the next episode.

We measure the empirical returns of the policy under the various conditions. This corresponds to the total rewards achieved by the robot. We measure the L1-distance, measured as $\sum_{s \in S} |b(s) - b'(s)|$. This is an indication of the accuracy of the estimated model. The smaller the distance between the real belief and the estimated belief, the more accurate the model is. We also measure the Observations-L1-distance, measured as $\sum_{z \in Z} |O(s, a, z) - O'(s, a, z)|$. This is an indication of the accuracy of the learnt observation parameters. The smaller the distance between the real observation parameters and the learnt observation parameters, the more accurate the model is. As both the L1-distance and the Observations-L1-distance vary depending on the episode, we take the maximum distance among all the episodes as the distance for the simulation.

5.2 Small POMDP Dialogue Manager

This problem was introduced in [16]. Here, a human operator instructs an assistive robot to move to one of two locations, bedroom or bathroom. Even though the human intent (the state) is one of these goals, the observation received by the robot through a speech recognizer is not accurate. The robot has the option to ask again to ensure the goal was understood correctly. In this model, we assume the probability of a wrong observation is 0.15. In fact, this model is similar to the classic Tiger problem [9]. There are four unknown parameters: OEE, OEA, OAE, OAA to learn.

OEE stands for $Pr(z = hearbedroom | s = user\ wants\ bedroom, a = ask) = 0.85$.

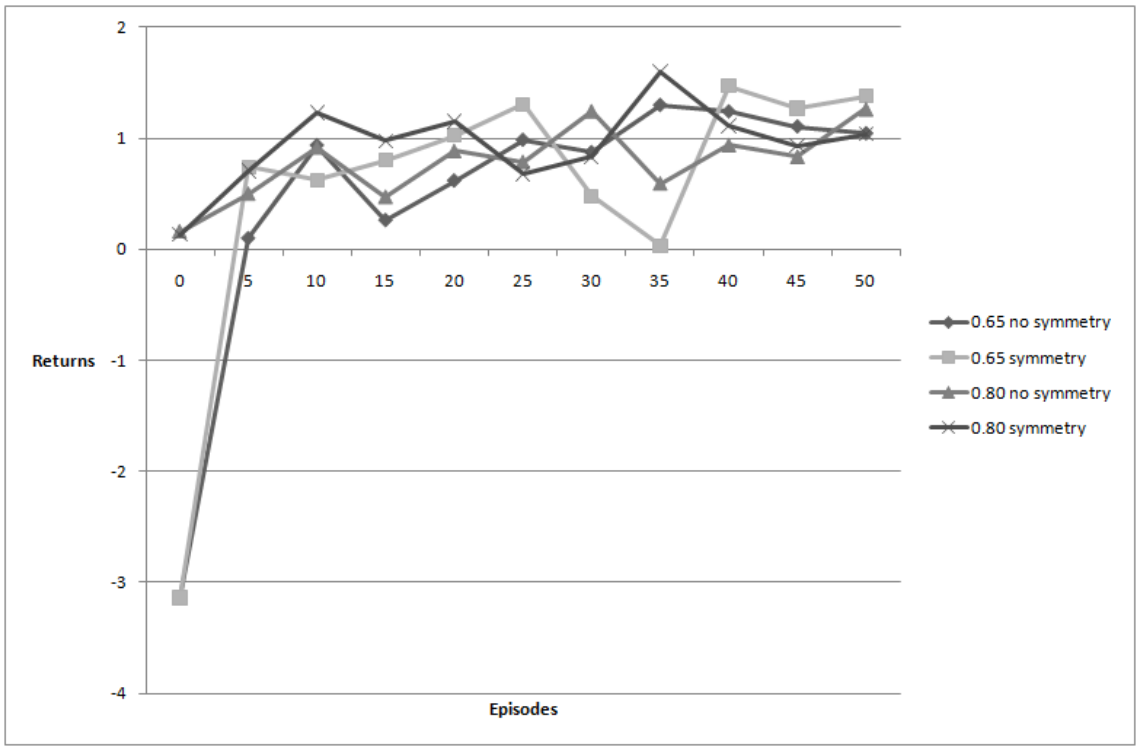


Figure 5–1: Small POMDP Dialogue Manager: Returns against different priors with and without symmetry

OEA stands for $Pr(z = hearbathroom|s = user\ wants\ bedroom, a = ask) = 0.15$.

OAE stands for $Pr(z = hearbedroom|s = user\ wants\ bathroom, a = ask) = 0.15$.

OAA stands for $Pr(z = hearbathroom|s = user\ wants\ bathroom, a = ask) = 0.85$.

Since $OEE = OAA$ and $OEA = OAE$, we can make use of symmetry in this model to perform parameter tying. Whenever the count for OEE is updated, OAA is also updated, and vice versa. Similarly, whenever the count for OAE is updated, OEA is also updated, and vice versa.

For this problem, we have a planning time of 1 second per action, 20 actions per episode, and we repeat 1000 trials. Experimental results are promising. For the

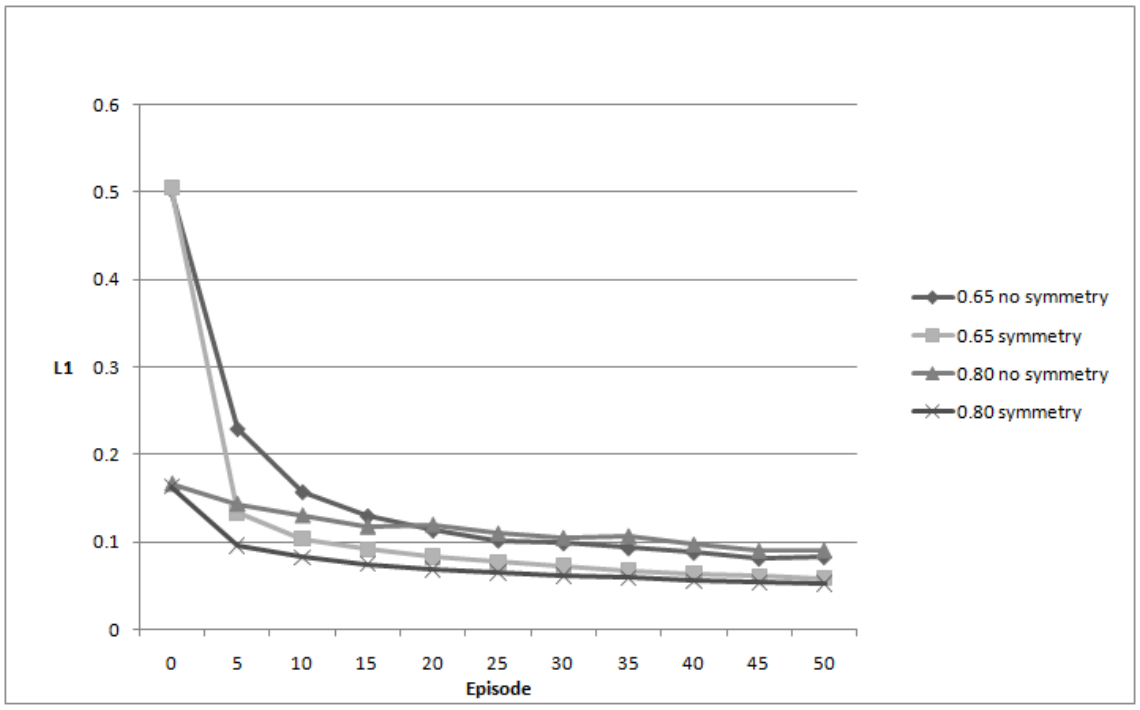


Figure 5–2: Small POMDP Dialogue Manager: L1-distance against different priors with and without symmetry

returns, as shown in Figure 5–1, using a prior of 0.80 gives better returns than a prior of 0.65 at the beginning, but the returns converge after around 10 episodes. Using symmetry leads to a faster convergence no matter whether we use a prior of 0.65 or 0.80. In Figure 5–2, the L1-distance is smaller when we use a prior of 0.80, as compared to a prior of 0.65, but the L1-distance converges after around 10 episodes. Using symmetry leads to a faster convergence no matter whether we use a prior of 0.65 or 0.80

These empirical results show that using symmetry in the model leads to more efficient learning of the model parameters. Even if the initial priors are more noisy, using parameters tying leads to a faster convergence of the model parameters, leading

to a faster convergence in both the returns and the L1-distance. This indicates that for learning in a dialogue system, it might be more effective to look for symmetry in the model than to come up with more accurate initial priors.

