Optimising Learning with Transferable Prior Information

by

Funlade Tajudeen Sunmola

A THESIS SUBMITTED TO THE FACULTY OF SCIENCE OF THE UNIVERSITY OF BIRMINGHAM FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

Intelligent Robotics Laboratory
School of Computer Science
University of Birmingham
Birmingham
United Kingdom
April 2012

Abstract

This thesis addresses the problem of how to incorporate user knowledge about an environment, or information acquired during previous learning in that environment or a similar one, to make future learning more effective. The problem is tackled within the framework of learning from rewards while acting in a Markov Decision Process (MDP). Being able to appropriately incorporate user knowledge and prior experience into learning is useful because it should lead to better performance during learning (the exploitation-exploration trade-off), and offer a better solution at the end of the learning period. In this thesis, we show how to incorporate both user knowledge and prior experience of the learning agent in an integrated framework for transfer learning.

In general, transfer learning exploits prior experience on different but related tasks to improve performance while learning. It is conventional wisdom that when faced with new tasks, learners seek to improve performance by transferring prior experience from related tasks. In this thesis, we focus on the use of explicitly available prior information about the transition matrix of the MDP. We work in a Bayesian setting and consider two main types of transferable information namely historical data and user knowledge expressed about constraints involving absolute and relative restrictions on process dynamics. We study situations where there exists a pro-forma prior (a.k.a. recipient) distribution formulated for a new task alongside pre-prior (a.k.a. donor) information that may include historical data and transition constraints. We present new algorithms for reasoning with the transition constraints based on Monte Carlo sampling and maximum likelihood methods that include sampling from truncated Dirichlet-distributed transition models and parameter inference for ordered transition parameters.

In addition to showing how to revise beliefs about the MDP transition matrix using constraints and prior knowledge, we also show how to use the resulting beliefs to control exploration. First we extend an existing algorithm for the exploration-exploitation trade-off called Optimistic Model Selection to work with constrained Dirichlet models. Second we present a new exploration control algorithm that allows us to select optimistic models from these constrained densities whilst incorporating historical data via power priors. Power priors use a relative precision parameter to control the influence of historical data relative to new observations. One question is how to set this precision parameter. We show how the constraints can be used to acquire a precision parameter. Essentially the historical data is valued to the extent that it matches the user knowledge about constraints on the transition matrix for the new task. The algorithm works by measuring the degree of match between the historical data and imaginary data drawn from the user specified constraints.

We apply our algorithms to grid-world MDPs and to a model of a maintenance intervention task, in order to show how user constraints might in practice be generated. To demonstrate the benefits of historical information we also show how to use process templates to transfer information from one environment to a second with related local process dynamics. We present results showing that incorporating historical data and constraints on state transitions in uncertain environments, either separately or collectively, can improve learning performance.

Acknowledgements

This work could not have been undertaken without God's blessing and support from a number of people. I am deeply grateful to my thesis supervisor Dr Jeremy Wyatt, Reader in AI & Robotics and Head of Intelligent Robotics (IR) Laboratory, University of Birmingham UK. I very much appreciate his support, advice and inspiration throughout my research - working with Jeremy has been a real pleasure.

I thank members of my thesis committee consisting of Professor Ela Claridge, Dr Ata Kaban and my supervisor Jeremy for all of their time and effort during my thesis group meetings. The IR Lab is a great place to work, study and enjoy academia in a friendly environment that also provides administrative, financial, and logistic support.

My special thanks to my family - my wife Mrs Olabisi Olufunmilayo Sunmola, my children Omolade and Olufadebi, nieces and nephews, brothers and sisters including Professor Adegbenga & Mrs Temitope Sunmola, Professor Adewale & Professor Morenike Dipeolu, Dr Timothy & Mrs Opeyemi Osadiya, Mrs Yetunde Oshodi, Dr Bolanle Abudu, Ms. Titilayo Oshodi, Mr Adeniran & Mrs Mojisola Paul, Ms. Yejide Oluwakemi Oshodi, Dr Gbenga & Mrs Modupe Ojumu, Mr Babajide & Mrs Yetunde Oyesoji Olaleye and also my wonderful friends including in the English Midlands particularly Dr Adesegun & Dr Titilola Abudu, Dr Adedeji & Mrs Oludotun Okubadejo, Dr Olaide & Mrs Mojisola Olaoye, Dr Olayiwola & Mrs Omolara Olawale and elsewhere, too numerous to list here, for their encouragement.

Finally, I am indebted to my parents Chief Olaitan & Chief Olufadebi Sunmola for starting me out in life with sound training and sponsorship to an excellent engineering school (ABU Zaria) for my first degree in Civil Engineering, and for allowing me to realize my own potential. Their support over the years has been priceless. My thanks also go to my father-in-law Chief Theophilus Oshodi. This thesis is dedicated to my Parents. To God be the Glory.

Outline of the Thesis

Contents

List of Algorithms

List of Figures

List of Tables

1 Introduction

This thesis explores approaches to learning in sequential decision processes when opportunities exist for exploiting prior information from diverse sources. In particular, the thesis is geared towards endowing agents with ability to autonomously recombine prior information from separate and possibly diverse sources in order to help them learn more quickly and effectively on a new task in unknown or partially known environments. A primary motivation is to reduce the total number of data, resources and learning steps needed to establish the models of the new process necessary to find an optimal or near optimal way of behaving. This chapter introduces the research that lead to this thesis, summarises the problem and approach of the thesis, the thesis contributions and its structure.

1.1 Background

Many real world tasks are sequential in nature and involve decisions under uncertainty, where actions do not entail certain outcomes. The outcomes are often partly random and partly under the control of the decision maker (i.e. an *'agent'*). Examples include a robot navigating through hospital wards, attending to patients in emergencies, deciding on a sequence of actions to take, and making up its mind on where and when to dock. The sequential nature of the tasks offers an opportunity to use accumulating experience to make advantageous and valuable decisions as learning progresses. At the heart of many computational techniques used for accomplishing sequential decision making lie the closely related frameworks of *Markov Decision Process* (MDP) and *Reinforcement Learning*

(RL).

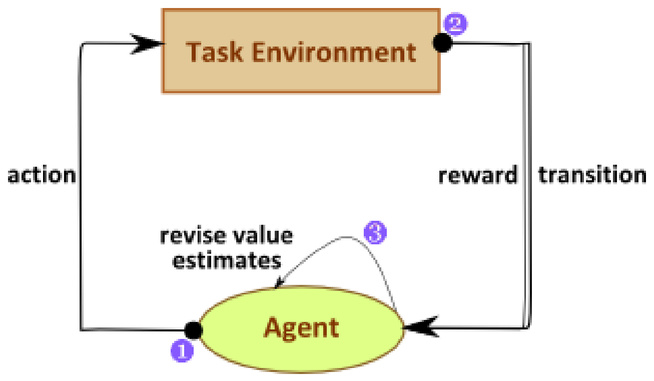
An MDP is a stochastic control process characterized by a set of states that can be perceived exactly, actions, and a probabilistic transition function that specifies the probabilities of moving from one state to the next. In MDPs after each transition the system moves into a new state, in which one can choose an action, which may incur an immediate reward (i.e. revenue or cost) and which, in addition, affects the next state transition. Essentially, the process is Markovian in the sense that, the outcome of applying an action to a state depends only on the current state and the current action. This means that if the current state of the MDP at time t is known, transitions to a new state at the next time step $(t+1)$ are independent of all previous states.

The Markov framework (of which MDPs are a part) is rich in capturing the essence of purposeful activity in a wide variety of tasks and can be quite expressive, as demonstrated by structured formalisms such as factored (Boutilier, Dean & Hanks 1999, Boutilier, Dearden & Goldszmidt 2000, Guestrin, Patrascu & Schuurmans 2002) and relational (Dzeroski, De Raedt & Driessens 2001) MDPs. Increasingly, the framework is finding considerable popularity in artificial intelligence most especially machine learning and sequential decision making with applications in a variety of areas, including engineering, economics, management and medicine (White 1985, White 1988, White 1993, Feinberg & Shwartz 2002).

Solving a known MDP, i.e. one in which transition and reward functions are available directly to the agent, translates simply to finding an optimal policy such that the expected value (such as total discounted reward) will be maximized. Generally, a policy, optimal or not, is a rule that specifies for each state how to choose an action given a state. For example, a policy in the hospital robot we mentioned earlier could be specified as follows: if robot is idle, has low fuel and is positioned near a docking station then dock. Executing the dock action could, with a probability, change the robot state to one in which the robot is switched off, still has low fuel and is docked in the docking station. On transition to the new state, the robot may be rewarded for not wasting fuel given that it is idle.

Unfortunately, the transition dynamics required for the MDP are rarely known precisely in practical applications. Whilst MDPs underlying real world tasks may be fully observable, they are often characterized by non-stationarity in the world, and knowledge of the actual transition probabilities and reward functions that has limited precision. For tasks involving MDPs with unknown or partially known transition and/or reward functions, the problem becomes one of reinforcement learning (Sutton & Barto 1998).

Reinforcement learning is the problem faced by an agent that must learn behaviour through either systematic or trial-and-error interactions with an environment (Kaelbling, Littman & Moore 1996, Wyatt 2002). As illustrated in Figure 1.1, an agent learning from reinforcement would perform actions in states, observe transitions and rewards, and use the accrued experience to revise value and policy estimates. Solving an unknown or partially known MDP, via reinforcement learning, may



1. The agent uses an appropriate *action selection* method to choose an action to perform based upon its current value and policy estimates.
2. The agent *acquires experience* by observing the effects of its action in the task environment.
3. The agent uses the experience it acquires to *revise its value estimates* and *update its policy*.

Figure 1.1: An illustration of Reinforcement Learning Framework

be tackled by first inferring a model of the process to determine the unknown transition and reward functions and then acting using the model. An alternative is to follow a direct reinforcement learning approach that relies on an ability to solve the underlying Markov decision process without computing transition probabilities. The two approaches differ fundamentally - the former is model-based while the latter is model-free. Model based reinforcement learning approaches are advantageous in domains where real-world actions are expensive and computation time is relatively cheap.

For several decades, researchers have studied techniques for inferring process models and success have been recorded, most notably and recently, in new algorithms for dealing with large state spaces with inherent structure (Boutilier et al. 1999). While progress in modelling stochastic processes has been considerable, it has also exposed a number of challenging and complementary problems one of which centres on how to take full advantage of useful information about known, previously solved, tasks in order to improve performance on new tasks in unknown environments. This particular challenge, often generally referred to as *'Transfer of Learning'* turns out to be quite formidable for MDPs, especially in unknown or partially known task environments.

The goal of the research documented in this thesis is to extend the study of MDPs to encompass transfer of learning. Specifically we study approaches to learning process models for MDP-based tasks in unknown environments when opportunities exist for exploiting prior information from diverse sources. Beyond studying whether learners can transfer what they have learnt from one task to another, we also explore the techniques underlying how such transfer takes place. This chapter contains an introduction to the thesis. It covers the motivation for the research, a statement of the research problem and solution approach, thesis contributions and an outline of the thesis structure. In the next two sections, we explain the challenges of learning a new task and introduce a way of addressing the challenges.

1.1.1 Challenges of Learning

Learning new tasks in unknown environments can be quite challenging. Learning, in this sense, refers to changes to the decision maker's perception and motor skills based on their experience and interaction with their environment leading to improvements in their expected future performance. There are multiple impediments including the obvious ones such as barriers to learning imposed by poor quality models and inability to make inferences about the dynamics of an unknown environment until enough data is acquired over time through interactions. Reinforcement learners typically face the following challenges.

Exploration-Exploitation tradeoffs: the learner needs to decide actions to perform while maximising current value and simultaneously seeking to benefit appropriately from learning opportunities. The agent must balance exploitation of the knowledge it already has with exploration in the hopes of learning something or discovering new knowledge that might lead to better performance in the future.

Bellman's curse of dimensionality: many interesting environments in which agents may perform do display structure that should allow them to model and reason in a compact way. Using structure allows agents to exploit prior knowledge about the regularity of the world. For example, knowing that the effect of actions in two or more different states are similar may allow one to make simultaneous inference about behaviour in those states. Enumerated state MDP approaches do not account for structure and as the number of domain variables grows the number of states increases exponentially. This phenomenon is known as *Bellman's curse of dimensionality* (Bellman 1957). Through efficient aggregation of domain variables, structure provides opportunities to dampen the curse of dimensionality.

Incorporating user and/or designer knowledge: when user and/or designer knowledge on rewards, policy, values and transition functions are available to the agent they should be incorporated into learning. A potential drawback of MDPs and reinforcement learning is their inability to easily incorporate user and/or designer knowledge into an agent's learning algorithm in order to improve performance.

Clearly, agents require the ability to deal with these three challenges simultaneously to allow them to systematically develop their knowledge of unknown environments and productively reuse their experiences for new tasks. In our world, such capabilities matter immensely – it is a fundamental and fascinating aspect of human learning. No one would seriously think of trying to solve any problem in everyday life without bringing to bear their prior experiences and, equally importantly, those of others.

1.1.2 Transferable Prior Information

The previous section alludes to one of the most important premise for success in life which is the fundamental idea that when learning to accomplish a new task the learner should take account of all he/she already knows in order to decide the next course of action. This includes both building upon his/her prior learning and exploiting strength borrowed from the experiences of others. The underpinning philosophy is that new knowledge builds on the old and can be directed to positively influence performance. As the saying often goes *'one cannot learn anything unless one almost knows it already'* (Winston 1984). For us, in this thesis, this philosophical saying influences our approach to two important and significantly intertwined subjects: a) the integration of rich and diverse sources of prior information into learning processes, and b) model transfer – the significance of which cannot be over-emphasised.

1.1.2.1 Possible Types of Transferable Prior Information

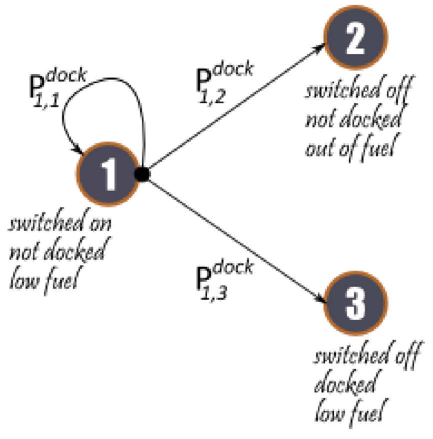
In realistic decision making tasks, there often is access to transferable (i.e. explicitly available) prior information that is relevant to the task at hand and exploiting such information wisely can lead to improved performance. The efficient use of prior information is seldom trivial and it is one of the main differences between humans and machines. The principal issues involve: a) the striking of a balance between the type and amount of built in versus learned information, and b) an ability to harness prior information emerging from diverse sources. There is in addition the paradoxical nature of prior knowledge to contend with, that is, inaccurate knowledge hinders development, and lack of it makes progress impossible or inefficient (Pintrich, Marx & Boyle 1993).

Machine learning has emphasised the importance of prior information, with an understanding that the incorporation of (implicit or explicit) priors is fundamental to any learning situation where we have a hope of being successful. This understanding motivates the development of learning systems which capitalise efficiently on pieces of prior information. The importance of prior knowledge in machine learning is summarised in the much acclaimed no free lunch theorem. Loosely, the no free lunch theorem states that all search algorithms have the same average performance over all problems, and thus implies that to gain in performance on a certain application one must use a specialized algorithm that includes some prior knowledge about the problem (Wolpert & Macready 1997).

Knowledge comes in many forms. For example, we could be provided with prior knowledge about: a) relevance of certain other variables that can help us ignore certain variables when building our model, b) preferences such as constraints on model parameters that can both guide our search over possible models and speed up inference, and c) uncertainties and lack of accuracy of our models. These knowledge forms often come from diverse sources that can include:

Historical Data: from previous tasks is available when there is access to recorded experiences. When such information is available, it is natural to incorporate it into learning process of a new task by quantifying the information with appropriate prior distribution.

Constraints (Absolute or Relative): that are expressions of conditions to be satisfied or hypotheses regarding aspects of a task environment. The expressions are usually established on the basis of physical, theoretical, customary and economic considerations, amongst others. They convey a degree of determinism in uncertain task environments. For example, consider the hospital robot we mentioned earlier and assume that the robot's environment is characterised by a model of system dynamics we illustrate in Figure 1.2. Reasoning with



Transition constraints:

$0.87 \leq p^{dock}_{1,3} \leq 0.97$; i.e. $p^{dock}_{1,3} = 0.92 \pm 0.05$

$p^{dock}_{1,3} \geq 10.0 p^{dock}_{1,2} \geq p^{dock}_{1,1}$

$p^{dock}_{1,1} = 0.07$

$0.0 \leq p^{dock}_{1,2} \leq 1.0$

$p^{dock}_{1,1} + p^{dock}_{1,2} + p^{dock}_{1,3} = 1.0$

Figure 1.2: An example of Transition Constraints

the transition constraints specified by the model can help simplify inferences and sometimes permit exact deductions. In the example, we can reasonably infer that $0.007 \leq p_{1,2}^{dock} \leq 0.06$, $0.87 \leq p_{1,3}^{dock} \leq 0.923$ with bounds on $p_{1,2}^{dock}$ significantly tighter in comparison to the conventional $[0,1]$ probability bound we would ordinarily have assumed.

The availability of transferable prior information should encourage the learner to value and harness the differences in the information set, drawing upon the widest possible range of views and experiences, to improve learning performance. This can be accomplished through model transfer.

1.1.2.2 Model Transfer

Model transfer concerns the ability to learn more quickly and develop a deeper understanding of the task environment by bringing some knowledge or skills from models of related tasks. Model transfer provides an alternative to undertaking complete data collection and model development in every task instance. This helps an agent to acquire new views on a task by looking at the task from a different approach, which strengthens our understanding of the task. Not only do such capabilities help speed up decision making processes they also provide solid basis for effective decisions. Model transfer provides a convenient approach to the problem of incorporating parameter domain knowledge into model estimation steps. A primary motivation for model transfer is that it helps in reducing the total number of data, resources and learning steps needed to establish the models of related processes. Model transfer also facilitates the use of prior distributions formulated for simpler, economical, and more easily understood models to modify the prior distribution of a more complex model (Neal 2001).

Model transfer can be thought of as transfer learning in the sense that it leverages learned knowledge in one context to enhance learning in different contexts and thus establish a channel between previous isolated learned tasks. A variety of model transfer studies have been reported including in engineering (Schaible 1999, Karasmaa 2003), statistical analysis and modelling (Koppelman, Kuah & Wilmot 1985, Neal 2001). Model transfer in reinforcement learning is a much more recent subject and its conspicuous absence in literature is posited by Price (2003) in his study of imitation in reinforcement learning domains. Transfer learning research in MDPs include, for example, Singh (1991) on transfer across composition of sequential tasks, Dixon, Malak & Khosla (2000) on incorporating prior knowledge and previously learned information into reinforcement learning agents, Tanaka & Yamamura (2003), Wilson, Fern, Ray & Tadepalli (2007) on multitask reinforcement learning, Sunmola & Wyatt (2006) on model transfer in Markov decision processes using parameter matching techniques, Taylor (2008) on autonomous inter-task transfer in reinforcement learning, Taylor, Jong & Stone (2008) on transferring of instances in model based reinforcement learning, Lazaric, Restelli & Bonarini (2008), Lazaric (2008) on the transfer of samples in batch reinforcement learning, Torrey (2009) on relational transfer based largely on advice taking, Castro (2011) on the use of bisimulation for policy transfer in MDPs, and a survey of publications in the subject area (Taylor & Stone 2009).

1.2 About this Thesis

In this section we informally describe the problem of the thesis and describe the hypothesis we explore in this thesis. In addition we present the thesis contributions and its structure.

1.2.1 Problem and Approach of this Thesis

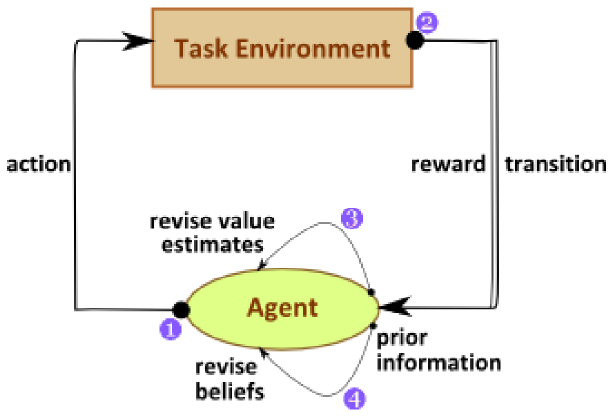
We informally describe the problem of this thesis as follows. Our agent has knowledge of transitions for a set of related tasks. The agent is presented with a new task and a new reward function but do not yet have a policy and a model for that task. The problem is to transfer relevant information from the models of the previous tasks to help improve its performance on the new task in comparison to learning the new task from scratch. We focus in this thesis on transition knowledge and consider that the transition knowledge from previous tasks is available in two main forms namely historical data and constraints involving absolute and relative restrictions on process dynamics. We consider situations where there exists a pro-forma prior (a.k.a. recipient) distribution formulated for a new task alongside pre-prior (a.k.a. donor) information that may include historical data and transition constraints derived from related tasks. We make a simplifying assumption that the transition constraints are known and consistent with the new task environment. What we do not know is how to appropriately reason with the transition constraints, the historical data and the agent's pro-forma prior information in order to optimise learning performance on a new task. The following characterises the thesis.

- We introduce inferential bias into the learner. We focus primarily on an *'Optimistic'* agent that uses a principle of maximum expected utility but with biases in its probabilistic estimates of transitions.
- We utilise a variety of types of prior knowledge available to the agent, and
- We use a Bayesian paradigm for information transfer.

In integrating various sources of information we adopt the standard Bayesian methodology for sequential decision making, Figure 1.3, shows how to incorporate prior information about transitions into reinforcement learning. The Bayesian approach arises naturally when information is available *a priori* for planning and estimation. The Bayesian approach has several advantages over other methods. First, it allows natural incorporation of priors over transition and reward parameters. Second, it provides an elegant solution to the exploration/exploitation problem arising from action selection especially in structured domains. A downside however is that the Bayesian solution to the problem is intractable because it involves dynamic programming over a tree of information states. As a result, approximate solutions have been popular due largely to the availability of fast computing and advance in stochastic sampling algorithms e.g. Dearden (2000), Strens (2000),Wyatt (2001),Castro & Precup (2007), and Asmuth & Littman (2011).

1.2.2 Contributions

The research situates model transfer in a Bayesian framework for learning Markov decision tasks leading to the transfer learning methodology we propose and study in this thesis. The methodology is illustrated in Figure 1.4.



1. The agent uses an appropriate *action selection* method to choose an action to perform based upon its current value and policy estimates.
2. The agent *acquires experience* by observing the effects of its action in the task environment.
3. The agent uses the experience it acquires to *revise its value estimates* and *update policy*.
4. The agent *revises its beliefs* about transition probability and rewards based on its *prior information* and *experience*.

Figure 1.3: Bayesian Learning in Markov Decision Processes

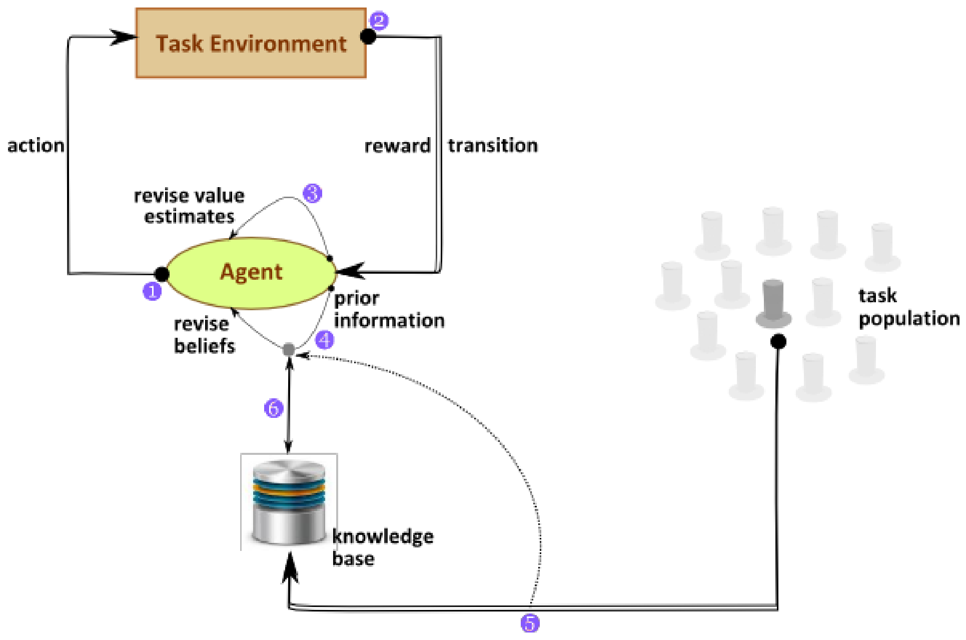
Based on the framework of Figure 1.4 we explore the following hypotheses:

1. Extending power prior analysis to Markov chains allows learners to incorporate historical information in a sensible way. Power priors provide a useful class of informative priors for Bayesian inference. This leads to better performance on some new tasks while learning (less regret).

2. Incorporating absolute constraints on state transitions in uncertain Markov decision process environments can improve performance.

3. Incorporating relative constraints on state transitions in uncertain environments modelled as Markov decision processes can improve performance.

4. Integrating historical data with transition constraints improves learning performance still further.



1. The agent uses an appropriate *action selection* method to choose an action to perform based upon its current value and policy estimates.
2. The agent *acquires experience* by observing the effects of its action in the task environment.
3. The agent uses the experience it acquires to *revise its value estimates* and *update policy*.
4. The agent *revises its beliefs* about transition probability and rewards based on its *quantification of prior information* and *experience*.
5. Information from the task population and their models are transferred to the new task and added to knowledge base.
6. Information may flow between the new task environment and the knowledge base.

Figure 1.4: A Framework for Optimising Learning with Transferrable Prior Information

leading to the following contributions of this thesis:

1. We present a Bayesian methodology for incorporating transferable prior information into steps for learning Markov decision tasks. We work in a Bayesian setting and consider two main types of transferable information namely historical data and user knowledge expressed about constraints involving absolute and relative restrictions on process dynamics. We study situations where there exists a pro-forma prior (a.k.a. recipient) distribution formulated for a new task alongside pre-prior (a.k.a. donor) information that may include historical data and

transition constraints.

2. We present new algorithms for reasoning with the transition constraints based on Monte Carlo sampling and maximum likelihood methods that include sampling from truncated Dirichlet-distributed transition models and parameter inference for ordered transition parameters.

3. In addition to showing how to revise beliefs about the MDP transition matrix using constraints and prior knowledge, we also show how to use the resulting beliefs to control exploration.

    - First we extend an existing algorithm for the exploration-exploitation trade-off called Optimistic Model Selection (OMS) to work with constrained Dirichlet models. We describe how OMS can be formulated as a lexicographic goal program that allows us to straightforwardly incorporate a variety of transition constraints into the model selection steps.

    - Second we present a new exploration control algorithm that allows us to select optimistic models from these constrained densities whilst incorporating historical data via power priors. Power priors use a relative precision parameter to control the influence of historical data relative to new observations. One question is how to set the precision parameter required in power prior analysis. We show how the constraints can be used to acquire a precision parameter. Essentially the historical data is valued to the extent that it matches the user knowledge about constraints on the transition matrix for the new task. The algorithm works by measuring the degree of match between the historical data and imaginary data drawn from the user specified constraints.

4. We apply our algorithms to grid-world MDPs and to a model of a maintenance intervention task, in order to show how user constraints might in practice be generated. To demonstrate the benefits of historical information we also show how to use process templates to transfer information from one environment to a second with related local process dynamics.

1.2.3 Structure of the Thesis

The thesis is structured into five parts - *Research background, Exploration Control and Model Selection, Transfer Methodology, Transfer Techniques,* and *Conclusions*.

Part one contains three chapters – one to three – consisting of the concepts and background material that constitute the foundation we build upon in subsequent chapters of the thesis. This chapter, i.e. chapter one, contains an introduction to the thesis. In chapter two we describe the framework of Markov decision processes and look at techniques for solving Markov Decision tasks when there is a complete model of the task environment. The chapter also describe methods of learning how to solve Markov decision tasks when a complete model of the task environment is not available. In chapter three, we look more closely at the issue of uncertainty in system dynamics and describe Bayesian approaches to learning MDPs in uncertain environments.

Part two consists of one chapter - four - that presents procedures we use in this thesis for selecting models in Bayes adaptive Markov decision processes. The selected model is used as a vehicle for explaining and predicting observations, estimating value functions, controlling exploration and deciding how to behave. We describe in the chapter an existing algorithm for the exploration-exploitation trade-off called Optimistic Model Selection (Wyatt 2001) and describe how optimistic model selection can be formulated as a lexicographic goal program to allow it to be extended to handle constraints on transition probability.

Part three consists of two chapters – five and six – with a focus on the methodology of transferring prior information from previous tasks to new Markov decision tasks. In chapter five we review existing approaches to prior information transfer. In the context of model transfer, we review probabilistic expressions that underpin the transfer approaches we adopted in the thesis. Next, in chapter six, we present our transfer framework. We described in the chapter the transfer context i.e. the domain, task, and specific types of the transferable prior information and explain how we incorporate the transferable prior information into Bayesian steps for estimating expected returns.

Part four consists of three chapters – seven, eight and nine – which, in addition to chapter four, represent the main contributions of this thesis. In chapter seven we turn to the problem of reasoning with transition constraints and describe constrained Bayesian inference methods for accomplishing this. We also present in the chapter empirical results that support our hypotheses that constraints on state transitions in uncertain environments can improve performance. In chapter eight we describe power priors as a means of discounting historical information in Markov chains. In the chapter, we focus on the use of historical data from previous tasks in the construction of prior distributions for a new Markov decision task. The construction is accomplished through power prior Bayesian analysis. We start by describing the basics of power prior distributions. We then describe how power priors can be used to construct priors for transition models whilst incorporating historical data. The problem that we attend to in chapter nine is of a slightly different nature. We assume that a reference pattern (template) from donor models of source tasks is available and the agent have to decide how best to use the template in leveraging learning performance on a new target task.

The thesis concludes in part five - chapter ten - with a summary of the research and the conclusions drawn. In addition the chapter contains a list of areas of future work.

2 Markov Decision Tasks

In this chapter we describe Markov decision process (MDP), a powerful framework for controlling systems that evolve in a stochastic way. We first introduce the framework of Markov decision theory and then look at dynamic programming techniques for solving Markov decision tasks. This applies when we have a complete MDP model of the task. Following this we then consider learning how to solve Markov decision tasks when a complete model of the MDP is not available. In that case we have to either learn without maintaining a model or attempt to build a model while learning. The concepts and background introduced here will constitute the foundations that we build upon in all the subsequent chapters.

2.1 Introduction

Let us begin this chapter by considering an example decision task involving an agent whose well-being may deteriorate over time in an uncertain environment. Reward is accrued on the basis of the agent's state of wellbeing. The agent can select actions that could improve its wellbeing, albeit at some cost. The agent can perfectly sense its environment and fully observe the effects of its actions. Sometimes its actions fail, could lead to undesired outcomes or unexpected results. This example, which we refer to as the *'agent wellbeing task'* is a stochastic process i.e. a sequence of random events governed by a set of probability distributions rather than being deterministic. As we shall see in this chapter, the task can be framed as a Markov decision process (MDP).

MDPs constitute a powerful framework for representing (and thus reasoning about and controlling) systems that evolve in a stochastic way. It is well suited for the robot wellbeing task we described above. MDPs provide the theoretical foundations for sequential decision-making in completely observable environments. An MDP specifies an environment and task. The problem in an MDP is to find an optimal policy (or way of behaving) and there exist a variety of known solution algorithms for this problem.

MDPs were first introduced in 1957 by Richard Bellman as a variant of his more general 'dynamic programming' theory of optimal control, itself based on work by Hamilton and Jacobi in the 1800s (Bellman 1957, Bellman 1961, Sutton & Barto 1998). Since its invention, MDP have been applied in economics, operations research, control systems design, and artificial intelligence (AI) among other areas. In AI research, MDP theory has gained favour as a model of rational decision making in well-defined circumstances where an intelligent agent's outcome preferences can be expressed as a trajectory utility function (Russell & Norvig 2003). In unknown task environments, MDPs can sometimes be combined with reinforcement learning techniques to avoid the need to directly specify reward functions and process dynamics.

Russell & Norvig (2003) give a good introduction to MDPs from an AI point of view. Other popular references include Bertsekas (1987) and Puterman (2005). Much of the content of this chapter is a recapitulation of work in the operations research and control engineering literature e.g. (Hernández-Lerma & Lasserre 1991, Sennott 1999, Feinberg & Shwartz 2002, Heyman & Sobel 2003), and in the reinforcement-learning literature e.g. (Watkins 1989, Kaelbling et al. 1996, Sutton & Barto 1998, Wyatt 2002, Szepesvári 2010). The concepts and background introduced here will be built upon in the subsequent chapters of this thesis.

2.2 The Framework of Markov Decision Theory

The sequential decision-making problems formalised by Markov decision processes are described as follows. At a specified point in time, a decision-maker, agent, or controller observes the current state of a system. Based on the observed state, the decision maker chooses an action. The consequence of the action choice is twofold – the decision-maker receives an immediate reward (or incurs an immediate cost), and at the subsequent point in time the system evolves to a new state according to the underlying transition dynamics of the process determined by the chosen action. At the new state, the decision-maker faces a similar problem but this time the new state could be a state that is different to that the agent was in during the previous time step and there may be a different set of actions to consider.

In this section we describe the framework of Markov decision processes (MDPs).

2.2.1 Components of the Framework

Mathematically, a Markov decision process is a 5-tuple (S, A, Ψ, P, R). In the tuple, S is a set of $|S|$ distinct states, A is a set of $|A|$ distinct actions, $\Psi \subseteq S \times A$ is the set of admissible state-action pairs, $P : \Psi \times S \longrightarrow [0,1]$ is a Markovian transition model that specifies in a probabilistic form the process dynamics and $R : \Psi \longrightarrow \mathbb{R}$ is the expected reward function mapping S and A into real-valued returns.

We explain these concepts i.e. state, transition and Markov chain in Section 2.2.1.1. We describe the other component of the framework i.e. the reward in Section 2.2.1.2.

2.2.1.1 The Concepts of State, Transition & Markov Chain

The problem of modelling a decision task is greatly simplified by the concept of state and, in specific applications, the modelling *'art'* is to find an adequate state description that characterises the task's underlying stochastic process. The state of a system at a particular time is a description of the condition of the system at that time that is sufficient to determine all aspects of the future behaviour of the system when combined with knowledge of the system's future controlled actions. The history of the system that is relevant to its future behaviour is summed up in its current state. Essentially, future behaviour does not depend on how the system arrived at its current state, a property sometimes called *'path independence'* of the system description.

The concept of path independence derives from a general notion of *independent processes*. To describe the notions of independence let us first define a stochastic process by a set of random variables $\{X_t,\ t \in T\}$ where the random variable X_t denotes the outcome at time step t and $T = \{0, 1, 2\ldots\}$. The outcomes are referred to as the states of the process. The domain of X_t is the set of possible outcomes denoted $S = \{s_1, s_2, \ldots, s_{|S|}\}$. In the general case the outcome at time $t+1$ is dependent on the prior sequence of outcomes $x_0, x_1, \ldots, x_{t-1}, x_t$. The likelihood of the outcome at time $t+1$ being s' given the prior sequence of outcomes $x_t, x_{t-1}, x_{t-2}, \ldots x_0$ is stated as follows:

$$Pr(X_{t+1} = s' | x_t \wedge x_{t-1} \wedge \ldots \wedge x_0). \tag{2.1}$$

If the outcome at each time is independent of the outcome at all prior stages then the process is said to be an independent process. That is,

$$Pr(X_{t+1} = s' | x_t \wedge x_{t-1} \wedge x_{t-2} \wedge \ldots \wedge x_0) = Pr(X_{t+1} = s'). \tag{2.2}$$

The independence assumption allows calculations of transition probabilities to be simplified. The more independence assumptions the more the possibility of explicit calculations but the more questionable is the realism of the model. When modelling a stochastic process, the challenge is to have dependencies which allow for sufficient realism but can be analytically tamed to permit sufficient mathematical tractability. The Markov assumption frequently balances these two demands nicely. A Markov process weakens the independence assumption marginally by permitting the outcome at time $t+1$ to be independent of all previous outcomes except that immediately, prior at the time t. That is,

$$Pr(X_{t+1} = s' | (X_t = s) \wedge x_{t-1} \wedge x_{t-2} \wedge x_{t-3} \wedge \ldots \wedge x_0) = Pr(X_{t+1} = s' | X_t = s). \tag{2.3}$$

Equation 2.3 is the Markov property of a stochastic dynamical system. The property establishes that, the way the system develops probabilistically in the future does not depend on the whole history but only on the present. The property stated in Equation 2.3 illustrates another use of the concept of state in describing the evolution of dynamically controlled processes that operate according to probabilistic rules. The system state and future controlled actions determine the probabilities of all aspects of the future behaviour of the system independently of how the state was reached.

The conditional probabilities $Pr(X_{t+1} = s'|X_t = s)$ in Equation 2.3 are called the single step transition probabilities, these will be denoted by $s \rightsquigarrow s'$ and written in shorthand to mean the probability of the transition from state s to s':

$$p_{ss'} = Pr(X_{t+1} = s'|X_t = s). \tag{2.4}$$

We refer to $p_{ss'}$'s simply as transition probabilities. Transitions from state s to state s' occurring as a result of an action a taken at state s will be denoted as $p^a_{ss'}$, expressed as follows:

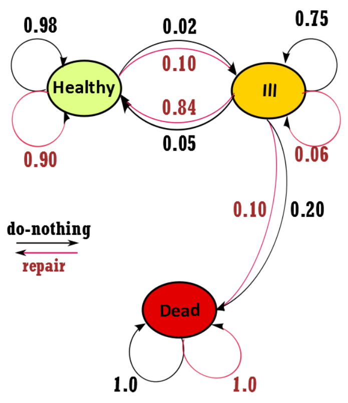
$$p^a_{ss'} = Pr(X_{t+1} = s'|X_t = s, A_t = a). \tag{2.5}$$

and we denote by $s \overset{a}{\rightsquigarrow} s'$ to incorporate the action taken at state s. The transition probabilities for each action a can be represented in the form of a square array called the matrix of transition probabilities, or the transition matrix, P^a in which the $(s,s')^{\text{th}}$ element is $p^a_{ss'}$. In finite state spaces P^a is $(|S| \times |S|)$ dimensional, written out as

$$P^a = \begin{bmatrix} p^a_{11} & p^a_{12} & p^a_{13} & \cdots & p^a_{1|S|} \\ p^a_{21} & p^a_{22} & p^a_{23} & \cdots & p^a_{2|S|} \\ p^a_{31} & p^a_{32} & p^a_{33} & \cdots & p^a_{3|S|} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p^a_{|S|1} & p^a_{|S|2} & p^a_{|S|3} & \cdots & p^a_{|S||S|} \end{bmatrix}$$

and the entries $p^a_{ss'}$ satisfy standard stochastic constraints $0 \le p^a_{ss'} \le 1; \sum_{s' \in S} p^a_{ss'} = 1 \quad \forall s \in S$.