5.3 SACTI1

This domain was first introduced in `http://mi.eng.cam.ac.uk/projects/sacti/corpora` [53]. It contains 144 dialogues between 36 users and 12 experts (who have the role of a dialogue manager), covering 24 tasks. The utterances from users to human experts are first confused using a speech recognition error simulator. Hidden Topic Markov Models (HTMM) was used to design dialogue POMDP for the SACTI1 dialogues [8]. The generated dialogue POMDP consists of 5 states, 14 actions, and 5 meta observations drawn by HTMM using 817 primitive observations (words). In the generated model, users can have 3 different intentions. They represent the machine states: transportation, visiting area, and food. There are also 2 other states: success and failure, depending on whether the dialogue finished successfully or not. The 14 actions include inform, request, greeting farewell, request repeat, and others. We assume that we do not know the observation parameters for the action 'inform', and we want to learn them. For this problem, we have a planning time of 1 second per action, 20 actions per episode, and we repeat 1000 trials.

Our objective here is to investigate the effects of the depth of the online search on the returns obtained. In solving a POMDP problem, we usually obtain better returns as we increase the depth of a search. Increasing the depth of a search leads to better actions being chosen because we take into account further actions in the future. However, in a BAPOMDP model, our search tree is a lot larger than a

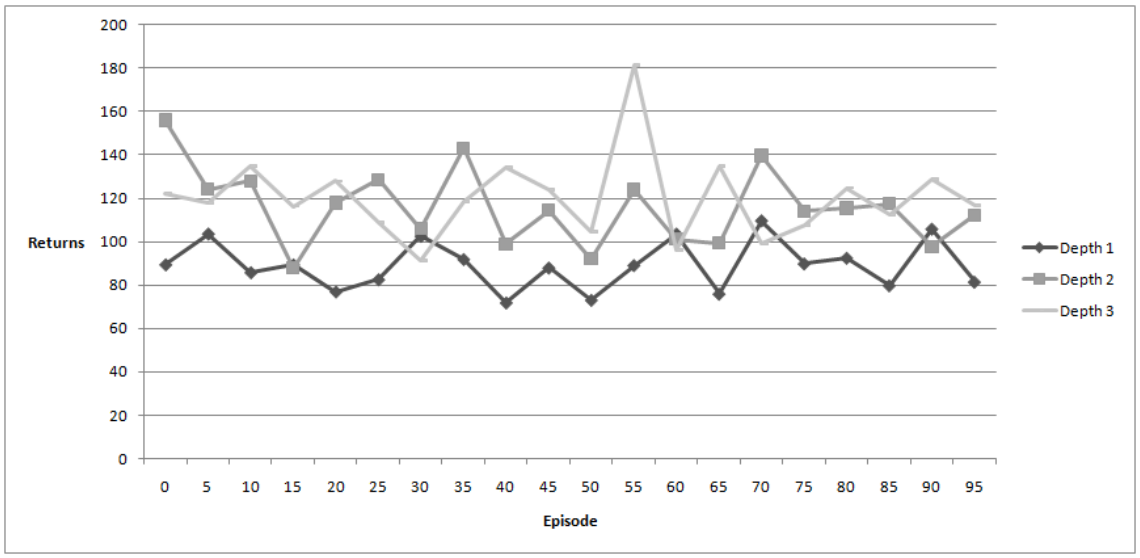


Figure 5–3: Sacti1: Returns against different depths

similar POMDP problem because a BAPOMDP model has to take into account the Dirichlet counts. Solving a BAPOMDP exactly for all belief states is impossible in practice due to the dimensionnality of the state space. In particular, the Dirichlet count vectors can grow unbounded. It is interesting for us to observe if increasing the depth of the online search will lead to better returns in a BAPOMDP.

In our experiments, we only use a depth of 1 to 3 because beyond depth 4, the computations are too slow and time-consuming. We note some interesting experimental results. In Figure 5–3, we note that using an online search of depth 2 and 3 seems to gives slightly better returns than using depth 1. Increasing the search beyond depth 2 does not give any obvious improvements. This is interesting for further investigations, as it is indicative whether it is necessary and useful to learn how to plan for many decision-making problems.

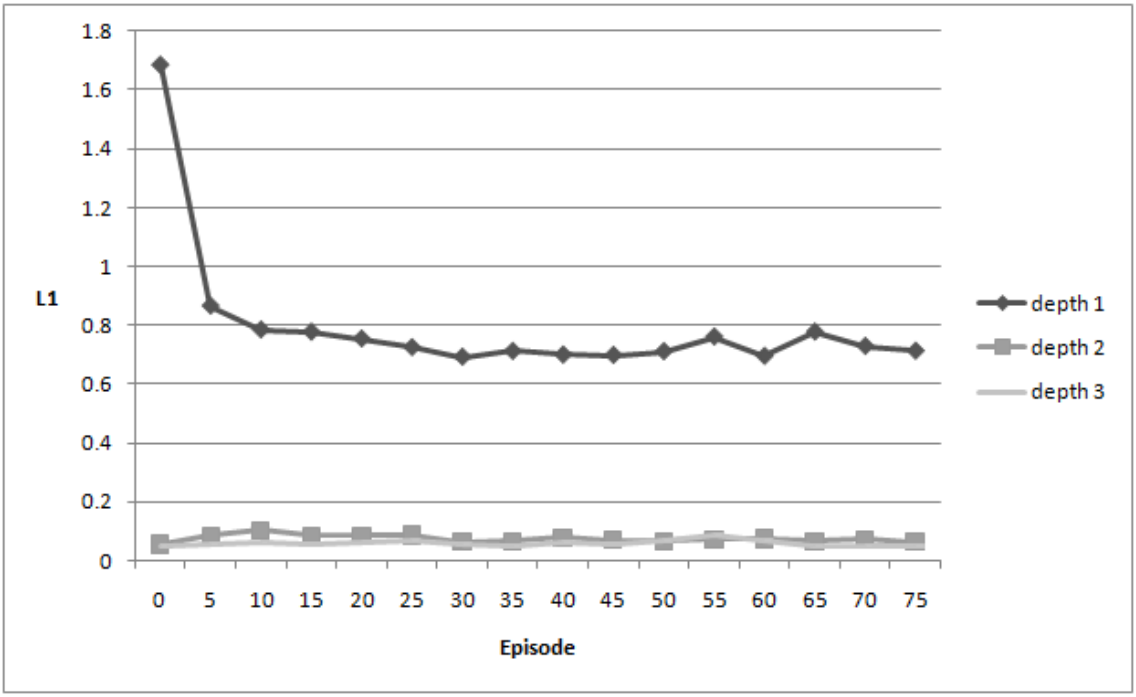


Figure 5–4: Sacti1: L1-distance against different depths

In Figure 5–4, we note that for the L1-distance, using an online search of depth 1 gives drastically poorer results as compared to using depth 2 and depth 3. This is a surprise, as we do not expect to see such a big difference. This difference is actually due to the the actions selected by the search with depth 1. When we use an online search of depth 1, our search always returns action "inform", which is the action with observation parameters we want to learn. Due to the noise in these parameters, our L1-distance increases drastically if we perform action "inform" right at the beginning. As we take the maximum L1-distance among all the episodes as our L1-distance for the simulation, this L1-distance we observe at the very beginning becomes our L1-distance for the simulation. However, when we perform a search of

depth 2 or depth 3, the search tends to give us actions other than "inform". Since these actions give almost perfect observations, our L1-distance decreases, and they do not increase much after that. This results in the large disparity shown in Figure 5–4.