In the agent wellbeing task, assume that the agent can choose from a set of two actions $\{repair, do-nothing\}$ and let the possible states of wellbeing be $\{Healthy, Ill,$ and $Dead\}$. An example of system dynamics for the agent wellbeing task is shown in Figure 2.1.



$$P^{do-nothing} = \begin{array}{c} \\ Healthy \\ Ill \\ Dead \end{array} \begin{array}{c} \begin{array}{ccc} Healthy & Ill & Dead \end{array} \\ \begin{bmatrix} 0.98 & 0.02 & 0.00 \\ 0.05 & 0.75 & 0.20 \\ 0.00 & 0.00 & 1.00 \end{bmatrix} \end{array}$$

$$P^{repair} = \begin{array}{c} \\ Healthy \\ Ill \\ Dead \end{array} \begin{array}{c} \begin{array}{ccc} Healthy & Ill & Dead \end{array} \\ \begin{bmatrix} 0.90 & 0.10 & 0.00 \\ 0.84 & 0.06 & 0.10 \\ 0.00 & 0.00 & 1.00 \end{bmatrix} \end{array}$$

Figure 2.1: A simple instance of the Agent Wellbeing task

In the figure, the states represent whether the robot is healthy, ill, or dead during a period captured by the model. According to the figure, under a do-nothing action, a healthy state is followed by a) another healthy state with a probability of 0.98, b) an ill state with a probability of 0.02, and c) a dead state with a probability of 0.0. That is, according to the model when the agent follows a do-nothing action in a healthy state, there is very small probability that its state will change to ill and it would not be expected to die. From this figure, it is possible to calculate the long-term fraction of time during which the robot is healthy, or on average how long it will take to go from being healthy to being ill.

Andrei Markov, a Russian mathematician, was the first one to study these matrices. At the beginning of this century he developed the fundamentals of the Markov chain theory. A Markov chain is a stochastic process that consists of a finite number of states and some known probabilities $p_{ss'}$, where $p_{ss'}$ is the probability of moving from state s to state s'. We describe a Markov chain as follows: We have a set of states, $S = \{s_1, s_2, \ldots, s_{|S|}\}$. The process starts in one of these states and moves successively from one state to another. Each move is called a step. If the chain is currently in state s, then it moves to state s' at the next step with a transition probability denoted by $p_{ss'}$, and this probability does not depend upon which states the chain was in before the current state. The process can remain in the state it is in, and this occurs with probability p_{ss}. An initial probability distribution, defined on S, specifies the starting state. Usually this is done by specifying a particular state as the starting state. So a discrete time Markov chain can be completely described by the tuple $< S, P, u >$. where u is the transition probabilities across initial starting states and, as before, S is the finite set of states the process can be in, P are the state transition probabilities.

The subject of Markov chains is best studied by considering special types of Markov chains; a more common type is *Absorbing Markov chain* and it involves the notion of *Transient Markov state*.

Definition 2.1 (Absorbing Markov Chain:) *A state s of a Markov chain is called absorbing if it is impossible to leave it (i.e., $p_{ss} = 1$). A Markov chain is absorbing if it has at least one absorbing state and if from every state it is possible to go to an absorbing state in one or more steps (not necessarily in one step).*

Definition 2.2 (Transient Markov State:) *In an absorbing Markov chain, a state which is not absorbing is called transient.*

In our agent wellbeing model of Figure 2.1, dead is an absorbing state. Obvious questions that can be asked about such a chain include: (a) What is the probability that the process will end up in a given absorbing state? (b) On average, how long will it take for the process to be absorbed? and (c) On average, how many times will the process be in each transient state? The answers to all these questions depend, in general, on the state from which the process starts as well as the transition probabilities.

2.2.1.2 Rewards

As recompense in recognition of worthy behaviours, rewards are generated in MDPs, following actions and state transitions. A reward at time t is a random variable $R_t = R(s, a, s')$ that is dependent on the current state s at time t, the action taken a and the next state s' at time $t+1$. The actual sequence of rewards generated at times $t = (0, 1, 2, \ldots)$ is denoted as $(r_0, r_1, r_2, \ldots)$. At each time step, an observation resulting from action a at state s can be represented in short form as $s \overset{a,r_t}{\rightsquigarrow} s'$ for the transition $s \overset{a}{\rightsquigarrow} s'$ and reward $r_t = r^a_{ss'}$.

The general form of the stochastic reward R_t we have described can be relaxed in a variety of ways. The stochastic reward variable can be defined on either $S \times A \times S$, as we have done above, or defined on $S \times A$. In the MDP framework, there is no distinction between the two forms as the destination state can be averaged out with no effect on planning. In addition, the stochastic reward variable can sometimes be associated more conveniently with states, i.e. r_s, on the assumption that the reward accrues for being in the state.

The actual reward generated at a particular instant can be specified by a reward function. Three types of reward functions are described as follows (Narendra & Thathachar 1989). The simplest reward function, termed a *P-model*, is when the set of possible rewards is Boolean, R = {0,1}. Any task with well-defined criteria for success and failure can be represented as a P-model. For example, assuming the aim of the robot wellbeing task is to recompense the robot only for being in a healthy state, it is easy to specify the reward function as a P-model as follows, taking 1 to be a success for being at a healthy state:

$$r_t = \begin{cases} 1 & if \text{ robot is healthy} \\ 0 & \text{otherwise} \end{cases}$$

A minimal extension of the P-model is a *Q-model* that allows any finite number of reward values in the interval $[0,1]$. Tasks with real-valued rewards can be expressed in this form by means of normalisation and quantisation. The most general case is when the reward can take any real value in the interval $[0,1]$. Such a model is termed an S-model. By normalisation any problem with bounded reward can be expressed as an *S-model.*

2.2.2 Objective of the Markov Decision Task

An objective function in an MDP maps a set of possible states and action sequences and their probabilities to a single number known as *value*. MDP solution algorithms seek results that optimise the value, so the choice of this function is a critical part of the statement of a task. In Markov decision processes, value is defined in terms of cumulative reward received over time. In other words, the goal of the agent in terms of MDPs would be to implement a control policy that maximizes some measure of cumulative reward to be received in the future. In this section, we look at how such measures are defined and explain what we mean by optimal solutions in terms of value functions.

2.2.2.1 Return as a Measure of Cumulative Reward

We described reward functions in Section 2.2.1.2 and noted that such functions only specify rewards for particular times in a process. Operating an MDP over time would result in a series of reward accruals over a series of time steps and a measure of performance beyond immediate rewards will be required to account for the resulting accruals. Return **R** has been suggested, and widely used, as a long term measure of reward (Barto, Sutton & Watkins 1990, Sutton & Barto 1998). The following are three possible definitions of return.

Finite horizon criterion: The simplest *finite horizon criterion* takes into account the expected reward for the next N steps to form the undiscounted finite horizon return: $\mathbf{R} = \sum_{t=0}^{N} r_t$. This criterion often is not appropriate, since in most cases an agent does not know the length of its life.

Average reward criterion: The average reward criterion considers the average reward over an infinite life span to form the undiscounted infinite horizon return:

$$\mathbf{R} = lim_{N\to\infty}\frac{1}{N}\sum_{t=0}^{N} r_t \tag{2.6}$$

One problem with this criterion is that an agent's extremely ineffective early behaviour could be overlooked due to the averaging with the long run reward.

Infinite horizon discounted reward: In the *infinite horizon discounted reward* criterion, behaviour is optimized in the following way: the aim is to maximize the long-run reward, but rewards that are received in the future are geometrically discounted by the discount factor $\gamma \in (0,1)$:

$$\mathbf{R} = \sum_{t=0}^{\infty} \gamma^t r_t. \tag{2.7}$$

The discount factor $0 \leq \gamma < 1$ could be interpreted as the probability of existing for another step.

The most commonly used return function is the discounted model. This has been extensively used within work on learning from delayed reinforcement and is the return model adopted throughout subsequent portions of this thesis. The choice of a value for γ determines the relative weighting of close and distant future reward values. As $\gamma \rightarrow 0$ rewards close in time contribute more towards the return value. When $\gamma = 0$ only immediate rewards matter and return is reward so $\mathbf{R}_t = r_t$. Not only is this method elegant it has been shown to make certain problems tractable.

2.2.2.2 Policies and Optimal Policies

The behaviour of an agent in the MDP framework can be completely and sufficiently described by the agent's policy. The policy (π) controls the actions an agent takes. The function $\pi(s)$ that describes the action choice as a function of the state is called the policy function.

There are several types of policies and the most common distinctions are stationary versus non-stationary and deterministic versus non-deterministic policies. A stationary policy $(\pi : S \rightarrow A)$ is a mapping between states in the world and actions to be taken in the states. The action decision depends only on the state the agent occupies and is independent of time. A stochastic policy $(\pi : S \times A \rightarrow [0,1])$, is a mapping from states into probability distribution over actions $f(\pi|s)$ with probability $Pr(a|s)$ in state s. Under a stochastic policy the action selection in each state varies, but the distribution with which selection varies is fixed. A non-stationary policy $(\pi_t : S \rightarrow A)$ is again a mapping between states and actions, but it is indexed by time. The policy π_t is used to choose an action on the t^{th} to last time step as a function of the current state s at time t. The time indexing for non-stationary policies is slightly counter intuitive as it does not refer to the number of steps past but the number of steps from the final step of the process.

In general, a policy and the transition matrix together induce a probability distribution over the set of all possible sequences of states and actions for each possible initial state of the environment. In other words, we should in principle be able to take a description of a policy, an environment, and an initial state, and compute the probability that a given sequence of states and actions will occur.

We need some ways of distinguishing between the quality of different policies and how well they fulfil the agent's goals. The return models introduced previously in Section 2.2.2.1, allows an ordering to be defined over policies. A policy π^x is at least as good as a policy π^y if the expected return for all the process states under π^x is equal to or greater than at under π^y. In Markov decision processes, optimality is defined in terms of cumulative reward received over time. The agent's goal is typically directed towards implementing a control policy that maximizes some measure of the total reward to be received in the future. An optimal policy maximises the agent's expected return.

2.2.2.3 Value Functions

In this section we describe how to predict the expected return for a given state in a Markov decision process and we define value functions for doing so. A value function gives a measure of the expected return for the states in a Markov decision process. Since the value functions are measures of expected return they are defined with respect to particular policies. We describe two types of value functions:

State-Value Functions: Assuming an agent has a policy π, the state value function of a state s under the policy can be defined as:

$$\begin{aligned} V^{\pi}(s) &= E_{\pi}(\mathbf{R}|X_t = s) \\ &= E_{\pi}(R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots |X_t = s) \\ &= E_{\pi}(R_{t+1} + \gamma V^{\pi}(s')|X_t = s) \\ &= r_s^{\pi(s)} + \gamma \sum_{s'} p_{ss'}^{\pi(s)} V^{\pi}(s') \qquad (2.8) \end{aligned}$$

where $E_\pi(\cdot)$ denotes the expected value given that the agent follows policy π and $V^\pi(s)$ denotes the value function i.e. the expected return when starting in state s and following π thereafter. Without loss of generality, the expectation of immediate reward $E_\pi(R_{t+1})$ is expressed as $r_s^{\pi(s)}$. The expected return for each state under the policy can be calculated in both finite and infinite horizon cases. In the finite horizon case $V_t^\pi(s)$ is the expected rewards gained for starting in state s and executing the policy π for t time steps. The simplest case is when there is only one step

$$V_1^\pi(s) = r_s^{\pi(s)}, \tag{2.9}$$

whereas the recursive form for t-steps is

$$V_t^\pi(s) = r_s^{\pi(s)} + \gamma \sum_{s'} p_{ss'}^{\pi(s)} V_{t-1}^\pi(s'). \tag{2.10}$$

The $t-$step value of being in state s and executing the policy π is the expectation of the immediate reward, $r_s^{\pi(s)} = E[R(s,\pi(s))]$, plus the discounted expected value for the remaining $t-1$ steps. In the evaluation of the future all possible resulting states s' must be considered, the likelihood of their occurrence $p_{ss'}^{\pi(s)}$, and their $(t-1)$ step value under policy π, $V_{t-1}^\pi(s')$.

The optimal state-action value function, denoted V^*, is defined as:

$$\begin{aligned} V^*(s) &= \max_\pi V^\pi(s) \\ &= \max_a \{r_s^a + \gamma \sum_{s'} p_{ss'}^a V^*(s')\} \end{aligned} \tag{2.11}$$

State-Action Value Functions: The state-action value function Q^π corresponding to a policy π is the mapping of state-action pairs to their values and satisfies:

$$\begin{aligned} Q^\pi(s,a) &= E_\pi(\mathbf{R}|X_t = s, A_t = a) \\ &= r_s^a + \gamma \sum_{s'} p_{ss'}^a Q^\pi(s',a) \end{aligned} \tag{2.12}$$

$Q^{\pi}(s,a)$ is a function that calculates the quality of a state-action combination. The optimal state-action value function, denoted Q^*, is defined as:

$$\begin{aligned} Q^*(s,a) &= \max_{\pi} Q^{\pi}(s,a) \\ &= \max_{a} \{r_s^a + \gamma \sum_{s'} p_{ss'}^a Q^*(s',a)\} \end{aligned} \tag{2.13}$$

Intuitively, Equation 2.13 says that the state-action value, $Q^*(s,a)$, is the expected total discounted return resulting from taking action a in state s and continuing with the optimal policy thereafter.

2.3 Solving Markov Decision Tasks

The solution to a Markov Decision Process can be expressed as a policy π, a function from states to actions. In the example agent wellbeing task above, a specification to do-nothing in a healthy state specifies the policy for an agent to follow in that state.

The optimal solution of an MDP is an optimal policy π^* that dominates all other possible policies for that MDP. There exists a well established family of algorithms known as *dynamic programming* for calculating optimal policies (Bellman 1957, Bertsekas 1987). Dynamic Programming works via the calculation of successive approximations of the value function to evaluate a specific policy. The algorithms typically transform Bellman equations such as those of (2.14) and (2.15) into update rules for improving approximations of the desired value functions. The updates are repeated in some order for all the states until no further changes take place.

$$\pi(s) := \arg\max_{a} \{r_s^a + \gamma \sum_{s'} p_{ss'}^a V(s')\} \tag{2.14}$$

$$V(s) := r_s^a + \gamma \sum_{s'} p_{ss'}^{\pi(s)} V(s') \tag{2.15}$$

The order in which the updates are made depends on the variant of the algorithm; we could do them for all states at once, or state by state, and more often to some states than others. As long as no state is permanently excluded from either of the steps, the algorithm will eventually converge to an optimal solution.

In the sub-sections that follows we look at how Markov decision tasks can be achieved in completely observable environments mostly via dynamic programming techniques. We start by looking at the case where a perfect model of the task is available i.e. the transition probabilities and reward functions are fully determined. Following this we consider learning how to behave when we have no model and we do not maintain a model. Finally we will look at how we can build a model whilst learning and use the constructed model to decide how to behave. Our organisation of the section is motivated by the distinction made in much of the reinforcement learning literature, this being the difference between knowing the true model of the task environment a-priori and learning a model. Clearly, if the model is known the agent does not need to learn. However, when the agent does not have access to the correct model it would have to learn about the task environment in order to act appropriately. When the MDP model of the task environment is completely specified the search for an optimal policy reduces to a planning problem.

2.3.1 Planning in MDP

There are many methods for finding optimal policies for MDPs when we have a complete model. In this section we will describe three classical approaches commonly used namely *value iteration*, *policy iteration* and *linear programming* methods. All of the methods can be found in standard textbooks on MDPs e.g. (Bertsekas 1987, Puterman 2005).

2.3.1.1 Value Iteration

Value Iteration (VI) (Bellman 1957) is a simple iterative procedure that computes the value of each state based on the values assigned to neighbouring states, iterating the loop for each state until either

a) the value estimation converges to exactly the correct V^* values, formally requiring an infinite number of iterations, or b) the VI steps satisfies surrogate convergence criteria. The challenge in VI and many other iterative approaches lies in knowing exactly when to stop iterating and return a correct solution. An approach that relies on infinite number of iterations is less desirable and in VI the surrogate option is often used, thanks to a significant result for the VI algorithm that bounds the performance of the current greedy policy as a function of the maximum difference between two consecutive value functions $V_t(s), V_{t-1}(s)$ (Sutton & Barto 1998).

The quantity $|V_t(s) - V_{t-1}(s)|$ is known as the *Bellman error magnitude* and also sometimes called the *Bellman residual*. Based on the notion of Bellman residuals, the VI algorithm quite simply iteratively compute the values of each state until the maximum difference between the value estimates of two successive steps are close enough, i.e. less than a pre-specified threshold ε value. In other words, the iterative loop terminates when $|V_t(s) - V_{t-1}(s)| < \varepsilon \ \forall s \in S$. The VI algorithm is illustrated in Algorithm 2.1.

The precision of the algorithm increases with smaller thresholds albeit resulting in lengthier steps to convergence. On the completion of the value iteration algorithm an agent would be supplied with the optimal policy to control its behaviour, which is just the greedy policy with respect to the converged value function. It is important to note that often $\pi = \pi^*$ long before V_t is near V^*, so optimal behaviour can be found before the value function has converged. If $|V_t(s) - V_{t-1}(s)| < \varepsilon$ $\forall s \in S$, then the value of the greedy policy with respect to $V(s)$ does not differ from $V^*(s)$ by more than $2\varepsilon\frac{\gamma}{1-\gamma}$ at any state (Sutton & Barto 1998).

```
1: procedure $[V, \pi]$ = VALUE ITERATION($S, A, P, R, \varepsilon$)
2:     Initialise state values $V(s) \quad \forall s \in S$
3:     set $\Delta \leftarrow 0$
4:     repeat
5:         for each $s \in S$ do
6:             $v \leftarrow V(s)$
7:             $V(s) \leftarrow \max_a \{r_s^a + \gamma \sum_{s'} p_{ss'}^a V^*(s')\}$
8:             $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
9:         end for
10:    until $\Delta < \varepsilon$
11:    $V^*(s) \leftarrow V(s) \; \forall s \in S$
12:    $\pi(s) \leftarrow \arg\max_a r_s^a + \sum_{s'} p_{ss'}^a V^*(s')$
13:    return $[V^*(s), \pi(s)] \quad \forall s \in S$
14: end procedure
```

Algorithm 2.1: The Value Iteration Algorithm

2.3.1.2 Policy Iteration

The value iteration algorithm we described in Section 2.3.1.1 operates by iteratively updating state-values directly on the state space. An alternative approach is to iteratively search the policy space directly and this is the basis of *policy iteration*, our focus in this section.

Policy iteration (Howard 1960) iteratively computes the value function for a given policy π and if the policy can be improved then a replacement policy that is strictly better than the current one is obtained and the iteration continues until convergence when no further policy improvement can be found. The policy iteration algorithm should converge in at most an exponential number of steps since a) there are at most $|S|^{|A|}$ distinct policies, and b) the sequence of policies improves at each step (Puterman 2005). The policy iteration algorithm has two main phases (Algorithm 2.2):

Policy Evaluation: in which we compute the state-value function V^{π} for a given policy π. The job in this phase is sometimes referred to as a *prediction problem*. Recall from our discussion in Section 2.2.2.3 that the state value functions can be computed for all $s \in S$ as $V^{\pi}(s) = r_s^{\pi(s)} + \gamma \sum_{s'} p_{ss'}^{a} V^{\pi}(s')$ and, when the transition probabilities and reward functions are known, the evaluation accomplished using a number of methods including that of Section 2.3.1.1.

Policy Improvement: allows us to find a policy π' that are strictly better than the current policy π, if the optimal policy has not been found. From the policy evaluation step we know the worth of the current policy through its state value function $V^{\pi}(s)$. The issue raised in this phase is whether there would be a gain in changing to a new policy and the issue can be addressed via the policy improvement theorem (Sutton & Barto 1998) whose conditions are met by the greedy policy given by $\pi'(s) = \text{argmax}_a Q(s,a)$.

```
1:  procedure [$V,\pi$] = POLICY ITERATION($S,A,P,R$)
2:      Initialise policies $\pi(s)$ and state values $V^{\pi}(s)$  $\forall s \in S$
3:      repeat
4:          Set $stablePolicy \leftarrow true$
5:          solve $V^{\pi}(s) = r_s^{\pi(s)} + \gamma \sum_{s'} p_{ss'}^{\pi(s)} V^{\pi}(s')$  $\forall s \in S$.
6:          for each $s \in S$ do
7:              $\pi'(s) \leftarrow \text{argmax}_a Q^{\pi}(s,a)$
                    where  $Q^{\pi}(s,a) \leftarrow r_s^{\pi(s)} + \gamma \sum_{s'} p_{ss'}^{a} V^{\pi}(s')$  $\forall a \in A$
8:              if $\pi'(s) \neq \pi(s)$ then $stablePolicy \leftarrow false$ end if
9:          end for
10:         $\pi(s) \leftarrow \pi'(s)$  $\forall s \in S$
11:     until $stablePolicy$
12:     return [$V(s),\pi(s)$]  $\forall s \in S$
13: end procedure
```

Algorithm 2.2: The Policy Iteration Algorithm

Note that in the illustration the policy evaluation phase is accomplished by solving linear equations in $V^{\pi}(s)$ for all $s \in S$. In principle, other methods for computing the state value function of a policy can be used including the value iteration (VI) algorithm. If using VI, the initial value functions can be set to those from previous policy evaluation phase.

2.3.1.3 Linear Programming

In this section we describe *linear programming* (LP) approaches to Markov decision tasks. A linear program, consists of a set of variables, a set of linear equations over the variables and an objective function, A linear programming problem may be defined as finding an assignment to the variables in a way that the objective function is maximised or minimised over all the variable assignments that satisfy each of the linear equations, The linear equations constrain the space of acceptable solutions and may be equalities or inequalities.

In LP approaches to MDPs, the variables are the value functions $V(s)$ for every state $s \in S$ of the decision process. An LP formulation of MDP is as follows.

$$\begin{aligned} &Minimise \sum_{s} V(s) \\ &\text{subject to:} \quad V(s) \geq r_s^a + \gamma \sum_{s} p_{ss'}^a V(s') \quad \forall s \in S,\ a \in A \end{aligned} \tag{2.16}$$

where each pair (s,a) corresponds to a constraint. This LP formulation has $|S|$ optimisation variables and $|S| \times |A|$ constraints. The intuition here is that, for each state s, the optimal value from s is no less than what would be achieved by first taking action a, for each $a \in A$. The minimization ensures that we choose the least upper bound (the maximum, in other words) for each of the $V(s)$ variables.

An important fact from the theory of linear programming is that every linear program has an equivalent linear program in which the roles of the variables and the constraints are reversed. The resulting linear program, known as the *dual*, can also be used to solve MDPs. One advantage of the dual formulation is that it makes it possible to express and incorporate additional constraints on the form of the policy found.

Associated with the LP for MDP is a corresponding dual LP (*D-LP*) given by:

$$\begin{aligned}
&\mathit{Minimise}\ \sum_s \sum_a r_s^a y(s,a) \\
&\text{subject to:} \\
&\sum_s \sum_a V(s')y(s,a) = 0 \quad \forall s' \in S \\
&\sum_s \sum_a y(s,a) = 1 \\
&y(s,a) \geq 0 \quad \forall s \in S,\ \forall a \in A
\end{aligned} \tag{2.17}$$

The variables $y(s,a)$ constitute a feasible solution to the dual if they jointly satisfy the constraints in the D-LP problem. An optimal solution $y^*(s,a)$ requires the variables to be feasible and satisfy the following expression.

$$\sum_s \sum_a r_s^a y^*(s,a) \geq \sum_s \sum_a r_s^a y(s,a) \tag{2.18}$$

for all the feasible solutions to the D-LP. The optimal solution $y^*(s,a)$ could be used to establish the optimal policies.

In general, the LP approach to MDPs is appealing because it is valid in a very general context. It produces an exact solution without the need to specify stopping criterion. LP approaches offers a) an elegant way of solving planning problems where maximisation of expected total discounted reward is subject to additional constraints on expected rewards, and b) a family of mathematical programming algorithms for approximating value function using lower dimensional functions.

2.3.2 Learning in MDP

In the previous section we described methods an agent may use to find optimal policies for Markov decision processes. In order to use the methods described, the agent must have a-priori a complete model of the decision task i.e. state transition probabilities and reward functions must be known. In such instances, there is no learning involved. The known rewards and transition probabilities are simply plugged into the methods to obtain the optimal policy.

As we know, life is not always that simple. Often there are uncertainties and the agent would not always know a-priori the complete task environment. That is, all or some elements of the reward and transition probabilities would be unknown to the agent. When uncertainty exists the agent would have to learn from its interaction with the environment, using the resulting information to update its knowledge of the environment and decide future actions. There are two classical approaches as follows.

Model-free: *learn to perform a task without learning a process model.*

Model-based: *learn to perform a task by acquiring one or more process models of the task and using the acquired model(s) to establish the appropriate policies.*

The model-free approach is also known as a direct, *'cached values'*, method and the model-based approach is sometimes called an indirect method. There is no consensus as to the better of the two approaches. Whilst friendly debates continues in the field on the better of the two approaches, it is generally opined that the choice of an appropriate approach depends on the particular task at hand and the resources available. For example, model-based approaches may be more appropriate in well defined control-like tasks that presents opportunities for modelling the system. In addition, model-based methods may be the most suited when interaction between a learning agent and the real system is limited, not feasible and/or dangerous. On the other hand, model-free methods may be more applicable to agents learning constructional tasks, i.e. tasks that require coordinated, predictive actions over multiple time steps, for which there is no existing model and there may be no clear path towards developing a model (Gaskett 2002).

Although empirical reinforcement learning research provides success stories for both model-based and model-free methods (e.g. Kearns & Singh (1999)), the weight of evidence supports a view that model-based methods outperforms model-free methods in terms of total cumulative rewards received (e.g. Koppejan & Whiteson (2009)), albeit the importance of learning a reliable model cannot be overemphasised. Compared to model free reinforcement learning, Atkeson & Santamaria (1997) noted that model based reinforcement learning is more data efficient, finds better trajectories, plans and policies, and handles changing goals more efficiently.

Whilst model-free methods may have obvious benefits for some tasks (Wyatt, Hayes & Hallam 1999), especially in ensuring that state action values are attributed to, and behaviour driven directly by cues predicting rewards, they do have significant limitations (McDannald, Takahashi, Lopatina, Pietras, Jones & Schoenbaum 2012). Two significant limitations of model-free methods according to McDannald et. al. (2012) are: i) due to the fact that model-free value is represented in a common currency, the predictions used to guide behaviour are blind to specific features of rewards, and ii) model-free methods do not straightforwardly allow for 'new learning' to occur when these specific features of the predicted rewards are changed, so long as general or cached 'value' is maintained. Model-based methods addresses these issues much better particularly in facilitating the transfer of prior information about systems dynamics which is the subject of this thesis.

In summary, model-free methods are known to be appealing when there are computational resource constraints, they are simple to implement and its performance, though inferior to that of the model-based based methods, is still quite strong. Adaptation to changes in the reward structure and their specific features is much slower with model-free methods. Model-based methods are more flexible, uses dynamic programming algorithms on the estimated models to compute values, and can more easily accommodate changes in state-reward pairings directly. Model-free methods may not explore as efficiently as model-based methods and, as we know, the speed of convergence of an agent rests heavily on the exploration policy employed (Wyatt 1997). These differences are highlighted in Table 2.1.

We describe in this section the techniques used in the two approaches, starting in Section 2.3.2.1 with the model-free methods and turning to model-based methods in section 2.3.2.2.

Model-based Methods	**Model-free Methods**
• Indirectly learn state values.	• Directly learn state action values.
• Explicitly estimate a model from experience.	• Takes longer to learn especially in more complex environments.
• Use dynamic-programming algorithms on the estimated model.	• No guarantees about explore/exploit trade-off.
• Effective use of experience.	• Low memory and computational costs.
• High computational costs.	• Adaptation to changes in the reward structure and their specific features is much slower.
• Facilitates transfer of prior information relating to systems dynamics	

Table 2.1: Model-based versus Model-free Methods

2.3.2.1 Model-free Methods

In this section we describe solution methods for Markov decision processes on the basis that a) there is no prior knowledge of the task environment i.e. the transition and reward functions are unknown, and b) the task will be performed without inferring or using any process model of the task's environment. We describe model-free solutions that allows a learning agent to interact with task environments, shifting our discussion from the off-line methods introduced in the previous sections to on-line solution approaches. The on-line approaches permit real and/or simulated interaction with the task's environment and they generally involve iterative steps of taking an action in a state, acquiring experience by observing the effect of the action taken, revising value and/or policy estimates based on the acquired experience, and moving on to the next state. The iteration is illustrated in Algorithm 2.3.

```
1: procedure OnlineLearningBasedOnStateValues
2:     Initialise state values $V(s)$ and policy $\pi(s)$ $\forall s \in S$.
3:     loop (for each episode):
4:         Initialise state $s$.
5:         repeat (for each step of the episode):
6:             Select an action $a$ from state $s$ using $\pi$ and execute the action.
7:             Receive immediate reward $r = r_s^a$ and observe the new state $s'$ i.e. acquire experience $s \overset{a,r}{\rightsquigarrow} s'$.
8:             Update state-value $V(s)$ and policy function $\pi$ using the acquired experience $s \overset{a,r}{\rightsquigarrow} s'$.
9:             Advance to next state $s \leftarrow s'$.
10:        until end of episode e.g. $s$ is terminal.
11:    end loop
12: end procedure
```

Algorithm 2.3: An online algorithm for learning a policy and value function based on state values

There are known variations to the on-line Algorithm 2.3 including, for example, using state-action values instead of state-values as we illustrate in Algorithm 2.4.

In both Algorithms 2.3 and 2.4 the agent must establish how to a) in *step 6* select actions i.e. find policies that would optimise predicted performance, and b) in *step 8* update value functions i.e. estimate expected future performances. The former is sometimes referred to as a control problem and the latter a prediction problem. If the agent has a model then the prediction problem can be solved by using the planning techniques we discussed in Section 2.3.1. We will shortly, in Section 2.3.2.2, describe how the agent can acquire a model of the task environment. If the agent decides to operate model-free it can build estimates of the value function directly from experience. This can be done primarily in two main ways via Monte Carlo and temporal difference learning. We begin with a brief discussion of Monte Carlo methods followed by a description of temporal difference learning.

```
1: procedure OnlineLearningBasedOnStateActionValues
2:     Initialise state-action values $Q(s,a)$ $\forall s \in S$, $\forall a \in A$.
3:     loop (for each episode):
4:         Initialise state $s$.
5:         repeat (for each step of the episode):
6:             Select an action $a$ from state $s$ using the policy derived from $Q(s,a)$ and execute the action.
7:             Receive immediate reward $r = r_s^a$ and observe the new state $s'$ i.e. acquire experience $s \overset{a,r}{\rightsquigarrow} s'$.
8:             Update the state-action values $Q(s,a)$ using the acquired experience $s \overset{a,r}{\rightsquigarrow} s'$.
9:             Advance to next state $s \leftarrow s'$.
10:        until end of episode e.g. $s$ is terminal.
11:    end loop
12: end procedure
```

Algorithm 2.4: An online learning algorithm for learning a policy and value function based on state-action values

Monte Carlo methods solve the model-free prediction problem by averaging sample returns. In a basic Monte Carlo approach we sample a sequence of experience tuples $\{s_1 \overset{a_1,r_1}{\rightsquigarrow} s'_1, s_2 \overset{a_2,r_2}{\rightsquigarrow} s'_2, \ldots\}$ from the task environment and use the sampled experience to calculate the actual return from each state. We then repeat the sampling many times to obtain multiple independent realisations of the experience sequences and associated actual return for each sequence from each state. The value estimate is then based on the average of the actual returns.

To implement the basic Monte Carlo method it is typically assumed that the sampled experience is divided into episodes and all the episodes eventually terminate irrespective of the actions selected. This is because value estimates and policies are revised only at the end of an episode. Hence, Monte Carlo methods are suited to episodic tasks and they are incremental on an episode-by-episode sense. There are two established Monte Carlo methods for estimating expected return namely every visit Monte Carlo and the first visit Monte Carlo (Sutton & Barto 1998). As the name suggests, the approaches differs mainly in the way the sampled experience tuples are collated.

Simple Monte Carlo methods are important in that they give a performance baseline from which to work, and aspects of them have been important in developing sophisticated algorithms. The difficulty with simple Monte Carlo estimators is that their standard errors decline very slowly as the sample size rises. The variance of returns can be high which makes convergence slow (Singh & Sutton 1996). Also when interacting with a system in closed-loop, it is often impossible to reset the system to some particular state which would be necessary to obtain sufficient independent realisations of the process from that state. When this is not possible, it is not clear how to apply the Monte-Carlo techniques without introducing bias. Temporal difference learning which is without doubt one of the most significant ideas in reinforcement learning, addresses these issues. Methods of temporal differences were invented to perform prediction and optimization in exactly these circumstances.

Temporal difference (TD) learning (Sutton 1988) is a prediction method that allows an agent to learn in unknown environments by using past experience to predict the mean goodness of each state of the environment given that a certain policy is followed. Temporal difference methods were invented to account for the behaviour of animals in psychological experiments involving prediction, the links with dynamic programming were only made later. The idea of temporal difference learning draws from both dynamic programming and Monte Carlo methods. The former samples the environment according to a pre-specified policy whilst the latter approximates value estimates based on old estimates previously learned.

Temporal difference methods are predicated on a notion that value predictions across time steps are often correlated. Temporal difference methods quantify the difference between two subsequent value estimates and applies an update equation to revise the mean goodness of each state. The temporal difference quantity is simply

$$(r+\gamma V(s')) - V(s) \tag{2.19}$$

where $V(s)$ is the value estimate at state s and $r+\gamma V(s')$ is a different estimate of the value of state s arising from acquiring an experience $s \overset{a,r}{\rightsquigarrow} s'$ from an interaction with the environment, r is reward accrued from the transition under action a in state s i.e. $r = r_s^a$. Compared to $V(s)$, the revised estimate $r+\gamma V(s')$ facilitates a better estimate of the value of state s and the temporal difference between the two estimates can be used to update the value of $V(s)$ as follows:

$$V(s) \leftarrow V(s) + \alpha\left[r+\gamma V(s') - V(s)\right] \tag{2.20}$$

where $0 < \alpha \leq 1$ is the learning rate. The temporal difference learning procedure is illustrated in Algorithm 2.5.

```
 1: procedure TEMPORALDIFFERENCE TD(0) ($\pi$, $\alpha$)
 2:     Initialise state values $V(s)$  $\forall s \in S$.
 3:     loop (for each episode):
 4:         Initialise state $s$.
 5:         repeat (for each step of the episode):
 6:             Select an action $a$ from state $s$ using $\pi$ and execute the action.
 7:             Receive immediate reward $r = r_s^a$ and observe the new state $s'$ i.e. $s \overset{a,r}{\rightsquigarrow} s'$.
 8:             Update state-value $V(s) \leftarrow V(s) + \alpha[r+\gamma V(s') - V(s)]$.
 9:             Advance to next state $s \leftarrow s'$.
10:         until end of episode e.g. $s$ is terminal.
11:     end loop
12: end procedure
```

Algorithm 2.5: Estimation of state values via temporal difference $TD(0)$ method

In TD methods the value of state s depends on the value of state s' and not only on the final result. This makes it possible for TD methods to improve their predictions during a process without having to wait for the final result. There are flavours of temporal difference learning for control.

An actor-critic approach (Barto, Sutton & Anderson 1983) which parallels policy iteration has been suggested as being implemented in biological RL, and Q-learning (Watkins 1989) which parallels value iteration.

Actor-Critic

Actor-critic (AC), first proposed by Barto et al. (1983), is temporal difference learning TD method that have a separate memory structure to explicitly represent the policy independent of the value function. It consists of two components - actor and critic. The role of the actor is to select actions and that of the critic is to evaluate the performance of the actor. The critic's evaluation is used to provide the actor with a signal that allows it to improve its performance, typically by updating its parameters along an estimate of the gradient of some measure of performance, with respect to the actor's parameters. When the representations used for the actor and the critic are compatible, the resulting AC algorithm is simple, elegant and provably convergent to a local maximum of the performance measure used by the critic (under appropriate conditions) (Sutton, McAllester, Singh & Mansour 1999, Konda & Tsitsiklis 2000, Konda 2002).

Learning in the Actor-Critic architecture is on-policy: the critic must learn about and critique whatever policy is currently being followed by the actor. The critique takes the form of a TD error. This scalar signal is the sole output of the critic and drives all learning in both actor and critic. Typically, the critic is a state-value function. After each action selection, the critic evaluates the new state to determine whether things have gone better or worse than expected. That evaluation is the TD error:

$$\delta = r + \gamma V(s') - V(s) \tag{2.21}$$

where V is the current value function implemented by the critic, $s, s' \in S$ are the states at time t and $t+1$. This TD error can be used to evaluate the action just selected, the action a taken in state s at time t. If the TD error is positive then it suggests the tendency to select a should be strengthened for the future. Else if the TD error is negative then it suggests the tendency should be weakened. The strengthening or weakening steps can be implemented by increasing or decreasing action probabilities $p(s,a)$, for instance, by

$$p(s,a) \leftarrow p(s,a) + \beta\delta \tag{2.22}$$

where β is another positive step-size parameter. This AC method we described is just one variant in a family of actor-critic methods. There are other variations such as those that select actions in different ways or use a notion of eligibility traces (Sutton 1988, Sutton & Barto 1998). Additional factors considered by others include varying the amount of credit assigned to the action a taken which, for example, allows for a probability of selecting actions $Pr(a|s)$ in the update rule:

$$p(s,a) \leftarrow p(s,a) + \beta\delta\left[1 - Pr(a|s)\right]. \tag{2.23}$$

Algorithm 2.6 shows the actor critic algorithm in schematic form. Actor-critic methods offer the following advantages:

- They require minimal computation in order to select actions. Methods that lack a separate data structure for the policy typically require a repeated search for the action with the best value, and this search can become computationally prohibitive, especially for real valued actions.

- They can learn stochastic policies; that is, they can learn the optimal probabilities of selecting various actions.

- The separate actor in actor-critic methods makes them more appealing in some respects as psychological and biological models. In some cases it may also make it easier to impose domain-specific constraints on the set of allowed policies. An actor maintains a separately parameterized stochastic action-selection policy.

```
1: procedure ActorCriticAlgorithm ($\beta$)
2:     Initialise probability of selecting actions $Pr(a|s)$ and the state value $V(s)$ , $\forall s \in S$ and $\forall a \in A$.
3:     loop (for each episode):
4:         Initialise state $s$.
5:         repeat (for each step of the episode):
6:             Select an action $a$ from state $s$ using $Pr(a|s)$ and execute the action.
7:             Receive immediate reward $r = r_s^a$ and observe the new state $s'$ i.e. acquire experience $s \overset{a;r}{\rightsquigarrow} s'$.
8:             Calculate the TD error $\delta \leftarrow [r + \gamma V(s')] - V(s)$.
9:             with TD error $\delta$ do
10:                Update actor by adjusting the action probabilities for state $s$.
                       $p(s,a) \leftarrow p(s,a) + \delta\beta\,[1 - Pr(a|s)]$.
11:                Update critic by adjusting the value of state $s$:
                       $V(s) \leftarrow V(s) + \delta\beta$.
12:            end with
13:            Advance to next state $s \leftarrow s'$.
14:        until end of episode e.g. $s$ is terminal.
15:    end loop
16: end procedure
```

Algorithm 2.6: An Actor Critic Algorithm

Q-Learning

Q-learning (Watkins 1989, Watkins & Dayan 1992) is an off-policy temporal difference algorithm. This was a very important development in reinforcement learning as the learned action value function Q converges on the optimal action value function Q^* independently of the policy being followed. In this case, the learned action-value function directly approximates the optimal action-value function, independent of the policy being followed. This dramatically simplifies the analysis of the algorithm and enabled early convergence proofs. The policy still has an effect in that it determines which state-action pairs are visited and updated. However, a basic requirement for correct convergence is that all pairs continue to be updated:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r_s^a + \gamma \max_{a'} Q(s',a') - Q(s,a)] \tag{2.24}$$

The Q-learning algorithm is shown in Figure 2.7. The algorithm is guaranteed to converge to the correct Q-values with probability one if the environment is stationary and depends on the next state only, every state-action pair continues to be visited, and the learning rate α is decreased appropriately over time.

```
1: procedure Q-LEARNING($\alpha$)
2:     Initialise state–action values $Q(s,a) \quad \forall s \in S, \quad \forall a \in A$
3:     loop (for each episode):
4:         Initialise state $s$.
5:         repeat (for each step of the episode):
6:             Select an action $a$ from state $s$ using $\pi$ derived from the current Q-values and execute the action.
7:             Receive immediate reward $r = r_s^a$ and observe the new state $s'$ i.e. acquire experience $s \overset{a,r}{\rightsquigarrow} s'$.
8:             Update state-action-value $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$
9:             Advance to next state $s \leftarrow s'$
10:        until end of episode e.g. $s$ is terminal.
11:    end loop
12: end procedure
```

Algorithm 2.7: The Q-Learning Algorithm

SARSA

Another approach to model learning similar to Q-learning is by means of SARSA. The SARSA algorithm was first explored by Rummery & Niranjan (1994) as modified Q-learning. The acronym

'SARSA' stands for *S*tate-*A*ction-*R*eward-*S*tate-*A*ction (Sutton 1996). The principles of SARSA and Q-Learning are quite similar. However, SARSA updates $Q(s,a)$ for the policy π that it is actually executing. In essence, unlike Q-Learning, SARSA is an on-policy algorithm. The Q-value that SARSA updates depends on the current state s, the action a selected in that state, the reward received on executing the action, the next state s' following the execution of action a in s and the action a' to be taken in state s'. The Q-value updates are based on the quintuple (s,a,r,s',a') using the following formula.

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r_s^a + \gamma Q(s',a') - Q(s,a)] \tag{2.25}$$

The parameter α in Equation 2.25 has same meaning as in Q-learning. The SARSA learning algorithm is shown in Algorithm 2.8.

```
 1: procedure SARSA($\alpha$)
 2:     Initialise state–action values $Q(s,a)$  $\forall s \in S$,  $\forall a \in A$
 3:     loop (for each episode):
 4:         Initialise state $s$.
 5:         Select action $a$ from $s$ using policy derived from Q-values.
 6:         repeat (for each step of the episode):
 7:             Execute action $a$.
 8:             Receive immediate reward $r = r_s^a$ and observe the new state $s'$ i.e. acquire experience $s \overset{a,r}{\rightsquigarrow} s'$.
 9:             Select action $a'$ from $s'$ using policy derived from Q-values.
10:             Update state-action-value $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)]$
11:             Advance to next state $s \leftarrow s'$
12:             Advance to next action $a \leftarrow a'$
13:         until end of episode e.g. $s$ is terminal.
14:     end loop
15: end procedure
```

Algorithm 2.8: The SARSA Learning Algorithm

The SARSA algorithm has two action selection steps 5 and 9 in Algorithm 2.8.

2.3.2.2 Model-based Methods

In this section we look at how an agent may solve a Markov decision task when it does not know in advance the accurate model of the task but wants to use a model-based approach supplemented by the data it collects. Unlike the model-free methods we described previously, the agent will need to rely on algorithms that operate by learning the true MDP model during its interaction with the task environment. Model-based approaches are generally based around the following:

- *Experience Acquisition:* the agent acquires experience from the task environment through the data it collects during its interaction,
- *Model Construction:* the agent uses the experience it acquires to construct estimates of model structure and parameters, and
- *Acting using the Constructed Model:* the agent uses the estimated model to inform and speed up learning.

Constructing estimates of model parameters can be done by counting the frequencies of observed experiences. We focus on system dynamics. To compute a maximum likelihood model of system dynamics an agent will usually require the following variables

$$\begin{aligned}
\#(s \overset{a}{\rightsquigarrow} s') &= \text{Number of transitions from state } s \text{ to } s' \text{ after executing action } a \\
&= \sum_t I(X_t = s, A_t = a, X_{t+1} = s') \\
\#(s \overset{a}{\rightsquigarrow} succ(s)) &= \text{Number of times the agent has executed action a in state } s \\
&= \sum_t I(X_t = s, A_t = a, X_{t+1} = s' \in succ(s)) \qquad (2.26) \\
& \qquad (2.27)
\end{aligned}$$

in which $succ(s)$ is the set of states that succeeds state s and $I(.)$is an indicator function that returns 1 if $(\cdot)$ occurs and 0 otherwise. A maximum likelihood estimate of transition probabilities is estimated as:

$$\hat{p}^a_{ss'} = \frac{\#(s \overset{a}{\rightsquigarrow} s')}{\#(s \overset{a}{\rightsquigarrow} succ(s))} \tag{2.28}$$

where the estimates $\hat{p}^a_{ss'}$ are the maximum likelihood transition probabilities that enforces the sum to one constraint for the probability measures. The parameter estimates are computed as normalised transition counts.

Given that we can now reasonably estimate the transition probabilities from data the next step is to understand how the estimates can be used to speed up learning. There are three main approaches.

Two Phase Learning: in which the agent separates the process into learning and acting phases, and the division can be arbitrary. In the first phase, the agent concentrates on the acquisition of experience without worrying about using the acquired experience to moderate behaviour. It uses pre-defined policies, or simply act randomly, and builds an estimate of transition probabilities with the data collected. Following the data collection, the agent then turns its attention to acting, using the constructed model to compute value and policy functions.

With luck the agent may be able to gather data to construct a reasonable model that will lead to good policies. However this approach is fraught with difficulties that include inefficiencies in the way data is gathered, because the agent can get stuck in states, and may suffer from appropriate depth and coverage of the task space

Learning via Off-line Dynamic Programming: in which the agent learns the model continuously through the agent's lifetime and at each time step it uses the current model to compute value functions and optimal policies. The agent interleaves learning with acting making efficient use of data, which is better than the two phase approach. However, the approach is still computationally demanding and does not address the exploration question adequately.

Learning via On-line Dynamic Programming: in which the agent learns the model continuously through the agent's lifetime and incorporates phases of model estimation with phases of value estimation. The approach was motivated by the theory of asynchronous dynamic programming (ADP). The ADP approaches are geared towards addressing a major drawback of conventional dynamic programming (DP) methods wherein computations involve operations over the entire state set of the MDP i.e. requiring sweeps of the state set. ADP algorithms are iterative DP algorithms that operate by backing up state values in a particular order, using currently available estimates of the values of other states. The order of back-ups may be rule based. The values of some states may be backed up a number of times before those of other states are backed up once. However, it is important that the values of all the states must continue to be backed up in order to achieve convergence correctly; no state should be ignored. Developing the concepts of ADP further, a class of value iteration algorithms termed adaptive real-time dynamic programming (ARTDP) algorithms has been defined (Kaelbling et al. 1996, Sutton & Barto 1998) and it is this class of algorithms that we work with in this thesis.

2.4 Chapter Summary

In this chapter we have described the framework of Markov decision processes and discussed how a performance criterion can be coupled with the process to specify a Markov decision task. We have described approaches to solving Markov decision tasks, organising the description around the notions of planning and learning. We described solving a Markov decision task with a known correct model of the underlying MDP and this we explained can be accomplished through value iteration, policy iteration and linear programming algorithms. We also described learning in MDP and explained that this occurs when we don't have a complete model of the underlying MDP. The approaches we described for learning in MDP are split between a) model-free methods using Monte Carlo sampling and temporal difference, and b) model-based learning using normalised transition counts to build the model and adaptive dynamic programming algorithms to act using the constructed model. Our focus in the rest of the thesis shall be on model-based methods and particularly how the agent could improve its estimates of model parameters using transferable prior information.

3

Capturing Uncertainty about the Task-Environment Model

In the previous chapters we highlighted the need for learning when there exist uncertainty over task environments. A simple frequentist approach was presented in chapter two to illustrate the learning involved. In this chapter and the following ones we go into more detail about learning in uncertain task environments particularly in relation to system dynamics. We explore common descriptions for modelling transition uncertainty and present a Bayesian view of learning in Markov decision processes with uncertain transition probabilities.

3.1 Introduction

As discussed in Chapter two, sequential decision making tasks can be formalised in terms of Markov decision processes (MDP) as maximizing a value function jointly determined by the policy chosen by the decision agent and the MDP parameters. That is, the decision agent attempts to solve the following optimisation problem:

$$\max_{\pi} V^{\pi}(s) = u\{\gamma, R, P, V^{\pi}(s')\} \quad \forall (s,s') \in S, \tag{3.1}$$

where π is a policy i.e. a statement of actions to take in each state s of the task environment, $V^{\pi}(s)$ is the value of state s from following policy π with a discount factor γ, and R, P are the

MDP's reward and transition parameters respectively. It is sometimes assumed that the agent has a complete knowledge of the environment in which case the MDP is assumed known with certainty and the policy can be determined straightforwardly using, for example, the planning algorithms we described in Chapter two. Unfortunately, in real world, the parameters of the MDP are often subject to uncertainty.

The uncertainty can be intrinsic and unavoidable, may arise from many other reasons such as: a) imprecise or conflicting elicitations from experts, b) insufficient data from which to estimate the models, c) incorrect measurements, or d) nonstationary variations in the model values. In the face of uncertainty, establishing the MDP parameters and the optimal policy is no longer trivial. The simplest type of uncertainty arises when interactions within the world can be stochastic. At the other end of the spectrum is uncertainty that arises when we do not know exactly how the world works. Decision-making tasks under such strong or genuine uncertainty are generally difficult to tackle; in such environments it might be hard to evaluate actions and plans and, consequently, to find the optimal action or policy or plan to follow. It has long been known that neglecting uncertainty in the MDP parameters and instead solving the decision problem of Equation 3.1 with some roughly right parameters or certainty equivalents could over-simplify the decision task, render a computed solution highly sub-optimal with possibly conflicting and incorrect policies.

In this and subsequent chapters of the thesis we go into more detail about learning in uncertain task environments. We focus specifically on transition uncertainty and Bayesian approaches to learning in Markov decision processes with uncertain transition probabilities. The remainder of this chapter is organised into three sections. In Section 3.2 we explore common descriptions for modelling transition uncertainty and describe how we estimate state transition probabilities using multinomial distributions. In addition, we describe in the section a more precise Bayesian notion of transition uncertainty based on the idea of credibility region and discuss how to measure distances between two transition models. We introduce in Section 3.3 Bayesian learning in MDPs with uncertain transition models. The chapter ends in Section 3.4 with a chapter summary.

3.2 Transition Uncertainty

We now focus on the problem of transition uncertainty in Markov decision processes. As introduced in the preceding section, transition uncertainty exists naturally in tasks where a fully determined transition model P of system dynamics is not readily available and the model must be estimated from experimentation. One of the very first steps in handling transition uncertainty is describing the uncertainty in the transition model. There are many ways of representing transition uncertainty and the more common descriptions are discussed in the next section.

3.2.1 Common Descriptions for Modelling Transition Uncertainty

This section describes how uncertainties in transition probabilities are modelled. The approaches we describe vary from a partial knowledge of transition probabilities to Bayesian methods that place a probability density over the space of all possible transition models. Recall from Chapter two that the transition probability matrix P of an MDP under action a in $(|S| \times |S|)$ dimensional finite state space is

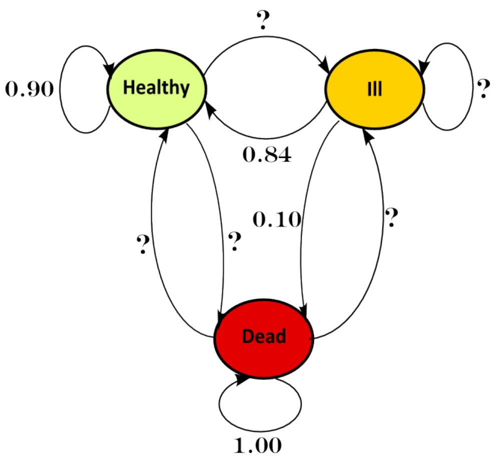
$$P^a = \begin{bmatrix} p^a_{11} & p^a_{12} & p^a_{13} & \cdots & p^a_{1|S|} \\ p^a_{21} & p^a_{22} & p^a_{23} & \cdots & p^a_{2|S|} \\ p^a_{31} & p^a_{32} & p^a_{33} & \cdots & p^a_{3|S|} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p^a_{|S|1} & p^a_{|S|2} & p^a_{|S|3} & \cdots & p^a_{|S||S|} \end{bmatrix}$$

and the entries $p^a_{ss'}$ are the conditional probabilities $Pr(X_{t+1} = s'|X_t = s)$ at the single step t to $t+1$, with $s, s' \in S$. The transition probabilities naturally satisfy the constraints $0 \leq p^a_{ss'} \leq 1$; $\sum_{s' \in S} p^a_{ss'} = 1$ $\forall\ s \in S$. For a given state $s \in S$ the constraints depend exclusively on the s-th row of the transition matrix. That is, for each $s \in S$, the constraints are not coupled by $p^a_{ss'}$ $\forall rows\ s' \neq s \in S$. As a result, the transition matrix P can be split by rows as follows.

$$P^a = \left[\vec{p}^{\,a}_1, \vec{p}^{\,a}_2, \ldots, \vec{p}^{\,a}_{|S|}\right]$$

where $\vec{p}^{\,a}_s$ is the row of the transition matrix in state s.

One way of incorporating uncertainty in the transition matrix P is to consider that the transition probabilities are partially available, that is, some elements of the matrix are unknown (Zhang & Lam 2010). Recall the agent wellbeing task we described in Figure 2.1 of Chapter two and take, for example, a situation in which an incomplete description is available for the transition matrix under the repair action illustrated in Figure 3.1 where each unknown element is labelled as '?'. An



$$P^{repair} = \begin{array}{c|ccc} & Healthy & Ill & Dead \\ \hline Healthy & 0.90 & ? & ? \\ Ill & 0.84 & ? & 0.10 \\ Dead & ? & ? & 1.00 \end{array}$$

Figure 3.1: A simple instance of the agent wellbeing task with partially known transition probability matrix

equivalent description of uncertainty can be obtained through a classical approach that assume that the transition probabilities are unknown but that they exist in a finite region of n-dimensional space bounded by known hyperplanes. This leads us nicely to the subject of polytopic transition models which we describe next.

3.2.1.1 Polytopic Transition Models

A common approach for describing uncertainty in task environments is to consider the transition matrix to lie in a given set, most typically a polytope. Coxeter (1973) defines polytope as geometrical entity represented by the general term of the infinite sequence 'point, line segment, polygon, polyhedron, ...' or, more specifically, as a finite region of n-dimensional space bounded by a finite number of hyperplanes. Using a general polytopic models to handle uncertainty in process dynamics is often not tractable as it incurs a significant additional computation effort and sometimes a poor representation of statistical uncertainty that leads to conservative policies. These issues with polytrophic representations of transition models are alleviated when the uncertainty is described by an interval matrix or sets (e.g. convex sets in Goncalves, Fioravanti & Geromel (2010), and compact sets in Kalyanasundaram, Chong & Shroff (2002)) or when the polytope is described by its vertices.

The following rows define each of the uncertain polytope vertices in the example of Figure 3.1.

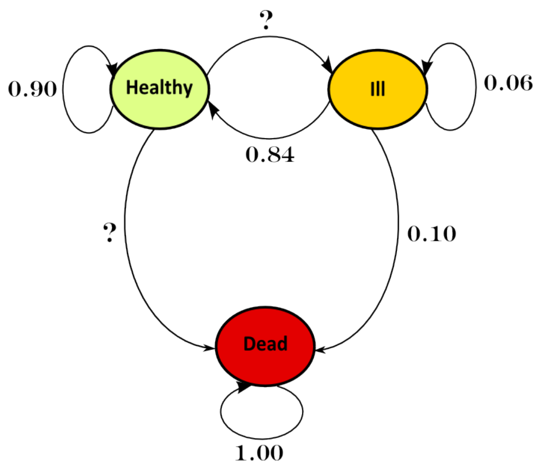
$$\begin{aligned}
&\vec{p}_1^{\,repair,1} = [0.90, 0.10, 0.00] \quad \vec{p}_1^{\,repair,2} = [0.90, 0.00, 0.10] \\
&\vec{p}_2^{\,repair,1} = [0.84, 0.06, 0.10] \\
&\vec{p}_3^{\,repair,1} = [0.00, 0.00, 1.00]
\end{aligned}$$

It is important to note that whenever a row contains none or one '?' or when an element in the row equals 1.0 then, as a result of the normalisation constraint, only one row is generated. This reduces the uncertainty in the process, as illustrated in Figure 3.2. Two or more '?' appearing in a row will produce the same number of vertices when none of its elements have a transition probability that is equal to 1, as it is in the case of $\vec{p}_1^{\,repair,1}$ and $\vec{p}_1^{\,repair,2}$ above.

Interval based approaches to handling uncertainty provides upper and lower bounds on the transition probability where

$$\vec{p}_s^{\,a} = \{p_{ss'}^a | p_{\downarrow,ss'}^a \leq p_{ss'}^a \leq p_{\uparrow,ss'}^a, \; \forall s' \in S\} \tag{3.2}$$

where $p_{\downarrow,ss'}^a$ and $p_{\uparrow,ss'}^a$ are the lower and upper bounds on $p_{ss'}^a$ respectively. Although the concept of



$$P^{repair} = \begin{array}{r} \\ Healthy \\ Ill \\ Dead \end{array} \begin{array}{c} \begin{array}{ccc} Healthy & Ill & Dead \end{array} \\ \begin{bmatrix} 0.90 & ? & ? \\ 0.84 & 0.06 & 0.10 \\ 0.0 & 0.0 & 1.00 \end{bmatrix} \end{array}$$

Figure 3.2: The simple instance of the agent wellbeing task with reduced transition uncertainty.

lower and upper bounds presents a relatively simpler and intuitive representation of uncertainty it is subsumed by representations using convex sets. A convex set approach to incorporating uncertainty in transition probabilities (Goncalves et al. 2010) usually consider the vector $\vec{p}_s^a$ unknown but belonging to a convex set with known vertices for all $k \in K_s$, specified as:

$$\vec{p}_s^a \in \{\sum_{k=1}^{K_s} \lambda_s^k \vec{p}_s^{a,k} : \lambda_s^k \in \Lambda\} \tag{3.3}$$

for all $s \in S$ and $a \in A$, where $\lambda_s^k \in \Lambda$ are the weights applied to each of the vertices $\vec{p}_s^{a,k}$. Liu (2000) used convex sets to model simplex constraints, such as monotonicity and convexity or concavity, on the probabilities of a set of discrete distributions, including multinomial, and described implementations using Expectation-maximization (EM) and Data Augmentation (DA) algorithms. The simplex approach is particularly relevant for handling statistical uncertainty in process dynamics considering that transition probabilities can be estimated using multinomial distributions.

One can also resort to using imprecise probabilities to represent uncertainty in MDPs. The term MDP with Imprecise Probabilities (MDP-IP) was proposed by White & Eldeib (1994). Satia & Lave (1973) described a related concept they termed MDP with uncertain transition probabilities. To specify an MDP-IP, all elements of an MDP must be specified except the transition probabilities which for MDP-IPs are specified as a set of probabilities for each transition between states. These sets are referred to as transition credal sets. MDP-IPs are identical to MDPs, except that the transition function is replaced with a set of distributions. Solutions to the MDP-IP require nonlinear optimization and can be extremely time-consuming in practical applications. The main computational problem in the solution of MDP-IPs is the need to repeatedly solve nonlinear constrained optimization problems.

3.2.1.2 Probability Distribution over Possible Models

In a Bayesian approach to handling of transition uncertainty in MDPs, we assume that there is a space $\mathbb{P}$ of all the possible transition functions (parametric models) for the MDP and that there exists a *belief state* over this space. The belief state defines a probability density $f(P|M)$ over the MDPs. The density is parameterised by $M \in \mathbb{M}$. In the Bayesian approach, the unknown parameter P is treated as a random variable, with prior distribution $f(P|M)$ which represents what one knows about the parameter before observing transitions.

The choice of prior distribution is a relevant issue in Bayesian methods. In practical applications, when a prior is specified, it is usually said that it approximately reflects an experimenter's prior opinion. There are two main reasons why prior information sometimes requires approximation. Firstly, there are situations in which the task of translating prior information into a probability distribution may be quite hard task and use of an approximation is necessary. Secondly, even when an expression of the true prior distribution is available, it might happen that the posterior cannot be exactly evaluated. From robust Bayesian viewpoint, Good (1950) proposed that subjective information can be quantified in terms of a class of possible distributions from which has emerged an $\varepsilon-$contaminated classes of prior distributions which is a mixture of base prior and a class of prior distributions e.g. Berger & Berliner (1986). Procedures that select the priors in the $\varepsilon-$contaminated classes acts as an automatic 'robustifier' for the base prior in the sense that the resulting posteriors and Bayes estimators are more robust to misspecification in base prior elicitation (Berger & Berliner 1986, Berger 1990).

3.2.2 Estimating Transition Probabilities

This section describes how state transition probabilities can be estimated using multinomial distributions. A multinomial distribution is the probability distribution of outcomes from a multinomial experiment. It is a frequently used distribution in statistics with the following properties:

- The experiment consists of a number of repeated trials.
- Each trial has a discrete number of possible outcomes; two or more outcomes in a multinomial experiment.
- On any given trial, the probability that a particular outcome will occur is constant.
- The trials are independent; that is, the outcome on one trial does not affect the outcome on other trials.

Let us consider an experiment consisting of n independent trials and for each trial assume the probability of a particular outcome k is p_k; $p_1, p_2, \ldots, p_k, \ldots, p_K$ where $p_k \geq 0$ for $k = 1, 2, \ldots, K$ and $\sum_k p_k = 1$. Let x_k indicate the number of times outcome k was observed over the n trials. The vector $\vec{x} = (x_1, x_2, \ldots, x_k, \ldots, x_K)$ follows a multinomial distribution with parameters n and $\vec{p}$ where $\vec{p} = (p_1, p_2, \ldots, p_k, \ldots p_K)$. The probability mass function of this multinomial distribution is given by:

$$f(x_1, x_2, \ldots, x_k, \ldots, x_K | n, p_1, p_2, \ldots, p_k, \ldots p_K) = \frac{n!}{x_1! x_2! \ldots, x_k!, \ldots, x_K!} \prod_{k=1}^{K} p_k^{x_k} \tag{3.4}$$

for non-negative integers $(x_1, x_2, \ldots, x_k, \ldots, x_K)$. The experiment and resulting distribution is binomial where there are only two outcomes.

The modelling of transition probabilities using multinomial distributions can be thought of as follows: a) in state s, we observe m^a_{s0} transitions under action a, b) the observed counts are stored in an $|S| \times |S|$ matrix of observation counts M^a in which $m^a_{ss'}$ is the total number of times we have observed transition $s \overset{a}{\rightsquigarrow} s'$, c) the transition model $\vec{p}^a_s \in P$ is a multinomial with probability mass function

$$f(\vec{m}^a_s | m^a_{s0}, \vec{p}^a_s) = \frac{m^a_{s0}!}{m^a_{s_1}! m^a_{s_2}! \dots m^a_{s_N}!} \prod_{s'=1}^{N} (p^a_{ss'})^{m^a_{ss'}} \tag{3.5}$$

where $\vec{m}^a_s = (m^a_{s_1}, m^a_{s_2}, \dots, m^a_{s_N})$ are observation counts ($m^a_{s0} = \sum_{s'} m^a_{ss'}$) and, as before, the transition probabilities must satisfy the constraints that $0 \leq p^a_{ss'} \leq 1; \sum_{s'} p^a_{ss'} = 1$ for $s = 1, 2, \dots N$, and finally d) an estimate of transition probability $\hat{p}^a_s$ can be extracted from the observation counts as follows:

$$\hat{p}^a_{ss'} = \frac{m^a_{ss'}}{\sum_{s' \in S} m^a_{ss'}} \quad \forall s \in S. \tag{3.6}$$

In Bayesian Settings, a useful prior for the probability parameter p of the multinomial distribution is *Dirichlet distribution*. We describe the Dirichlet distribution approach to modelling transition probabilities in the next section.

3.2.2.1 Dirichlet Distribution of Transition Models

We saw in the previous section that the transition model P of an MDP can be regarded as a generalisation of an independent multinomial model. A convenient prior distribution for the model is Dirichlet. The Dirichlet distribution is a natural conjugate prior of the parameters of the multinomial distribution. The probability density of the Dirichlet distribution for variables $\vec{p}^a_s = (p^a_{s_1}, p^a_{s_2}, \dots, p^a_{s_N})$ with parameters $\vec{m}^a_s = (m^a_{s_1}, m^a_{s_2}, \dots, m^a_{s_N})$ is defined by:

$$f(\vec{p}^a_s | \vec{m}^a_s) = \frac{1}{Z(\vec{m}^a_s)} \prod_{s'=1}^{N} (p^a_{ss'})^{m^a_{ss'} - 1} \tag{3.7}$$

for the possible N successor states of state $s \in S$ and action $a \in A$, given that $m^a_{ss'} > 0 \; \forall s, s' \in S$. The parameters $\vec{m}^a_s$ can be interpreted as *prior observation counts* for events governed by $\vec{p}^a_s$. The normalisation constant $Z(\vec{m}^a_s)$ is:

$$Z(\vec{m}^a_s) = \frac{\prod_{s'=1}^{N} \Gamma(m^a_{ss'})}{\Gamma(\sum_{s'=1}^{N} m^a_{ss'})} \tag{3.8}$$

As before, let $m^a_{s0} = \sum_{s'=1}^{N} m^a_{ss'}$. Table 3.1 contains expressions for the mean and variance of the Dirichlet distribution in the context of transition models.

Expectation	$E[p^a_{ss'}]$	$\frac{m^a_{ss'}}{m^a_{s0}}$
Variance	$Var[p^a_{ss'}]$	$\frac{m^a_{ss'}(m^a_{s0}-m^a_{ss'})}{(m^a_{s0})^2(m^a_{s0}+1)}$

Table 3.1: Some Properties of the Dirichlet Distribution

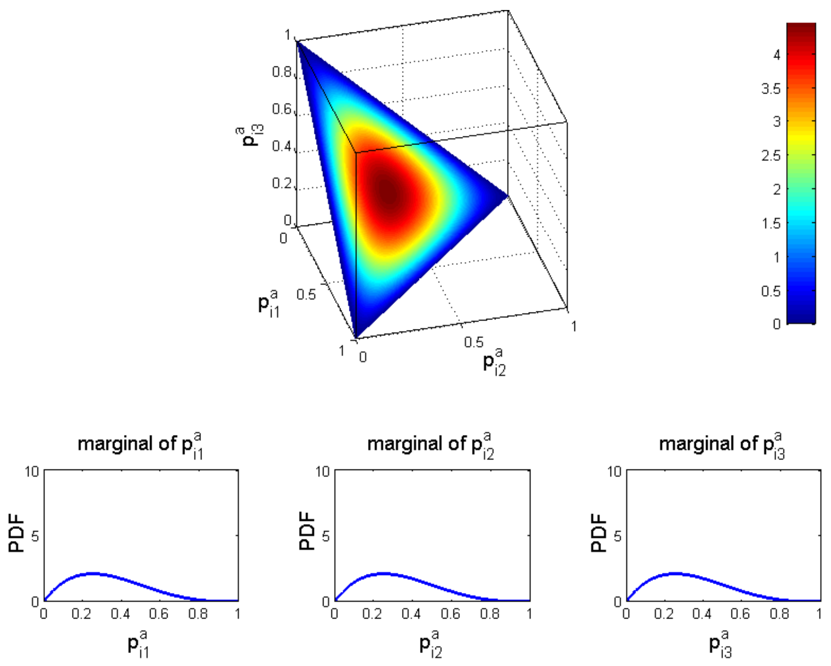
When $m^a_{ss'} \to 0$, the Dirichlet distribution becomes non-informative. This means that all the $p^a_{ss'}$ will stay the same if all the parameters $m^a_{ss'}$ are scaled with the same multiplicative constant. The variances will, however, get smaller as $m^a_{ss'}$ grows.

Example 3.1 *Lets consider the estimation of transitions from ill-health state of our wellbeing task under repair action. Assuming Dirichlet prior probability distribution on P, the resulting density functions are shown in Figure 3.3.*

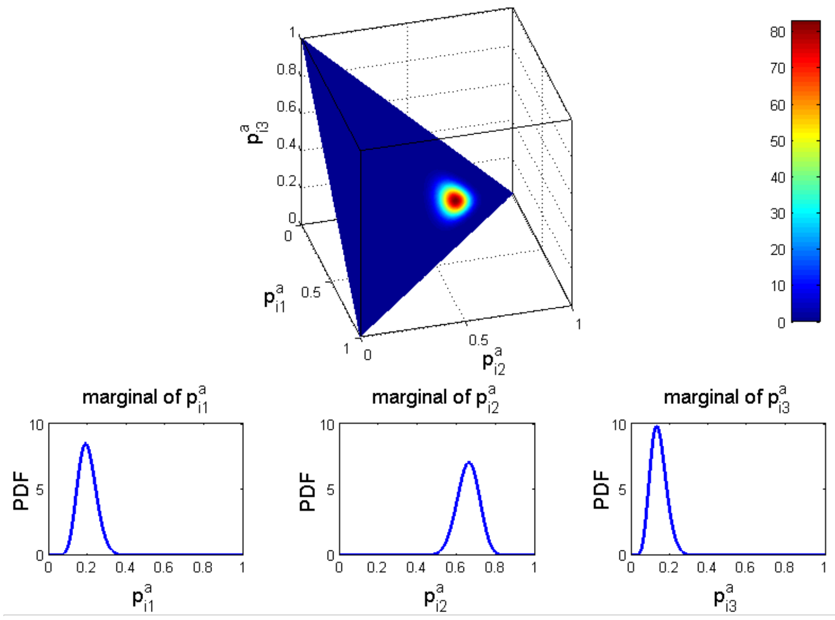
The Dirichlet distribution entertains several properties that are known to be very useful in statistical inference. In particular, estimates derived using Dirichlet priors are *consistent* (the estimate converges with probability one to the true distribution), *conjugate* (the posterior distribution is also a Dirichlet distribution), and can be computed efficiently (all queries of interest have a closed-form solution). Furthermore, theoretical studies of online predictions of individual sequences show that prediction using Dirichlet priors is competitive with any other prior distribution; with the multinomial distribution it is easy to encode a prior by assigning the initial counts in M appropriately. Complete uncertainty of the domain can be setting M to the constant matrix with entries equal to 1. Events which are more likely to occur can be assigned a higher value.

Dirichlet priors are known to be unwieldy in some key applications. These applications are characterized by several distinct features, such as follows.

- The set of possible outcomes is very large and often not known in advance.



(a) $\vec{m}^a_i = (2,2,2)$, $\vec{p}^a_i = (0.3333, 0.3333, 0.3333)$



(b) $\vec{m}^a_i = (14,46,10)$, $\vec{p}^a_i = (0.2000, 0.6571, 0.1429)$

Figure 3.3: Dirichlet distributions for the example transition models a) $\vec{m}^a_i = (2,2,2)$ and b) $\vec{m}^a_i = (14,46,10)$ having mean transition probabilities $[0.3333, 0.3333, 0.3333]$ and $[0.2000, 0.6571, 0.1429]$ respectively.

- The number of training examples is small compared to the number of possible outcomes.
- The approaches to multinomial estimation tend to be domain–independent; they make little use of prior knowledge about a specific domain. In many domains where multinomial distributions are estimated there is often at least weak prior knowledge about the potential structure of distributions.
- The outcomes that have positive probability constitute a relatively small subset of the possible outcomes. This subset is not known in advance.
- There usually is knowledge of the pattern that underlies the possible outcomes.

3.2.2.2 Generating Samples of Transition Models

How do we generate samples from Dirichlet distributions? As we shall see later in this thesis particularly in our treatment of truncated Dirichlet models, we require a good method of generating samples of transition models from Dirichlet distributions. In Bayesian settings, sampling approaches aim to provide estimates of both the model parameters and their uncertainty by generating samples of models that are distributed according to the true posterior probability distribution of the model parameters conditioned on the data.

There exist several algorithms for generating samples from Dirichlet distributions and they include (Ştefănescu 1989, Narayanan 1990, Hung, Balakrishnan & Cheng 2011, Ng, Tian & Tang 2011):

- The Gamma method based on the relation between Dirichlet and Gamma distributions,
- Different rejection techniques or the use of the classical inverse method, and
- The use of a transformation of a uniform random vector over a bounded domain.

In addition, Frigyik, Kapila & Gupta (2010) discussed a method commonly referred to as Pòlya's urn and a 'stick breaking' approach. They compared the two methods to that based on transforming Gamma-distributed random variables. Performance analysis of the algorithms indicate that the Gamma approach is fast and computationally more efficient in comparison to others (Narayanan 1990, Frigyik et al. 2010). We use the Gamma approach in this thesis. Given a Dirichlet distribution, Dir($\vec{m}$), the sampling algorithm based on the Gamma approach can be straightforwardly accomplished in two steps as follows:

step 1: Generate gamma realisations: for $i = 1, 2, \ldots, k$, draw a number ζ_i from $\Gamma(m_i, 1)$,

step 2: Normalise them to form a probability mass function: for $i = 1, 2, \ldots, k$, set $x_i = \frac{\zeta_i}{\sum_{j=1}^{k} \zeta_j}$. Then $\vec{x}$ is a realisation of Dir ($\vec{m}$).

The Gamma distribution $\Gamma(\kappa, \theta)$ is defined by the following probability density:

$$f(x|\kappa, \theta) = x^{\kappa-1} \frac{e^{-x/\theta}}{\theta^{\kappa}\Gamma(\kappa)} \tag{3.9}$$

The density is in terms of the gamma function $\Gamma(\cdot)$ and is parameterized by $\kappa > 0$ a shape parameter and $\theta > 0$ a scale parameter. The scale that is used to generate the Gamma variates is irrelevant, as it cancels in the ratio. Incidentally, only $k-1$ variates need to be generated as the k^{th} is obtained from the fact that $\sum_j x_j = 1$. Given the efficiency of modern gamma generators (Marsaglia & Tsang 2000), the generation of independent Dirichlet variates is particularly undemanding.

3.2.3 Credible Set of Transition Models

Whilst the Dirichlet density provides us with a good measure of transition uncertainty through its characterisation of the hyperparameters $\vec{m}_i^a$ for each row $i = 1, 2, \ldots, |S|$ of the transition model, we will be interested in a more precise notion of uncertainty in terms of posterior intervals that are expressed in probability statements of the form:

$$Pr(p^a_{\downarrow,ij} \leq p^a_{ij} \leq p^a_{\uparrow,ij}) = 1 - \alpha \tag{3.10}$$

where $p^{a}_{\downarrow,ij}$ and $p^{a}_{\uparrow,ij}$ are lower and upper bounds respectively on p^{a}_{ij}. The interval $\left[p^{a}_{\downarrow,ij}, p^{a}_{\uparrow,ij}\right]$ is called a $100(1-\alpha)\%$ *Bayesian confidence interval* or *credible set* for the parameter of interest p^{a}_{ij}. It is the Bayes version of the classical confidence intervals used in frequentist statistics.

The Bayesian counterpart of the frequentist idea of a confidence interval is usually referred to as a '(posterior) credible interval' and corresponds to $100(1-\alpha)\%$ of the posterior probability $f(P|M)$. Commonly used are central posterior intervals and regions of highest posterior density. In contrast to frequentist confidence intervals, Bayesian credible intervals possess individual coverage probability. For example, a single 95% Bayesian-credible interval for a parameter is interpreted as having 95% probability of containing the true parameter value. A single frequentist confidence interval, on the other hand, either contains the parameter value of not – there is no probability statement to be made. Frequentist coverage probabilities arise from the (possibly hypothetical) replication of a procedure and taking the ratio of favourable outcomes.

The idea of credibility regions (Berger 1985, Chen & Shao 1999) offers a more precise notion of the uncertainty in a row of transition matrix and is more formally defined as follows:

Definition 3.1 (The Credibility Region:) *An* $100\times\eta\%$ *credibility region for parameter p is a subset* $\mathbf{P}$ *of* $\mathbb{P}$ *of the form:*

$$\mathbf{P} = \{\vec{p} \in \mathbb{P} | f(\vec{p}|\vec{m}) \geq k(\eta)\}$$

and $k(\eta)$ *is the largest constant such that:*

$$\int_{\mathbf{p}} f(\vec{p}|\vec{m})dp \geq \eta. \tag{3.11}$$

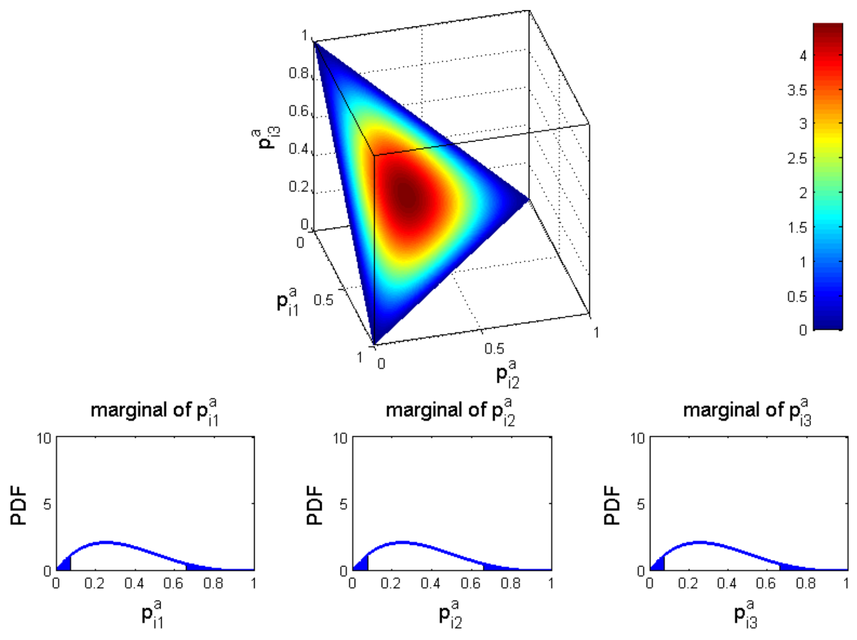
That is, for a given prior distribution, the credibility region $\mathbf{p}$ is such that the overall probability mass of the density covers a $100\times\eta\%$ region and the likelihood of the density is at least $k(\eta)$. Solution to the integration problem (Equation 3.11) for the credibility region cannot be achieved in closed form for the Dirichlet density. There have been several suggested approaches for constructing an highest posterior density (HPD) credible region for continuous distributions including from a general univariate density (Chen & Shao 1999, Gewali, Ntafos & Singh 2002). Even though some of the approaches are fairly efficient for Beta density, extending to higher dimensions like Dirichlet is difficult and computationally expensive. Our focus in this thesis is on algorithms that use Bayesian credible intervals and to estimate the credible intervals we use an F-distribution table method (Algorithm 3.1, adapted from Nicholson (1985)) and a simple Monte Carlo sampling method (Algorithm 3.2). Figure 3.4 shows estimates of credible intervals for two sample transition models.

```
1: procedure $\left[p^a_{\downarrow,ik}, p^a_{\uparrow,ik}\right]$ = F-DISTRIBUTION (CI)($\vec{m}^a_i$, $k$, $\alpha$)
2:     $m^a_{i-} \leftarrow \sum_{j \neq k} m^a_{ij}$
3:     $z_1 \leftarrow 2 \times (m^a_{ik} + 1)$
4:     $z_2 \leftarrow 2 \times (m^a_{i-} + 1)$
5:     $z \leftarrow F_{1-\alpha, z_1, z_2}$
6:     $p^a_{\uparrow,ik} \leftarrow (1 + z \times \frac{z_1}{z_2})^{-1}$
7:     Redo steps 2 through 6 for the second credible limit
8: end procedure
```

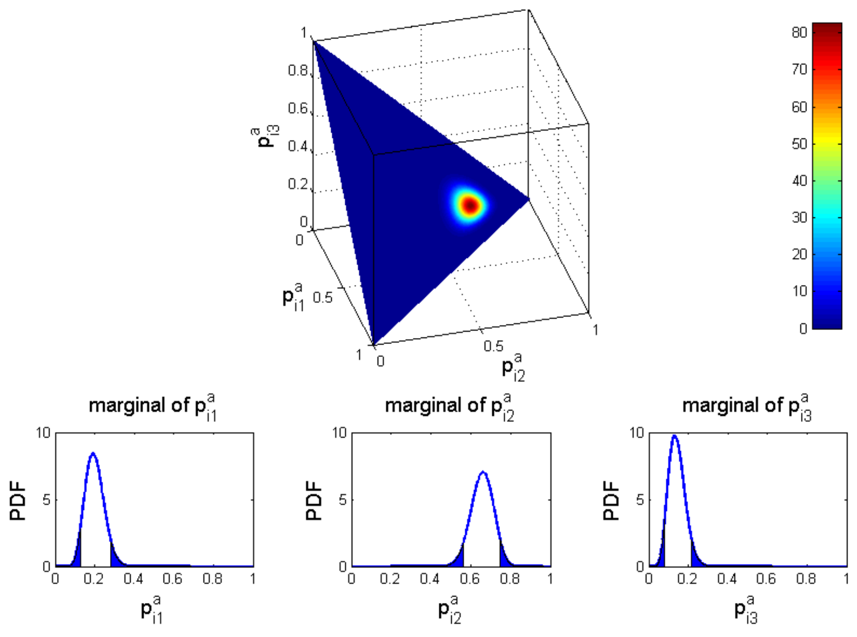
Algorithm 3.1: Credible Interval (CI) using F-Distribution Tables. Adapted from Nicholson (1985)

```
1: procedure $\left[p^a_{\downarrow,ik}, p^a_{\uparrow,ik}\right]$ = MONTE CARLO CI($\vec{m}^a_i$, $k$, $N$, $\alpha$)
2:     $m^a_{i-} \leftarrow \sum_{j \neq k} m^a_{ij}$
3:     $rands \leftarrow$ random (Beta, $N$, $m^a_{i-}$, $m^a_{ik}$)
4:     $\left[p^a_{\downarrow,ik}, p^a_{\uparrow,ik}\right] \leftarrow Quantile(rands, 1 - \alpha, \alpha)$
5: end procedure
```

Algorithm 3.2: Credible Interval (CI) through Simple Monte Carlo



(a) $\vec{m}^a_i = (2,2,2)$, $\vec{p}^a_i = (0.3333, 0.3333, 0.3333)$



(b) $\vec{m}^a_i = (14,46,10)$, $\vec{p}^a_i = (0.2000, 0.6571, 0.1429)$

Figure 3.4: Credible regions for the example transition models a) $\vec{m}^a_i = (2,2,2)$ and b) $\vec{m}^a_i = (14,46,10)$ having mean transition probabilities (0.3333,0.3333,0.3333) and (0.2000,0.6571,0.1429) respectively.