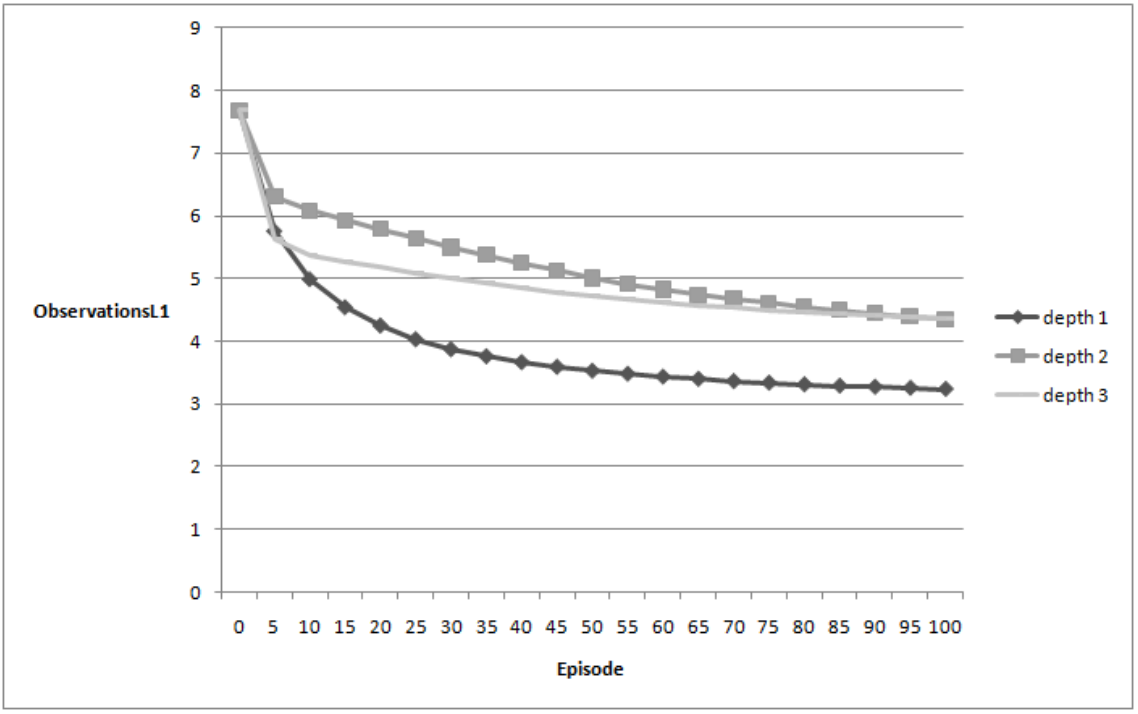


Figure 5–5: Sacti1: Observations-L1-distance against different depths

It is also interesting to note that even though using a search of depth 1 gives us poor returns and L1-distance, it actually gives better observation L1-distance as seen in Figure 5–5. Upon further investigations, we realise this is also due to the number of times the "inform" action is performed. As observed in Figure 5–6, using an online search of depth 1 leads to performing the "inform" action many more times than using depth 2 and depth 3. These results are consistent because peforming"inform"

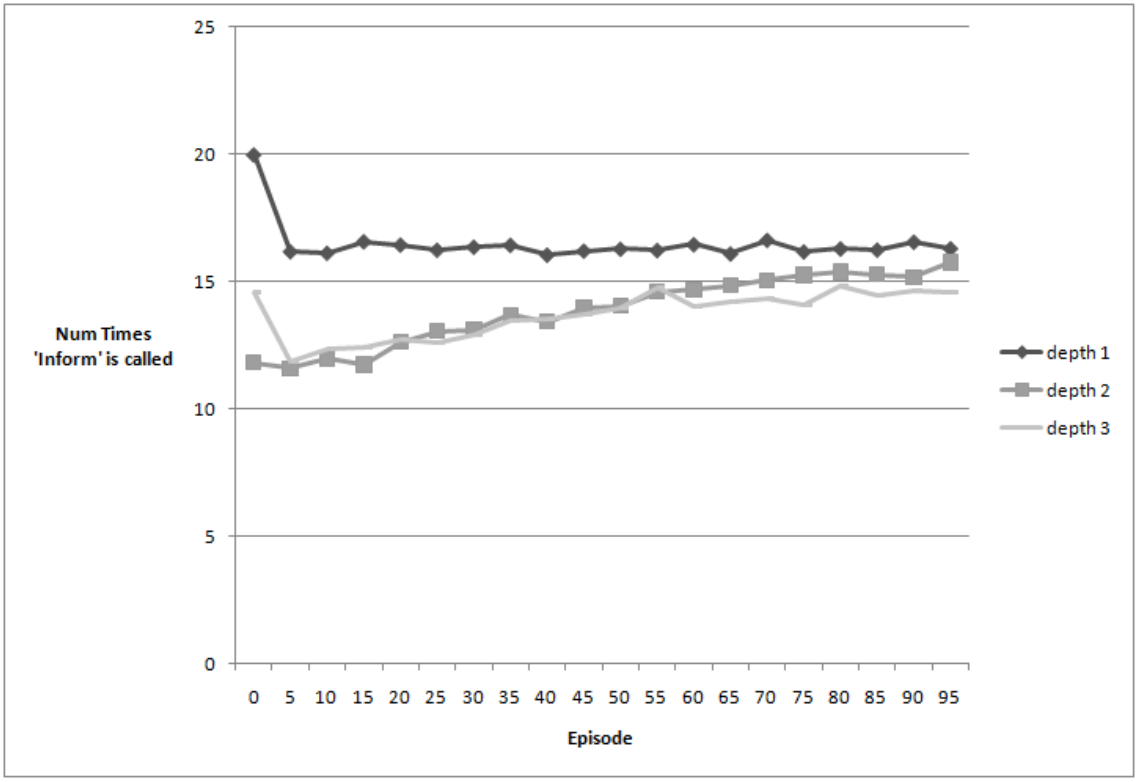


Figure 5–6: Sacti1: Number of times "inform" action was performed against different depths

actions more times will lead to better learning of its observation parameters. In Figure 5–7, we show how the Observation-l1-distance varies according to the number of times the "inform" action was performed.

These empirical results suggest that using the L1-distance is a better way to judge how well we learn a model than using the Observations-L1-distance. The improvements in L1-distance correspond to the improvements in the returns, as the

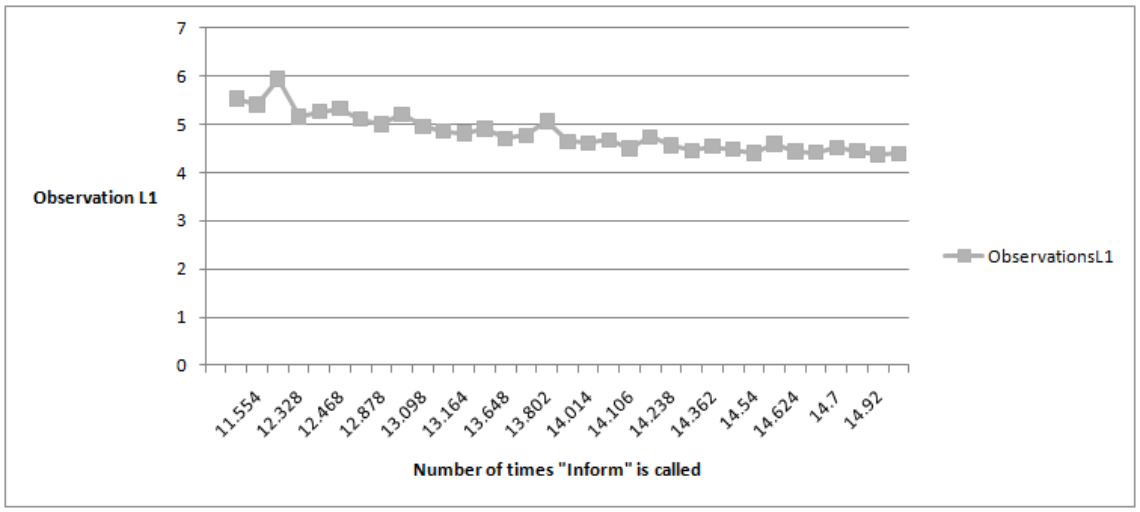


Figure 5–7: Sacti1: Observation-l1-distance against the number of times the "inform" action was performed

results indicate that depth 2 and depth 3 give better returns than depth 1. Improvements in Observations-L1-distance, however, are mainly due to the action (with observation parameters we want to learn) being chosen more frequently.

5.4 HandWashing

This problem was first introduced in [7]. The problem can be downloaded at `http://www.cs.uwaterloo.ca/~ppoupart/software/symbolicPerseus/problems/HandWashing/cppo3.txt`. This is a reduced version of the original HandWashing problem and has 180 states, 6 observations, and 6 actions.

The goal of the interaction manager in this domain is to monitor a user with Alzheimer's disease.

The interaction manager offers assistance to the user in the form of task guidance such as prompts or reminders when he attempts a HandWashing task. The user can perform 6 actions: "nothing", "wet hand", "turn on water", "turn off water",

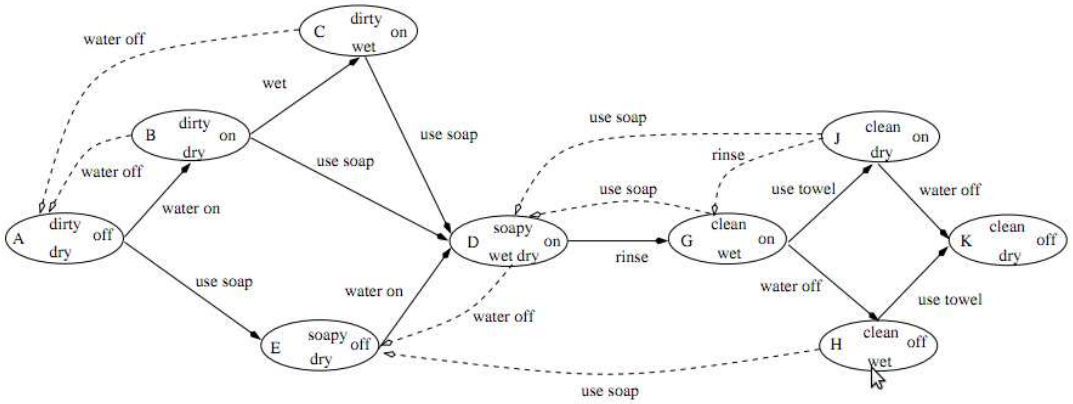


Figure 5–8: Plan Graph. Each node is labeled A,B,C,D,E,G,H,J,Kg. The state of the hands is indicated as either "dirty", "soapy", "clean", and whether the hands are "wet" or "dry". The water is indicated as either "on" or "off". Actions label the arcs between nodes, and they are necessary to move between the nodes. Solid arcs denote actions that make progress, while dashed arcs indicate regressions to an earlier state.