3.2.4 Probabilistic Distance Measures

Probabilistic distance measures between two probability distributions are very important metrics to evaluate the similarity for data of statistical nature. If the parameters of two probability density functions are known or can reliably be estimated, a single numerical value can be calculated that assesses how far or close two stochastic information sources are from each other. Rauber, Braun & Berns (2008) analyse probabilistic distances of the Dirichlet distribution and its particular two-parameter instantiation, the Beta distribution. Their work is motivated by the fact that the distribution has a variety of possible applications where similarity measures between sample sets are required and that probabilistic distances between sample sets have not been investigated so far for this particular distribution.

Consider two probability density functions $f_1(P|M_1)$ and $f_2(P|M_2)$ of a $|S|$ dimensional continuous random variable P defined by their functional forms f_1, f_2 and parameters M_1, M_2 respectively. A probabilistic distance measure J between the two probability density functions is a functional that measures the difference Δ integrated over the domain of P.

$$J(f_1, f_2, M_1, M_2) = \int_P \Delta[(f_1, f_2, M_1, M_2)]dP \tag{3.12}$$

Rauber et al. (2008) gives the analytical definitions of the Chernoff, Bhattacharyya and Jeffreys-Matusita probabilistic distances between two Dirichlet distributions and two Beta distributions as its special case. They showed the inappropriateness of some other measures including the Kullback-Leibler Divergence for calculating distances between Dirichlet distributions.

The Bhattacharyya coefficient ρ between two probability distributions described in the functional forms f_1, f_2 and their respective parameters M_1, M_2 is defined as (Bhattacharyya 1943, Kailath 1967):

$$\rho(M_1, M_2) = \int \sqrt{f_1(P|M_1) f_2(P|M_2)} dP \tag{3.13}$$

From Equation 3.7, we express the Dirichlet probability density function of transition probabilities $\vec{p}_s^a$ from state s under action a as follows:

$$f(\vec{p}_s^a|\vec{m}_s^a) = \frac{1}{Z(\vec{m}_s^a)} \prod_{s'=1}^{N} (p_{ss'}^a)^{m_{ss'}^a - 1} \tag{3.14}$$

with Dirichlet parameters $\vec{m}_s^a = (m_{s1}^a, m_{s2}^a, \ldots, m_{s|S|}^a)$, $N = |S|$, constraints $m_{ss'}^a > 0$ and $\sum_{s'} p_{ss'}^a = 1$. The normalising constant $Z(\vec{m}_s^a)$ which forces $f(\vec{p}_s^a|\vec{m}_s^a)$ as a probability density function to integrate to unity over the domain of P is the multinomial beta function with value:

$$Z(\vec{m}_s^a) = \frac{\prod_{s'=1}^{N} \Gamma(m_{ss'}^a)}{\Gamma(\sum_{s'=1}^{N} m_{ss'}^a)} \tag{3.15}$$

Following Rauber et al. (2008) in their definition of Bhattacharyya coefficient ρ for Dirichlet distributed densities, we write the Bhattacharyya coefficient ρ between two Dirichlet distributed transition models $\vec{p}_s^a$, $\vec{m}_s^{a,1}$ and $\vec{p}_s^a$, $\vec{m}_s^{a,2}$ as follows:

$$\rho(\vec{m}_s^{a,1}, \vec{m}_s^{a,2}) = \frac{1}{Z(\vec{m}_s^{a,1})^{\frac{1}{2}} Z(\vec{m}_s^{a,2})^{\frac{1}{2}}} \int \prod_{s'=1}^{N} (p_{ss'}^a)^{\frac{m_{ss'}^{a,1}}{2} + \frac{m_{ss'}^{a,2}}{2} - 1} d\vec{p}_s^a \tag{3.16}$$

Let $\beta_{ss'}^a = \frac{m_{ss'}^{a,1}}{2} + \frac{m_{ss'}^{a,2}}{2}$, $s' = (1, 2, \ldots, |S|)$. Then $f(\vec{p}_s^a|\vec{\beta}_s^a)$ can be seen as a Dirichlet distribution with parameters $\vec{\beta}_s^a$ in which $\beta_{ss'}^a > 0\ \forall s'$. In effect, $\int f(\vec{p}_s^a|\vec{\beta}_s^a) = 1$ and $\int \prod_{s'=1}^{N} (p_{ss'}^a)^{\beta_{ss'}^a - 1} d\vec{p}_s^a = Z(\vec{\beta}_s^a)$. Hence Equation 3.16 can be expressed as follows:

$$\rho(\vec{m}_s^{a,1}, \vec{m}_s^{a,2}) = \frac{Z(\frac{\vec{m}_s^{a,1}}{2} + \frac{\vec{m}_s^{a,2}}{2})}{\sqrt{Z(\vec{m}_s^{a,1}) Z(\vec{m}_s^{a,2})}} \tag{3.17}$$

and from the Bhattacharyya coefficient ρ defined in Equation 3.13 we can express the Bhattacharyya distance between the two Dirichlet densities, in logarithm form, as follows:

$$J_B(\vec{m}_s^{a,1}, \vec{m}_s^{a,2}) = -ln\ \rho(\vec{m}_s^{a,1}, \vec{m}_s^{a,2}) \tag{3.18}$$

The logarithm form prevents numerical overflows in the computation of Gamma $(\cdot)$ when $(\cdot)$ comes near 171.61.

In Chapter eight of this thesis we will need to calculate distances between Dirichlet distributions for the degree of match between historical data and imaginary data drawn from the user specified constraints. To do this however the measure would be required to be bounded between [0, 1]. A bounded measure is reported by Roman, Jolad & Shastry (2012) that uses base 2 for the logarithm, normalising the maximum value to 1.

$$J'_B(\vec{m}_s^{a,1}, \vec{m}_s^{a,2}) = -log_2 \left[\frac{1 + \rho(\vec{m}_s^{a,1}, \vec{m}_s^{a,2})}{2} \right] \tag{3.19}$$

We are adopting the bounded measure in this thesis. We validated the measure for a range of probability distributions and the result is shown in Figure 3.5.

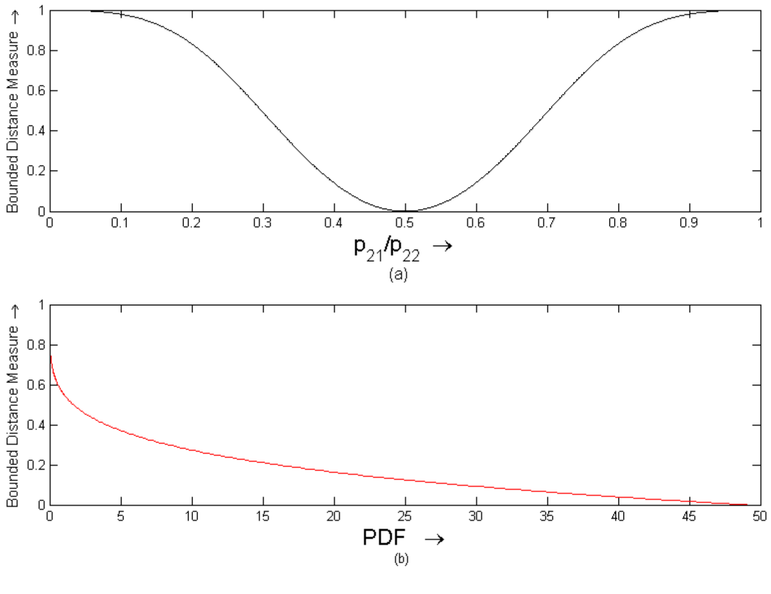


Figure 3.5: An illustration of the bounded distance measures between two Dirichlet distributions. In this figure, we consider two Dirichlet densities $\vec{m}_1 = \vec{p}_1 * 60$ and $\vec{m}_2 = \vec{p}_2 * 60$ where $\vec{p}_1 = [0.33334, 0.33333, 0.33333]$ and $\vec{p}_2 = [0.33334, p_{21}, p_{22}]$.
(a) plots $\frac{p_{21}}{p_{22}} \in [0,1]$ against the bounded measure $J'_B(\vec{m}_1, \vec{m}_2)$
(b) plots the probability density function $f(\vec{p}_2|\vec{m}_1)$ against the bounded measure $J'_B(\vec{m}_1, \vec{m}_2)$.

3.3 Bayesian Learning with Uncertain Process Models

As discussed in Chapter two, an agent that is learning to perform a task in unknown or partially known MDPs will start with a parameterised *'world model'* or a skeletal description of the task environment. The job of the agent is to acquire experience while interacting with the environment and use the experience to estimate the unknown parameters of the world model. The MDP that results from the estimated world model can be solved optimally using the planning techniques we described in Chapter two. The agent can approach the model based learning task using either frequentist methods that we described in Chapter two or alternatively the agent could use Bayesian techniques. Our focus in the remainder of this chapter is on Bayesian approaches.

Bayesian learning in MDPs can be traced back to (Bellman & Kalaba 1959*a*, Bellman & Kalaba 1959*b*) where adaptive control processes are introduced. Bayesian Learning in MDPs have two distinctive features namely, a) a *prior* belief state which specifies our prior density over the space of possible models, and b) the concept of hyperstates, introduced in Bellman & Kalaba (1959*b*), that refers to the system as an adaptive control process. Informally an hyperstate pairs a current model estimate (the information state) with the current state in the MDP (the physical state). A Hyper MDP is then created by allowing the transition probabilities between hyper states to be determined by the current model estimate at the hyper state. It has been shown that if the agent considers all the possible hyper states it could reach and computes the value function for all these hyper states, this value function produces an optimal exploration strategy. Using hyperstates as the model for exploration overcomes myopic behaviour.

3.3.1 Bayesian Estimator of expected return

In a Bayesian formulation of the estimation problem, we assume that there is a space $\mathbb{P}$ of all the possible transition functions (parametric models) for the MDP and that there exists a *belief state* over this space. The belief state defines a probability density $f(P|M)$ over the MDPs. The density is parameterised by $M \in \mathbb{M}$. In the Bayesian approach we treat the unknown parameter P as a random variable, with prior distribution $f(P|M)$ which represents what one knows about the parameter before observing transitions.

We choose a *prior* belief state which specifies our prior density over the space of possible models. At each step in the environment, we start at state s, choose an action a and then observe a new state s' and a reward r. We summarise our experience by a sequence of experience tuples $< s, a, r, s' >$ stored in D. When we observe transitions, we update the prior with the new experience. Given an experience tuple D we can compute the *posterior* belief state by Bayes rule:

$$\begin{aligned} f(P|M) &= \frac{f(D|P)f(P|M)}{f(D)} \\ &= \frac{1}{Z} f(D|P)f(P|M) \end{aligned} \tag{3.20}$$

in which Z is a normalising constant. Thus, the Bayesian approach starts with some prior probability distribution over all possible MDPs (we assume that the sets of possible states, actions, and rewards are delimited in advance). As we gain experience, the approach focuses the mass of the posterior distribution on those MDPs in which the observed experience tuples are most probable.

As mentioned in the previous section, we need to choose a distribution that is closed under updates and which can be indexed by a set of parameters M. The multinomial distribution is a natural choice for finite state and action spaces.

The Bayesian estimator of expected return under the optimal policy is the expectation of the value function:

$$V_s(M) = \int_{\mathbb{P}} V_s(P) f(P|M) dP \tag{3.21}$$

where $V_s(P)$ is the value of state s given the transition function P. $\mathbb{P}$ is the set of all the possible transition functions. We know from the central result of both Bellman and Martin that when this integral is evaluated we transform our problem into one of solving a MDP with known transition probabilities, defined on the information space $\mathbb{M} \times \mathbb{S}$:

$$V_s(M) = \max_a \ \{\sum_{s'} \bar{p}^a_{ss'}(M)(r^a_s + \gamma V_{s'}(T^a_{ss'}(M)))\} \tag{3.22}$$

in which, for convenience, the transformation on M due to a single observed transition $s \overset{a,r}{\leadsto} s'$ is denoted $T^a_{ss'}(M)$, $\bar{p}^a_{ss'}(M)$ is the marginal expectation of the Dirichlet, and r is the reward associated with the transition $s \overset{a,r}{\leadsto} s'$. The optimal solution is to act greedily with respect to the Bayes values (Bellman 1961).

This shows how the Bayesian estimate of value elegantly incorporates the value of future information. The optimal solution to the well-known exploration-exploitation trade-off is thus to act greedily with respect to the Bayes values. Because the solution involves dynamic programming over a tree of information states the problem is intractable. A simple approximation to this is the certainty equivalent (CE) estimate. We could also approximate the value of the integral by random sampling (Dearden 2000, Strens 2000, Strens 2003). As previously noted, the successive-approximation technique of dynamic programming can be used to solve Equation 3.22. However, as pointed out by Satia & Lave (1973), in this case, the state space includes the parameter space of the matrix-beta distribution and even for small size Markovian decision processes this approach has heavy computational requirements.

Other approaches to the optimal learning problem include: a) sampling from the infinite hypertree to produce a small, more manageable, tree (Wang, Lizotte, Bowling & Schuurmans 2005, Asmuth & Littman 2011) and using linear programming to compute the value of the hyperstates (Castro & Precup 2007), b) exploration control based on optimistic model selection (OMS) from a density over possible models (Wyatt 2001), and c) an improved characterization of the underlying decision problem to be solved, recognising that Bayesian RL can be cast as a partially observable Markov decision process (Duff 2002), and taking advantage of the fact that the optimal value function can be parameterized by a set of multivariate polynomials thereby allowing efficient offline approximate policy optimisation techniques to be derived (Poupart, Vlassis, Hoey & Regan 2006).

3.4 Chapter Summary

We described in this chapter the notion of uncertainty regarding the transition probabilities of Markov decision processes and explored the common descriptions for modelling transition uncertainty. We then focused on Bayesian methods. Bayesian inference derives the posterior probability as a consequence of two antecedents, a prior probability and a 'likelihood function' derived from a probability model for the data to be observed. Bayesian inference computes the posterior probability according to Bayes' rule. We highlighted the Bayesian approach to optimal learning in the framework of Markov decision processes and, in particular, described how total expected returns can be calculated using Bayesian techniques.

4

Exploration Control through Model Selection

Usually when making decisions in an unknown or partially known world there often exist a set of possible world models from which an agent may select one to work with. The selected model can be used as a vehicle for explaining and predicting observations, estimating value functions, controlling exploration and deciding how to behave. This chapter presents procedures for selecting models in Bayes adaptive Markov decision processes.

4.1 Introduction

Models provide a concrete vehicle for explaining the occurrence of observations. Often the observations could be seen to have been produced by a set of competing candidate models and the role of model selection is to identify the one model, from a set of candidate models that best captures the regularities underlying the observations. Model selection is fundamental to scientific enquiry, problem solving and decision making. An example of model selection is that of value function approximation in large scale reinforcement learning problems where a set of real world observations is used to select a function that predicts state action values. Whilst picking a model from amongst a set of competing models may seemingly appear straightforward, the problems with model selection typically include the following: a) quantifying what is meant by *'best'*, b) establishing the appropriate set of assumptions that are applicable to the selection process, and c) knowing how to search the model space for the *'best'* model. A good model selection method will typically balance goodness of fit with simplicity and be efficient in its search of the model space.

Exploration control in optimal learning can be achieved through model selection. In this chapter we shall describe a method for controlling exploration via model selection. The method is known as Optimistic Model Selection (Wyatt 2001), abbreviated as *OMS*. We shall review two established algorithms used by OMS to select optimistic models in Bayes adaptive MDPs. We shall then formulate OMS as a multi-objective program. The multi-objective program will allow us to a) use OMS in the presence of transition constraints, and b) exploit existing multi-objective programming solution techniques to efficiently search the model space for the *'best'* model.

The rest of the chapter is organised into five main sections. We introduce in Section 4.2 the general set up for model selection in optimal learning and in Section 4.3 we describe the OMS method. Our multi-objective programming approach to optimistic model selection is presented in Section 4.4 and we show in Section 4.5 worked examples of the OMS methods we describe. We conclude the chapter in Section 4.6 with a summary. The original contribution of this chapter is to show how to extend OMS to work with constraints on the model using multi-objective programming techniques.

4.2 The General Set up

Recall from Chapter three the Bayesian estimator of return under the optimal policy:

$$V_s(M) = \int_{\mathbb{P}} V_s(P) f(P|M) dP \tag{4.1}$$

where $V_s(P)$ is the value of state s given a transition function P. $\mathbb{P}$ is the set of all the possible transition functions, the Dirichlet distribution over the space of possible models is parameterised by M and $f(P|M)$ is the probability density for the possible transition model $P \in \mathbb{P}$ given M. Also recall from Chapter three the central result of both Bellman and Martin (Bellman 1961, Martin 1967) that when the integral 4.1 is evaluated the problem is transformed into one of solving an MDP with known transition probabilities, defined on the information space $\mathbb{M} \times \mathbb{S}$:

$$V_s(M) = \max_a \ \{\sum_{s'} \bar{p}^a_{ss'}(M)(r^a_{ss'} + \gamma V_{s'}(T^a_{ss'}(M)))\} \tag{4.2}$$

in which, for convenience, the transformation on M due to a single observed transition $s \overset{a,r}{\leadsto} s'$ is denoted $T^a_{ss'}(M)$, $\bar{p}^a_{ss'}(M)$ is the marginal expectation of the Dirichlet distribution, and $r^a_{ss'}$ the reward associated with the transition $s \overset{a,r}{\leadsto} s'$. The optimal solution is to act greedily with respect to the Bayes values defined in Equations 4.1 and 4.2. A way to evaluate equation 4.2 in unknown or partially known environments is to default the transition model P to models that we select from M. Our focus in this chapter and the rest of the thesis is on a model selection approach, with the following general set-up.

- There is a set of candidate models $P_1, P_2, \ldots, P_K$, where K is size of the model space. If K is reasonably small an agent may choose to compare models to one another in a sequential experiment. Unfortunately the set is usually infinite, i.e. $K \to \infty$.

- There is a prior distribution for the model and it is Dirichlet with known parameters M. The probability density for a given model $P \in \mathbb{P}$ given the Dirichlet, i.e. $f(P|M)$ can be computed using the methods we described in Chapter three.

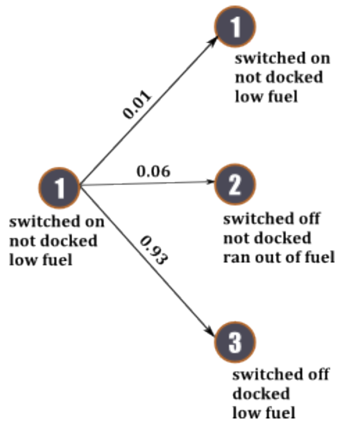
- Following a new observation regarding transition from a current state s to a new state s' under an action a, the prior distribution can be revised into posterior distribution M using Bayes rule.

- The successor states $succ^a_s$ to the current state s under action a will become fully established as learning progresses.

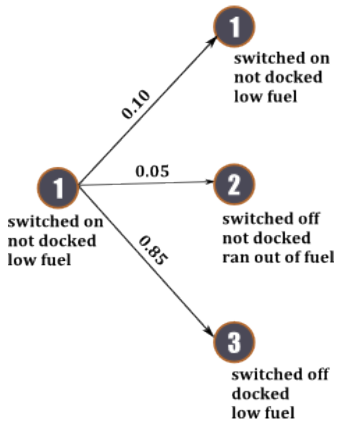
- The probability interval for each of the transition probabilities of model P is established using the algorithms we described in Chapter three for calculating credible intervals.

- The back-up values of successor states can be calculated using the value iteration methods we described in Chapter two.
- One of the candidate models $P_1, P_2 \dots P_K$. will be the best model given the estimated state values and the credible intervals

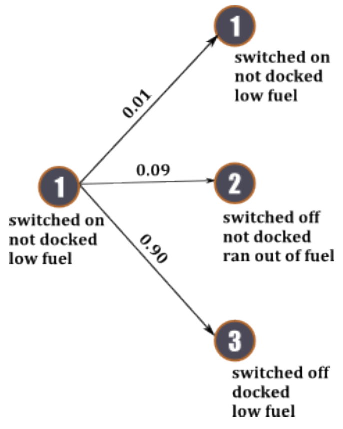
Let us revisit the hospital robot task we mentioned earlier in Chapter one and consider transitions



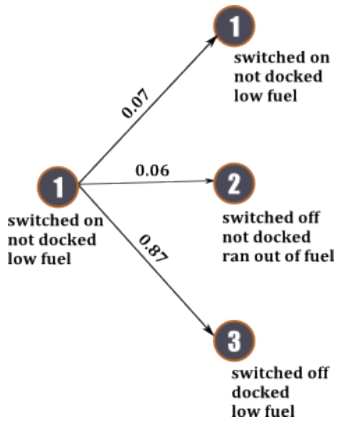
(a) $\vec{p}_1^{\,\text{dock}} = (0.01, 0.06, 0.93)$



(b) $\vec{p}_1^{\,\text{dock}} = (0.10, 0.05, 0.85)$



(c) $\vec{p}_1^{\,\text{dock}} = (0.01, 0.09, 0.90)$



(d) $\vec{p}_1^{\,\text{dock}} = (0.07, 0.06, 0.87)$

Figure 4.1: Examples of transition models in the Wellbeing task. The models will appeal differently to an optimistic learning agent that uses a principle of maximum expected utility with biases in its probabilistic estimates of transitions.

under a dock action (Figure 4.1). Let the current state be 1, i.e. the robot is switched on, not docked and has low fuel and let successor states to the current state be states 1, 2, and 3. We assume the corresponding value functions to be 120, 50, and 20 under $1 \overset{dock}{\rightsquigarrow} 1$, $1 \overset{dock}{\rightsquigarrow} 2$, and $1 \overset{dock}{\rightsquigarrow} 3$ respectively inducing the ordering shown in example models of Figure 4.1. In addition the probability intervals for each of the successor states are assumed to be $[0.01, 0.3]$, $[0.05, 0.4]$, and $[0.85, 0.93]$ for p_{11}^{dock}, p_{12}^{dock}, and p_{13}^{dock} respectively. The candidate models in the model space described by these probability intervals include those shown in Figure 4.1 and, as we shall show in the subsequent sections of this chapter, an optimistic model is contained in this set of the candidate models. The example models shown in Figure 4.1 will appeal differently to an optimistic learning agent that uses a principle of maximum expected utility with biases in its probabilistic estimates of transitions. In next section, we shall describe the OMS method as a specific instance of the general set-up we itemised above.

4.3 Optimistic Model Selection (OMS)

The OMS method (Wyatt 2001) integrates ideas from a popular family of approximate approaches to the exploration-exploitation trade off which typically use some instantiation of the heuristic 'be optimistic in the face of uncertainty' (Kaelbling et al. 1996, Wiering & Schmidhuber 1998, Meuleau & Bourgine 1999). OMS integrates the instantiation idea with the Bayesian view of exploration by selecting an optimistic model P_{opt} from $\mathbb{P}$ using probability intervals calculated based on $f(P|M)$. The OMS method has the following two components:

Augmented MDP: Similar to R-max (Brafman & Tennenholtz 2002), OMS uses an augmented state space. It adds a hypothetical state k to the underlying MDP resulting in an augmented state space $k \cup S$. The hypothetical state is sometimes referred to as the *'Garden of Eden'* (Szita & Lőrincz 2008). k is an absorbing state and once visited the agent remains in the state indefinitely with reward R_{max} for each time step taken at the garden of Eden. State k represents possible unobserved transitions (Kearns & Singh 2002) and by making the state highly rewarding it also induces a distal exploration value function (Wyatt 1997) that will drive the learner toward novel state action pairs.

Exploration value function: The value function for the augmented MDP can differ from the true underlying state values that we described in Chapter two. The value function for the augmented MDP is called the *exploration value function* and the relevant Bellman equation is:

$$\xi_s^a(M) = \sum_{s'} p^a_{opt,ss'}(M)(r^a_{ss'} + \gamma \max_{a'}\{\xi_{s'}^{a'}(M)\}) \tag{4.3}$$

where $p^a_{opt,ss'}(M)$ are the transition probabilities according to P_{opt}. The agent acts greedily with respect to $\xi_s^a(M)$ by selecting the action with the highest optimistic value.

4.3.1 The Main Loop of OMS

The main loop of the OMS method is shown in Algorithm 4.1. The procedure takes as input a known specification of an MDP i.e. the states S, actions A and reward function R of the MDP. Also passed as input to the procedure are the prior distribution M, the discount rate γ for infinite horizon tasks, α that signifies the level of confidence attached to probability intervals, and the initial state values V. The procedure returns an optimistic model P^a_{opt}.

Lines 2-5 are the initialisation steps. The true value function for the hypothetical state V_k is initialised in line 2 and the prior information for the hypothetical state is initialised in line 3. The exploration values are initialised in lines 4 and 5. The underlying MDP is augmented with a hypothetical state k during the initialisation steps.

Lines 6-18 contains the main learning steps. The agent observes the current state x in line 7 and selects an action a to perform given the current state and the current exploration values. In line 8, the agent selects an action a with the highest optimistic value, breaking ties randomly. In line 9 the agent executes the action selected and observes the transition to the next state i.e. $x \overset{a,r}{\rightsquigarrow} y$. The observed transition is used to update beliefs in line 10. The updated belief is the posterior distribution defined by M. The agent runs its asynchronous real time dynamic programming (ARTDP) algorithm in lines 11-17. Since the value function may change as the agent perform asynchronous back-ups it would be necessary to perform model selection every time a state action pair is backed up. The optimistic

```
1: procedure $P^a_{opt} = OMS(S, R, A, V, M, \gamma, \alpha)$
2:     Initialise $V_k \;\; \forall i, j \in \mathbb{S} \cup k$ and $\forall a \in \mathbb{A}$
3:     Initialise $m^a_{ik}, m^a_{kj} \;\; \forall i, j \in \mathbb{S} \cup k$ according to $M$
4:     $\xi^a_i = \gamma V_k, \;\; \forall i \in \mathbb{S}$ and $\forall a \in \mathbb{A}$ where $m^a_{ik} > 0$
5:     $\xi^a_k = V_k, \;\; \forall a \in \mathbb{A}$
6:     loop
7:         observe current state $x$
8:         select action $a = \arg\max_b \{\xi^b_x\}$ breaking ties randomly
9:         execute action $a$ in state $x$ and observe the transition $x \overset{a,r}{\rightsquigarrow} y$
10:        update belief $m^a_{xy} = m^a_{xy} + 1$
11:        repeat
12:            choose i
13:            for each action b do
14:                find $\vec{p}^{\,b}_{opt,i}$ using Algorithm 4.2 or 4.3
15:                update $\xi^b_i$ using Equation 4.3
16:            end for
17:        until ARTDP algorithm stops
18:    end loop
19: end procedure
```

Algorithm 4.1: The Main Loop of the Optimistic Model Selection (OMS) Algorithm. Adapted from Wyatt (2001).
Algorithms 4.2 and 4.3 are shown in pages 80 and 81 respectively.

model is selected in line 14 and the exploration value function is revised in line 15 to reflect the newly selected model. The optimistic estimate can be calculated using any form of ARTDP.

To complete our description of the OMS algorithm we need to explain how exactly an optimistic model is selected. Algorithm 4.1 cited two algorithms 4.2 and 4.3 for selecting an optimistic model. We shall describe the two algorithms in the next section.

4.3.2 Algorithms for Selecting Optimistic Models

Wyatt (2001) suggests two ways of selecting P_{opt}, which were termed *'simple'* and *'full'* OMS. In simple OMS the agent is optimistic only about hypothesised transition to the garden of Eden state. In full OMS the agent can be optimistic about transitions to other states too.

Simple OMS

In simple OMS (Algorithm 4.2), when an agent receives in state i a new observation, it re-calculates the upper bound of the $(1-\alpha)$ probability interval for the transition probability to the state k for each state action pair. The marginal density required for this computation is simply a Beta density, always following Beta$(m^a_{ik}, \sum_{j\neq k} m^a_{ij})$, and can be calculated using the algorithms we describe in Chapter three for computing credible intervals. The other probabilities are then renormalized to give an optimistic one step transition model from state action pair i, a. Applied to all states the result is an optimistic MDP P^a_{opt}, under action $a \in A$.

```
1: procedure $\vec{p}^a_{opt,i} = OMS - Simple(i, a, k, \vec{m}^a_i, \alpha)$
2:     for $i, a$ construct $\vec{p}^a_{opt,i}$:
3:         $p^a_{opt,ik} =$ upperbound(Beta,$m^a_{ik}, \sum_{j\neq k} m^a_{ij}, \alpha)$
4:         $p^a_{opt,ij} = \frac{1-p^a_{opt,ik}}{1-\bar{p}^a_{ik}} \bar{p}^a_{ij}, \quad \forall j \neq k$
               where $\bar{p}^a_{ij} = \frac{m^a_{ij}}{\sum_{x\in\mathbb{S}\cup k} m^a_{ix}}$
5: end procedure
```

Algorithm 4.2: Simple Optimistic Model Selection.
Adapted from Wyatt (2001).

Simple OMS can be seen as a relation of Kearns and Singh's E^3 algorithm (Kearns & Singh 2002) in which the learner chooses either to identify the model by taking actions that drive it toward the unknown state set, or to exploit within the set of known states. In the $OMS - Simple$ algorithm as soon as a state action pair is tried it is considered known, and can be used in exploitation if it is appealing enough.

Full OMS

In full OMS (Algorithm 4.3), an agent can be optimistic about the transition probabilities for any of the successors of state i under action a. OMS employs the ideas of bounded parameter MDPs (Givan, Leach & Dean 2000). However, instead of performing interval value iteration, full OMS computes only the optimistic value function. Given state action pair i, a we order the successor states by the current estimate of the value function, in descending order (Line 4) and then calculate the lower and upper bounds of the $(1-\alpha)$ probability interval for each transition (Lines 5 and 6) using techniques we described in Chapter three. An optimistic transition function is then constructed by sending as much probability mass as possible to the states early in the ordering, while keeping all probabilities within their lower and upper bounds.

```
1:  procedure $\vec{p}^{a}_{opt,i} = OMS - Full(i, a, k, \vec{m}^{a}_{i}, \alpha)$
2:      for $i, a$ construct the model $\vec{p}^{a}_{opt,i}$:
3:      obtain the $u$ successor states $succ^{a}_{i}$ from $S \cup k$.
4:      order the $u$ states in $succ^{a}_{i}$ to give
        $(j_1, j_2, \ldots, j_u)$ such that $V_{j_1} \geq V_{j_2} \geq \ldots \geq V_{j_u}$.
        where $V_j = \max_b \{\xi^{b}_{j}\}$
5:      $p^{a}_{\uparrow,ix}$ =upperbound(Beta,$m^{a}_{ix}, \sum_{y \neq x} m^{a}_{iy}, \alpha$),  $\forall x$
6:      $p^{a}_{\downarrow,ix}$ =lowerbound(Beta,$m^{a}_{ix}, \sum_{y \neq x} m^{a}_{iy}, \alpha$),  $\forall x$
7:      set $s$ to be as large as possible while
        $p^{a}_{\downarrow,ij_s} \leq 1 - (\sum_{p<s} p^{a}_{\uparrow,ij_p} + \sum_{q>s} p^{a}_{\downarrow,ij_q}) \leq p^{a}_{\uparrow,ij_s}$
8:      set $p^{a}_{opt,ij_p} = p^{a}_{\uparrow,ij_p}$,  $\forall p < s$
9:      set $p^{a}_{opt,ij_q} = p^{a}_{\downarrow,ij_q}$,  $\forall q > s$
10:     set $p^{a}_{opt,ij_s} = 1 - \sum_{j \neq j_s} p^{a}_{opt,ij}$
11: end procedure
```

Algorithm 4.3: Full Optimistic Model Selection.
Adapted from Wyatt (2001).

Full OMS can be seen as an extension of Wiering and Schmidhuber's method (Wiering & Schmidhuber 1998) which uses a more appealing density to represent uncertainty about the model; utilises this density in exploration control from the outset; and takes account of all successors in calculating the optimistic model.

4.4 Multi-objective Programming Approach

In this section we shall describe how OMS can be extended to work with constraints on models using multi-objective programming. Whilst the OMS algorithms we described in Section 4.3 satisfy the probability constraints on transition models, that is, $0 \leq p^a_{ss'} \leq 1$; $\sum_{s'=1}^{|S|} p^a_{ss'} = 1 \; \forall \; s = 1, 2, \ldots, |S|$ it does not handle other possible constraints such as absolute and/or relative restrictions on transition probabilities.

In contrast to optimisation involving a single objective, multi-objective programming involves recognition that an agent is responding to multiple objectives. Generally, objectives are conflicting, so that not all objectives can simultaneously arrive at their optimal levels. Problems involving multiple objectives can be solved using linear programming, where one of the objectives, the most important, is optimised and the others are considered in the restrictions. This procedure generates some disadvantages such as

- Representing the goals by means of restrictions of linear programming generally leads to intractable problems. In large problems it is difficult to find the restriction that causes the intractability.

- The choice of which objective should be optimised is sometimes difficult or subjective.

Multi-objective programming presents a way of solving these problems, where the optimum solution of the problem of linear programming is substituted by a set of solutions, not necessarily optimum in the sense of the linear programming, but efficient solutions. A multi-objective problem can be represented as follows:

$$max\{z_1(x), z_2(x), \ldots, z_n(x)\}, x \in \aleph \tag{4.4}$$

where x are the decision variables, $\aleph$ is the set of possible alternatives and $\{z_1, z_2, \ldots, z_n\}$ are the finite set of objectives. The obtained solution can be dominated or non-dominated. A solution is non-dominated when there is no other feasible solution that improves one of the objectives without decreasing at least another objective. Several methods exist to generate a set of non-dominated solutions. An assumed utility function is used to choose appropriate solutions. Several fundamentally different utility function forms have been used in multi-objective models. These may be divided into three classes: lexicographic, multi-attribute utility and unknown utility (Chankong & Haimes 1983). The method that will be used in this work is lexicographic goal programming and will be described next.

4.4.1 Goal Programming (GP)

The lexicographic approach (Charnes, Cooper & Ferguson 1955, Lee 1972, Ignizio 1976, Charnes & Cooper 1977) assumes the decision maker has a strictly ordered pre-emptive preference system among objectives with fixed target levels. For example, a lexicographic system could have its first priority goal as income of not less than £$10,000$; the second priority as leisure of no less than 20 hours a week; the third as costs of no more than £$6,500$, etc. This formulation is typical of goal programs (Lee 1972).

The various goals are dealt with in strict sequential order - higher goals before lower order goals. Once a goal has been dealt with (meeting or failing to meet the target level), its satisfaction remains fixed and the next lower order goal is considered. Consideration of the lower level goals does not alter the satisfaction of higher level goals and cannot damage the higher level goals with respect to target level attainment. The characteristic that distinguishes the formulation of goal programming is that one or more goals are directly incorporated in the function objective, through deviation variables, that is, the objectives are written in the form of goals restrictions, where each goal represents the value that intends to be reached. The central construct in goal programming is the deviation variable. The goals cannot be reached completely and, to allow this flexibility, deviation variables are used d^+ and d^-, indicating how much the objective was surpassed or was lacked by that value respectively. Goal programming expresses a form of reaching the goals as closest as possible; the objective of this technique is to minimize the sum of the deviations for all the goals.

The general model of goal programming can be written as follows:

$$\begin{aligned} \min \quad & z = \mathbf{w}^+\mathbf{d}^+ + \mathbf{w}^-\mathbf{d}^- \\ \text{s.t.} \quad & C_G X - \mathbf{d}^+ + \mathbf{d}^- = G \\ & C_B X \overset{\geq}{\underset{\leq}{=}} B. \\ & x_k \geq 0,\ \ d_k^+ \geq 0,\ \ d_k^- \geq 0,\ \ x_k \in X \ \ \forall k \in K. \end{aligned} \tag{4.5}$$

where:

z	objective function
$\mathbf{w}^+, \mathbf{w}^-$	are the vector of weights associated with the positive and negative deviations of the goals.
C_G	is an $(m \times n)$ matrix of the decision variable coefficients associated with goal constraints.
C_B	is an $(m \times n)$ matrix of the decision variable coefficients associated with other constraints.
G	is an $(m \times 1)$ vector that represents the goals that are to be reached.
B	is an $(m \times 1)$ vector that represents bounds on the other constraints.
$\mathbf{d}^+, \mathbf{d}^-$	are $(m \times 1)$ vectors that represent the positive and negative deviations of the m goals.
X	is an $(m \times 1)$ vector of decision variables.

There are two main approaches to handling the objective function in goal programming: Non-pre-emptive and Pre-emptive (Lee 1972, Ignizio 1976).

Non-pre-emptive Goal Programming: In this approach, we put all the goals in the objective function and solve the linear program a single time. The objective for the problem is to minimise the weighted sum of the deviation variables. The penalty measures the relative importance of the goals. Because the goals are very often measured on different scales, the penalties play the double role of transforming all the goals to the same dimensional units as well as specifying their relative importance. In this approach the subjective step is the determination of the weights. Different weights will often yield very different solutions.

Pre-emptive Goal Programming: here the goals are divided into sets and each set is given a priority: i.e. first, second, and so on. The assumption is that a higher goal is absolutely more important than a lower priority goal. The solution is obtained by initially optimising, with respect to the first priority goals without regard to the values of lower priority objectives. Then, holding constant the value of the first priority objective function by adding the constraint $z_1(d_1^+, d_1^-) = z_1^*$, the optimal solution is obtained for the second-priority goals. The feasible solution space for this second problem is the set of alternative optima for the first problem. The process continues until all priorities are considered. If no alternative optima exist at the end of a particular stage, we have reached the end of the computations so we must be satisfied with the current values of the lower priority objectives. If several goals have about the same priority we include them in the set in the objective at the appropriate step of the process. The relative importance of the goals within any set are reflected by the specification of the penalty weights, as in the non-pre-emptive case. The subjective part of this procedure is the division of the goals into priority sets and the selection of penalties within a priority set.

4.4.2 Pre-emptive GP Approach to OMS

To formulate optimistic model selection as a pre-emptive goal program we let $\vec{p}^{\,a}_{opt,s}$ be the decision variables of interest and goals are the ordered successor states to state s.

$$\begin{aligned}
\min \quad & z = \mathbf{w}^+\mathbf{d}^+ + \mathbf{w}^-\mathbf{d}^- \\
\text{s.t.} \quad & p^a_{opt,ss'} - d^+_{s'} + d^-_{s'} = p^a_{\uparrow,ss'}, \ \ \forall s' \in S \\
& p^a_{opt,ss'} \geq p^a_{\downarrow,ss'}, \ \ \forall s' \in S \\
& \textstyle\sum_{s'} p^a_{opt,ss'} = 1. \\
& p^a_{opt,ss'} \geq 0, \ \ d^+_{s'} \geq 0, \ \ d^-_{s'} \geq 0 \ \ \forall s' \in S.
\end{aligned} \tag{4.6}$$

where:

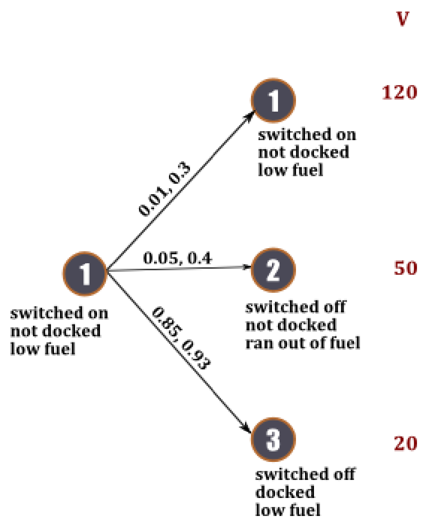
z	objective function
$\mathbf{w}^+, \mathbf{w}^-$	are the vector of weights associated with the positive and negative deviations of the goals.
$p^a_{opt,ss'}$	is a decision variable representing transition probability from state s to state s' under action a.
$p^a_{\downarrow,ss'}, p^a_{\uparrow,ss'}$	are the lower and upper bounds on $p^a_{ss'}$ respectively.
$\mathbf{d}^+, \mathbf{d}^-$	represent the positive and negative deviations of the goals.

The goals are divided into sets according to decreasing order of the state values and each set is given a priority. Once the goal program of Equation 4.6 is solved, the resulting $\vec{p}^{\,a}_{opt,s}$ can then be used in the main loop of the OMS, line 14 of Algorithm 4.1 in place of Algorithms 4.2 and 4.3.

4.5 Worked Examples

In this section we use two simple Examples 4.1 and 4.2 to illustrate how to select optimistic models using the full OMS Algorithm 4.3. We also show that optimistic models can be selected for the same set of examples using the pre-emptive GP formulation of Equation 4.6.

Example 4.1 *We return to the hospital robot task we presented in Section 4.2 and select an optimistic model from the model space, illustrated in Figure 4.2.*



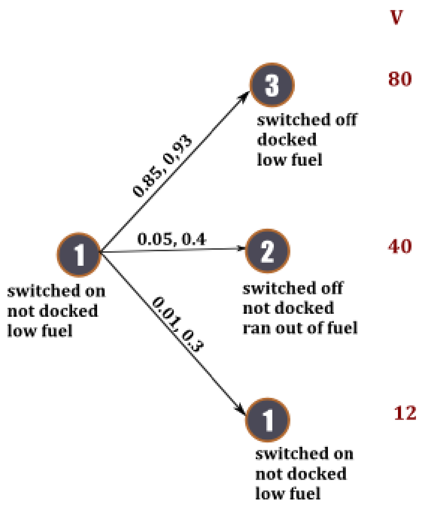
Transition constraints:

$$0.0 \leq p^{dock}_{11} \leq 1.0$$
$$0.0 \leq p^{dock}_{12} \leq 1.0$$
$$0.0 \leq p^{dock}_{13} \leq 1.0$$
$$p^{dock}_{11} + p^{dock}_{12} + p^{dock}_{13} = 1.0$$

Figure 4.2: An example of Model Selection

To pick an optimistic model $\vec{p}^{dock}_{opt,1}$, full OMS sends as much probability mass as possible to $p^{dock}_{opt,1,1}$ while keeping $p^{dock}_{opt,1,2}$ and $p^{dock}_{opt,1,3}$ to their lower bounds. That is $p^{dock}_{opt,11}$ is set to $\min(0.3, 1-0.05-0.85) = 0.1$. *In line 7 of the full OMS Algorithm 4.3, s is computed to be 1 hence $p^{dock}_{opt,12}$ and $p^{dock}_{opt,13}$ are kept to their lower bounds i.e. 0.05 and 0.85 respectively. The $\vec{p}^{dock}_{opt,1}$ selected by the full OMS Algorithm 4.3 is therefore* $[0.1, 0.05, 0.85]$ *for* $[p^{dock}_{opt,11}, p^{dock}_{opt,12}, p^{dock}_{opt,13}]$. *We also used goal programming Equation 4.6 to select an optimistic model $\vec{p}^{dock}_{opt,1}$ for this process. The model selected by the goal programming approach is the same as that selected by the full OMS algorithm.*

Example 4.2 *Assume the value functions in Example 4.1 are changed to* $[12, 40, 80]$ *for* $[\xi^{dock}_{11}, \xi^{dock}_{12}, \xi^{dock}_{13}]$ *while keeping the probability intervals the same as shown in Figure 4.3. Observe that compared to*



Transition constraints:

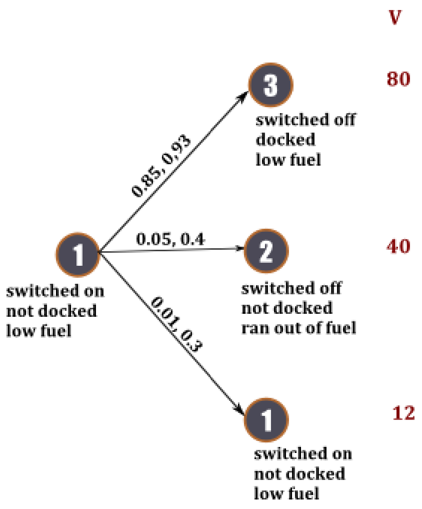
$$0.0 \leq p^{dock}_{11} \leq 1.0$$
$$0.0 \leq p^{dock}_{12} \leq 1.0$$
$$0.0 \leq p^{dock}_{13} \leq 1.0$$
$$p^{dock}_{11} + p^{dock}_{12} + p^{dock}_{13} = 1.0$$

Figure 4.3: An example of Model Selection

Example 4.1, the state ordering has now changed because of the changes we have made to the exploration value functions. As a result, the full OMS algorithm will now push $\min(0.93, 1-0.01-0.05)$ $= 0.93 > 0.85$ *to* $p^{dock}_{opt,13}$, $\min(0.4, 1-0.93-0.01) = 0.06 > 0.05$ *to* $p^{dock}_{opt,12}$ *and keep* $p^{dock}_{opt,11}$ *to its lower bound which is* 0.01. *Hence the optimistic model selected in this case by Full OMS is* $[0.01, 0.06, 0.93]$ *corresponding to* $[p^{dock}_{opt,11}, p^{dock}_{opt,12}, p^{dock}_{opt,13}]$ *which, as expected, differs from model we picked in Example 4.1. We also used the goal program of Equation 4.6 to select an optimistic model $\vec{p}^{dock}_{opt,1}$ for this process. The model selected by the goal programming approach is the same as that selected by the full OMS algorithm.*

Finally, to illustrate the appeal of the goal programming approach, we applied it to an extension of Example 4.2 incorporating absolute and relative constraint on transitions.

Example 4.3 *Staying with Example 4.2, let's assume we now have an additional constraint in the form of an order restriction on the transition probabilities which states that* $1.0 \geq p_{13}^{dock} \geq p_{11}^{dock} \geq p_{12}^{dock} \geq 0.0$*, Figure 4.4. Due to the order constraint, the OMS Algorithms 4.2 & 4.3 cannot be applied*



Transition constraints:

$$1.0 \geq p_{13}^{dock} \geq p_{11}^{dock} \geq p_{12}^{dock} \geq 0.0$$
$$p_{11}^{dock} + p_{12}^{dock} + p_{13}^{dock} = 1.0$$

Figure 4.4: An example of Model Selection with Transition Constraints Illustrating the GP approach

to this problem. However, by including the order restriction as an additional constraint in the goal program we solved in Example 4.2 we are able to select the following model. The model selected is $[0.05, 0.05, 0.9]$ *corresponding to* $[p_{opt,11}^{dock}, p_{opt,12}^{dock}, p_{opt,13}^{dock}]$ *respectively. The selected model differs from that we obtained in Example 4.2, the goal programming solution we obtained in this example reflects the order restriction placed on the transition probabilities whilst attempting to satisfy the goals we set out to achieve in terms of the ordering of successive states in terms of exploration values.*

4.6 Chapter Summary

Our focus in this chapter has been on model selection in Bayes Adaptive Markov decision processes (BAMDP). The role of model selection is to identify the one model, from a set of competing models that best captures the regularities underlying the process of interest. We described the optimistic model selection (OMS) approach to exploration control in BAMDPs with two algorithms for selecting optimistic models termed *'simple'* and *'full'* OMS. In simple OMS the agent is optimistic only about hypothesised transition to a *'garden of Eden'* state. In full OMS the agent can be optimistic about transitions to other states too. We also developed a multi-objective programming approach to model selection in BAMDPs to extend OMS to work with additional constraints on the models such as orderings on the transition probabilities. Specifically, we presented a pre-emptive goal programming approach to OMS and showed that it gives the same result as those of the full OMS Algorithm in two of our examples. In the third example we imposed order restrictions on state transition probabilities and noted that whilst the goal programming approach was able to select an optimistic model given the order restriction the simple and full OMS Algorithms are not equipped to handle such constraints. In such situations, where constraints are imposed on transitions, the goal programming approach is the preferred choice.

5

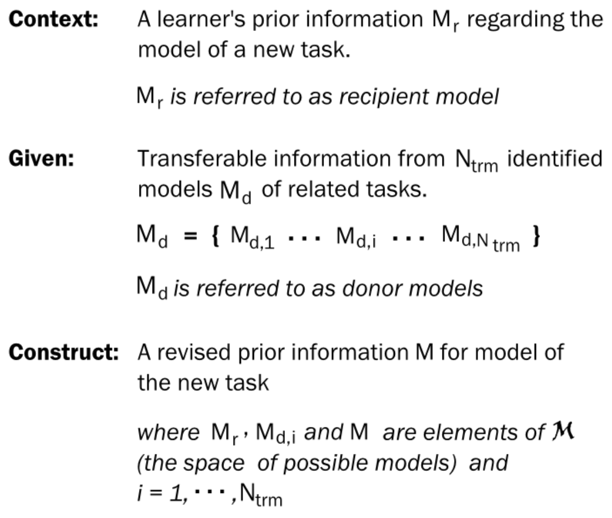
Approaches to Prior Information Transfer

In this chapter we will review techniques for transferring prior information between models. We will focus primarily on model transfer and the associated probability expressions.

5.1 Introduction

Consider an unknown model P informed by n diverse data set $D_1, \ldots, D_n$. These data have been evaluated for their individual information content related to the unknown model through the elementary probabilities $f(P|D_i)$. The challenge is then to recombine the prior probability $f(P)$ and the n single-event conditional probabilities $f(P|D_i)$ into the posterior probability $f(P|D_1, \ldots, D_n)$ while accounting for the interaction among data. From a probabilistic point of view, the general problem is one of meaningfully summarising into a posterior probability distribution the prior and pre-posterior probabilities from the diverse sources of information (Genest & Zidek 1986). The primary goal is to build a new composite probabilistic model by objectively aggregating all available information while respecting the different nature and uncertainty levels present in the information sources (Xu & Golay 2006).

Model transfer is the practice of taking a model identified for one process and, *after some adjustment*, reusing it to predict a different process. The idea behind this concept assumes that we can take advantage of relationships that are common to both processes. Model transfer does not typically include adaptive methods where the transfer from process to process occurs gradually. A typical model transfer problem can be characterised as shown in Figure 5.1. This contrasts with a) classical machine learning problem in which the source (donor) tasks are absent, and b) multi-task learning in which the output labels are also allowed to change the problem. Research in transfer



Context: A learner's prior information M_r regarding the model of a new task.

M_r is referred to as recipient model

Given: Transferable information from N_{trm} identified models M_d of related tasks.

$M_d = \{ M_{d,1} \cdots M_{d,i} \cdots M_{d,N_{trm}} \}$

M_d is referred to as donor models

Construct: A revised prior information M for model of the new task

where M_r, $M_{d,i}$ and M are elements of $\mathcal{M}$ (the space of possible models) and $i = 1, \cdots, N_{trm}$

Figure 5.1: A typical model transfer problem

learning (Taylor & Stone 2009, Pan & Yang 2010) promises mechanisms that let systems improve with experience particularly when they are across domains. Whilst cross-domain transfer is hugely important, a fundamental issue in model transfer is whether adjustments will be needed to donor models before transfer and, if yes what sort of adjustments are required and how. The resolution of this issue remains the main driver of research in model transfer.

5.2 Overview of Transfer Methods

This section provides an overview of the main methods of model transfer reported in the literature.

5.2.1 Naive Transfer

In naive transfers the model specification and parameter estimates from donor tasks are applied directly (without any change) to a new recipient task. This is the simplest method of transferring prior information. It requires minimum effort and does not require adjustments to the donor information. The assumption here is that all factors relevant to the recipient are captured by the donor. Such assumptions may be difficult to justify in practical applications. In the next chapter (Chapter six) we shall present a naive method that transfers donor information from related tasks to a new task through a simple addition operator and without any change to the donor information. There are alternatives to the naive approach which include scaling, discounting and quality adjustments of the donor information which we discuss in the next section.

5.2.2 Transfer Scaling, Discounting & Quality Adjustments

In transfer scaling it is assumed that the parameters of the donor models are transferrable to the recipient up to a certain 'scale'. The scale is an indicator of transfer bias (the difference in the true parameters between the two contexts), it is a representation of the unobserved 'hidden' factors in the transfer contexts.

A number of authors have described techniques for using 'imaginary' or 'fictitious' data to modify a pre-prior distribution (Karny 1984, Neal 2001, Tesař 1996). Karny (1984) focused on deriving quantitative expressions of prior information contained both in prior data and individual pieces of expert information. They expressed the individual pieces of expert information in a common form called 'fictitious' data. Neal (2001) showed how a prior distribution formulated for a simpler, more easily understood, model can be used to modify the prior distribution of a more complex model. He used imaginary data drawn from the simpler 'donor' model to condition the more complex 'recipient' model. The approach of Tesař (1996) centres on minimizing Kullback-Leibler distance between empirical and model distributions

A common problem in information transfer is that of making inference about a parameter that would govern an ideal (or *paradigm*) study for a particular purpose – one for the population of interest to the investigator, without misclassification or other weaknesses, on the basis of non–ideal studies whose conditions do vary from the ideal in some important ways (Zellner 1988, Zellner 1997, Zellner 2002, Wolpert & Mengersen 2004). There could be significant variation in the quality of a) the inputs from the donor, b) the prior information of the recipient and c) the sample information. Zellner (2002) suggested *'quality-adjustment'* methods that adjust the prior and the likelihood of data such that the posterior distribution f' is proportional to a function of the quality adjustments, i.e. $f' \propto Adj_1(f(\theta))Adj_2(l(\theta|D))$, where $Adj_1(f(\theta))$ is the quality-adjusted prior and $Adj_2(l(\theta|D))$ is the quality-adjusted likelihood function. It is also appropriate to use quality-adjusted posteriors when the prior and sample information are to be weighed differently (Zellner 2002).

Wolpert & Mengersen (2004) described a parametric adjustment approach. Assuming the ith study does offer direct evidence about a parameter θ^i through a likelihood $L_i(\theta^i)$, and if each θ^i (including θ^0) is related to a hyperparameter θ through a known functional relationship $\theta^i = \phi_i(\theta)$, then we can 'adjust' the evidence from the ith study to bear directly on θ (and hence on θ^0) through the relationship:

$$L_i^{\mathrm{Adj}}(\theta) = L_i(\phi_i(\theta)), \tag{5.1}$$

in which the function $\phi_i(\theta)$ represents the value of θ_i when the hyperparameter value is θ.

The *parametric adjustment model* can be generalised to cases when the adjustment function $\theta_i = \phi(\theta, \alpha_i)$ depends explicitly on a parameter α_i, resulting in (Wolpert & Mengersen 2004):

$$L_i^{\mathrm{Adj}}(\theta) = L_i(\phi_i(\theta, \alpha_i)), \tag{5.2}$$

Wolpert & Mengersen (2004) noted that if the parameter α_i in (5.2) is regarded as uncertain and therefore random with a prior probability distribution $f_i^{\alpha}(d\alpha_i|\theta)$ then a conditional distribution for θ^i given θ can be calculated by averaging over the possible values of α_i,

$$f_i(d\theta^i|\theta) = \int \delta(\theta^i - \phi_i(\theta, \alpha_i)) f_i^{\alpha}(d\alpha_i|\theta), \tag{5.3}$$

and an adjusted likelihood function:

$$\begin{aligned} L_i^{Adj}(\theta) &= \int L_i(\theta^i) f_i(d\theta^i|\theta), \\ &= \int L_i(\phi(\theta,\alpha_i)) f_i^{\alpha}(d\alpha_i|\theta). \end{aligned} \tag{5.4}$$

Akin to the parametric adjustment model, the Confidence profile method (CPM) in both its classical and its Bayesian forms (Eddy 1989, Eddy, Hasselblad & Shachter 1990*a*, Eddy, Hasselblad & Shachter 1990*b*) was intended to apply to conventional fixed and random-effects meta-analysis, as well as to multi-parameter evidence synthesis. Adjustment for bias was strongly emphasized, and the CPM literature sets out formulae for adjusting basic or functional parameters. The confidence profile method (CPM) can correct for this by defining a function that relates a donor parameter θ^{ω} to a recipient parameter θ. Call this function $\vartheta(\theta^{\omega})$. This function can be substituted for θ^{ω} in the likelihood function restoring the correctness of Bayes' formula.

$$f(\theta|D) \propto L(\vartheta(\theta^{\omega})|D) f(\theta) \tag{5.5}$$

This last formula illustrates the three basic ingredients of the Confidence Profile Method. The method requires prior distributions, likelihood functions, and functions that describe biases. Although the purpose of bias adjustment is clearly to recover the 'true' underlying effects from messy data, it is also viewed as a method for dealing with heterogeneity in trial results.

Another important way of transferring information, especially those arising from historical data, is by using power priors. A power prior discounts the historical data by raising the likelihood of the historical data to a fractional power. Let the historical data set be D and $L(\theta|D)$ the likelihood of θ based on the historical data. The power prior is expressed as $L(\theta|D)^{\delta} f(\theta)$ (Ibrahim & Chen 2000), in which $f(\theta)$ is the prior distribution about θ that is specified by the agent before any historical data is made available and $0 \leq \delta \leq 1$ is a scalar precision parameter that weights the historical data relative to the likelihood of the current task. Under power prior rules, the posterior probability of θ

given the historical data is expressed for a fixed δ as follows:

$$f(\theta|D,\delta) = \frac{L(\theta|D)^{\delta} f(\theta)}{\int_{\Theta} L(\theta|D)^{\delta} f(\theta) d\theta} \tag{5.6}$$

Ibrahim, Chen & Sinha (2003) show that the power priors are optimal in the sense that the distribution in Equation 5.6 minimizes a convex sum of Kullback-Leibler divergence between the following two posterior densities based on a) 'pooled historical and current data' $(\delta = 1)$ and b) 'not using the historical data at all' $(\delta = 0)$. Bhattacharya (2009) showed that both quality-adjusted and power prior rules can be derived as special cases of a unified procedure. They also showed that a well-justified value of the precision parameter δ in power priors can be obtained by considering constraints relating divergence from a specified distribution given a single historical dataset.

5.2.3 Bayesian Melding

Bayesian melding was proposed by Raftery, Givens & Zeh (1995) and Poole & Raftery (2000) as a technique for combining or melding information from three sources in order to arrive at Bayesian posterior distributions. The three sources of information are direct, indirect and the model itself. Direct information involves observations that are made directly on a population of interest. Indirect information is obtained from outside sources somehow related to the population of interest. The indirect information is typically expressed as probability density functions that reflect knowledge and uncertainty about the unknown model quantities.

A basic premise about Bayesian melding is that information is available for some inputs θ and outputs ϕ, and it is likely that some uncertainty are associated with the information. The uncertainty is captured as probability distributions before and after applying the model. Bayesian pre-model priors are denoted as $f_{\theta}(\theta)$ and $f_{\phi}(\phi)$ for the inputs and outputs respectively. The likelihoods for the inputs and outputs are denoted by $L_{\theta}(\theta)$ and $L_{\phi}(\phi)$. Assuming conditional independence of inputs and outputs, the joint pre-model posterior distribution of θ and ϕ according to Bayes theorem is:

$$f(\theta,\phi) \propto f_\theta(\theta)L_\theta(\theta)f_\phi(\phi)L_\phi(\phi) \tag{5.7}$$

Considering the model Φ as a mapping of inputs to outputs: $\Phi(\theta) \to \phi$. The joint post-model posterior probability distribution $f'(\theta,\phi)$ will have a non-zero probability only when $\phi = \Phi(\theta)$. The Bayesian melding principle define the joint distribution of θ and ϕ given the model as the 'restriction of the pre-model distribution to the submanifold $(\theta,\phi) : \phi = \Phi(\theta)$', that is:

$$f'(\theta,\phi) \propto \begin{cases} f(\theta,\Phi(\theta)) & \text{if } \phi = \Phi(\theta), \\ 0 & \text{otherwise} \end{cases} \tag{5.8}$$

5.3 A Note on Transfer Performance

Transfer methods are generally evaluated experimentally rather than theoretically, including the algorithms we developed in this thesis. Only very little work has been done on theoretical evaluation of transfer algorithms and they focus on highly restricted transfer scenarios where the relationships between the source and target tasks are mathematically well-defined. Optimal learning is much more involved with several parameters to contend with, hence most of the current research at the intersection of transfer learning and optimal learning does not contain theoretical analysis.

Empirical analysis of learning performance is straightforward. We will discuss this using an acronym we refer to as *SEE - Start-up*, *Efficiency* and *Effectiveness*. Model transfer is important at the start of learning and initial performance can be measured against an expert or an ignorant agent. Efficiency relates to the time, effort or expense spent in accomplishing transfer. For example one could be interested in the amount of time it takes to fully learn the target task given the transferred knowledge compared to the amount of time to learn it from scratch.

Finally we have effectiveness i.e. the quality of being able to bring about an effect, which in transfer learning, captures the overall performance improvement brought about as a result of implementing a transfer learning scheme. A decrease in performance that is traced or associated with a transfer learning scheme means that negative transfer has occurred, avoiding negative transfer is a major challenge.

5.4 Chapter Summary

This chapter briefly summarised the methods of transferring prior information between models. By far the simplest method is the naive approach that transfers information without adjustments. We will look at an example of this method in chapter 6. More interesting are methods that do involve adjusting or discounting information before they are transferred thereby reducing or eliminating transfer bias. In chapter eight we will study one of such methods - the power prior Bayesian analysis approach.

6 The Transfer Framework

In Chapter five we saw that there are many different ways of exploiting transferable prior information by objectively aggregating all available information while respecting the different nature of the information sources and the uncertainty levels present in those sources. The methods for exploiting prior information reviewed in that chapter come from a wide variety of fields. In this chapter we start to develop our transfer methods for Bayes Adaptive Markov Decision Processes (BAMDP). The objectives of this chapter are: a) to establish the transfer context, i.e. the domain, task, and specific types of the transferable information, and b) to explain how we incorporate the transferable prior information into Bayesian estimates of expected returns.

6.1 Introduction

In Chapter five we saw that an agent can exploit the presence of information from related tasks to improve performance when learning a new task. A successful exploitation of the available information will depend on:

1. What constitutes the transfer context i.e. the specifications of the domain, task and available information, and

2. How prior information is to be quantified given all the available information and how the quantified prior information is to be incorporated into the agent's learning algorithm.

In this chapter we shall establish each of these items.

We consider situations in which an agent is learning to perform a task that can be formalised as a Bayes Adaptive Markov Decision process (BAMDP) in environments where additional information is explicitly available to the agent in the form of user knowledge and/or information acquired from previous learning. Specifically, suppose the agent can formulate a prior distribution for the transition parameters of the BAMDP but finds it difficult to come up with a well-specified prior distribution for the task-environment model. Although the agent is less confident that his/her prior distribution is an adequate representation of reality, we shall assume the agent is willing to exploit well specified *'donor'* information made available before the start of learning.

We use the term *'pro-forma'* to describe the agent's less-well specified prior distribution (Neal 2001) and refer to the well-specified prior distribution from the donors as *'pre-prior'* information. The transfer problem in our setup, for the agent, is how to integrate the well-specified pre-prior information with the less-well specified pro-forma distribution.

The rest of this chapter is structured into three main sections. In Section 6.2 we describe the transfer context and explain in Section 6.3 how we incorporate transferable information into Bayesian estimation steps of expected returns. The chapter ends in Section 6.4 with a summary.

6.2 The Transfer Context

We now describe the transfer context. To do so we shall first give the definition of the *domain* and a *task* in Section 6.2.1 and then turn to the definition of transferable information in Sections 6.2.2 and 6.2.3.

6.2.1 Domain and Task

A domain in the context of transfer learning in Markov decision processes (MDPs) consists of four of the five components that specify an MDP. These are the state space S, the action space A, a set of admissible state-action pairs Ψ and the transition function P. We adopt this definition of domain in this thesis. We assume that the transition function is completely or partially unknown and it is to be inferred from the agent's interaction with the domain. The reward function R is not considered part of the domain specification but instead forms part of the specification for the task.

Definition 6.1 (Domain:) *A domain υ is a tuple (S, A, Ψ, P). In the tuple, S is a set of $|S|$ distinct states, A is a set of $|A|$ distinct actions, $\Psi \subseteq S \times A$ is the set of admissible state-action pairs, $P : \Psi \times S \longrightarrow [0,1]$ is a Markovian transition model that specifies in a probabilistic form the process dynamics. The transition probabilities $p^a_{ss'} \in P$ naturally satisfy the constraints $0 \leq p^a_{ss'} \leq 1$; $\sum_{s'} p^a_{ss'} = 1 \; \forall s, s' \in S$.*

We define a task ω as a tuple consisting of a domain, a reward function and a performance measure.

Definition 6.2 (Task:) *A task $\omega \in \Omega$ is a tuple $\langle \upsilon, R, G \rangle$ where υ is the domain, $R : \Psi \longrightarrow \mathbb{R}$ is a reward function and G is the performance measure. Ω is the space of tasks.*

Performance G of the learning agent can be measured in several ways. To account for exploration and exploitation trade-off we will measure the discounted total reward to-go i.e. the average return at each point at each time step. Learning proceeds in multiple episodes. More precisely, suppose the agent receives the following rewards $r_1, r_2, \ldots, r_t, \ldots, r_n$ in a run of time length n. The return to go at time t is defined to be $\sum_{t' \geq t} \gamma^{(t'-t)} r_{t'}$. Each episode positions the learning agent in an initial state of the environment and terminates when an end condition is satisfied. For instance, an episode terminates when a maximum number of steps are achieved or when the agent reaches a terminal state. For simplicity in this thesis we primarily consider the case where both the source of the transferable information and the new task belong to the same domain.

6.2.2 Transferring Knowledge from Historical Data

It is often the case that decision makers may have access to data on previously accomplished tasks. Such data, which is generally referred to as historical data, can be very useful when learning to accomplish a new task. In lifelong tasks, for example, the data gathered from previous time periods may provide a useful prior distribution for new tasks. This is especially so when the new tasks only differ slightly from the previous tasks. In general, historical data may be elicited from diverse sources, of which raw historical data obtained from similar previous tasks is the most natural. Other sources include expert opinion, case-specific information, and functional model of data - empirical and/or theoretical.

Suppose while learning or experimenting on some previous tasks the experience acquired is stored and made explicitly available to an agent learning on a new but related task. Historical data in terms of experience acquired can be roughly divided into two groups: i) samples of transitions and ii) knowledge of solutions involving value and policy functions. Our focus in this thesis, in addition to transfer of constraints from experts, is on samples of transitions.

More formally, we consider situations in which samples of transition $\diamond$ data have been collected from previous tasks and made available to the learning agent. The samples of transitions $\diamond$ can be treated as historical counts and quantified into a pre-prior distribution $f(P|M^{\diamond})$. The transfer algorithm quantifies the prior distribution for a new learning task by combining the pre-prior distribution $f(P|M^{\diamond})$ of the donor with the pro-forma prior distribution $f(P|M^{\omega})$ of the recipient agent. The pro-forma distribution quantifies the agent's less-well specified prior information.

6.2.2.1 A Simple Transfer Algorithm

In this section we describe a simple algorithm for quantifying prior distribution M for a new task in the presence of a pre-prior information $M^{\diamond}$ and pro-forma prior information M^{ω}. We consider the simplest method discussed in Chapter five for transferring prior information between models. We assume the pre-prior information $M^{\diamond}$ is representative of the new task and apply a simple addition operator to combine the pre-prior and the pro-forma information as shown in Algorithm 6.1.

```
1: procedure $M$ = SIMPLE TRANSFER ALGORITHM($M^{\diamond}, M^{\omega}, S, A$)
2:     for each action $a$ in $A$ do
3:         for each state $s$ in $S$ do
4:             for each state $s'$ in $S$ do
5:                 $m^{a}_{ss'} = m^{\diamond,a}_{ss'} + m^{\omega,a}_{ss'}$
6:             end for
7:         end for
8:     end for
9:     return $M \leftarrow \left[m^{a}_{ss'}\right] \quad \forall s, s' \in S$ and $a \in A$.
10: end procedure
```

Algorithm 6.1: A simple transfer algorithm for combining pre-prior and pro-forma information. The algorithm uses an addition operator to combine the information. We shall consider alternatives to the simple transfer algorithm in subsequent chapters of this thesis.

The simple transfer algorithm takes as input the pre-prior information $M^{\diamond}$, the pro-forma prior information M^{ω} and the definition of states S and actions A. It returns a quantified prior M. The aggregation is achieved in line 5 through the use of an addition operator to combine each of the corresponding entries of the two matrices. The resulting prior can be used as input to an optimal learning algorithm for Bayes Adaptive MDPs. In the next section, we describe a set of simple experiments that uses the simple algorithm (Algorithm 6.1) with an optimistic model selection algorithm (Algorithm 4.3 of Chapter four) to learn a new task in the presence of historical data. We shall consider alternatives to the simple transfer algorithm in subsequent chapters of this thesis.

6.2.2.2 Experiment I: Grid World

We apply Algorithm 6.1 to quantify the prior distribution in a simple grid world domain. The domain is adapted from Russell and Norvig (2003). The environment is discrete and fully observable. It consists of a cellular grid with a known starting point and one or more exit points. Each cell in the grid is either a free space where the agent can move, or an obstacle. The agent can move in any of the free space in orthogonal directions using the usual four primitive actions of North, East, South and West, illustrated in Figure 6.1. There are uncertainties in the agent's actions, each action succeeds

Figure 6.1: Control Actions in the Grid World

in moving the agent in the chosen direction with a probability p and fails by moving the agent in a perpendicular direction with probability $1-p$. That is, if the agent selects a 'North' action then it will move north with probability p but will move east or west with probability $0.5 \times (1-p)$. If the agent hits an obstacle it will bounce back to its original location. There are positive and negative rewards for the actions. Moving from cell to cell in the grid incurs cost and the exits have rewards attached to them.

An agent on the grid is faced with the task of moving from a starting point to an exit point, selecting actions to take in each state of the environment with the objective of maximising expected rewards in the presence of obstacles and transition uncertainty. The agent is provided with the reward function but not the actual transition model of the task. The actual transition model is of dimension $S \times S$, but is sparse for this domain since the agent can only transition to adjacent cells.

Experiment Setup

We carried out two sets of experiments I-A and I-B to demonstrate the application of the simple transfer algorithm with optimistic model selection on the grid world. We also applied Q-learning to the grid world. The two sets of experiments differ in the actual probability p of success. The success probability is set to 0.9 and 0.5 in I-A and I-B respectively. We use a 4×3 Grid in both sets of experiments with a single start state at cell (1,1) and two exit cells (Hi - 4,3) and (Low - 4,2) as shown in Figure 6.2. The task consists of the simple 4×3 grid of Figure 6.2 and the reward function

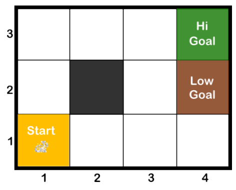
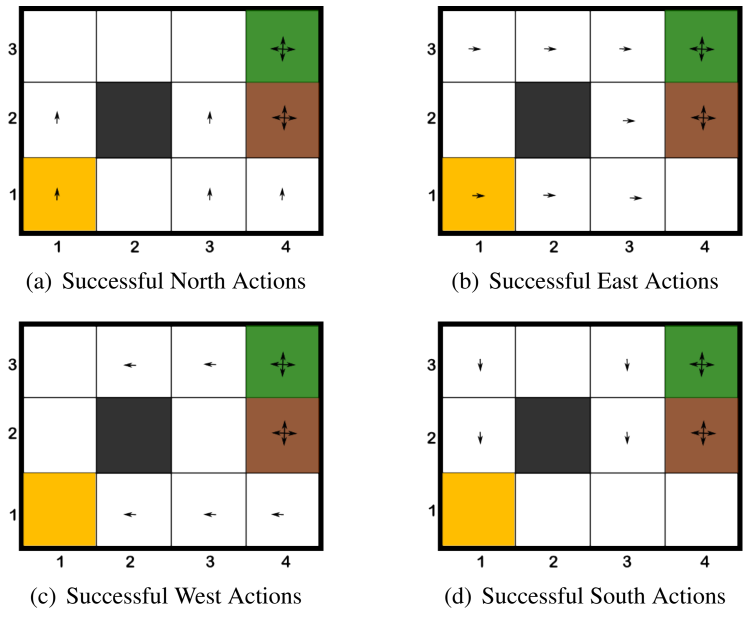


Figure 6.2: A 4×3 Grid World with High and Low Goal Cells. The black cell is an obstacle.

specified in Equation 6.1. The task in each of the experiment is kept the same.

$$r_s = \begin{cases} +1.0 & \text{for Hi Goal,} \\ -1.0 & \text{for Low Goal,} \\ -0.2 & \text{otherwise, for accessible cells} \end{cases} \tag{6.1}$$

The transitions that correspond to successful actions are shown in Figure 6.3.



(a) Successful North Actions (b) Successful East Actions

(c) Successful West Actions (d) Successful South Actions

Figure 6.3: Transitions Under Successful Actions in the 4×3 Grid World

The agent's pro-forma prior is Dirichlet whose parameters are set to one everywhere. The following historical data, Equations 6.2 - 6.5, in the form of a pre-prior information is supplied to the agent. The entries that correspond to admissible transitions carry more weight. For example, as illustrated in Figure 6.4 applying the North action in cell (1,1) the agent can either move to cell (1,2), cell (2,1) or remain at the same location i.e. cell (1,1). These three cells carry more weight in the pre-prior information row 1 of Equation 6.2.

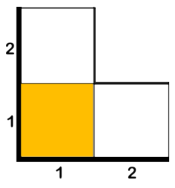


Figure 6.4: Admissible transitions under the North actions in cell (1,1).
The agent can move to cells (1,2),(2,1), or remain in (1,1)

$$M^{\diamond,North} = \begin{array}{c|ccccccccccc|} \mathit{Cells} & 1,1 & 2,1 & 3,1 & 4,1 & 1,2 & 3,2 & 4,2 & 1,3 & 2,3 & 3,3 & 4,3 \\ \hline 1,1 & 10.0 & 10.0 & 1.0 & 1.0 & 10.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 2,1 & 10.0 & 10.0 & 10.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 3,1 & 1.0 & 10.0 & 1.0 & 10.0 & 1.0 & 10.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 4,1 & 1.0 & 1.0 & 10.0 & 10.0 & 1.0 & 1.0 & 10.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 1,2 & 1.0 & 1.0 & 1.0 & 1.0 & 10.0 & 1.0 & 1.0 & 10.0 & 1.0 & 1.0 & 1.0 \\ 3,2 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 10.0 & 10.0 & 1.0 & 1.0 & 10.0 & 1.0 \\ 4,2 & 10.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 1,3 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 10.0 & 10.0 & 1.0 & 1.0 \\ 2,3 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 10.0 & 10.0 & 10.0 & 1.0 \\ 3,3 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 10.0 & 10.0 & 10.0 \\ 4,3 & 10.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \end{array} \tag{6.2}$$

$$M^{\diamond,East} = \begin{array}{c|ccccccccccc|}
Cells & 1,1 & 2,1 & 3,1 & 4,1 & 1,2 & 3,2 & 4,2 & 1,3 & 2,3 & 3,3 & 4,3 \\
1,1 & 10.0 & 10.0 & 1.0 & 1.0 & 10.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\
2,1 & 1.0 & 10.0 & 10.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\
3,1 & 1.0 & 1.0 & 10.0 & 10.0 & 1.0 & 10.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\
4,1 & 1.0 & 1.0 & 1.0 & 10.0 & 1.0 & 1.0 & 10.0 & 1.0 & 1.0 & 1.0 & 1.0 \\
1,2 & 10.0 & 1.0 & 1.0 & 1.0 & 10.0 & 1.0 & 1.0 & 10.0 & 1.0 & 1.0 & 1.0 \\
3,2 & 1.0 & 1.0 & 10.0 & 1.0 & 1.0 & 1.0 & 10.0 & 1.0 & 1.0 & 10.0 & 1.0 \\
4,2 & 10.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\
1,3 & 1.0 & 1.0 & 1.0 & 1.0 & 10.0 & 1.0 & 1.0 & 10.0 & 10.0 & 1.0 & 1.0 \\
2,3 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 10.0 & 10.0 & 1.0 \\
3,3 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 10.0 & 1.0 & 1.0 & 1.0 & 10.0 & 10.0 \\
4,3 & 10.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0
\end{array} \tag{6.3}$$

$$M^{\diamond,South} = \begin{array}{c|ccccccccccc|}
Cells & 1,1 & 2,1 & 3,1 & 4,1 & 1,2 & 3,2 & 4,2 & 1,3 & 2,3 & 3,3 & 4,3 \\
1,1 & 10.0 & 10.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\
2,1 & 10.0 & 10.0 & 10.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\
3,1 & 1.0 & 10.0 & 10.0 & 10.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\
4,1 & 1.0 & 1.0 & 10.0 & 10.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\
1,2 & 10.0 & 1.0 & 1.0 & 1.0 & 10.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\
3,2 & 1.0 & 1.0 & 10.0 & 1.0 & 1.0 & 10.0 & 10.0 & 1.0 & 1.0 & 1.0 & 1.0 \\
4,2 & 10.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\
1,3 & 1.0 & 1.0 & 1.0 & 1.0 & 10.0 & 1.0 & 1.0 & 10.0 & 10.0 & 1.0 & 1.0 \\
2,3 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 10.0 & 10.0 & 10.0 & 1.0 \\
3,3 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 10.0 & 1.0 & 1.0 & 10.0 & 1.0 & 10.0 \\
4,3 & 10.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0
\end{array} \tag{6.4}$$

$$M^{\diamond,West} = \begin{array}{c|ccccccccccc} \mathit{Cells} & 1,1 & 2,1 & 3,1 & 4,1 & 1,2 & 3,2 & 4,2 & 1,3 & 2,3 & 3,3 & 4,3 \\ \hline 1,1 & 10.0 & 1.0 & 1.0 & 1.0 & 10.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 2,1 & 10.0 & 10.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 3,1 & 1.0 & 10.0 & 10.0 & 1.0 & 1.0 & 10.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 4,1 & 1.0 & 1.0 & 10.0 & 10.0 & 1.0 & 1.0 & 10.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 1,2 & 10.0 & 1.0 & 1.0 & 1.0 & 10.0 & 1.0 & 1.0 & 10.0 & 1.0 & 1.0 & 1.0 \\ 3,2 & 1.0 & 1.0 & 10.0 & 1.0 & 1.0 & 10.0 & 1.0 & 1.0 & 1.0 & 10.0 & 1.0 \\ 4,2 & 10.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 1,3 & 1.0 & 1.0 & 1.0 & 1.0 & 10.0 & 1.0 & 1.0 & 10.0 & 1.0 & 1.0 & 1.0 \\ 2,3 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 10.0 & 10.0 & 1.0 & 1.0 \\ 3,3 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 10.0 & 1.0 & 1.0 & 10.0 & 10.0 & 1.0 \\ 4,3 & 10.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \end{array} \qquad (6.5)$$

The set up for the experiments are summarised in Tables 6.1 and 6.2. Each of I-A and I-B consists of four experiments (I-A-1, I-A-2, I-A-3, I-A-4) and (I-B-1, I-B-2, I-B-3, I-B-4). Experiments I-A-1 and I-B-1 were used as baselines in which the agent has access to the optimal policy for the task right from the start of the two experiments. We assume that the actual transition function is known to the agent in these two experiments. In contrast to Experiments I-A-1 and I-B-1, the transition model is unknown to the agent for the other experiments. In Experiments I-A-2 and I-B-2 the agent learns the transition function using the optimistic model selection algorithm and a non-informative Dirichlet pro-forma prior information whose parameters are set to 1 everywhere. Experiments I-A-3 and I-B-3 extends those of I-A-2 and I-B-2 by making available to the agent pre-prior information Equations 6.2 - 6.5 and the simple transfer algorithm. Finally, experiments I-A-4 and I-B-4 highlights the performance of a model free method (Q learning) applied to the grid world task. For Q learning, we use a learning rate $\alpha = \frac{1}{\#(s,a)}$ where $\#(s,a)$ is the number of times action a was executed in state s. For a balance between exploitation and exploration actions, we use in the Q learning experiments an ε-greedy exploration method with ε linearly reduced from 0.8 to 0. We carried out fifty trials per experiment and the discount factor is set to 0.95 in all the experiments.