"use soap", or "dry hands". The user's state can be factored into 4 different state variables. The user's behavior describes what the user has just done: "nothing", "wet", "taps", "soap", or "dry". The planstep is described in Figure 5–8. The user's responsiveness describes if the user will respond to a prompt if given, and it can be either "yes" or "no". The user's awareness describes how aware the user is of their current situation, and it can be either "yes" or "no". The 6 observations correspond to the position of the user's hand, such as "away", "sink", "water", "tap", "soap", or "towel".

With the domain knowledge in this HandWashing domain, we can perform parameter tying as mentioned in Section 4.1.1. We know that the observation we have depends on the position of the user's hand. The observation we get after taking an action depends only on the position of the user's hand, but not the type of action.

This means that we should have one identical set of observation parameters for all the 6 different actions. Even though the state space of the user is factored into 4 different state variables, the observation (the position of the user's hand) should depend on only one state variable, the user's behaviour. The user's behaviour describes what the user has just done, and it affects the observation we obtain after taking any action. For example, if the user's behaviour is "wet", we are more likely to observe that the user's hand is at "water". Similarly, if the user's behaviour is "dry", we are more likely to observe that the user's hand is at "towel". On the other hand, the other 3 state variables such as the planstep, the user's responsiveness, and the user's awareness, should not affect the observation we obtain after taking any action. Therefore, many observation parameters are identical, and we can make use of the symmetry in the model.

There are a total of $6 \times 180 = 1080$ observation parameters to learn. In Table 5–1, there are 5 different sets of observation parameters, as indicated by "a", "b", "c", "d", or "e". Each row corresponds to one of the 5 possible user's behaviour. Using known symmetry in the model as shown in Table 5–1, we perform parameter tying in our experiments. As seen in Table 5–1, even though there are a larger number of observation parameters, many of them are actually similar.

The aim of this experiment is to evaluate the performance of the BAPOMDP on the interaction manager in the HandWashing domain. Unlike the previous experiments, we assume no knowledge of the initial observation parameters. We generate numbers from a random number generator for each of the 1080 observation parameters. For each row of 5 observation parameters, we then normalise them so that they

Table 5–1: The observation parameters in the HandWashing domain for a particular action. Each row corresponds to a different state variable. Each column corresponds to a different observation variable. Note that we have the same set of observation parameters for all the different actions.

row 1	a1	a1	a2	a2	a2	a2
...	...	...	...	...	...	...
row 9	a1	a1	a2	a2	a2	a2
row 10	b1	b2	b3	b1	b1	b1
...	...	...	...	...	...	...
row 18	b1	b2	b3	b1	b1	b1
row 19	c1	c2	c3	c4	c1	c1
...	...	...	...	...	...	...
row 27	c1	c2	c3	c4	c1	c1
row 28	d1	d2	d1	d1	d3	d1
...	...	...	...	...	...	...
row 36	d1	d2	d1	d1	d3	d1
row 37	e1	e2	e3	e1	e1	e4
...	...	...	...	...	...	...
row 45	e1	e2	e3	e1	e1	e4
...	...	...	...	...	...	...
...	...	...	...	...	...	...
...	...	...	...	...	...	...
row 136	a1	a1	a2	a2	a2	a2
...	...	...	...	...	...	...
row 144	a1	a1	a2	a2	a2	a2
row 145	b1	b2	b3	b1	b1	b1
...	...	...	...	...	...	...
row 153	b1	b2	b3	b1	b1	b1
row 154	c1	c2	c3	c4	c1	c1
...	...	...	...	...	...	...
row 162	c1	c2	c3	c4	c1	c1
row 163	d1	d2	d1	d1	d3	d1
...	...	...	...	...	...	...
row 171	d1	d2	d1	d1	d3	d1
row 172	e1	e2	e3	e1	e1	e4
...	...	...	...	...	...	...
row 180	e1	e2	e3	e1	e1	e4

sum up to 1. Then, we compare the effects of using symmetry to update the counts. For this problem, we have a planning time of 6 seconds per action, 20 actions per episode, and we repeat 1000 trials.

In this experiment, when we measure the return of the learnt model, we find that the return does not improve with more episodes, in both the cases when we use the symmetry, and when we ignore the symmetry in the model. Further investigating this issue, we discovered that solving the actual model as a POMDP does not give us higher returns than solving the same model as a POMDP and replacing the original observation parameters with randomly generated probabilities. This is a surprising result, and it suggests that the observation parameters in this particular POMDP do not matter when we run simulated experiments with the optimal policy. This means that the returns do not seem to matter even if the observations are totally wrong.

We also solve the same POMDP problem but we remove all the observation parameters. This means that this POMDP problem is now a MDP problem. We discovered that solving the problem as a POMDP gives much higher returns than solving it as a MDP. This is illustrated in Table 5–2. This result is likely to be due to the nature of the problem domain, where the observations are not useful in getting an optimal policy. This belongs to a particular class of POMDP problems, where the presence of observations are useful, but the observation parameters are not useful in computing a optimal policy. It will be useful in future work to explore whether we can deduce whether any POMDP model belongs to such a particular class of problems.

Table 5–2: Comparing returns of the actual model with different observation parameters

Type	Returns	95% Confidence Interval
MDP	12.42	(10.97,13.87)
Actual POMDP	16.45	(14.85,18.05)
POMDP with randomly generated observation parameters	16.87	(15.27,18.46)

Since it is not obvious whether learning took place, we also measure the Observations-L1-Distance. As seen in Figure 5–9, the Observations-L1-Distance decreases much faster when we perform parameter tying. However, the Observations-L1-Distance is still quite high after 100 episodes. This is probably due to the large number of observation parameters that we are trying to learn. The Observations-L1-Distance is an indication of the accuracy of the learnt observation parameters. The smaller the distance between the real observation parameters and the learnt observation parameters, the more accurate the model is. This experiment shows that for the HandWashing problem, even though learning took place with the BAPOMDPs model, we do not obtain better results in terms of the returns. However, this seems to be due to the particular characteristics of the domain, not to our learning approach.

5.5 SmartWheeler dialogue Domain

The SmartWheeler dialogue domain is a POMDP model used for dialogue management between a user and an intelligent wheelchair. It is a modification of the POMDP model described in [4]. The problem can be downloaded at `http://www.cs.mcgill.ca/~smartwheeler/data/wheelchairdialogue25.POMDP`. In this domain, the user has an unknown intent, and the robot has to execute an action based on its guess of the user's intent. When it has identified the user's intent, the robot