	I-A-1	I-A-2	I-A-3	I-A-4
Domain	Grid Task 0.9 success probability	Grid Task 0.9 success probability	Grid Task 0.9 success probability	Grid Task 0.9 success probability
Task (Reward)	4×3 Grid $\gamma = 0.95$ Equation 6.1	4×3 Grid $\gamma = 0.95$ Equation 6.1	4×3 Grid $\gamma = 0.95$ Equation 6.1	4×3 Grid $\gamma = 0.95$ Equation 6.1
Learning Algorithm	Optimal Policy Known Model	OMS $\alpha = 95\%$	OMS $\alpha = 95\%$	Q Learning
Pro-Forma Prior	-	Uniform(1)	Uniform(1)	-
Pre-Prior	-	-	Equations 6.2 - 6.5	
Transfer Method	-	-	Simple Transfer Algorithm	-

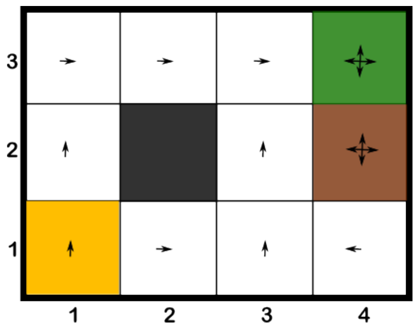
Table 6.1: Set up for I-A in a 4×3 grid world with 0.9 probability of successful action. I-A consists of four individual experiments - the baseline (I-A-1) in which the actual transition model is available to the agent and an optimal policy is used from start, I-A-2 involves learning using OMS with a non-informative prior information, I-A-3 involves learning with OMS using the simple transfer algorithm to combine the non-informative prior information supplied to the agent in I-A-2 and a pre-prior information Equations 6.2 - 6.5, and I-A-4 involves model free learning using the standard Q learning Algorithm.

	I-B-1	I-B-2	I-B-3	I-B-4
Domain	Grid Task 0.5 success probability	Grid Task 0.5 success probability	Grid Task 0.5 success probability	Grid Task 0.5 success probability
Task (Reward)	4×3 Grid $\gamma = 0.95$ Equation 6.1	4×3 Grid $\gamma = 0.95$ Equation 6.1	4×3 Grid $\gamma = 0.95$ Equation 6.1	4×3 Grid $\gamma = 0.95$ Equation 6.1
Learning Algorithm	Optimal Policy Known Model	OMS $\alpha = 95\%$	OMS $\alpha = 95\%$	Q Learning
Pro-Forma Prior	-	Uniform(1)	Uniform(1)	-
Pre-Prior	-	-	Equations 6.2 - 6.5	
Transfer Method	-	-	Simple Transfer Algorithm	-

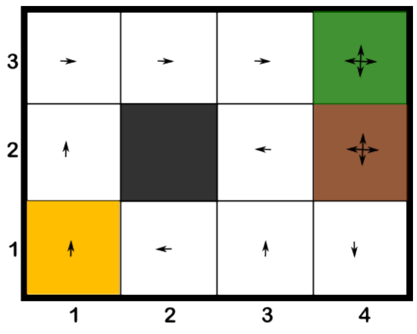
Table 6.2: Set up for I-B in a 4×3 grid world with 0.5 probability of successful action. I-B consists of four individual experiments - the baseline (I-B-1) in which the actual transition model is available to the agent and an optimal policy is used from start, I-B-2 involves learning using OMS with a non-informative prior information, I-B-3 involves learning with OMS using the simple transfer algorithm to combine the non-informative prior information supplied to the agent in I-B-2 and a pre-prior information Equations 6.2 - 6.5, and I-B-4 involves model free learning using the standard Q learning Algorithm.

Results and Discussion

The optimal policies are shown in Figures 6.5(a) and 6.5(b) for 0.9 and 0.5 probabilities of successful actions respectively. Due to the episodic nature of the tasks all of the four actions are optimal in the goal states. With the exceptions of cell locations (3,2) and (4,1) the optimal policies are the same in both sets of experiments. Figures 6.6 and 6.7 show the discounted total reward acquired by the agent



(a) Probability of successful action = 0.9.



(b) Probability of successful action = 0.5.

Figure 6.5: Optimal Policies for the Grid Task

over time in each of the experiments. The shaded area is estimates of standard deviation showing standard error around the mean. As we would expect, the agent is able to learn the task with and without transfer of the prior information. The agent's performances with optimal policies derived directly from the known models (I-A-1 and I-B-1) outperform all of the others that involve learning. In addition, as expected, the agent when using OMS with pro-forma prior is able to learn the task a bit quicker when the reliability of the action is greater i.e. convergence was achieved at around 500 steps when $p = 0.9$ in comparison to about 1000 steps when $p = 0.5$. More interesting is the role of the historical data i.e. the pre-prior. The agent benefits more from the historical data when $p = 0.5$ in comparison to when $p = 0.9$. One possible reason for this is that the transitions to each of the admissible cells are equally weighted in the pre-prior information supplied to the agent. Both the OMS with and without pre-prior outperforms the model free Q Learning method in terms of discounted total reward obtained. These differences can be seen in Figure 6.8 to be statistically significant.

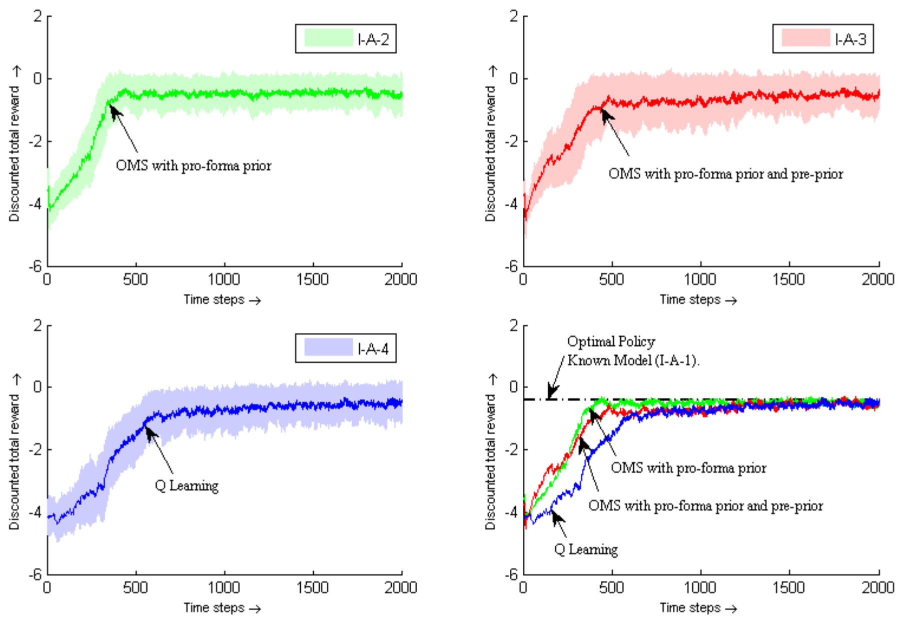


Figure 6.6: Discounted Total Rewards over time for Experiment I-A: I-A-1, I-A-2. I-A-3 & I-A-4. The shaded area is estimates of standard deviation showing standard error around the mean.

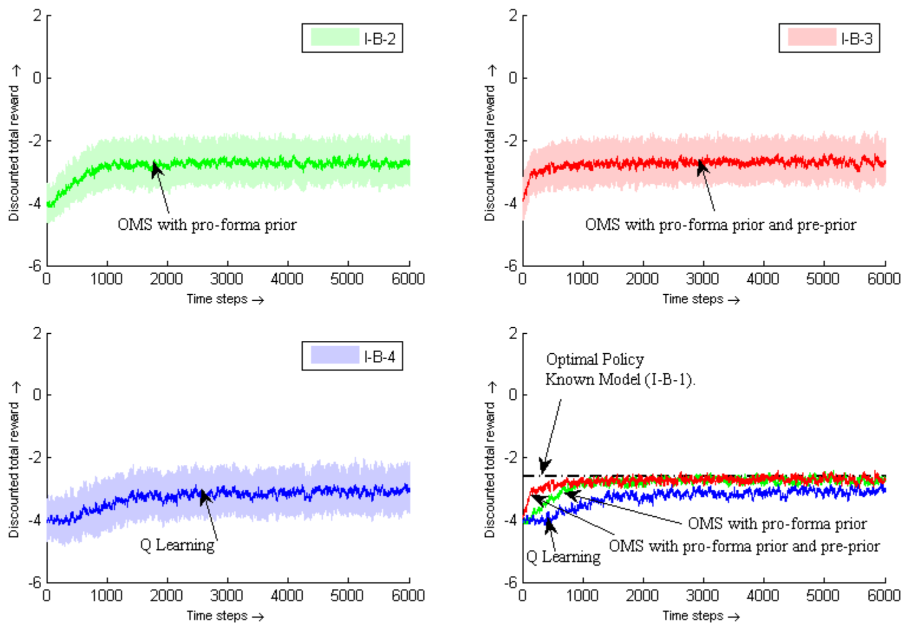


Figure 6.7: Discounted Total Rewards over time for Experiment I-B: I-B-1, I-B-2, I-B-3, & I-B-4. The shaded area is estimates of standard deviation showing standard error around the mean.

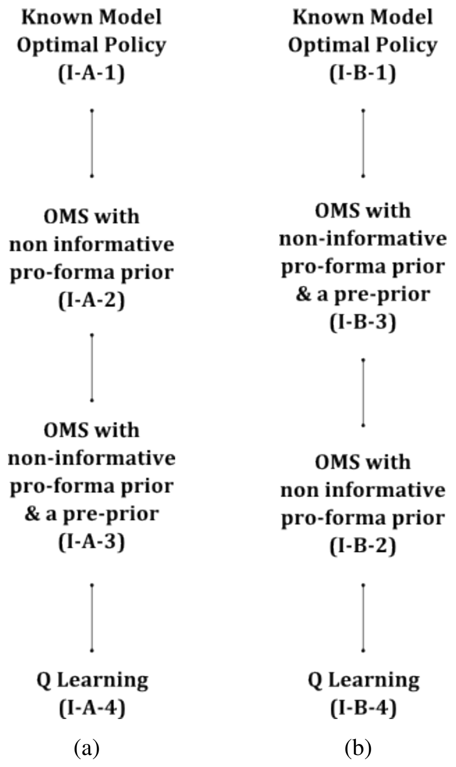


Figure 6.8: Ordering for discounted cumulative reward for Experiments I-A and I-B under (a) 0.9 probability of successful action and (b) 0.5 probability of successful action. The solid lines indicate statistical significance at 95% level.

6.2.3 Transferring Knowledge from Experts about Transition Constraints

Having shown how to incorporate historical data we now turn to the problem of how we can incorporate user knowledge in the form of constraints on transition models.

In many real-world applications, domain experts usually have valuable information about model parameters that can be expressed as transition constraints. A transition constraint refines a process model by expressing a condition or a restriction to which the model must conform. We gave simple examples of transition constraints in Chapter one. Our intention is to make the constraints as simple as possible, so that the experts can easily formalise their knowledge into these constraints. We distinguish two types of constraints, absolute and relative constraints. An absolute transition constraint allows domain experts to specify conditions for a transition parameter without having to worry about relating the specification to other transition parameters in the model. Relative constraints involve relative relationships between transition parameters.

In the context of absolute constraints, one can for example specify an upper bound and a lower bound for the probability of moving from state s to state s' under an action a such that $p^a_{ss'} \in \left[p^a_{\downarrow,ss'}, p^a_{\uparrow,ss'}\right]$ where $p^a_{\downarrow,ss'}$ and $p^a_{\uparrow,ss'}$ are the lower and upper bound restrictions placed on transition probability $p^a_{ss'}$. This type of constraint can be seen as a form of parameter tolerance that specifies the plausible deviations of the parameter from pre-specified values. A single known value of $p^a_{ss'}$ that collapses the bounds into a single estimate e.g. $p^a_{ss'} = 0.8$ can also be regarded as an absolute constraint, although such constraints are not common in practice. In between the two i.e. known parameter values and absolutely bounded transition constraints is yet another example, an interesting case that we refer to as significant inequalities. It is reasonable to sometimes expect in practice the availability of information relating to the size of a state transition probability. For example, in the grid task we described in section 6.2.2.2 the agent may be informed that the transition probability of a successful action is at least 0.8. This we regard as significant inequality because that single transition probability, say from state s to state s' under action a, is greater than the sum of the transition probabilities from s to all other successor states of s (i.e. excluding s'), under action a. Significant inequalities implicitly cross the boundary between absolute and relative constraints in that they give an indication of how significant the constraint is in comparison to some of the other transition parameters in the model.

A typical example of a relative constraint is an equality constraint on the transition probabilities, i.e. when we know that one or more state transitions share the same transition probability value. Another important class of relative constraints are order constraints on transition probabilities which we will cover in the next chapter.

6.2.3.1 Constrained Prior Distributions

A convenient choice of prior distribution over the transition parameters, given a set of transition constraints C_Ω over the tasks Ω, can be expressed as follows:

$$f(P|M) = \begin{cases} \frac{1}{Z_{C_\Omega}(M)} \prod_{a=1}^{A} \prod_{s=1}^{N} \prod_{s'=1}^{N} (p_{ss'}^a)^{m_{ss'}^a - 1} & \text{for } p_{ss'}^a \in \mathbb{P}_{C_\Omega} \\ 0 & \text{otherwise,} \end{cases} \tag{6.6a}$$

where $N = |S|$, $\vec{m}_s^a = \{m_{s1}^a, m_{s2}^a, \ldots, m_{s|S|}^a\}$ for the possible successor states of state $s \in S$ and action $a \in A$, given that $m_{ss'}^a > 0 \ \forall s, s' \in S$. The constraint space $\mathbb{P}_{C_\Omega}$ is a subset of the possible space of models $\mathbb{P}$ i.e. $\mathbb{P}_{C_\Omega} \in \mathbb{P}$. C_Ω is in two parts, the usual stochastic constraints on probabilities i.e. $0 \leq p_{ss'}^a \leq 1$; $\sum_{s'} p_{ss'}^a = 1 \ \forall s, s' \in S$, $a \in A$ and user-specified transition constraints ϕ_Ω in the space of tasks Ω. That is,

$$C_\Omega = \{\phi_\Omega; 0 \leq p_{ss'}^a \leq 1; \sum_{s'} p_{ss'}^a = 1 \ \forall s, s' \in S, \ a \in A\} \tag{6.6b}$$

and

$$Z_{C_\Omega}(M) = \int_{\mathbb{P}_{C_\Omega}} \prod_{a=1}^{A} \prod_{s=1}^{N} \prod_{s'=1}^{N} (p_{ss'}^a)^{m_{ss'}^a - 1} dp_{ss'}^a \tag{6.6c}$$

We have now shown how to incorporate user knowledge about constraints into a density over models that is also able to incorporate historical data. We now show how to modify the estimated expected return based on the resulting density.

6.3 Revised Bayesian Estimation of Expected Return

Recall from Chapter 3 the Bayesian estimator of expected return under optimal policy which we stated as follows:

$$V_s(M) = \int_{\mathbb{P}} V_s(P) f(P|M) dP \tag{6.7}$$

in which P is a model for the MDP in the space of all possible models $\mathbb{P}$, $f(P|M)$ is the density of the model P for the MDP and $V_s(M)$ is the estimated value given M. We wish to incorporate into our estimate of expected return the transferable prior information that is available to the learning agent, in the form of historical data characterised as pre-prior $M^\diamond$, pro-forma M^ω and transition constraints C_Ω. To do so we do need to modify the integral of Equation 6.7. The Bayesian estimator of expected return under the optimal policy then becomes:

$$V_s(M) = E[\tilde{V}_s|O,M^\diamond,M^\omega,C_\Omega] = \int_{\underbrace{\mathbb{P}_{C_\Omega}}_{\text{changes \textit{possible} to \textit{plausible} space of models}}} V_s(P)\underbrace{f(P|O,M^\diamond,M^\omega)}_{\textit{incorporates transferable information}}dP \tag{6.8}$$

The quantity of interest V is expressed as an expectation of a density over the transition functions P parameterised by a) a matrix M^ω of pro-forma transition counts and b) pre-prior historical data $M^\diamond$, and c) the observations O; subject to constraint $\mathbb{P}_{C_\Omega}$ on the model space. Equation 6.7 is modified in two significant ways.

1. After observing O a transition with experience tuple $< s,a,r,s' >$, the posterior over process models is, in a standard Bayesian context, decomposed into a likelihood and prior distribution as follows.

$$f(P|O,M^\diamond,M^\omega) \quad = \quad \frac{f(O|P)f(P|M^\omega,M^\diamond)}{f(O,M^\omega,M^\diamond)} \tag{6.9}$$

2. In contrast to Equation 6.7 in which the integral is over the space of possible models, the integral of Equation 6.8 is restricted to the space of plausible models. A question arising is how plausible or empirically adequate do the transition models have to be. We make a simplifying assumption that the transition constraints are known, fits with the way the environment works and neither violates the functional form nor what we know about the data.

The idea of plausible models is central to this thesis. The idea is not new. It is used for example in motion simulation including the exploitation of randomness to satisfy constraints as introduced by (Barzel, Hughes & Wood 1996). So what exactly is meant by plausible models? The adjective plausible has two meanings when used in conversations:

- Apparently reasonable and valid, and truthful,
- Given to or characterized by presenting specious arguments.

The notion of plausibility has been used by some researchers to solve inference problems e.g. (Chenney & Forsyth 2000, Fellows, Hartman, Hermelin, Landau, Rosamond & Rozenberg 2011). Chenney & Forsyth (2000) provides the following definition of plausibility.

> *A model, including its simulator, is plausible if the important statistics gathered from samples distributed according to $p(A)$ [a probabilistic model] are sufficiently close to the real world statistics we care about.*

This definition of plausibility is sufficient for our purpose even though we recognise that it is quite vague because it says nothing about which statistics we might care about, or what it means to be sufficiently close.

6.4 Chapter Summary

We described in this chapter our transfer setting and presented a framework for optimising learning in Bayes Adaptive MDPs for the transfer setting. The objectives in this chapter were twofold: a) to establish the transfer context, i.e. the domain, task, and specific types of the transferable information, and b) to explain how we incorporate the transferable prior information into Bayesian steps for estimating expected returns. We described the two main types of transferable information that we study in this thesis, namely historical data and transition constraints. The constraints are acquired from experts. It is previous knowledge that fits with the way the environment works and neither violates the functional form nor what we know about the data. We will describe in the next chapter how to reason with the transition constraints.

7

Reasoning with Constrained Transitions

Our focus in this chapter is on how to reason with transition constraints in the context of optimal learning. We will study computational algorithms for reasoning with the two types of transition constraints we distinguished in chapter six - absolute and relative constraints. Our original contribution in this chapter is that we show how to model both constraint types using the already known idea of truncated Dirichlet densities and how to use the resulting beliefs to control exploration.

7.1 Introduction

Constraints on transition parameters lead to a partitioning of the overall space of process models into feasible and infeasible regions. Knowledge about the unknown parameters of a transition model in form of constraints C_Ω, in a task space Ω, restricts the space $\mathbb{P}$ of possible models to a space of plausible models $\mathbb{P}_{C_\Omega}$ enabling us to discard non-conforming transition models. How to use information about constraints in the estimation procedure for optimal learning is the focus of this chapter.

There is a large literature on parameter constrained estimation, particularly in the areas of truncated distributions and order-restricted inferences, and comprehensive review of these areas includes Barlow, Bartholomew, Bremner & Brunk (1972). The two main approaches are constrained maximum likelihood estimation and Bayesian inference typically done via Monte Carlo Markov Chains. A simple approach to constrained parameter estimation problem is rejection sampling. Given a constrained distribution, we could sample from the parent (unconstrained) distribution and only accept samples that satisfy given constraints. Unfortunately, the rejection sampling algorithm may be too inefficient especially when the number of parameters is large and/or when the constraints are severe. In such cases, the algorithm will be too slow as a high percentage of the candidate variates will be rejected.

There are a number of constrained maximum likelihood estimation algorithms for multinomial distributions especially in situations where there is order restriction on model parameters. The most widely used of these algorithms is the pool adjacent violators algorithm (Ayer, Brunk, Ewing, Reid & Silverman 1955) which is applicable only in the case of a simple linear ordering or an amalgamation of simple linear orderings. Jewell & Kalbfleisch (2004) described a variant of the pool adjacent violators algorithm as a maximum likelihood estimator of a series of ordered multinomial parameters. They demonstrate convergence of the algorithm. A known computational routine that implements isotonic regression for the case of a simple linear ordering is credited to Cran (1980). Lim, Wang & Choi (2009) reformulated the maximum likelihood estimation problem of ordered multinomial probabilities as a geometric program and then solved the problem globally and efficiently. They reported the computational merits of the geometric programming approach to the pool adjacent violators algorithm of Jewell & Kalbfleisch (2004) and concluded that geometric programming based approach is computational faster and easier to use.

Compared with the vast literature on constrained maximum likelihood estimation particularly for order restricted inference, few Bayesian methods have been reported for solving constrained parameter problems. Typically, constraints in Bayesian estimation methods are imposed by choosing a prior distribution that has support on a restricted space. Bayesian techniques for constrained parameter inference include the use of truncated prior distribution (Gelfand, Smith & Lee 1992, Sedransk, Monahan & Chiu 1985), applying transformations e.g. isotonic regressions to unconstrained parameter estimates (Gunn & Dunson 2005). As we shall see later in this chapter, Monte Carlo Markov Chains, particularly Gibbs sampling can be straightforwardly implemented for Bayesian calculations to solve constrained parameter and truncated data problems (Gelfand et al. 1992). The sequential

Monte Carlo (SMC) algorithm of Lang, Chen, Bakshi, Goel & Ungarala (2007) for Bayesian estimation in constrained dynamic system provides an alternative to the classical Monte Carlo rejection sampling approach we mentioned earlier. The SMC algorithm enforces inequality constraints via acceptance / rejection algorithm but is more accurate and efficient. They showed that the SMC algorithm possesses the theoretical properties of unconstrained SMC.

Compared to the Bayesian approaches, the constrained maximum likelihood estimation faces the challenges of assessing uncertainty in the parameter estimates since standard asymptotic theory does not apply in this case (Geyer 1991). In addition, the constrained maximum likelihood estimates are boundary values, i.e., parameters violating the constraints are simply moved to their corresponding constraint boundaries. To support inference mechanisms in knowledge systems, White (1986) presented simple generalisations of Bayes' rule that allow both a priori probabilities and conditional probabilities to be described by linear inequalities. His generalisations also produce an extreme point description and a linear inequality description of a set containing all possible a posteriori probabilities.

In the context of process control, the role of constraints in optimal control problems for linear systems is recognised in the literature e.g. Dreyfus (1962), Ho (1962), and Chang & Seborg (1982). Linear programming has been shown (Chang & Seborg 1982) to be a powerful method for developing multivariate control schemes which explicitly includes linear equality and linear inequality constraints on the state, control and/or output variables. Typically, a linear programming problem is solved on-line at sampling instants to determine the values of the control variables which minimise or maximise a linear performance index while satisfying the constraints.

The main thrust of this chapter is the use of constrained parameter Bayesian inference methods to reason about transition constraints in Markov decision processes. To do this we rely primarily on truncated Dirichlet distribution. Before going on to describe the truncated distribution we shall first look at how we could convert a set of constraints to simple bounds through linear programming. We describe the conversion process in Section 7.2 and then move on to our probabilistic inference methods in Section 7.3. Equally important is how the result of the inference feeds into our exploration control algorithms. We describe the link in Section 7.4. We present our experiments and its results in Sections 7.5, 7.6, & 7.7 and end the chapter in Section 7.8 with concluding remarks

7.2 Converting a Convex Set into Simple Bounds

In this section we shall describe a method of converting a set of transition constraints into lower and upper bounds on transition probabilities. Let us consider a convex set C_Ω on the transition probabilities $\vec{p}^a_s$ of state s under action a in a Markov decision process. The constraints in C_Ω can be converted into lower and upper bounds, $p^a_{\downarrow,ss^*} \leq p^a_{ss^*} \leq p^a_{\uparrow,ss^*}$, by solving the following optimisation problem.

For lower bound,

$$\begin{aligned} p^a_{\downarrow,ss^*} = \quad & \min \quad p^a_{ss^*} \quad s^* \in S \\ & \text{s.t.} \quad p^a_{ss'} \in \mathbb{P}_{C_\Omega} \quad \forall s, s' \in S. \end{aligned} \tag{7.1}$$

and for upper bound,

$$\begin{aligned} p^a_{\uparrow,ss^*} = \quad & \max \quad p^a_{ss^*} \quad s^* \in S \\ & \text{s.t.} \quad p^a_{ss'} \in \mathbb{P}_{C_\Omega} \quad \forall s, s' \in S. \end{aligned} \tag{7.2}$$

The optimisation problem is a linear program for constraints C_Ω that consists of linear equalities and/or inequalities.

To illustrate, let us recall the hospital robot example we described in Chapter one, Page 7, in which a set of constraints is imposed on the robot's task-environment model. The lower and upper bounds on transition probabilities p^{dock}_{12} and p^{dock}_{13} can be calculated from the constraints by solving the linear programs specified in Equations 7.1 and 7.2. The linear program gave the same result we inferred in Chapter one, i.e. $0.007 \leq p^{dock}_{12} \leq 0.06$ and $0.87 \leq p^{dock}_{13} \leq 0.923$.

Converting constraints to simple bounds may present us with a solution when the constraints are many and/or severe. The bounds calculated from the linear programs in Equations 7.1 and 7.2 can, for example, be used to specify a truncated Dirichlet distribution over the space of the transition models. We shall turn our attention in the next section to truncated Dirichlet distribution.

7.3 Probabilistic Inference in Constrained Transition Models

This section introduces the truncated Dirichlet distribution approach to modelling transition probabilities. We shall use the truncated Dirichlet distribution to reason with transition constraints and use the resulting belief as input to our exploration control algorithms. In Section 7.3.1 we will first describe the density function of the truncated Dirichlet distribution for transition models. We will then describe how to draw samples from the distribution in Section 7.3.2. In Section 7.3.3 we will extend our knowledge of working with truncated Dirichlet distributions to probabilistic inferences on order restricted transition models.

7.3.1 Truncated Dirichlet Distributions

A truncated distribution is a conditional distribution that results from apriori restricting the domain of some other probability distribution. Not only do such distributions prevent values outside of truncated bounds, a proper truncated distribution should integrate to one within the truncated bounds. In contrast to a truncated distribution, a censored distribution occurs when the probability distribution is still allowed outside of a pre-specified range.

A random vector of transition probabilities $\vec{p}^a_s = \{p^a_{s1}, p^a_{s2}, \ldots, p^a_{sN}\}$ for the possible N successor states of state $s \in S$ under action $a \in A$ is said to be a *truncated Dirichlet distribution* (TDirichlet) if the density of $\vec{p}^a_s$ is:

$$\begin{cases} Z(\vec{m}^a_s)^{-1} \prod_{s'=1}^{N} (p^a_{ss'})^{m^a_{ss'}-1}, & \text{for } \vec{p}^a_s \in \mathbb{P}_{C_\Omega}(\vec{p}^a_{\downarrow,s}, \vec{p}^a_{\uparrow,s}) \\ 0, & \text{otherwise} \end{cases} \tag{7.3}$$

where $Z(\vec{m}^a_s)^{-1}$ is the normalizing constant, $\vec{p}^a_{\downarrow,s} = (p^a_{\downarrow,s1}, p^a_{\downarrow,s2}, \ldots, p^a_{\downarrow,sN})$, $\vec{p}^a_{\uparrow,s} = (p^a_{\uparrow,s1}, p^a_{\uparrow,s2}, \ldots, p^a_{\uparrow,sN})$, and the convex polyhedra

$$\mathbb{P}_{C_\Omega}(\vec{p}^a_{\downarrow,s}, \vec{p}^a_{\uparrow,s}) = \{\vec{p}^a_s : p^a_{\downarrow,ss'} \le p^a_{ss'} \le p^a_{\uparrow,ss'}, 1 \le s' \le N, \sum_{s'=1}^{N} p^a_{ss'} = 1\} \tag{7.4}$$

When $p^a_{\downarrow,ss'} = 0$ and $p^a_{\uparrow,ss'} = 1$, $\forall s,s' \in S$ the TDirichlet$(\vec{p}^a_{\downarrow,s}, \vec{p}^a_{\uparrow,s}; m^a_{s1}, m^a_{s2}, \dots, m^a_{sN})$ reduces to the Dirichlet distribution. When $N = 2$ the truncated Dirichlet distribution reduces to a truncated Beta distribution.

7.3.2 Sampling Truncated Dirichlet Distributions

Ng et al. (2011) presented conditional sampling methods and Gibbs samplers as two approaches that can be used for generating independent identically distributed samples from truncated Dirichlet distributions. We adopt the Gibbs sampler approach in this thesis and use it to generate samples of transition models from truncated Dirichlet distributions. The Gibbs sampler requires all full conditional distributions and relies on being able to simulate random draws from associated truncated Beta distributions. We will first describe how we draw samples from the truncated Beta distribution in Section 7.3.2.1 and then extend the presentation to the Gibbs sampler for the truncated distribution in Section 7.3.2.2.

7.3.2.1 Sampling Transition Models from Truncated Beta Distribution

It is easy to draw random samples from a truncated Beta distribution. Following Ng et al. (2011), we develop and implement our sampler as follows.

1. Assume we are interested in drawing random samples of a transition probability:

   $p^a_{s1} \sim TBeta(m^a_{s_1}, m^a_{s_2}, p^a_{\downarrow,s_1}, p^a_{\uparrow,s_1})$

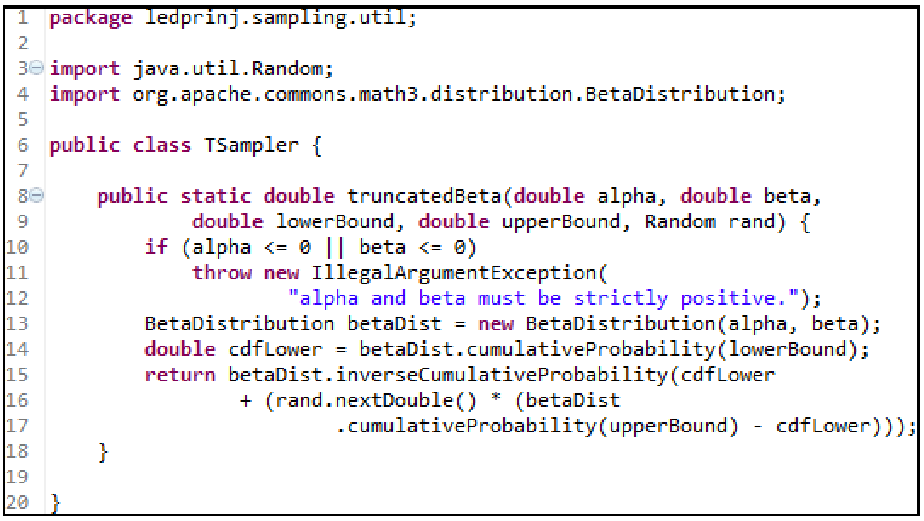
2. Let ζ be a random value drawn from a standard uniform distribution with $0 \leq \zeta \leq 1$.

3. The random transition probability p^a_{s1} is generated as follows:

   $p^a_{s1} \leftarrow F_B^{-1}(F_B(p^a_{\downarrow,s1}) + \zeta\{F_B(p^a_{\uparrow,s1}) - F_B(p^a_{\downarrow,s1})\})$,

   where $F_B(\cdot)$ denotes the cumulative density function (CDF) of the beta distribution with parameters (m^a_{s1}, m^a_{s2}) and $F_B^{-1}(\cdot)$ is its inverse function. $p^a_{s2} = 1 - p^a_{s1}$ and ζ is set in step 2.

Fortunately, evaluating $F_B(\cdot)$ and $F_B^{-1}(\cdot)$ are straightforward as software is readily available to do the calculations e.g. R or S-Plus. A simple implementation in Java using the Beta distribution class in Apache org.apache.commons.math3.distribution.BetaDistribution library is listed in Figure 7.1



```java
package ledprinj.sampling.util;

import java.util.Random;
import org.apache.commons.math3.distribution.BetaDistribution;

public class TSampler {

    public static double truncatedBeta(double alpha, double beta,
            double lowerBound, double upperBound, Random rand) {
        if (alpha <= 0 || beta <= 0)
            throw new IllegalArgumentException(
                    "alpha and beta must be strictly positive.");
        BetaDistribution betaDist = new BetaDistribution(alpha, beta);
        double cdfLower = betaDist.cumulativeProbability(lowerBound);
        return betaDist.inverseCumulativeProbability(cdfLower
                + (rand.nextDouble() * (betaDist
                        .cumulativeProbability(upperBound) - cdfLower)));
    }

}
```

Figure 7.1: A simple implementation of a Java function for sampling from a truncated Beta distribution.

7.3.2.2 Extending the Sampler to Truncated Dirichlet Distributions

To facilitate sampling from a truncated Dirichlet distribution TDirichlet$(\vec{p}^a_{\downarrow,s}, \vec{p}^a_{\uparrow,s}; m^a_{s1}, m^a_{s2}, \ldots, m^a_{sN})$ we adapt from the distribution theory the following (Ng et al. 2011):

$$y^a_{ss'*}|\vec{p}^a_{s(-N)} \sim TBeta(m^a_{ss'}, m^a_{sN}, p^a_{\downarrow,ss'*}, p^a_{\uparrow,ss'*}) \tag{7.5}$$

where $\vec{p}^a_{s(-N)} = \{p^a_{s1}, p^a_{s2}, \ldots, p^a_{sN-1}\}$

$$y^a_{ss'*} = \frac{p^a_{ss'}}{p^a_{ss'*}} \quad \forall s' = 1, 2, \ldots N-1, \tag{7.6}$$

$$p^a_{ss'*} = p^a_{sN} + p^a_{ss'} = 1 - \sum_{j=1, j\neq s'}^{N-1} p^a_{sj} \tag{7.7}$$

and

$$p^a_{\downarrow,ss'*} = \max(\frac{p^a_{\downarrow,ss'}}{y^a_{ss'*}}, 1 - \frac{p^a_{\uparrow,sN}}{y^a_{ss'*}}) \quad p^a_{\uparrow,ss'*} = \min(\frac{p^a_{\uparrow,ss'}}{y^a_{ss'*}}, 1 - \frac{p^a_{\downarrow,sN}}{y^a_{ss'*}}) \tag{7.8}$$

The sampling algorithm is therefore as follows: we proceed through the indices $s' = 1, 2, \ldots, N-1$, where we generate $y^a_{ss'*}$ from the corresponding truncated Beta distribution and set $p^a_{ss'} = p^a_{ss'*} \times y^a_{ss'*}$.

7.3.3 Order Restrictions on Transition Probabilities

Let us consider the following order restriction on the transition probabilities $\vec{p}^a_s = \{p^a_{s1}, p^a_{s2}, \ldots, p^a_{sN}\}$ for the possible N successor states of state $s \in S$ under action $a \in A$, and for a known k:

$$p^a_{s1} \leq p^a_{s2} \leq \ldots \leq p^a_{sk} \geq p^a_{sk+1} \geq \ldots \geq p^a_{sN}; \;\; 0 \leq p^a_{ss'} \leq 1, \;\; \sum_{s'=1}^{N} p^a_{ss'} = 1 \tag{7.9}$$

As we did for truncated Dirichlet distributions in Section 7.3.1, we will use the Gibbs sampler that requires the full conditional for $p^a_{ss'}$, $s' = 1, \ldots, N-1$ (with p^a_{sk} as fraction of these $p^a_{ss'}$) probabilities; $[p^a_{ss'} | p^a_{sj}, j = 1, \ldots, N-1, j \neq s', k]$ is a Beta distribution scaled to $[0, 1 - \sum_{j=1, j\neq s'}^{N-1} p^a_{sj}]$ and then suitably restricted according to the following constraints determined by k (Sedransk et al. 1985, Gelfand et al. 1992).

$$\begin{cases} \text{if } s' < k, & \max(p^a_{ss'-1}, b^a_{ss'} - p^a_{sN-1}) \leq p^a_{ss'} \leq \min(p^a_{ss'+1}, b^a_{ss'}); \\ \text{if } s' > k, & \max(p^a_{ss'+1}, b^a_{ss'} - p^a_{sN-1}) \leq p^a_{ss'} \leq \min(p^a_{ss'-1}, b^a_{ss'}); \\ \text{if } s' = k, & \max(p^a_{sk-1}, p^a_{sk+1}, b^a_{sk} - p^a_{sN-1}) \leq p^a_{sk} \leq b^a_{sk}. \end{cases} \tag{7.10}$$

where $b^a_{ss'} = 1 - \sum_{j=1, j\neq s'}^{N-1} p^a_{sj}$ and $p^a_{s0} \equiv 0$

As discussed in Section 7.1, an alternative approach to inference for order constrained parameters is through either pool adjacent violators algorithms or Geometric Programming (Jewell & Kalbfleisch 2004, Lim et al. 2009). The two algorithms are straightforward to implement for the order constraints we specified in Equation 7.9.

7.4 Constrained Optimistic Model Selection (COMS)

In this section we describe our extension to the optimistic model selection algorithm (Algorithm 4.1) to make it work with transition constraints. We refer to the new algorithm as constrained OMS (COMS) and its main loop is shown in Algorithm 7.1. The algorithm takes as input a known specification of an MDP i.e. the states S, actions A and reward function R of the MDP. Also passed as input to the procedure are the prior distribution M, the discount rate γ for infinite horizon tasks, α that signifies the level of confidence attached to probability intervals, and the initial state values V. The procedure returns an optimistic model P^a_{opt}. In addition, the algorithm takes as parameter the set of transition constraints C_Ω supplied to the agent.

```
1: procedure $P^a_{opt} = OMS(S,R,A,V,M,\gamma,\alpha,C_\Omega)$
2:     Initialise $V_k$  $\forall s,s' \in \mathbb{S} \cup k$ and $\forall a \in \mathbb{A}$
3:     Initialise $m^a_{sk}, m^a_{ks'}$  $\forall s,s' \in \mathbb{S} \cup k$ according to $M$
4:     $\xi^a_s = \gamma V_k$,  $\forall s \in \mathbb{S}$ and $\forall a \in \mathbb{A}$ where $m^a_{sk} > 0$
5:     $\xi^a_k = V_k$,  $\forall a \in \mathbb{A}$
6:     loop
7:         observe current state $x$
8:         select action $a = \arg\max_b\{\xi^b_x\}$ breaking ties randomly
9:         execute action $a$ in state $x$ and observe the transition $x \overset{a,r}{\rightsquigarrow} y$
10:        update belief $m^a_{xy} = [m^a_{xy} + 1]|C_\Omega$
11:        repeat
12:            choose i
13:            for each action b do
14:                find $\vec{p}^{\,b}_{opt,i}$ using the Goal Program Equation 4.6.
15:                update $\xi^b_i$ using Equation 4.3
16:            end for
17:        until ARTDP algorithm stops
18:    end loop
19: end procedure
```

Algorithm 7.1: The Main Loop of the Constrained Optimistic Model Selection (COMS) Algorithm.

The algorithm is in most part the same as that of the OMS except in Lines 10 and 14 where we take cognisance of the information about constraints.

Lines 2-5 are the initialisation steps. The true value function for the hypothetical state V_k is initialised in line 2 and the prior information for the hypothetical state is initialised in line 3. The exploration values are initialised in lines 4 and 5. The underlying MDP is augmented with a hypothetical state k during the initialisation steps.

Lines 6-18 contains the main learning steps. The agent observes the current state x in line 7 and selects an action a to perform given the current state and the current exploration values. In line 8, the agent selects an action a with the highest optimistic value, breaking ties randomly. In line 9 the agent executes the action selected and observes the transition to the next state i.e. $x \overset{a,r}{\rightsquigarrow} y$. The observed transition is used to update beliefs in line 10 subject to the constraints C_Ω imposed on the transition probabilities. This involves a recognition of the specific type of constraint involved and applying the techniques we described in Sections 7.2 and 7.3 to update M. The updated belief is the posterior distribution jointly defined by M and the constraints C_Ω.

The agent runs its asynchronous real time dynamic programming (ARTDP) algorithm in lines 11-17. Since the value function may change as the agent perform asynchronous back-ups it would be necessary to perform model selection every time a state action pair is backed up. The optimistic model is selected in line 14 using the Goal Program we formulated in Equation 4.6 (Chapter four) and the exploration value function is revised in line 15 to reflect the newly selected model. The optimistic estimate can be calculated using any form of ARTDP.

7.5 Experiment II: The Grid World with Transition Constraints

Recall the robot on a grid task we studied in Chapter six and in particular experiment I-A where we showed the performance of a uniform pro-forma prior on a 4×3 grid world and in which actions carry a 0.9 probability of success. We revisit experiment I-A in this chapter.

We keep the set up for the experiment the same as in Chapter six. The details of the experiments are recalled in Table 7.1 below in which experiments II-1, II-2 and II-3 are same as their counterparts in Chapter six i.e. I-A-1, I-A-2 and I-A-3 respectively.

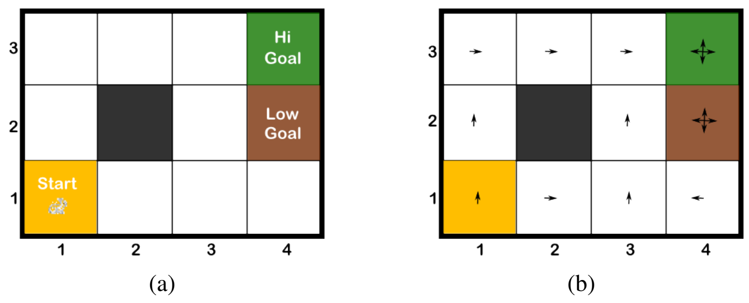


Figure 7.2: The 4×3 Grid World with Transition Constraints. The Grid contains one start cell and two exit cells - High and Low Goals. The black cell is inaccessible. a) The 4×3 Grid. b) Optimal Policy for 0.9 probability of success.

	II-1	II-2	II-3	II-4	II-5
Domain	Grid Task 0.9 success probability	Grid Task 0.9 success probability	Grid Task 0.9 success probability	Grid Task 0.9 success probability	Grid Task 0.9 success probability
Task (Reward)	4×3 Grid $\gamma = 0.95$ Equation 6.1	4×3 Grid $\gamma = 0.95$ Equation 6.1	4×3 Grid $\gamma = 0.95$ Equation 6.1	4×3 Grid $\gamma = 0.95$ Equation 6.1	4×3 Grid $\gamma = 0.95$ Equation 6.1
Learning Algorithm	Optimal Policy Known Model	OMS $\alpha = 95\%$	OMS $\alpha = 95\%$	COMS $\alpha = 95\%$	COMS $\alpha = 95\%$
Pro-Forma Prior	-	Uniform(1)	Uniform(1)	Uniform(1)	Uniform(1)
Pre-Prior Constraint	- - -	- - -	Equations 6.2 - 6.5 -	Equations 6.2 - 6.5 Tolerances Prob of Success $= 0.9 \pm 0.05$	- - Tolerances Prob of Success $= 0.9 \pm 0.05$
Transfer Method	-	-	Simple Transfer (ST) Algorithm	ST & Sampling from Truncated Dirichlet	Sampling from Truncated Dirichlet

Table 7.1: Set up for II in a 4×3 grid world with 0.9 probability of successful action. II consists of five individual experiments - the baseline (II-1) in which the actual transition model is available to the agent and an optimal policy is used from start, II-2 involves learning using OMS with a non-informative prior information, II-3 involves learning with OMS using the simple transfer algorithm to combine the non-informative prior information supplied to the agent in II-2 and a pre-prior information Equations 6.2 - 6.5, and II-4 & II-5 involve transition constraints using the COMS learning Algorithm.

Experiments II-4 and II-5 are new and they are designed to test the COMS algorithm. The set-up for experiment II-4 is basically the same as II-3 except that in both II-4 and II-5 the learning algorithm is COMS and there is knowledge of constraints. II-4 and II-5 differs with respect to pre-prior information. II-4 allows for pre-prior information (historical data) as in II-3 unlike II-5 in which there is no pre-prior information (historical data). The constraints are absolute and set to a tolerance of 0.9 ± 0.05 for transitions involving successful actions. The results of experiments II-4 and II-5 alongside those of II-1, II-2 and II-3 are shown in Figures 7.3 and 7.4.

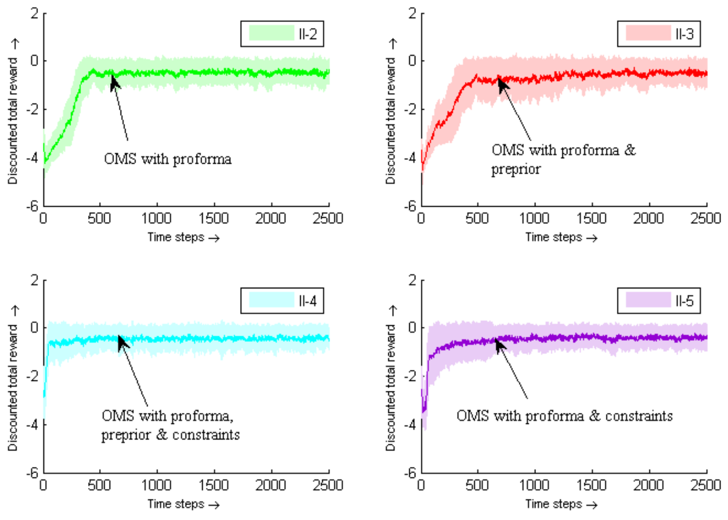


Figure 7.3: Discounted Total Rewards over time
for Experiment II-2, II-3, II-4 & II-5.

As before, the shaded area is estimates of standard deviation showing standard error around the mean. We saw in Chapter six that the agent is able to learn the task with and without transfer of the prior information. We also saw in Chapter six that both the OMS with and without pre-prior outperforms the model free Q Learning method in terms of discounted total reward obtained and the OMS with pro-forma information performed better than OMS with pro-forma and pre-prior information. As expected, the results of Experiments II-4 and II-5 show that the learning performance of the agent improved with the constraints. The pro-forma and pre-prior information when combined with constraints performed better than just the pro-forma information with constraints. The absolute constraint allowed the agent to quickly improve the precision of its estimates transition probabilities. The agent's performances with optimal policy derived directly from the known models (II-1) outperform all of the others that involve learning. These differences can be seen in Figure 7.5 to be statistically significant.

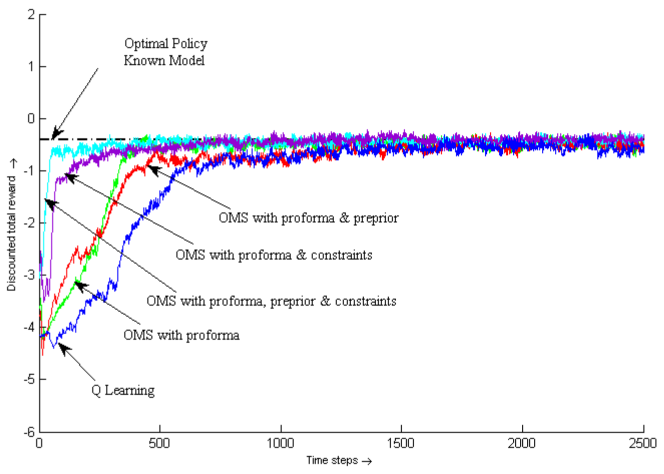


Figure 7.4: Comparison of Results for Experiment II-2, II-3, II-4 & II-5.
The Q Learning result is from Experiment I-A-4 in Chapter six.

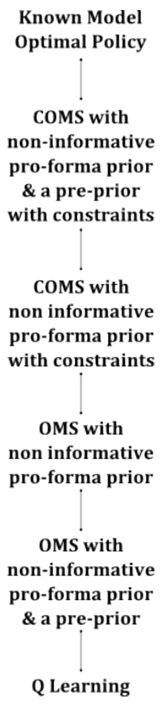


Figure 7.5: Ordering for methods in Experiment II based on the discounted cumulative reward obtained. The solid lines indicate statistical significance at 95% level.

7.6 Experiment III: Maintenance Intervention Task

An important challenge a decision maker often face is deciding when to intervene so as to alter or prevent the progression of a condition. Instances of this challenge abound. In health service, for example, physicians are often faced with decisions about medical interventions when precipitating events threaten patient's life or when patient's wellbeing is severely affected (Alagoz, Maillart, Schaefer & Roberts 2004, Magni, Quaglini, Marchetti & Barosi 2000, Shechter, Bailey, Schaefer & Roberts 2008). The challenge typically requires the decision maker to choose between two actions i) *'watchful waiting'* i.e. postpone the decision up to a critical point, and ii) *'intervene'*. While seemingly straightforward, such tasks (referred to as intervention timing tasks) involve uncertainty, complexity and dynamic change. A simplified abstraction of the intervention timing task is shown in Figure 7.6. The states of the task comprise of an intervention state 0 and health states $1, 2, \ldots, H+1$ in order of decreasing health. The action space consists of *watchful waiting* and *intervene*. We ex-

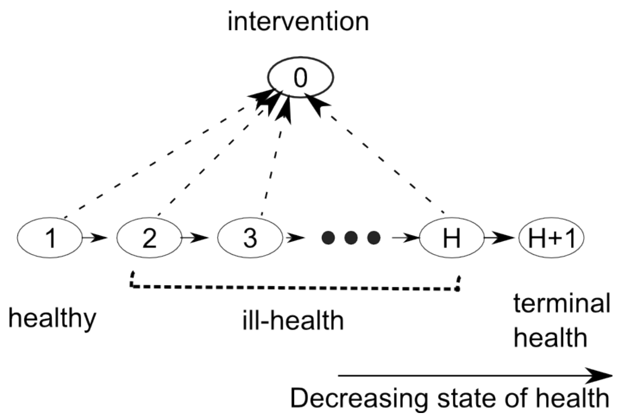


Figure 7.6: A simplified MDP model of the intervention timing task. Transitions under watchful waiting actions are shown as solid lines while those for intervene actions are shown in broken lines. Not all possible transitions are shown in the figure.

periment with a specific instance of the intervention timing task. We adapt a model of a maintenance intervention task from the literature Moustafa, Maksoud & Sadek (2004) and use it to highlight the benefits of transferable prior information in practical settings. The maintenance task consists of three actions, namely *do-nothing*, *minimal maintenance* and *replacement*. The do-nothing action corresponds to watchful waiting whilst the other two, i.e. *minimal maintenance* and *replacement*, are interventions. The transition matrix for the deteriorating system is shown in Equation 7.11.

$$P^a = \begin{bmatrix} 0.28 & 0.20 & 0.15 & 0.12 & 0.09 & 0.07 & 0.05 & 0.03 & 0.01 \\ & 0.40 & 0.19 & 0.15 & 0.10 & 0.07 & 0.05 & 0.03 & 0.01 \\ & & 0.32 & 0.23 & 0.17 & 0.12 & 0.08 & 0.05 & 0.03 \\ & & & 0.32 & 0.26 & 0.20 & 0.13 & 0.07 & 0.02 \\ & & & & 0.56 & 0.20 & 0.14 & 0.08 & 0.02 \\ & & & & & 0.60 & 0.25 & 0.13 & 0.02 \\ & & & & & & 0.65 & 0.22 & 0.13 \\ & & & & & & & 0.70 & 0.30 \\ & & & & & & & & 1.0 \end{bmatrix} \tag{7.11}$$

The matrix exhibits structure that can be exploited during learning. First we validated the model and obtained the same results reported in Moustafa et al. (2004), the paper was only concerned with planning. We carried out the experiments described below on the Maintenance task. In Experiment III-1 we simulated the optimal policy for use as a benchmark for evaluating learning performance, assuming the agent knows the actual transition probabilities (Equation 7.11) from start. Unlike Experiment III-1, Experiments III-2 and III-3 involve learning as the agent has no knowledge of the actual transition probabilities. In experiment III-2 we applied the OMS algorithm with a uniform pro-forma prior. Finally in experiment III-3 we applied order restriction on state transition probabilities given as simple order (SO) in which, at any given state $s \in S$ under action $a \in A$, $p^a_{si} \geq p^a_{sj}$ for all $i \leq j$. The pool adjacent violators algorithm (PAVA) was used in experiment III-3 for Bayesian Inference and to control exploration we use COMS. The set-up is summarised in Table 7.2. The result of the experiments is shown in Figure 7.7. The results are averaged over fifty trials. The shaded area is estimates of standard deviation showing standard error around the mean.

The agent is able to learn the task using the OMS algorithm although it took over 50000 steps to converge to optimal solution. Performance of the COMS is quite remarkable, it shows significant

	III-1	III-2	III-3
Domain	ITD	ITD	ITD
Task	$\gamma = 0.99$	$\gamma = 0.99$	$\gamma = 0.99$
Learning Algorithm	Optimal Policy	OMS [$\alpha = 0.95\%$]	COMS [$\alpha = 0.95\%$]
Pro-Forma Prior	-	Uniform (1)	Uniform (1)
Pre-Prior Constraint	-	-	SO
Inference Method	-	-	PAVA

Table 7.2: Summary of the Set up for Experiment III in the Intervention Timing Doman (ITD)

improvement over the OMS Algorithm on this task. These differences can be seen in Figure 7.8 to be statistically significant. Much of the improvements is due to the transition constraints which is indicative of the potential benefits of our approach in practical applications. For example, the notions of order restrictions and increasing / decreasing failure rates are known to engineers and scientists and the knowledge that such information could optimise learning performance is significant.

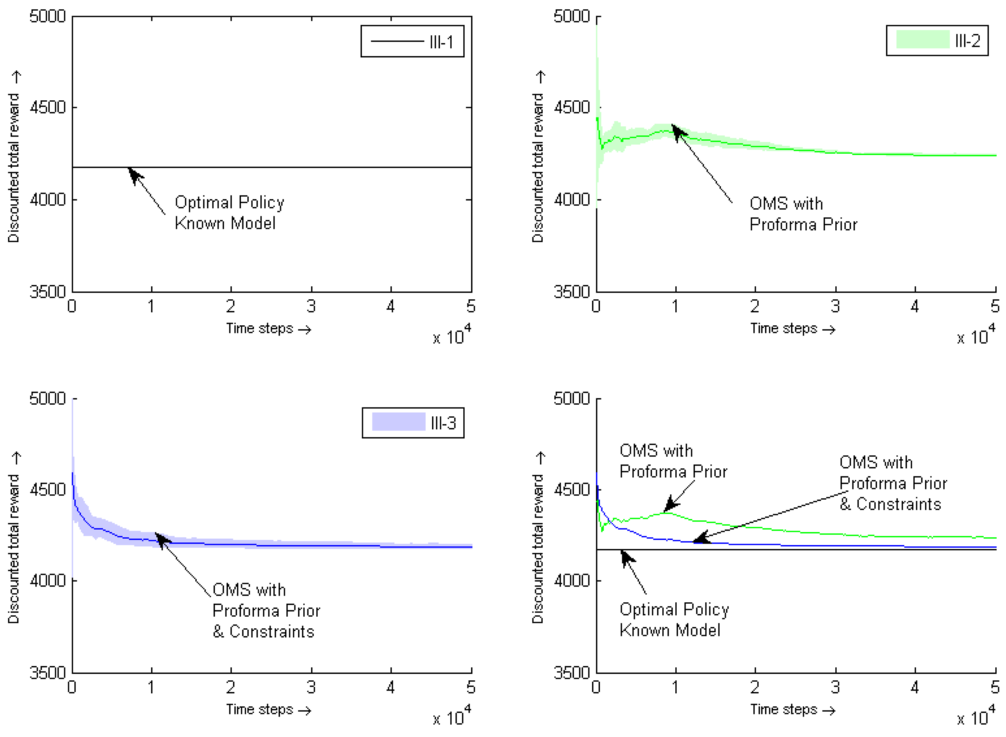


Figure 7.7: Discounted Total Rewards (Costs) over time for Experiment III

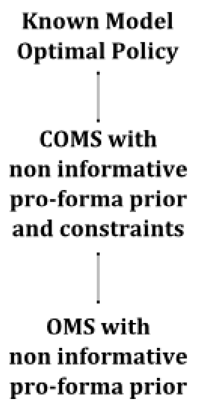

Known Model Optimal Policy
COMS with non informative pro-forma prior and constraints
OMS with non informative pro-forma prior

Figure 7.8: Ordering for discounted cumulative reward for Experiments III. The solid lines indicate statistical significance at 95% level.

7.7 Experiment IV: Additional Experiments

We carried out experiments on a 15×15 grid to study the performance of the COMS algorithm on a larger state space. The grid is shown in Figure 7.9(a). An agent on the grid is faced with the task of

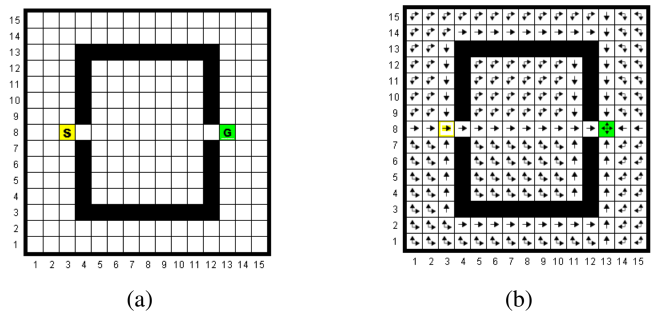


Figure 7.9: The 15×15 Grid World. The Grid contains one start cell and one goal cell. The black cell is inaccessible. a) The 15×15 Grid. b) Optimal Policy for 0.94 probability of success.

moving from a starting position, cell (3,8), to a goal position, cell (13,8). The agent can move in any of the free space in orthogonal directions using the usual four primitive actions North, East, South and West. Each action succeeds in moving the agent in the chosen direction with a probability of 0.94 or fails and remains in the current cell with a probability 0.06. If the agent tries to take an action going through an obstacle, it bounces back to the current cell. The reward function is specified in

Equation 7.12.

$$r_s = \begin{cases} +10.0 & \text{for goal state,} \\ +0.0 & \text{for accessible cells,} \\ -1.0 & \text{penalty for attempting to go through an obstacle.} \end{cases} \tag{7.12}$$

We carried out three experiments IV-1, IV-2, and IV-3. The set up of the experiments are summarised in Table 7.3. Experiment IV-1 is the baseline in which the optimal policy is applied throughout given that the actual transition model is known. In contrast to Experiment IV-1, Experiments IV-2 and IV-3 involve learning. The agent uses the COMS algorithm in Experiment IV-2 with a pro-forma prior that is set to 1 everywhere and a knowledge of absolute transition constraints that sets the probability of successful actions to 0.94. The agent needs to learn the other effects of its actions. The agent uses a model free Q Learning algorithm in Experiment IV-3. The optimal policies are shown in Figure

	IV-1	IV-2	IV-3
Domain	Grid Task 0.94 success probability	Grid Task 0.94 success probability	Grid Task 0.94 success probability
Task (Reward)	15×15 Grid $\gamma = 0.99$ Equation 7.12	15×15 Grid $\gamma = 0.99$ Equation 7.12	15×15 Grid $\gamma = 0.99$ Equation 7.12
Learning Algorithm	Optimal Policy Known Model	COMS $\alpha = 95\%$	Q Learning -
Pro-Forma Prior	-	Uniform(1)	-
Pre-Prior Constraint	-	Absolute Probability of Successful action = 0.94	-

Table 7.3: Set up for IV in a 15×15 grid world with 0.94 probability of successful action. IV consists of three individual experiments - the baseline (IV-1) in which the actual transition model is available to the agent and an optimal policy is used from start, IV-2 involves learning using OMS with a non-informative prior information and an Absolute Constraint, and IV-3 uses model free Q learning algorithm.

7.9(b). Figure 7.10 shows the discounted total reward acquired by the agent over time in each of the experiments. As we would expect, the agent is able to learn the task in both IV-2 and IV-3. The COMS algorithm (IV-2) performed better than the Q-Learning algorithm (IV-3), the former exploiting the knowledge of the absolute constraints.

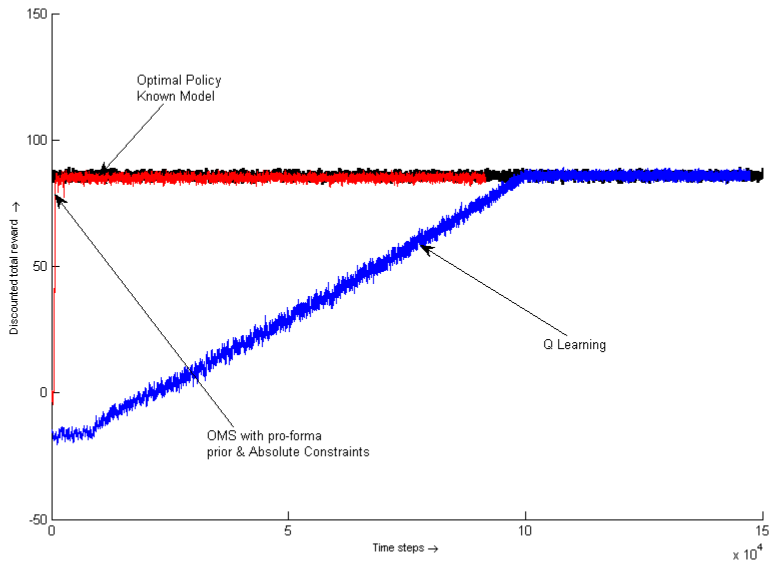


Figure 7.10: Discounted Total Rewards over time for Experiment IV

7.8 Chapter Summary

We described in this chapter algorithms for reasoning with transition constraints and showed how we applied them to improve learning performance. The contributions in this chapter closely relate to those of chapter four where we first covered the subject of constraints and optimistic model selection. We showed in this chapter that there are many ways in which an agent can reason with constraints and the effect of constraints on learning performance could be significant. In the next chapter we will present another perspective on how transition constraints can be used when transferring prior information in MDPs.

8 Accounting for Historical Data

In this chapter, we focus on how an agent can incorporate historical data into prior distribution of a new task. The new task is an MDP with unknown or partially known transition probabilities and the historical data constitutes pre-prior information available to the agent before the commencement of learning on the new task. We will describe in this chapter methods of constructing a prior distribution for the new task that integrates the historical data with other available information. The construction is accomplished through power prior Bayesian analysis. We start by describing the basics of power prior distributions. We then describe how power priors can be used to construct priors for transition models. We use a model of the grid world task and the maintenance intervention task to demonstrate the power prior approach.

8.1 Introduction

When an agent is faced with a task that involves learning and has at its disposal historical data from related tasks the agent should incorporate the historical data in a way that will improve learning performance. This can be done by weighing the historical data relative to its other beliefs when calculating prior distribution for the model parameters of the new task. As we discussed in Chapter five, a way to accomplish this is through power prior Bayesian analysis (Ibrahim & Chen 2000).

The power prior approach provides a useful class of informative priors for Bayesian inference. The basic idea of a power prior is to introduce into the inference algorithm a relative precision parameter that controls the influence of the historical data on the new task. The power prior is constructed by raising the *likelihood function* of the historical data to a suitable power to discount the historical data relative to that of the new task.

Following the seminal work of Ibrahim, Chen and Sinha (2003) in their extensive study of the theoretical properties of power priors, the power prior approach has gained popularity and has been applied to a wide variety of Bayesian inference problems. Our contribution in this chapter is to extend the power prior approach to Bayesian inference in unknown or partially known Markov decision processes. We also present a method for acquiring the precision parameter and demonstrate the benefit of our power prior method on the maintenance intervention task and the grid world task we studied in the previous chapter.

The remainder of the chapter is organised as follows. In Section 8.2 we describe the idea of power priors. We present our extension of power priors to process models in Section 8.2.1 and describe in Section 8.2.2 how we calculate the relative precision parameter. Experiments illustrating the performance of the power prior approach are reported in Sections 8.3 and 8.4. The chapter ends in Section 8.5 with a summary.

8.2 Power Priors

In this section we briefly describe power prior Bayesian analysis for single and multiple historical data sets. Let us start by considering a model that has unknown parameters θ. We would like to incorporate historical data, denoted by $D^{\diamond}$, when making inference about θ. We assume that θ follows a probability distribution and that, given θ, the historical data $D^{\diamond}$ and current data D are independent random samples from an exponential family.

Let $L(\theta|D^{\diamond})$ be the likelihood function of θ based on the historical data. Ibrahim & Chen (2000) define the power prior of θ as:

$$f(\theta|D^{\diamond},\delta) \propto L(\theta|D^{\diamond})^{\delta} f(\theta) \tag{8.1}$$

in which, $f(\theta)$ is the pro-forma prior distribution about θ that is specified by the agent before any historical data is made available and δ is a relative precision parameter that weights the historical data relative to the likelihood of the current task.

The precision parameter δ can be assumed fixed. It is constrained to lie between 0 and 1. The boundary values of δ, that is 0 and 1, give two interesting cases. The contribution of historical data to the power prior is nil when $\delta = 0$. This means that the whole of the power prior equates to the agent's pro-forma prior distribution. On the other hand, the case of $\delta = 1$ results in full incorporation of the historical data. That is, equal weight is given to both the likelihood $L(\theta|D^{\diamond})$ and the pro-forma prior distribution $f(\theta)$.

Taking account of the marginal probability of the historical data, the posterior probability of θ given the historical data is expressed for a fixed δ as follows:

$$f(\theta|D^{\diamond},\delta) = \frac{L(\theta|D^{\diamond})^{\delta} f(\theta)}{\int_{\Theta} L(\theta|D^{\diamond})^{\delta} f(\theta) d\theta} \tag{8.2}$$

Notice that in Equations 8.1 and 8.2 the relative precision parameter is assumed fixed and known. Typically the precision parameter is not necessarily pre-determined and more flexibility may be achieved by making the parameter a random variable. The power prior $f(\theta|D^{\diamond},\delta)$ in Equations 8.1 and 8.2 can be extended by specifying a prior distribution for δ and including the distribution in a joint power prior of (θ,δ). Ibrahim and Chen(2000) proposed a joint power prior distribution for (θ,δ) of the form

$$f(\theta,\delta|D^{\diamond}) \propto L(\theta|D^{\diamond})^{\delta} f(\theta) f(\delta) \tag{8.3}$$

in which $f(\delta)$ is the prior distribution for the precision parameter δ taken as a random variable. In the same vein, Equation 8.2 can be extended as follows,

$$f(\theta,\delta|D^{\diamond}) = \frac{L(\theta|D^{\diamond})^{\delta} f(\theta) f(\delta)}{\int_0^1 \int_{\Theta} L(\theta|D^{\diamond})^{\delta} f(\theta) f(\delta) d\theta d\delta} \tag{8.4}$$

There are advantages associated with making δ a random variable instead of a fixed variable. First, a random δ allows the tails of the marginal distribution of θ to be heavier than the tails with δ fixed, and this may be more desirable. Secondly, a random δ brings about flexibility in expressing uncertainty associated with δ via a prior distribution.

A natural prior for δ would be a Beta(α,β) distribution or, since $0 \le \delta \le 1$, simply a uniform distribution. The learning agent may influence the prior weight on the historical data by adjusting the hyper parameters α,β that specify the prior distribution for δ. The main reason often cited for taking a beta prior for δ is that the beta prior is quite simple and natural (Ibrahim & Chen 2000, Chen, Ibrahim & Shao 2000, Ibrahim et al. 2003, Duan 2005, Duan, Ye & Smith 2006). It is suggested that several sets of hyper parameters should be used and sensitivity analyses conducted when inferring the precision parameter (Chen et al. 2000).

When there are multiple historical data sets, it is possible to incorporate the whole of the historical data sets in the power prior formalism by generalising the specifications we described above for single historical data. Suppose there are K historical data sets, $D^{\diamond} = \{D_1^{\diamond}, D_2^{\diamond} \ldots, D_k^{\diamond}, \ldots, D_K^{\diamond}\}$ where $D_k^{\diamond}$ is the historical data set for the kth task. Chen et al. (2000) suggested defining a different weight parameter δ_k for the kth historical data set and taking the δ_k's to be independent identically distributed Beta random variables with hyper parameters (α,β). Let $\vec{\delta} = (\delta_1, \ldots, \delta_k, \ldots, \delta_K)$, then the power prior for multiple historical data sets takes the form

$$f(\theta,\vec{\delta}|D^{\diamond}) \propto \frac{(\prod_{k=1}^{K} L(\theta|D_k^{\diamond})^{\delta_k} f(\delta_k|\alpha,\beta)) f(\theta)}{\int_{\Theta} (\prod_{k=1}^{K} L(\theta|D_k^{\diamond})^{\delta_k}) f(\theta) d\theta} \tag{8.5}$$

This framework could accommodate potential heterogeneity among several historical data sets, and hence the role of historical data can be more accurately evaluated. Modifications (Duan 2005, Neuenschwander, Branson & Spiegelhalter 2009, Gajewski 2010) to the power prior formulations of Equation 8.3 and its extensions has been reported to resolve issues raised around: a) the tendency for δ to be close to zero meaning that much of a historical data set is not used, and b) the suggestions that the resulting power prior could be improper. Our focus in this thesis is primarily on the original power prior formulations, Equations 8.1 and 8.2.

8.2.1 Power Priors for Process Models

We now turn to how historical data can be incorporated in the quantification of prior distributions for process models. For ease of exposition we develop a power prior and the consequent posterior with only one historical data set. Suppose we are interested in a process model with unknown transition parameters P. We would like to incorporate historical data, denoted by $D^{\diamond}$, when making inference about P. The agent has a pro-forma prior information matrix M^{ω} that it formulated for P before the historical data is made available. We assume that $f(P|M^{\omega})$ follows a probability distribution and that, given P, the historical data $D^{\diamond}$ and the pro-forma prior information M^{ω} are independent quantities from an exponential family. Specifically, in line with previous chapters, we assume that P follows a multinomial distribution and the pro-forma prior distribution $f(P|M^{\omega})$ is a product of Dirichlet densities.

Let $L(P|D^{\diamond})$ be the likelihood function of the transition model P based on the historical data. Following Ibrahim and Chen (2000), we define the power prior of P as:

$$f(P|D^{\diamond},\delta,M^{\omega}) \propto L(P|D^{\diamond})^{\delta} f(P|M^{\omega}) \tag{8.6}$$

in which, $f(P|M^{\omega})$ is the initial prior distribution about P before any historical data is made available and δ is a relative precision parameter that weights the historical data relative to the likelihood of the current task. We consider the parameter δ to be fixed and constrained to lie between 0 and 1.

Taking account of the marginal probability of the historical data, the posterior probability of P

given the historical data is expressed for a fixed δ as follows

$$f(P|D^{\diamond},\delta,M^{\omega}) = \frac{L(P|D^{\diamond})^{\delta} f(P|M^{\omega})}{\int_{\mathbb{P}} L(P|D^{\diamond})^{\delta} f(P|M^{\omega}) dP} \tag{8.7}$$

where $f(P|M^{\omega})$ is the pro-forma prior distribution and $L(P|D^{\diamond})$ is the multinomial likelihood of P based on the historical data.

Consider a state s of the MDP and assume that the historical transition count data vector $\vec{d}_s^{a,\diamond}$ under action a in state s is a multinomial with a fixed number $|S|$ of possible outcomes (successor states) and $|\vec{d}_s^{a,\diamond}|$ independent trials. The pro-forma prior distribution for state s under action a, $\vec{m}_s^{a,\omega}$, is Dirichlet. The posterior probability of $\vec{p}_s^a$ given the historical data is expressed for a fixed δ_s as follows:

$$\begin{aligned} f(\vec{p}_s^a|\vec{d}_s^{a,\diamond},\delta_s,\vec{m}_s^{a,\omega}) &\propto \left(\frac{|\vec{d}_s^{a,\diamond}|!}{d_{s1}^{a,\diamond}! \times d_{s2}^{a,\diamond}! \times \ldots \times d_{s|S|}^{a,\diamond}!} \prod_{s'} (p_{ss'}^a)^{d_{ss'}^{a,\diamond}} \right)^{\delta_s} f(\vec{p}_s^a|\vec{m}_s^{a,\omega}) \\ &= \frac{1}{Z(\vec{m}_s^{a,\omega})} \left(\frac{|\vec{d}_s^{a,\diamond}|!}{d_{s1}^{a,\diamond}! \times d_{s2}^{a,\diamond}! \times \ldots \times d_{s|S|}^{a,\diamond}!} \prod_{s'} (p_{ss'}^a)^{d_{ss'}^{a,\diamond}} \right)^{\delta_s} \prod_{s'} (p_{ss'}^a)^{m_{ss'}^{a,\omega}-1} \end{aligned}$$

where $Z(\vec{m}_s^{a,\omega})$ is the normalisation constant.

Example 8.1 *Consider a two state MDP whose transition probabilities $\vec{p}_1^a$ from state 1 under action a are unknown. Assume historical data is available as transition counts [50, 15] for state transitions p_{11}^a and p_{12}^a respectively. The agent has a pro-forma prior distribution with Dirichlet parameters [40, 60]. We calculate the PDF for four example values of $\vec{p}_1^a$ on a range of precision parameters, using power prior Bayesian analysis. The result is shown in Figure 8.1 below. As the precision parameter varies so does the likelihood of any particular model varies.*

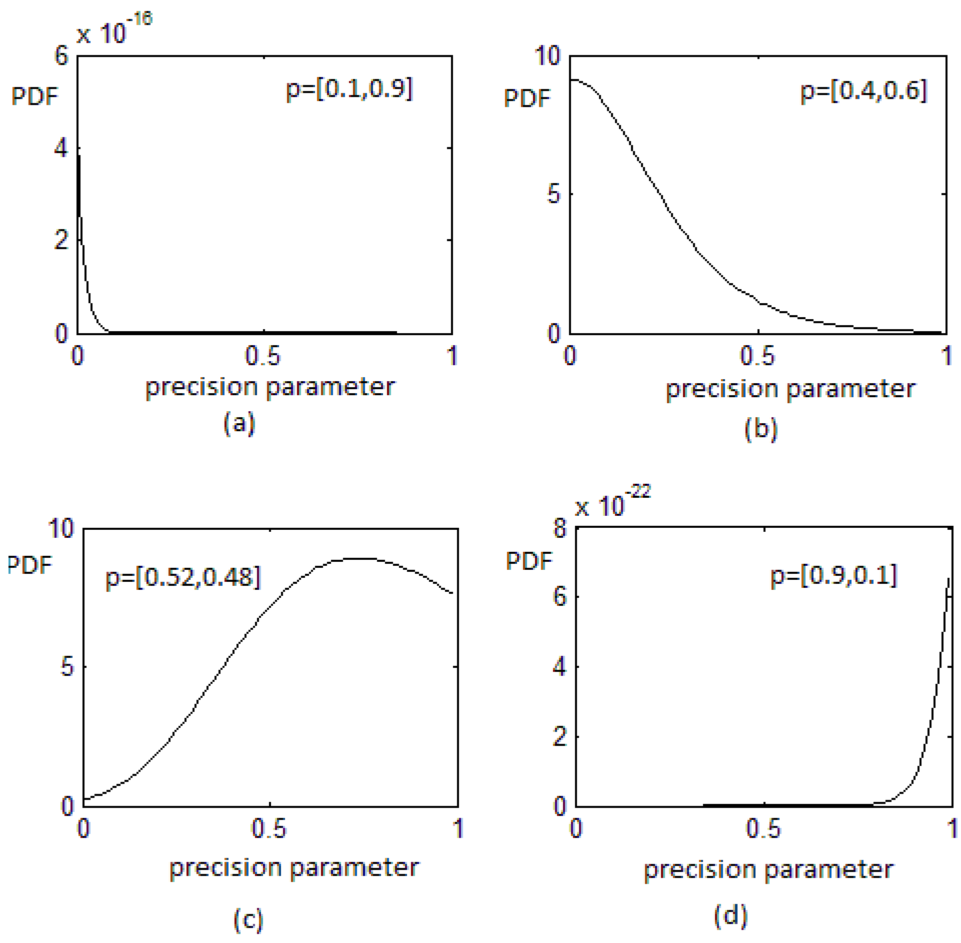


Figure 8.1: Plot of PDF versus precision parameter for power prior with historical data set to [50, 15] and a Dirichlet distributed pro-forma prior with parameters [40, 60]. The actual transition model for the task is assumed to be the p associated with each graph. The precision parameter varies for each of the four transition models.

8.2.2 Acquiring the Precision Parameter

To apply the power prior approach to a new task, a salient problem is how to establish a relative precision parameter δ at the start of learning when no current data is available. We address this problem in this section and show how transition constraints can be used to acquire a power prior precision parameter. Essentially, the historical data is valued to the extent that it matches the user knowledge about constraints on the transition matrix for the new task. The algorithm works by measuring the degree of match between the historical data and *imaginary data* drawn from the user specified constraints. The steps for computing the precision parameter is as follows.

1. We use the transition constraints C_Ω as a generative model for drawing random samples of state transitions. The resulting samples, which we refer to as *imaginary data*, are then used to infer parameters of a Dirichlet distribution $\vec{m}_s^{a,C_\Omega}$ in state s under action a.

2. We create a parameter matrix for a Dirichlet distribution $\vec{m}_s^{a,\diamond}$ from the historical data $\vec{d}_s^{a,\diamond}$.

3. We calculate the distance between the two Dirichlet distributions with parameters $\vec{m}_s^{a,C_\Omega}$ and $\vec{m}_s^{a,\diamond}$ using Equation 3.19 that we presented for bounded divergence measures (Rauber et al. 2008, Roman et al. 2012).