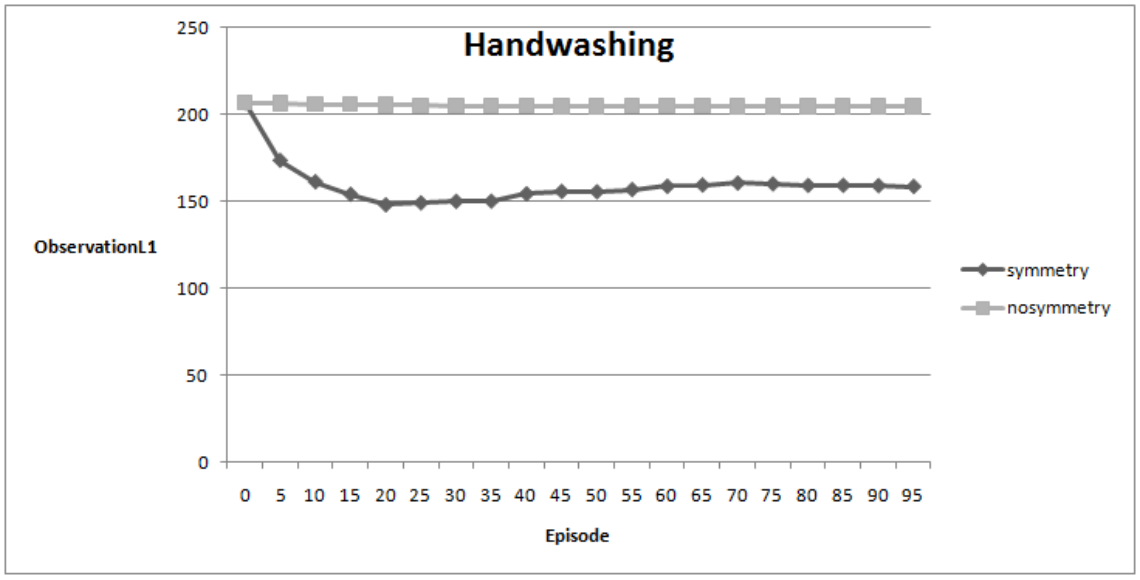
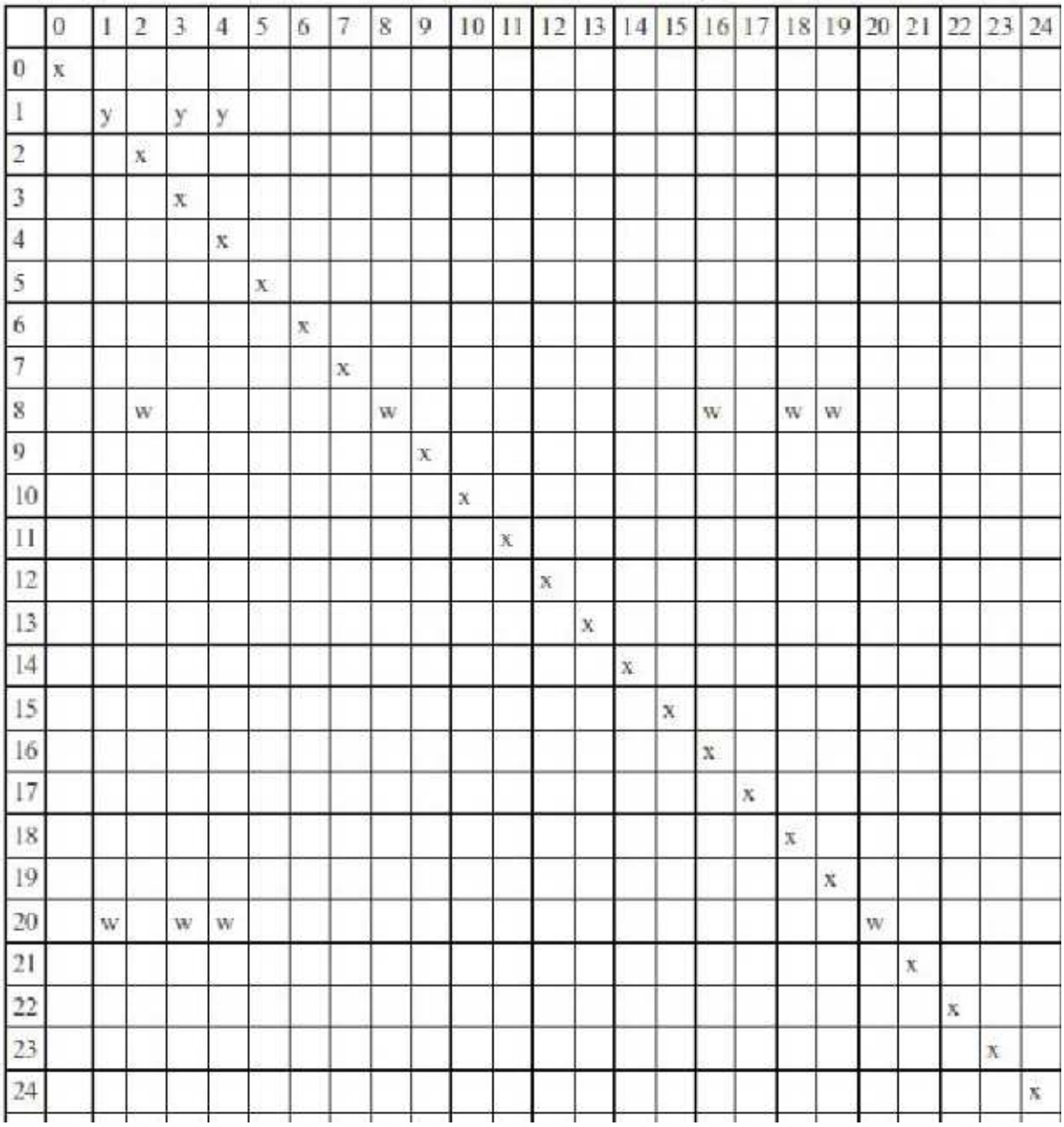


Figure 5–9: HandWashing: Observations-L1-distance against models with and without symmetry

can execute a command action, receiving a positive reward if correct, and zero reward otherwise.

The robot can also execute one query action, which is strictly information gathering. This returns an observation giving an indication of the user's intent. Observations are not fully accurate.

As mentioned in Section 2.6.1, the SmartWheeler can communicate with a user with its voice recognition device. In this domain, there are 25 states in the POMDP model. Each state corresponds to the user's intent, such as *"drive one meter forward"* or *"set speed to fast"*. We assume that the user may possibly want to operate the wheelchair in one of the 25 possible ways as listed in Table 5–3. For example, when the user wants to move forward one meter in his wheelchair, he might say, *"drive one meter forward"*.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
0	x																								
1		y		y	y																				
2			x																						
3				x																					
4					x																				
5						x																			
6							x																		
7								x																	
8			w						w								w		w	w					
9										x															
10											x														
11												x													
12													x												
13														x											
14															x										
15																x									
16																	x								
17																		x							
18																			x						
19																				x					
20		w		w	w																w				
21																						x			
22																							x		
23																								x	
24																									x

Figure 5–10: SmartWheeler: Matrix for observation probability parameters with symmetry

Table 5–3: A list of possible operations in SmartWheeler

avoid obstacle
drive slowly backward
drive slowly forward
drive slowly one meter backward
drive slowly two meters backward
go down ridge
move joystick away
move joystick back
roll forward
set speed to fast
set speed to medium
tilt seat backward
turn controller off
turn controller on
turn ninety degrees right
drive fast backward
drive fast forward
tilt seat forward
drive one meter forward
drive two meters forward
drive backward
align to wall
stop
veer left
veer right

Each action represents a user's intention and the set of possible operations defines the state space. With a semantic parser in the voice recognition device, observations are handled dynamically based on the output. If the phrase is parsed successfully, the belief state is then updated using statistics based on the semantic assignments. However, if the parsing fails, the phrase is merely treated as a bag-of-words. See [3] for more details. Statistics for both of these methods were obtained from a previously gathered data set. For each state, there is a corresponding "correct" action which can be executed by the wheelchair. Additionly, there are four query actions. There is a general query action which requests a phrase to be repeated, and three other action-specific queries which request clarification for a particular set of actions. This set includes movement actions, hardware actions (such as tilting the seat), and configuration actions (such as setting the drive mode).

In a typical dialogue system, many parameters are likely to be similar. For instance, the phrase *"drive slowly backward"* is similar to *"drive slowly forward"* and *"drive slowly one meter backward"*, but is very different from *"avoid obstacle"*. Using the knowledge that certain observation parameters have similar values, we can learn the parameters in a faster manner. In Fig 5–10, similar value parameters are represented by the same letters, w, x, y or o. Note that all the unlabeled squares are actually o.

There are $25 \times 25 = 625$ unknown observation parameters to learn. They are represented by the squares in Fig 5–10. Each corresponds to a probability $p(z|s, a)$. We learn each parameter separately by making use of Dirichlet counts, and updating the counts each time we make an observation. For instance, the square in row 1,

column 0 represents the probability of observing State 0 when we are in State 1. The main diagonal squares (i.e row 0 colum 0, row 1 column 1... row 25 column 25) are the probabilities that we make the correct observation. In this model, the state represented by the parameter x is a phrase that is distinct, such as "avoid obstacle". The parameter x has a high value. The parameters w and y exist due to phrases that are very similar. For example, "drive slowly backward" is similar to "drive slowly forward" and "drive slowly one meter backward". The parameter o is due to phrases that are very distinctly different. Hence, o has a low value. In this model, the paramater x is around 0.97, y is around 0.32, w is around 0.19, and o is around 0.001. These are data inferred from during preliminary user experiment with 15 user subjects, and they capture the basic noise level of the recognition system[3].

As mentioned in Section 4.1.1, we make use of parameter tying. This is how we make use of symmetry to update the Dirichlet parameters upon receiving a new observation. Since certain observation parameters have approximately the same values, whenever we have an observation for a particular observation parameter, besides updating the corresponding Dirichlet parameter, we also update the Dirichlet parameters corresponding to other similar observation parameters.

- If any x is observed (let's say row 0), all x are updated (this means all rows from 0 to 25). This is called parameter tying. We have 25 different multinomial distributions. Each distribution has 25 parameters, but one parameter is identical in all the distributions.

- If any y is observed (let's say row 1), we only update that particular y (row 1). In this case, the three ys in row 1 correspond to different parameters in one of the multinomial distributions.
- If any w is observed (let's say row 8), we update one of the w in the other row (in this case, row20). Here, the ws in row 8 are tied to the ws in row 20.
- If any of the non-labelled square is observed (let's say row 0 column 1), for all other rows except row 1 and row 8, we update one non-labelled square (i.e. row 2 column 0).

The aim of this experiment is to evaluate the performance of the BAPOMDP approach under different conditions. For this problem, we have a planning time of 3 seconds per action, 20 actions per episode, and we repeat 1000 trials. First, we investigate the effects of having different initial estimation of the observation parameters, and second, we measure the impact of using symmetry to update the counts.

In the actual dialogue model, the value of the observation parameter x is set to 0.97. In our experiments, we consider different priors with range from 0.5 to 0.9.

Experimental results are promising and indicate that our approach scales up to larger domains. For the returns, as shown in Figure 5–11, using a prior of 0.80 gives better returns than a prior of 0.60 at the beginning. However, using symmetry leads to a faster convergence no matter whether we use a prior of 0.60 or 0.80. Using an inaccurate prior of 0.60 with symmetry actually leads to a faster convergence than using a more accurate prior of 0.80 without symmetry. At the end of the 100 episodes, only the techniques that use symmetry in the model show a convergence in

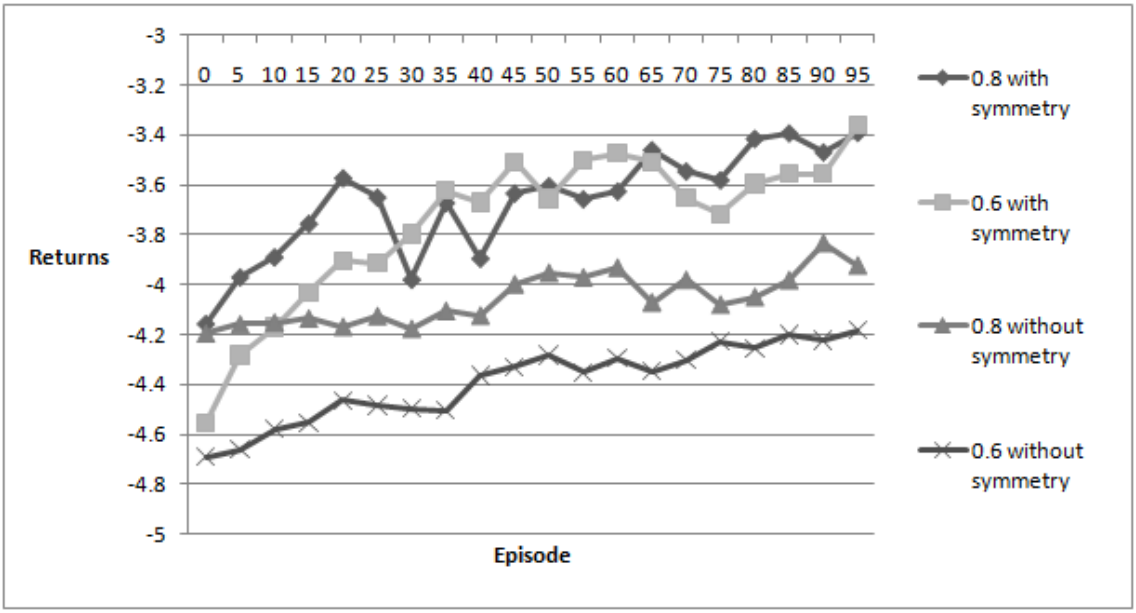


Figure 5–11: SmartWheeler: Returns against different priors with and without symmetry

the returns. This clearly shows that using symmetry results in a larger return, and leads to a faster convergence of the eventual returns.

Using symmetry to update the observation counts also results in a faster convergence to the correct model as shown in Figure 5–12, Figure 5–13 and Figure 5–14. In Figure 5–12 and Figure 5–13, the initial L1-distance is smaller as the prior is closer to the true value of 0.97, but the L1-distance converges faster if we use symmetry in the model. In Figure 5–14, we observe that using symmetry leads to a faster convergence no matter which priors are used.

These empirical results show that using symmetry in the model can be an efficient way of learning the model parameters in a larger domain such as the SmartWheeler

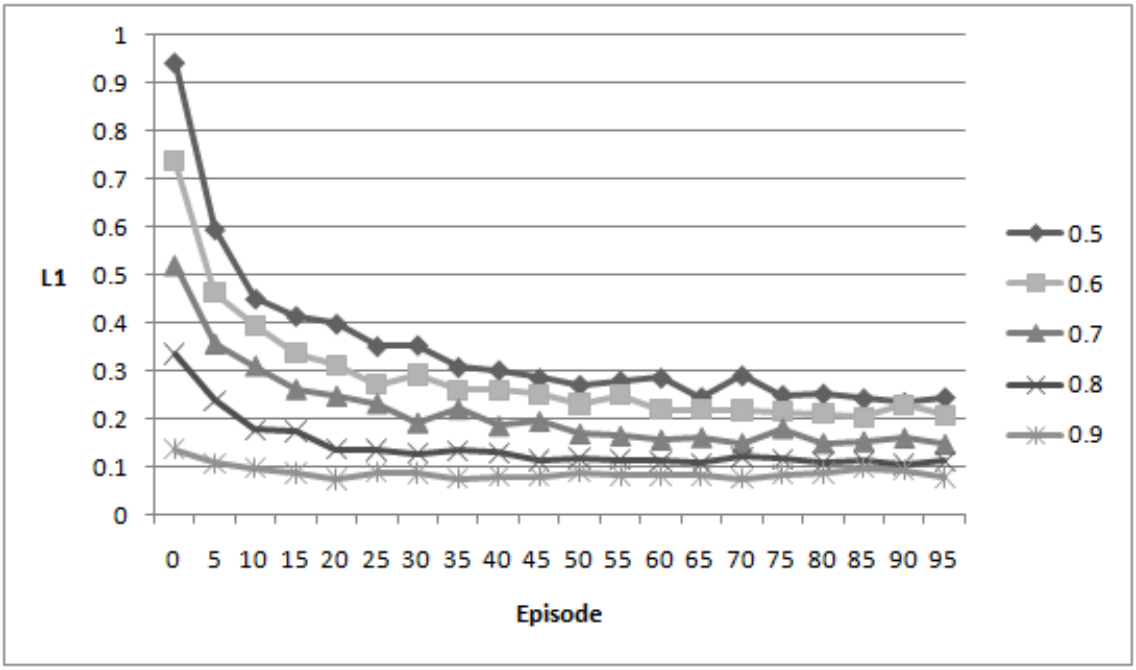


Figure 5–12: SmartWheeler: L1-distance against different priors with symmetry

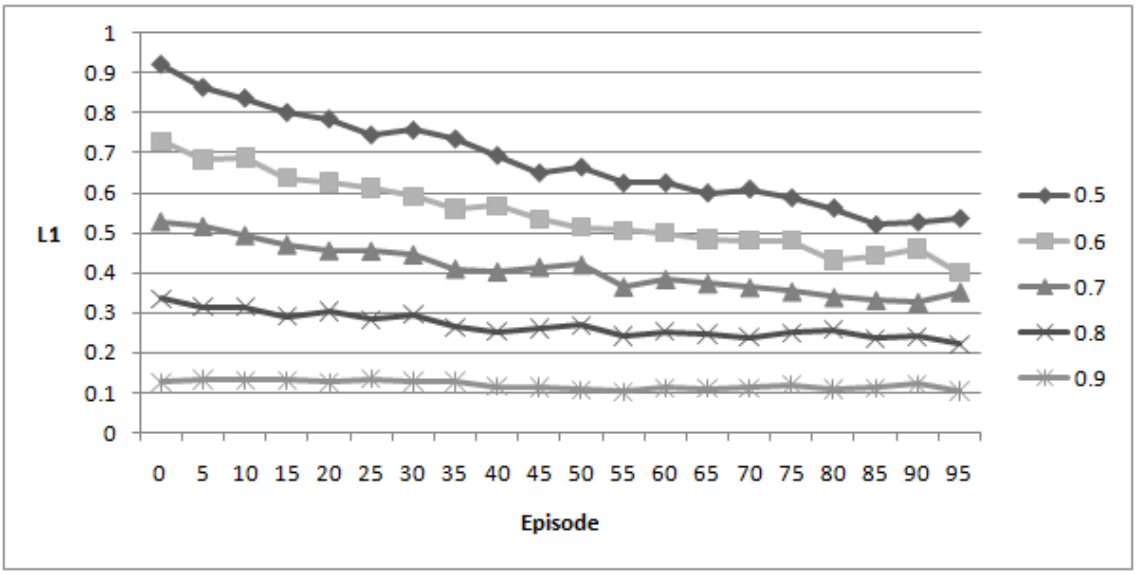


Figure 5–13: SmartWheeler: L1-distance against different priors without symmetry

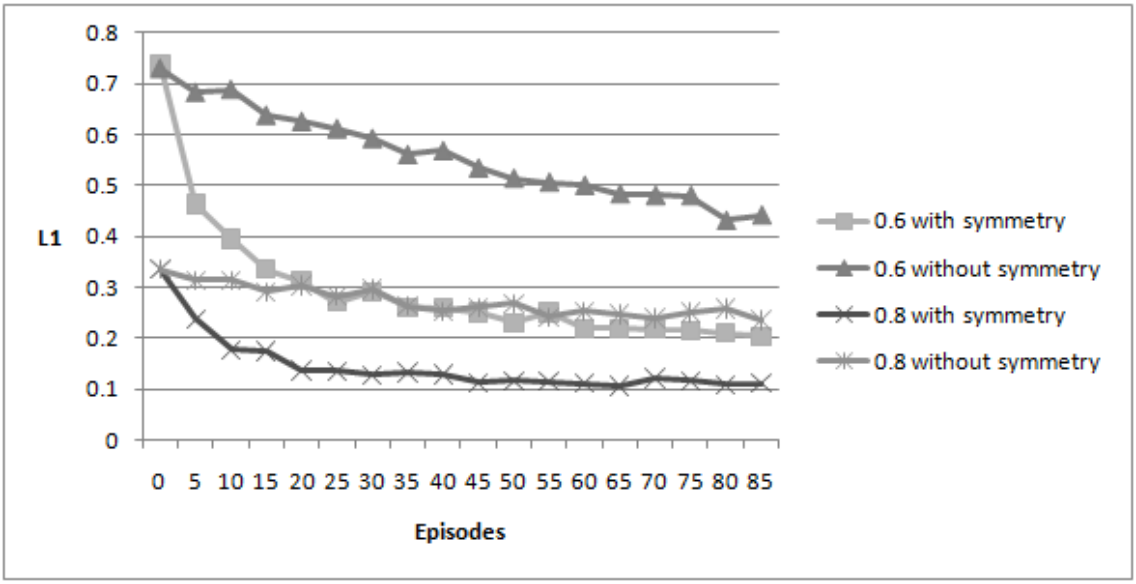


Figure 5–14: SmartWheeler: L1-distance against different priors with and without symmetry

problem. Even with poor intial estimation of the observation parameters (poor priors), we obtain good convergence to the presumed optimal returns and good model accuracy, assuming we leverage symmetry when updating the observation parameter counts. This again indicates that for learning in a dialogue system, it might be more effective to look for symmetry in the model than to come up with more accurate initial priors.

5.6 Discussion

We have demonstrated that by exploiting certain known components of the dialogue system, such as knowledge of symmetrical properties, and by using an approximate online planning algorithm (RTBSS) as mentioned in Section 4.1.2, we are able to apply Bayesian RL on several realistic spoken dialogue system domains.

In both Section 5.3 and Section 5.5, we show that using symmetry in the model is very efficient for learning the model parameters. Even when initial priors are more noisy, using parameters tying leads to a faster convergence of the model parameters, leading to a faster convergence in both the returns and the L1-distance. For learning in a dialogue system, it might be more effective to look for symmetry in the model than to come up with more accurate initial priors.

In Section 5.3, we note that the depth of the search may not affect the returns. We should further investigate whether it is necessary and useful to learn how to plan for many decision-making problems. We also learn that using the L1-distance is a better way to judge how well we learn a model than using the Observations-L1-distance.

In Section 5.4, we show that for a particular class of POMDP problems such as the HandWashing problem, noise in the observation parameters does not affect the returns. In such a case, the Observations-L1-Distance can be used to measure the progress of learning. We also illustrate that even though learning occurs in the BAPOMDPs model, we do not obtain better results in terms of the returns.

There are also other issues. In Section 5.3, we notice that the Observations-L1-distance depends only on the number of times the action (with the observation parameters to learn) is performed. This is not surprising. However, it is also possible that an optimal policy may pick such this action infrequently, resulting in a slower learning of the observation parameters. Also, having noisy initial priors might possibly bias the policy such that the action (with the observation parameters to learn)

is not performed at all. In such a case, there is no learning of the observation parameters in each episode, and we may be struck in a local minima.

In all of our problems, we are simply using our BAPOMDP framework on simulated models instead of real deployed dialogue systems. Our planning time for the SmartWheeler domain is 3 seconds, which is possibly slow for most real systems. With expected development of technology, the BAPOMDP model will likely be computationally fast enough for deployment on realistic dialogue systems in the near future. Of course, more planning time results in a better policy. However, the careful use of exploiting the symmetry in the model based on domain knowledge is an important key to scalability.

CHAPTER 6
Conclusion

This chapter concludes the thesis by providing an overview of the work presented and some discussion. Possible improvements to the BAPOMDP model are outlined in Section 6.2.

6.1 Discussion

In this paper, we propose a Bayesian reinforcement learning framework for simultaneous learning and decision making on a robust spoken dialogue management system. We present tractable algorithms for applying this framework in large domains. We also demonstrate the benefits of such an approach on four different dialogue systems including a human-robot interaction task.

This framework is mathematically sound, and the algorithms are tractable on large domains. Even though knowledge engineering in terms of defining the states, actions, priors and rewards, is still a challenge, this is still applicable in many domains.

As voice user interfaces become more ubiquitious in our daily lives, such as in our mobile devices and automated telephone operators, we believe this is a first step towards customizable user-specific interfaces. For example, spoken dialogue systems may be deployed without previous knowledge of variations in speech among different users. One example is the SmartWheeler domain. Previously, when more than one patient wanted to use a voice recognition wheelchair, the wheelchair had

to be trained to reduce the noise due to each patient's unique voice features. Now, with our framework, the voice recognition wheelchair can accomodate multiple users without adapting the algorithm for each individual user.

6.2 Future Work

The BAPOMDPs model has proven to be effective for solving realistic spoken dialogue systems, but there are still many ways to improve the existing framework. For example, in forumulating our problem, we have to express prior background and knowledge. It is not obvious what is the best way for specifying such domain knowledge, and priors defined as Dirichlet counts are not intuitive for people with little mathematical background. It is also easier for users with no technical knowledge to express their domain knowledge in a non-mathematical manner. In this case, our system may have to perform a preprocessing step to convert the domain knowledge into useful priors for the learning.

In this thesis, we do not know certain observation parameters and attempt to learn them with our improved BAPOMDPs framework. In situations where there are many unknown parameters, it is better if we have a method to determine which are the parameters we should learn. It is useful to know, if possible, which parameters are easier to learn, or which parameters are more crucial in obtaining the optimal policy for a particular POMDP problem.

In our experiments, we relied on symmetry in the model to perform parameter tying. Many spoken dialogue systems are likely to have existing symmetry in their models. Currently, we make use of domain knowledge about the system to decide on which parameters to perform parameter tying. However, it is not obvious which

parameters are similar without domain knowledge. It would be very useful if we could detect symmetry in the system while learning the model. It might be possible to keep a history of all the increments to the Dirichlet counts, and observing if any of them have the same rate of increments.

There are also other issues that have been mentioned earlier. We note that in Section 5.3, increasing the depth of the online search (beyond depth of 2) does not seem to improve the returns. This is interesting for further investigations, as it is indicative of whether it is necessary and useful to learn how to plan for many decision-making problems.

We also note that we can only learn the observation parameters for a particular action if that action is performed. However, it is also possible that an optimal policy may pick such an action infrequently, resulting in a slower learning of the observation parameters. Also, having noisy initial priors might possibly bias the policy such that the action (with the observation parameters to learn) is not performed at all. In such a case, there is no learning of the observation parameters in each episode. Perhaps we can improve the framework so that we can adopt a more exploratory approach while learning the model.

6.3 Evaluation with Human Subjects

So far, we have used our BAPOMDP framework on simulated models, where the ground truth is used to generate data. It is useful to conduct further experiments with human subjects. We can use the ground truth as the initial priors instead and use real data for Bayesian learning. We expect the BAPOMDP to be computationally fast enough for deployment in real dialogue systems since our planning time for the

SmartWheeler domain is 3 seconds. However, one problem that we might encounter is the lack of real data for learning. In our experiments, we usually run repeated trials for at least a hundred iterations. Without enough data for at least a hundred repeated trials, we might have problems evaluating our performance due to the large variance in the returns.

In the short-term, it would be interesting to deploy our BAPOMDP framework on the SmartWheeler project. We can then evaluate our framework with real human subjects and validate the framework more thoroughly.

6.4 Conclusion

In this thesis, we have applied Bayesian Reinforcement Learning techniques to realistic POMDP-based dialogue systems. To achieve this, we make enhancements to an algorithm based on the existing BAPOMDP framework in three ways. First, we make use of known symmetry in the model to perform parameters tying. Second, we incorporate a branch-and-bound technique called RTBSS. Last, we make an improvement to the way sampling is done in BAPOMDP. This opens up further work on learning in decision-theoretic dialogue systems.

References

[1] B. Anderson and A. Moore. Active learning for hidden markov models: Objective functions and algorithms. In *International Conference on Machine learning*, 2005.

[2] K.J. Astrom. Optimal control of Markov decision processes with incomplete state estimation. *Journal of Mathematical Analysis and Applications*, 10(1):174–205, 1965.

[3] A. Atrash. *A Bayesian Framework for Online Parameter Learning in POMDPs*. PhD thesis, McGill University, 2011.

[4] A. Atrash and J. Pineau. A bayesian reinforcement learning approach for customizing human-robot interfaces. In *International Conference on Intelligent User Interfaces*, 2009.

[5] D.P. Bertsekas. *Dynamic programming and optimal control*. Athena Scientific Belmont, MA, 1995.

[6] D.P. Bertsekas and D.A. Castanon. Rollout algorithms for stochastic scheduling problems. *Journal of Heuristics*, 5(1):89–108, 1999.

[7] J. Boger, P. Poupart, J. Hoey, C. Boutilier, G. Fernie, and A. Mihailidis. A decision-theoretic approach to task assistance for persons with dementia. In *International Joint Conference on Artificial Intelligence*, 2005.

[8] A. Boularias, H.R. Chinaei, and B. Chaib-draa. Learning the Reward Model of Dialogue POMDPs from Data. In *NIPS Workshop on Machine Learning for Assistive Techniques*, 2010.

[9] A.R. Cassandra, L.P. Kaelbling, and M.L. Littman. Acting optimally in partially observable stochastic domains. In *National Conference on Artificial Intelligence*, 1995.

[10] M. Danieli and E. Gerbino. Metrics for evaluating dialogue strategies in a spoken language system. In *AAAI Spring Symposium on Empirical Methods in Discourse Interpretation and Generation*, 1995.

[11] R. Dearden, N. Friedman, and D. Andre. Model based bayesian exploration. In *Uncertainty in Artificial Intelligence*, 1999.

[12] M. Denecke, K. Dohsaka, and M. Nakano. Learning dialogue policies using state aggregation in reinforcement learning. In *International Conference on Spoken Language Processing*, 2004.

[13] F. Doshi, J. Pineau, and N. Roy. Reinforcement learning with limited reinforcement: Using bayes risk for active learning in POMDPs. In *International Conference on Machine Learning*, 2008.

[14] Y.Z. Du, D. Hsu, H. Kurniawati, W.S. Lee, S.C.W. Ong, and S.W. Png. A POMDP approach to robot motion planning under uncertainty. In *ICAPS Workshop on POMDP Practitioners Workshop: solving real-world POMDP problems*, 2010.

[15] M.O.G. Duff. *Computational Procedures for Optimal Learning*. PhD thesis, University of Massachusetts Amherst, 2002.

[16] M. M Fard, J. Pineau, and P. Sun. A variance analysis for POMDP policy evaluation. In *Association for the Advancement of Artificial Intelligence*, 2008.

[17] K. Georgila, J. Henderson, and O. Lemon. User simulation for spoken dialogue systems: Learning and evaluation. In *International Conference on Spoken Language Processing*, 2006.

[18] D. Goddeau and J. Pineau. Fast reinforcement learning of dialog strategies. In *International Conference on Acoustics, Speech and Signal Processing*, 2000.

[19] M. Hauskrecht. Value-function approximations for Partially Observable Markov Decision Processes. *Journal of Artificial Intelligence Research*, 13:33–94, 2000.

[20] R.A. Howard. *Dynamic programming and Markov process*. Technology Press of Massachusetts Institute of Technology., 1960.

[21] R. Jaulmes, J. Pineau, and D. Precup. Active learning in Partially Observable Markov Decision Processes. 2005.

[22] L.P. Kaelbling, M.L. Littman, and A.R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134, 1998.

[23] S. Koenig. Agent-centered search. *Association for the Advancement of Artificial Intelligence AI Magazine*, 22(4):109, 2001.

[24] H. Kurniawati, D. Hsu, and W.S. Lee. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Robotics: Science and Systems*, 2008.

[25] E. Levin, R. Pieraccini, and W. Eckert. Learning dialogue strategies within the Markov decision process framework. In *Automatic Speech Recognition and Understanding*, 1997.

[26] D. Litman, S. Singh, M. Kearns, and M. Walker. NJFun: a reinforcement learning spoken dialogue system. In *ANLP/NAACL Workshop on Conversational Systems*, 2000.

[27] M.L. Littman, A.R. Cassandra, and L. Pack Kaelbling. Learning policies for partially observable environments: Scaling up. In *International Conference on Machine Learning*, 1995.

[28] R. Nisimura, T. Uchida, A. Lee, H. Saruwatari, K. Shikano, and Y. Matsumoto. Aska: Receptionist robot with speech dialogue system. In *International Conference on Intelligent Robots and Systems*, 2002.

[29] S. Ong, S. Png, D. Hsu, and W. Lee. POMDPs for robotic tasks with mixed observability. In *Robotics: Science and Systems*, 2009.

[30] S. Paquet. *Distributed Decision-Making and Task Coordination in Dynamic, Uncertain and Real-Time Multiagent Environments*. PhD thesis, Université Laval, 2006.

[31] S. Paquet, B. Chaib-draa, and S. Ross. Hybrid POMDP algorithms. In *AAMAS Workshop on Multi-Agent Sequential Decision Making in Uncertain Domains*, 2006.

[32] S. Paquet, L. Tobin, and B. Chaib-draa. Real-time decision making for large POMDPs. In *Advances in Artificial Intelligence*, 2005.

[33] J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In *International Joint Conference on Artificial Intelligence*, 2003.

[34] S. Png and J. Pineau. Bayesian reinforcement learning for POMDP-Based dialogue systems. In *International Conference on Acoustics, Speech and Signal Processing*, 2011.

[35] P. Poupart, N. Vlassis, J. Hoey, and K. Regan. An analytic solution to discrete bayesian reinforcement learning. In *International Conference on Machine Learning*, 2006.

[36] M.L. Puterman. *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons, Inc., 1994.

[37] S. Ross, B. Chaib-draa, and J. Pineau. Bayes-Adaptive POMDPs. In *Neural Information Processing Systems*, 2007.

[38] S. Ross, J. Pineau, S. Paquet, and B. Chaib-Draa. Online planning algorithms for POMDPs. *Journal of Artificial Intelligence Research*, 32(1):663–704, 2008.

[39] N. Roy, J. Pineau, and S. Thrun. Spoken dialogue management using probabilistic reasoning. In *Association for Computational Linguistics*, 2000.

[40] J. Schatzmann, K. Weilhammer, M. Stuttle, and S. Young. A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *The Knowledge Engineering Review*, 21(02):97–126, 2006.

[41] G. Shani, R. Brafman, and S. Shimony. Model-based online learning of POMDPs. *European Conference on Machine Learning*, 2005.

[42] D. Silver and J. Veness. Monte-Carlo Planning in Large POMDPs. In *Neural Information Processing Systems*, 2010.

[43] S.P. Singh, D.J. Litman, M.J. Kearns, and M.A. Walker. Optimizing dialogue management with reinforcement learning: Experiments with the NJFun system. *Journal of Artificial Intelligence Research*, 16(1):105–133, 2002.

[44] R.D. Smallwood and E.J. Sondik. The optimal control of Partially Observable Markov processes over a finite horizon. *Operations Research*, 21(5):1071–1088, 1973.

[45] T. Smith and R. Simmons. Heuristic search value iteration for POMDPs. In *Uncertainty in Artificial Intelligence*, 2004.

[46] E.J. Sondik. *The Optimal Control of Partially Observable Markov Processes.* PhD thesis, Stanford University, 1971.

[47] M.T.J. Spaan and N. Spaan. A point-based POMDP algorithm for robot planning. In *International Conference on Robotics and Automation*, 2004.

[48] M.A. Walker. An application of reinforcement learning to dialogue strategy selection in a spoken dialogue system for email. *Journal of Artificial Intelligence Research*, 12(1):387–416, 2000.

[49] M.A. Walker, J.C. Fromer, and S. Narayanan. Learning optimal dialogue strategies: A case study of a spoken dialogue agent for email. In *International Conference on Computational linguistics*, 1998.

[50] M.A. Walker, D.J. Litman, C.A. Kamm, and A. Abella. PARADISE: a framework for evaluating spoken dialogue agents. In *Association for Computational Linguistics*, 1997.

[51] J.D. Williams, P. Poupart, and S. Young. Partially Observable Markov Decision Processes with continuous observations for dialogue management. *Recent Trends in Discourse and Dialogue*, 2008.

[52] J.D. Williams and S. Young. Partially Observable Markov Decision Processes for spoken dialog systems. *Computer Speech & Language*, 21(2):393–422, 2007.

[53] J.D. Williams and SJ Young. The SACTI-1 corpus: guide for research users. 2005.

[54] S.J. Young. Probabilistic methods in spoken–dialogue systems. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 358(1769):1389, 2000.