4. The distance measure obtained in step 3 lies between [0, 1] and is returned as the precision parameter δ_s.

8.3 Experiment V: Maintenance Intervention with Power Priors

The model of the maintenance intervention task we introduced in Chapter seven is used in this section to study the performance of the power prior methods. Recall that the intervention task requires an agent to choose between two actions i) *'watchful waiting'* i.e. postpone the decision up to a critical point, and ii) *'intervene'*. The maintenance task is adapted from Moustafa, Maksoud & Sadek (2004). It consists of three actions, namely *do-nothing*, *minimal maintenance*, and *replacement*. The do-nothing action corresponds to watchful waiting whilst the other two, i.e. *minimal maintenance* and *replacement*, are interventions. The transition matrix for the deteriorating system is shown in

Equation 8.8. The matrix exhibits structure that can be exploited during learning.

$$P = \begin{bmatrix} 0.28 & 0.20 & 0.15 & 0.12 & 0.09 & 0.07 & 0.05 & 0.03 & 0.01 \\ & 0.40 & 0.19 & 0.15 & 0.10 & 0.07 & 0.05 & 0.03 & 0.01 \\ & & 0.32 & 0.23 & 0.17 & 0.12 & 0.08 & 0.05 & 0.03 \\ & & & 0.32 & 0.26 & 0.20 & 0.13 & 0.07 & 0.02 \\ & & & & 0.56 & 0.20 & 0.14 & 0.08 & 0.02 \\ & & & & & 0.60 & 0.25 & 0.13 & 0.02 \\ & & & & & & 0.65 & 0.22 & 0.13 \\ & & & & & & & 0.70 & 0.30 \\ & & & & & & & & 1.0 \end{bmatrix} \tag{8.8}$$

In the experiments we carry out in this section (i.e. Experiment V), we assume the transition probabilities are unknown but that the agent is presented with prior information in the form of historical data and constraints. We demonstrate the benefits of the power prior approach when applied to this task.

8.3.1 Experiment Setup

We keep all the parameters of the maintenance task the same as first introduced in Chapter seven. We perform two sets of experiments: V-A and V-B. The two sets of experiments differ in the way we generate the imaginary data used to acquire the power prior precision parameters. In experiment V-A, we use samples obtained directly from the actual transition probability matrix as the imaginary data. In experiment V-B the imaginary data are generated using the transition constraints. In both sets of experiments we assume that the agent has a uniform pro-forma prior distribution expressed in Equation 8.9.

$$M^{\omega} = \begin{bmatrix} 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ & & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ & & & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ & & & & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ & & & & & 1.0 & 1.0 & 1.0 & 1.0 \\ & & & & & & 1.0 & 1.0 & 1.0 \\ & & & & & & & 1.0 & 1.0 \\ & & & & & & & & 1.0 \end{bmatrix} \tag{8.9}$$

In addition, we assume that in both sets of experiments the agent is presented with transition constraints associated with the domain. The constraints consist of: i) order restriction on state transition probabilities given as simple order (SO) in which, at any given state $s \in S$ under action $a \in A$, $p^a_{si} \geq p^a_{sj}$ for all $i \leq j$, and ii) significant inequality constraints on transition probabilities in states $5, 6, 7,$ and 8, such that $p^a_{ss} \geq 0.5$ where $s = [5, 6, 7, 8]$ $\forall a$. We assume that the following historical data (Equation 8.10) is available to the agent (highlighted in green are entries where the data and the actual transition probability match).

$$D^{\diamond} = 10 \times \begin{bmatrix} 0.28 & 0.2 & 0.15 & 0.12 & 0.09 & 0.07 & 0.05 & 0.03 & 0.01 \\ & 0.40 & 0.19 & 0.15 & 0.1 & 0.07 & 0.05 & 0.03 & 0.01 \\ & & 0.12 & 0.33 & 0.27 & 0.12 & 0.08 & 0.05 & 0.03 \\ & & & 0.02 & 0.16 & 0.32 & 0.20 & 0.08 & 0.22 \\ & & & & 0.36 & 0.12 & 0.14 & 0.26 & 0.12 \\ & & & & & 0.25 & 0.25 & 0.25 & 0.25 \\ & & & & & & 0.05 & 0.65 & 0.30 \\ & & & & & & & 0.10 & 0.90 \\ & & & & & & & & 1.00 \end{bmatrix} \tag{8.10}$$

The details of the experiments are summarised in Table 8.1.

	V-1	V-2	V-3	V-4
Domain	ITD	ITD	ITD	ITD
Task	$\gamma = 0.99$	$\gamma = 0.99$	$\gamma = 0.99$	$\gamma = 0.99$
Learning Algorithm	Optimal Policy Known Model	OMS $\alpha = 95\%$	OMS $\alpha = 95\%$	OMS $\alpha = 95\%$
Pro-Forma Prior	-	Uniform Equation 8.9	Uniform Equation 8.9	Uniform Equation 8.9
Pre-Prior	-	-	Equation 8.10	Equation 8.10
Constraint	-	-	SO & Significant Inequality	SO & Significant Inequality
Inference Method	-	-	Power Prior Precision = 1.0 for historical data	Power Prior Precision acquired

Table 8.1: Summary of the Set up for Experiment V in the Intervention Timing Domain (ITD). The task is the maintenance intervention task we adapted from Moustafa, Maksoud & Sadek (2004). As before, V-1 is the baseline in which the actual transition model is available to the agent and an optimal policy is used from start. V-2, V-3 and V-4 involve model-based learning. V-2 uses OMS with a non-informative proforma prior. V-3 and V-4 allows for power priors.

8.3.2 Results and Discussion

The power prior precision parameters were calculated using the method we described in Section 8.2.2. The result of the calculation is presented in Table 8.2. The result shows that the acquired power prior is sensitive to the imaginary data used to acquire them. As expected, the precision parameters are closer to 1 in states where the historical data closely matches the actual transition probabilities (i.e. where the historical & imaginary data match)

State	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Imaginary Data	Acquired Precision								
V-A-4: Obtained from actual transitions	1.00	0.77	0.61	0.60	0.22	0.21	0.61	0.0	1.0
V-B-4: Obtained from transition constraints	0.32	0.13	0.28	0.63	0.70	0.55	0.32	0.0	1.0

Table 8.2: The power prior precision parameters acquired in Experiments V-A-4 & V-B-4. The precision parameters differ for imaginary data obtained from the actual transition probabilities V-A-4 and those derived from transition constraints V-B-4.

The discounted total rewards obtained for the experiments are shown in Figures 8.2 and 8.3. The results are averaged over fifteen trials. The results show that learning performance improves with the use of power prior and the improvement is higher for the case where the imaginary data is obtained from the actual transitions. The performance of OMS with proforma and pre-prior under power priors using the global precision and local precision settings are better than OMS with proforma prior when the imaginary data was drawn from actual transition probabilities. Only the OMS with proforma and pre-prior under power priors using the local precision settings performed better than OMS with proforma prior in the case of the imaginary data drawn from transition constraints. In addition, using local precision parameters appear better than defaulting to a global precision parameter of 1. Notice that although the pro-forma prior (Equation 8.9) is uniform i.e. non-informative, it nonetheless correctly specifies the successor states and does not violate order constraints on the transitions. These differences are statistically significant as shown in Figure 8.4. As expected, performance under the optimal policy given known model dominates all the other methods.

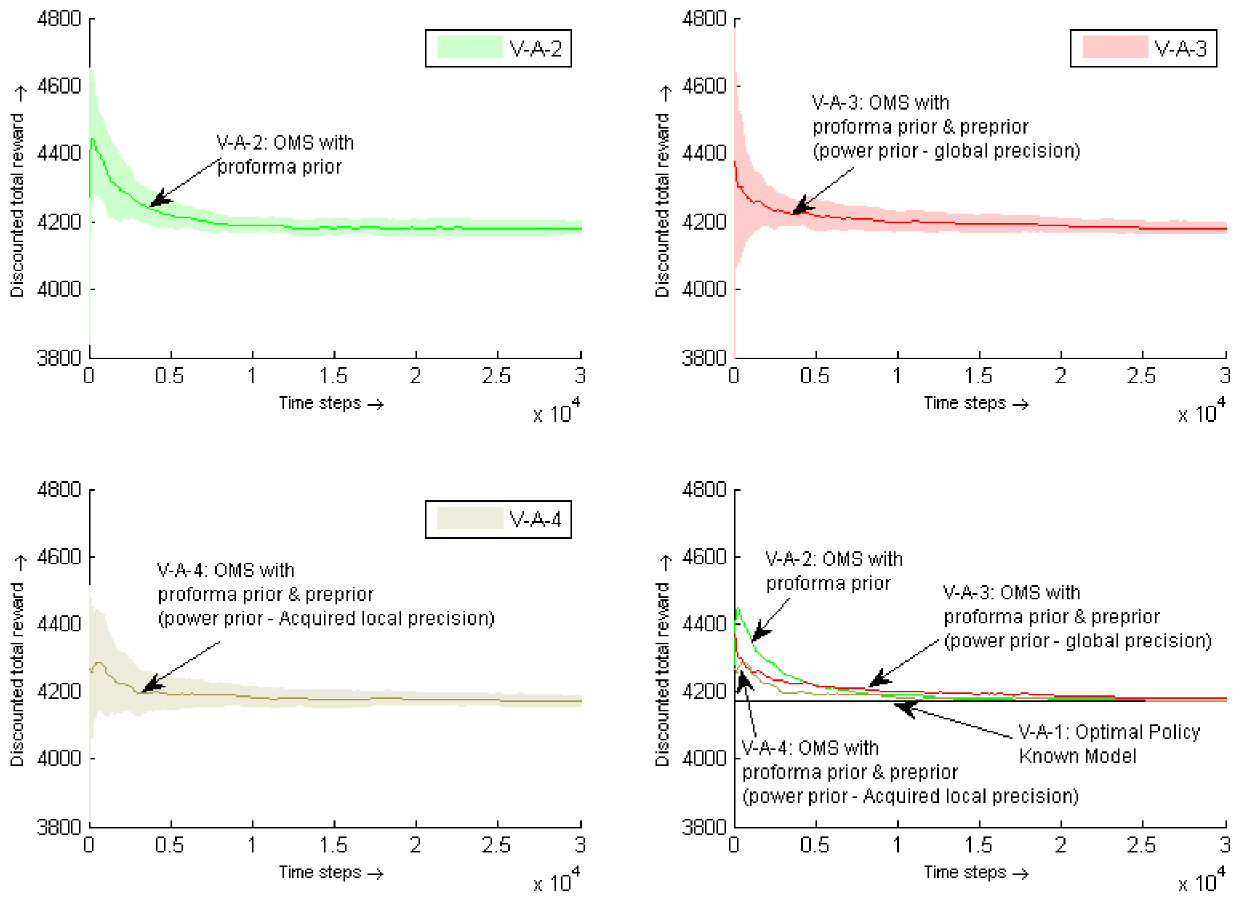


Figure 8.2: Discounted Total Rewards (costs) over time for Experiment V-A. Imaginary data from the actual transition probabilities

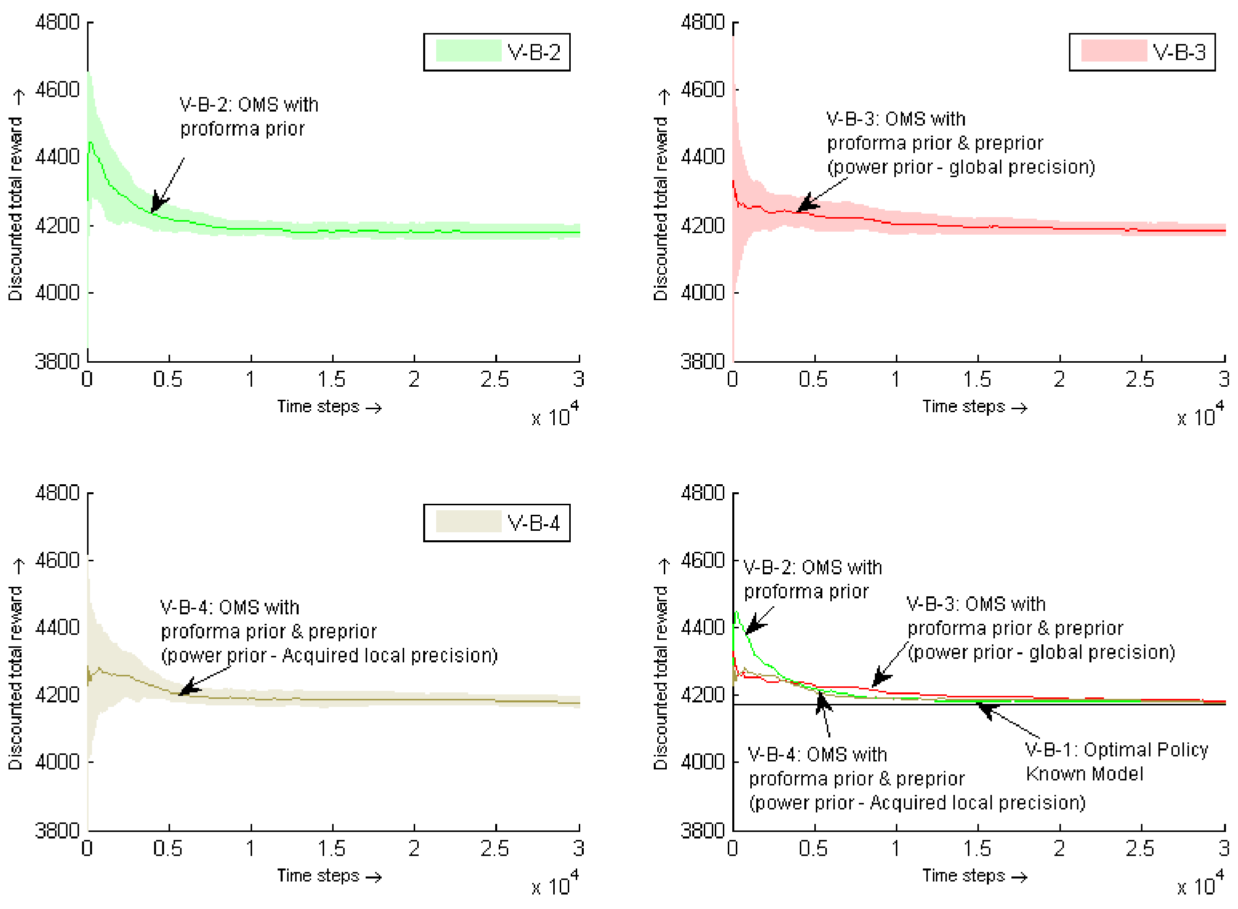
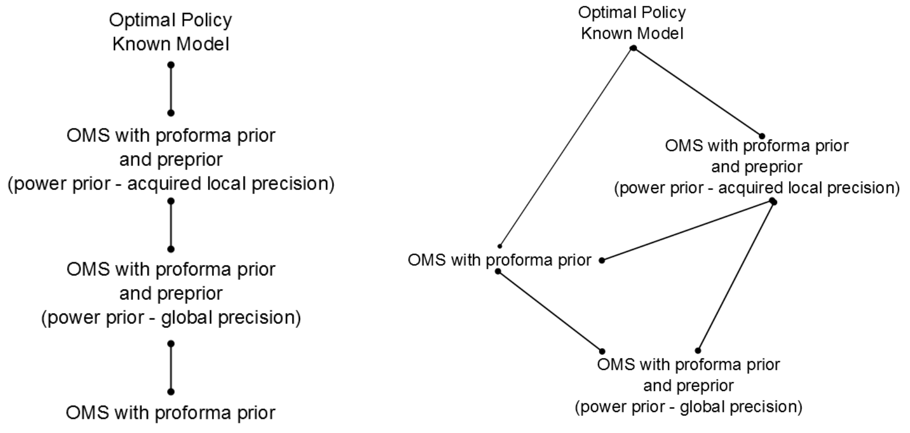


Figure 8.3: Discounted Total Rewards (costs) over time for Experiment V-B. Imaginary data from transition constraints



(a) Imaginary Data from Actual Transitions

(b) Imaginary Data from Transition Constraints

Figure 8.4: Partial dominance ordering for methods used in Experiment V at 95% level of significance.

8.4 Experiment VI: The Grid World with Power Priors

We also study the performance of the power prior methods on the simple grid world we studied in Chapters six and seven, particularly experiments I-A and II. We kept all the parameters of the task as the same, the grid world and the associated optimal policy are reproduced in Figure 8.5.

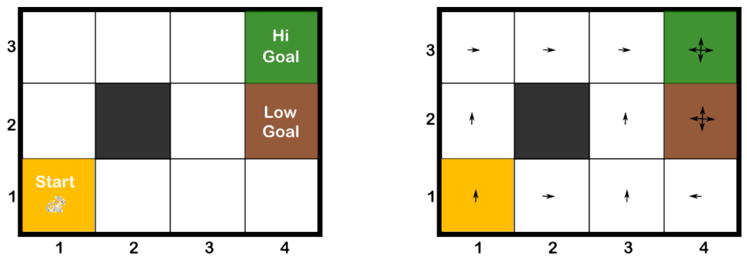


Figure 8.5: The 4×3 Grid World with Transition Constraints. The Grid contains one start cell and two exit cells - High and Low Goals. The black cell is inaccessible. a) The 4×3 Grid. b) Optimal Policy for 0.9 probability of success.

In addition to experiments II-1, II-2, and II-5 of Section 7.5 (Chapter seven), we carried out in this section three further experiments applying power priors in the Grid domain. In the first experiment we use power priors to combine the pre-prior information Equations 6.2 - 6.5 with the non-informative proforma prior using imaginary data drawn from the actual transition probabilities. The resulting local precision parameter δ_s is zero for all states suggesting that the historical data i.e. the pre-prior should not be used. In such a case, the inference on P is not much different from the inference when the historical data is ignored. The second experiment is similar to the first experiment except that the imaginary data from transition constraints is as specified in Equation 8.11, $\forall s, s' \in S$ and $\forall a \in A$.

$$\text{Constraints } \phi_\Omega = \begin{cases} 0.85 \leq p^a_{ss'} \leq 0.95 & \text{transitions arising from successful action} \\ 0.04 \leq p^a_{ss'} \leq 0.06 & \text{transitions arising from wall collisions} \\ 0.0001 \leq p^a_{ss'} \leq 0.0002 & \text{otherwise} \end{cases} \tag{8.11}$$

	VI-1	VI-2	VI-3	VI-4
Domain	Grid Task 0.9 success probability	Grid Task 0.9 success probability	Grid Task 0.9 success probability	Grid Task 0.9 success probability
Task (Reward)	4×3 Grid $\gamma = 0.95$ Equation 6.1	4×3 Grid $\gamma = 0.95$ Equation 6.1	4×3 Grid $\gamma = 0.95$ Equation 6.1	4×3 Grid $\gamma = 0.95$ Equation 6.1
Learning Algorithm	Optimal Policy Known Model	OMS $\alpha = 95\%$	COMS $\alpha = 95\%$	OMS $\alpha = 95\%$
Pro-Forma Prior	-	Uniform(1)	Uniform(1)	Uniform(1)
Pre-Prior	- -	- -	- -	Samples from Transition Constraints
Constraint	-	-	Tolerances Prob of Success $= 0.9 \pm 0.05$	Equation 8.11
Transfer Method	-	-	Sampling from Truncated Dirichlet	Power Prior Precision acquired

Table 8.3: Set up for Experiment VI.

Like the first experiment, the resulting local precision parameter δ_s is zero for all states, again suggesting that the historical data i.e. the pre-prior should not be used. Finally, in the third experiment, we use the imaginary data from the transition constraints as the historical data and transferred information via power priors with δ_s set to 1 for all states. This means that the agent fully incorporates the historical data into its prior distribution with equal weight given to both the likelihood of the historical data and the proforma prior distribution. The set-up for this third experiment and those relating to II-1, II-2, and II-5 of Section 7.5 is shown in Table 8.3.

The discounted total reward obtained is reported in Figure 8.6. As expected, the use of optimal policy derived from known model dominates all the other methods.

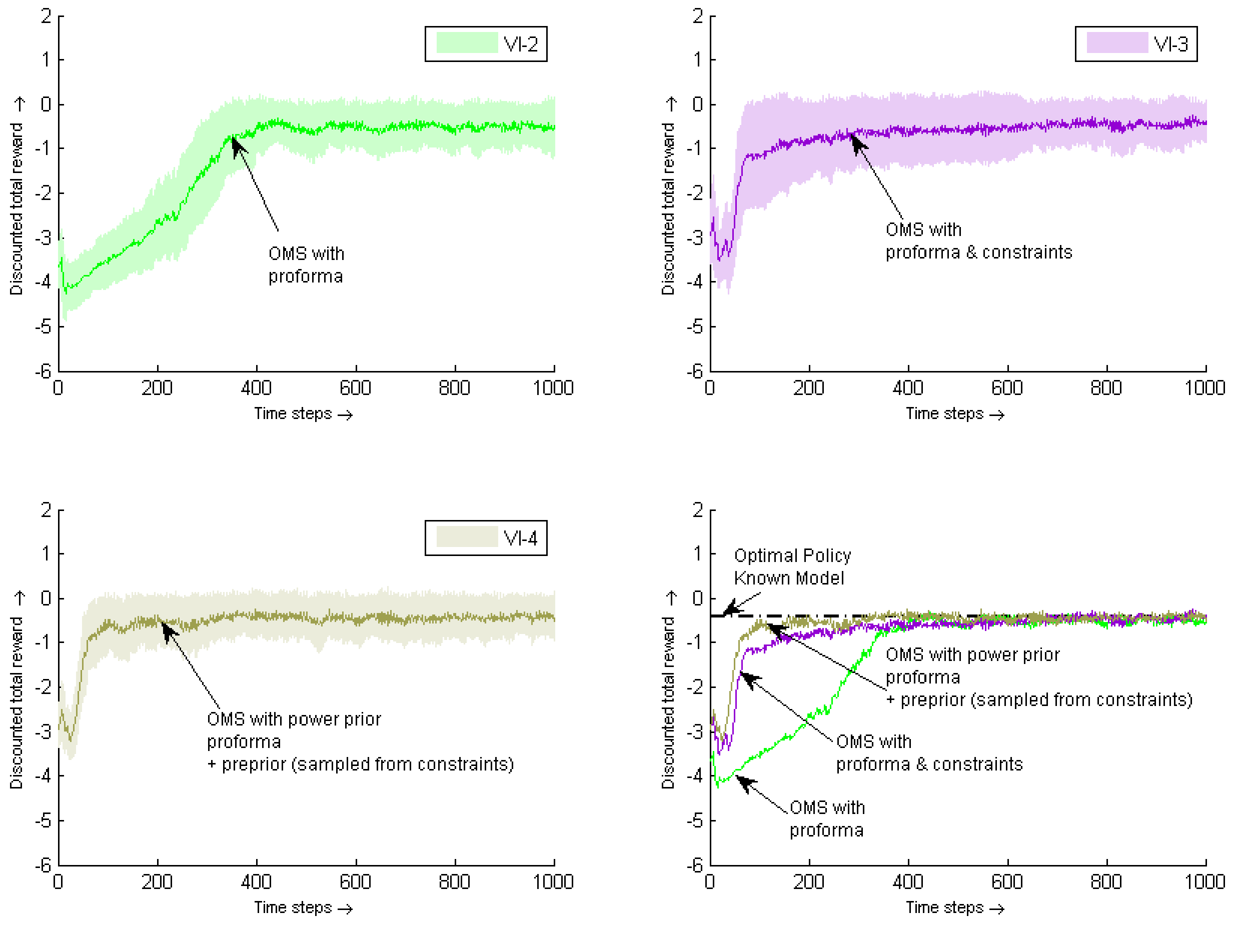


Figure 8.6: Discounted Total Rewards over time for Experiment VI.

In addition, the method using power priors performed better than the approach based on OMS with non-informative proforma. We also observed that the OMS with power prior (proforma & pre-prior from transition constraints) [VI-4] performed slightly better than the approach based on

OMS with proforma and transition constraints [VI-3]. One reason for this is the slight differences in the specification of the transition constraints used in VI-3 and VI-4. The differences in the total discounted rewards are significant as shown in Figure 8.7.

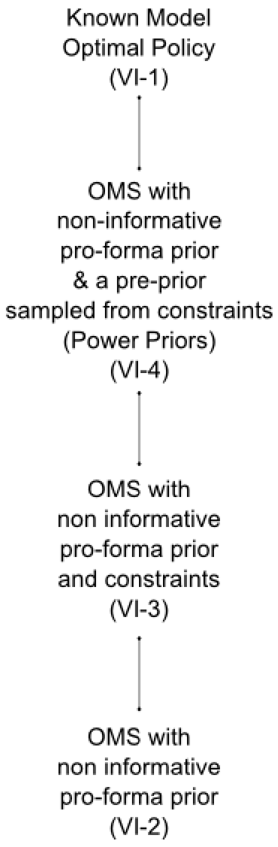


Figure 8.7: Partial dominance ordering for methods used in Experiment VI at 95% level of significance.

8.5 Chapter Summary

This chapter has presented a method of incorporating historical data into specifications of prior distribution for process models of new tasks. This is done by extending the power prior approach of (Ibrahim & Chen 2000) to Bayes adaptive Markov decision processes. Power priors allow us to control the influence of historical data relative to new observations by using a relative precision parameter to down weight the historical data. This chapter addressed the problem of how to set the precision parameter by assessing the degree of match between the historical data and imaginary data drawn from the user specified constraints. The main points are:

- When an agent has at its disposal historical data from related tasks the agent should appropriately incorporate the historical data into its learning procedure. It is convenient to achieve this through power prior Bayesian analysis. Power priors use a relative precision parameter to control the influence of historical data relative to new observations.
- In the context of Bayes Adaptive Markov decision processes, power priors are constructed by raising the multinomial *likelihood function* of the transition model parameters, based on the historical data, to a suitable power (i.e. the precision parameter) to discount the historical data relative to the pro-forma Dirichlet distribution of the learning agent.
- A salient question is how to set the precision parameter. The chapter shows how the constraints on process dynamics can be used to acquire a precision parameter. A useful answer is to set the precision parameter to the degree of match between the historical data and imaginary data drawn from the user specified constraints. In a sense, the historical data is valued to the extent that it matches the user knowledge about constraints on the transition matrix for the new task.
- The quantified power prior can be used to augment existing exploration control algorithms allowing the agent to exploit the presence of historical data when learning a new task from reinforcement.
- Although the power prior methodology we present in this chapter is quite general, the most natural specification of historical data arises when the raw data from a similar previous study are available.

The power prior approach presents us with an alternative easier way of dealing with constraints through the generative process. However the approach raises some issues for future work including, importantly, i) how to detect automatically whether or not the imaginary data is of sufficient quantity and quality, and ii) how much does it harm when similarity is weaker.

9

Leveraging with Process Templates

The algorithms and experiments presented so far in this thesis have all used flat, enumerated state, representations as the basis of model transfer. The problem that we attend to in this chapter is of a slightly different nature. We will assume that a reference pattern (template) from donor models of source tasks is available to a learning agent and the agent has to decide how best to use the template to leverage learning on a recipient model of a new target task. A pattern is conceptualised simply as a set of 'locally-defined' features of the donor models, each of which can be located independently of the rest by analysing small regions of the model space.

9.1 Introduction

A template is a reusable definition of a part of task's process model. It is considered as a repository of process knowledge which holds information that can be transferred to new tasks. In some cases, the template can be conceptualised as a historical datum, created for its value as a repository of previous experience without regard to its potentials for reuse in future tasks. In most cases however templates are created to be reused in future tasks, created specifically to capture the knowledge in a domain. A good template should easily facilitate knowledge transfer.

In this chapter, we present an algorithm for learning a new task using information contained in a template. The information is initially collected as ground truth for state-transitions in various tasks. For a new and different task, our template-based approach would only need to add some follow-up experiences into the transferred information rather than starting from scratch. In the next section, we briefly summarise related work on the use templates in reinforcement learning. In Section 9.3 we describe the model assumptions and characteristics of the Markov decision processes that define our templates. A method of using the templates in the context of transfer learning is presented in Section 9.4 and in Section 9.5 we illustrate through experiments the benefits of using the templates. The chapter ends with a summary in Section 9.6.

9.2 Use of Templates in Reinforcement Learning

There is a rich collection of publications on the use of templates in artificial intelligence, most especially in the field of machine vision using, for example, rigid and deformable templates (Jain, Zhong & Lakshmanan 1996). A rigid template is not capable of adapting itself to data through transformation other than by rotations and scaling whilst a deformable template, on the other hand, is able to *'deform'* itself by adapting to data through transformations that are jointly more complex than transition, rotations and scaling. Deformable templates are usually of two types: a) *'free-form'* i.e. there is no global structure to the template, the template is constrained by only local continuity, and b) *'parametric'* in which there is some prior information on the possible deformations which can be encoded using a small number of parameters (Jain et al. 1996, Zhong, Jain & Dubuisson-Jolly 2000).

A number of researchers have reported their use of templates for information transfer and reuse in reinforcement learning including, for example, the use of local state descriptions (Lin & Li 2003, Stolle & Atkeson 2007), static templates in the form of snakes (Drummond 1997), parameterised relational templates (Proper & Tadepalli 2009, Proper & Tadepalli 2010), and family of value surfaces (Shannon, Santiago & George 2003) as viable template-based representations of transferable knowledge in reinforcement learning.

Lin & Li (2003) presented an algorithm called Feature-SARSA for learning to optimally solve Markov decision problems in domains that has particular characteristics attributable to each state such that an agent might be able to take advantage of those characteristics to direct future learning. They define a state feature function that generates local state features in each state. To control exploration, a weight function is used in conjunction with the feature function to adjust current policy to the actions worth exploring.

Silver, Sutton & Müller (2007) in their application of reinforcement learning to the game of Go employed a template based representation. They specified a template as a configuration of features for a rectangular region of the board. A basic template specifies a colour (black, white or empty) for each intersection within the rectangle. The template is matched in a given position if the rectangle on the board contains exactly the same configuration as the template. A local shape feature simply returns a binary value indicating whether the template matches the current position. They use weight sharing to exploit several symmetries of the Go board. All rotationally and reflectionally symmetric shapes share the same weights. Colour symmetry is represented by inverting the colour of all stones when evaluating a white move. These invariances define the class of location dependent shapes. A second class of configuration independent shapes also incorporate translation invariance. Weights are shared between all local shape features that have the same template, regardless of its location on the board. For each type of shape, all possible templates are exhaustively enumerated to give a shape set.

In considering the mechanics of reinforcement learning and adaptive dynamic programming (ADP), Shannon et al. (2003) suggested that the appropriate site for integration of a priori template-based information is the critic. They specifically looked at the critic as an approximator of the value function from an ADP framework and opined that a priori information serves to constrain the family of functions over which the critic must search to find an accurate approximation. They describe this family of functions as being composed of a single template and set of perceptual transformations of that template. Illustrating with an example of a family of value surfaces described by a quadratic surface, an isotropic quadratic surface is in their framework considered a template, while the set of parameterized transformations which connect this surface to the state space over which it is to be evaluated are perceptual transformations of the template. In general, templates facilitate abstraction and the knowledge of the template could be exploited in learning.

9.3 Model Assumptions and Characteristics

A task that is to be performed using templates must first be decomposed into a set of templates that includes all the actions needed to perform the task. This can be done by an expert off-line and made available to the agent or the learning agent can discover the templates by itself automatically during its interactions with the environment. In this work, we assume the templates are defined offline by an expert and supplied to the learning agent. In this section we describe the assumptions and characteristics of the Markov decision processes that define our templates.

9.3.1 Configuration of Features in the MDP State Space

Assume the states $s \in S$ in an MDP domain υ have a set of K local state features and the features contain transferable information that influences action selection. We consider a grid world domain. For any state in the grid we reference it in two ways. s_i is the enumerate state and there is one to one correspondence between this and the Cartesian coordinate (x_i, y_i). Our templates will exploit geometric regularities in the grid world that are most easily represented by Cartesian coordinates. As a short hand, we will refer to the coordinate (x_i, y_i) as the configuration corresponding to s_i. We sometimes use u_i as a shorthand for (x_i, y_i) so that we are not tied to a fixed dimension for the grid world (one could make it 3D, 4D etc.).

For each distinct configuration $u_i \in \mathbb{U}$ in the state space, suppose there is a feature vector $\vec{f_i} = \{f_{i1}, f_{i2}, \ldots, f_{ik}, \ldots, f_{iK}\}$ that captures the particular characteristics attributable to that configuration u_i, $i = \{1, 2, \ldots |S|\}$ in which f_{ik} is a local state feature. A state s is expressed in terms of the configuration variable u_i and the feature vector $\vec{f_i}$. A mapping from the state to the feature vector, given the configuration variable, namely $\nabla_\upsilon : s \in S|u_i \to \vec{f_i}$ is called a feature function of s. Given two states s_1 and s_2 in domain υ, if $s_1, s_2 \in S$ and $\nabla_\upsilon(s_1|u_1) = \nabla_\upsilon(s_2|u_2)$ then we say that s_1, s_2 have the same local state features and feature vectors, that is $\vec{f_1} = \vec{f_2}$.

Associated with each feature vector is matrix $\vec{F_i} \in \mathbb{F}$ of transition counts. The set of all feature vectors and the associated count matrices constitute the transferable information for the domain. This information is assumed to be available to the agent at the start of learning.

9.3.2 An Illustration using the Grid Domain

The following example from a grid domain illustrates the configuration we described in the previous section. Assume the natural grid coordinates of a grid are represented as (x_i, y_i) in which $x_i \in \{1, 2, \ldots, N_x\}$ $\;y_i \in \{1, 2, \ldots, N_y\}$, such that N_x is the number of cells wide and N_y is the number of cells deep in the grid. We index a cell configuration in the world by i where $i \in \{1, 2, \ldots, N\}$ and in an obstacle free grid $N = N_x \times N_y$ is the total number of states in the world. Figure 9.1 shows an example 5×5 grid. We refer to the example grid as a baby maze. There are occupied (obstacle)

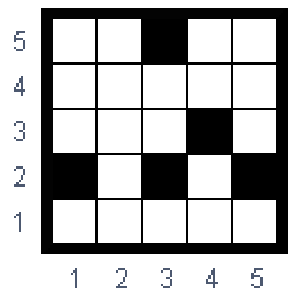


Figure 9.1: A simple 5×5 grid with five occupied (obstacle) cells

cells in the baby maze i.e. cells $(1,2), (3,2), (3,5), (4,3)$ and $(5,2)$. The number of states in the baby maze is $5 \times 5 - 5 = 20$. Other notations used in the model are as follows.

$\vec{f_i}$: is the set of local state features, $\vec{f_i} = \langle status(x_i, y_i), status(x_i, y_i + 1), status(x_i + 1, y_i), status(x_i, y_i - 1), status(x_i - 1, y_i) \rangle$ comprising of the local state features when the agent is at cell configuration (x_i, y_i). We use binary indicators e.g. $status(x_i, y_i) \in [0|1]$ $\forall i$ where values 0 and 1 indicate empty and occupied (obstacle) cells respectively.

$s \in S$: the set of $|S|$ distinct states, $s = \{(x_i, y_i), \vec{f_i}\}$. We set the configuration variable u_i to (x_i, y_i)

making $s = \{u_i, \vec{f_i}\}$.

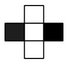
$a \in A$: actions the agent can take in the domain, which in the grid is specified as the usual primitive actions that would move the agent $North(N), South(S), East(E)$, and $West(W)$.

$p^a_{ss'} \in P$: Probability that the process will be in state s' at time $t+1$ given that it is in state s at time t and an action a is taken at time t.

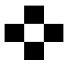
$R : S \times A \longrightarrow \mathbb{R}$: is the expected reward function mapping S into real-valued returns with $r^a_s \in R$ being the reward for performing action a in state s.

γ: the discount rate, $0 \leq \gamma < 1$.

The following is an example of a template at cell configuration $(2,2)$ in the baby maze.



The feature vector for this template is $< status(2,2), status(2,3), status(3,2), status(2,1), status(1,2) >$ which equates to 00101 and bit encoding 5. A full list of the feature vectors in the grid domain is shown in Figure 9.2 and Table 9.1. Figure 9.3 shows the correspondence of the templates to cell configurations in the grid. Notice that the following template has no match in the grid, it is listed only for completeness.



9.4 Template-Based Transfer Algorithm

Once a template and associated data is made available to an agent, the agent should have a way of using the information when learning a new task. There are at least three considerations. First, in each time step, the learning agent needs to decide whether or not to use a template and if so which one. Second, once the template choice is made the learning agent must decide how to apply the

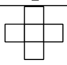
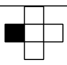
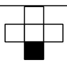
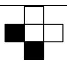
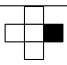
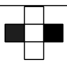
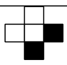
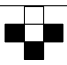
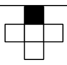
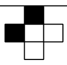
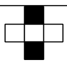
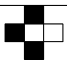
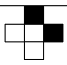
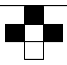
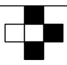
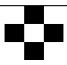
Template		Index
	00000	0
	00001	1
	00010	2
	00011	3
	00100	4
	00101	5
	00110	6
	00111	7
	01000	8
	01001	9
	01010	10
	01011	11
	01100	12
	01101	13
	01110	14
	01111	15

Table 9.1: Bit Encoding and Indexing of the Templates

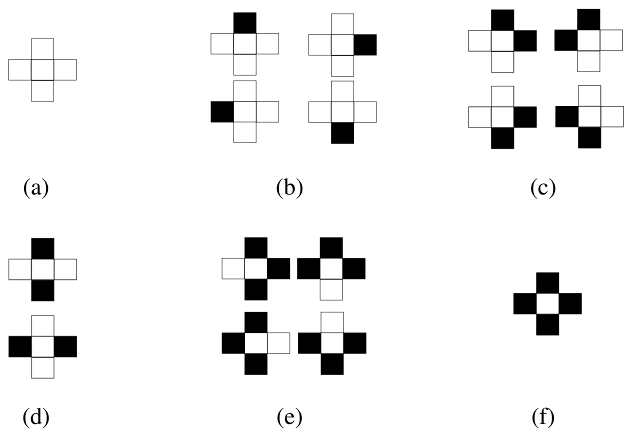


Figure 9.2: Local templates in the grid domain

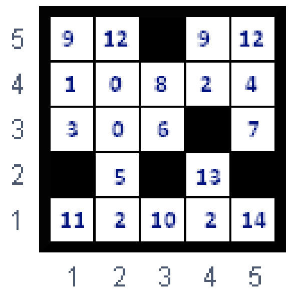


Figure 9.3: The 5×5 grid showing correspondence of templates to cell configurations (Indexes in Table 9.1.)

information in the template to the new task. Finally, to close the loop, new information acquired from learning on the new task may be used to enrich the template definition for the domain. We focus on the first two considerations and adopt the following method.

- The agent specifies belief about the unknown transition probabilities as a pro-forma prior distribution $f(P|M^{\omega})$.

- Historical data consisting of a feature vectors and associated transition count matrix relating to the domain is made available to the agent. The agent also receives information about transition constraints that apply to the domain.

- At each time step in the learning process, the agent observes its current state. If the agent is visiting the state for the first time it uses the feature function of the domain ∇_υ to map the current state to a feature vector then transfer the transition count for the matched feature vector to its prior distribution. The resulting distribution is then used to update value and policy functions.

We extend the constrained optimistic model selection COMS algorithm described in Chapter 7 to incorporate template-based transfer as shown in Algorithm 9.1.

```
 1: procedure $TemplateBasedOMS(S, R, A, \gamma, \alpha, M^{\omega}, F_\upsilon, C_\Omega)$
 2:     Initialise $V_k$  $\forall i, j \in \mathbb{S} \cup k$ and $\forall a \in \mathbb{A}$
 3:     Initialise $m^a_{ij} \leftarrow m^{\omega,a}_{ij}$  $\forall i, j \in \mathbb{S}$
 4:     Initialise $m^a_{ik}, m^a_{kj}$  $\forall i, j \in \mathbb{S} \cup k$
 5:     $\xi^a_i = \gamma V_k$,  $\forall i \in \mathbb{S}$ and $\forall a \in \mathbb{A}$ where $m^a_{ik} > 0$
 6:     $\xi^a_k = V_k$,  $\forall a \in \mathbb{A}$
 7:     loop
 8:         observe current state $x$ and the configuration $u_x$
 9:         select action $a = \arg\max_b \{\xi^b_x\}$ breaking ties randomly
10:         execute action $a$ in state $x$ and observe the transition $x \overset{a,r}{\rightsquigarrow} y$
11:         if transition $x \overset{a}{\rightsquigarrow} y$ is observed the very first time then
12:             retrieve feature vector $\vec{f}_x \leftarrow \nabla_\upsilon : x \in S|u_x$ and the associated transition counts $\vec{F}^a_x|\vec{f}_x, a$
13:             transfer information $\vec{m}^a_x \leftarrow \vec{m}^{\omega,a}_x + \vec{F}^a_x$
14:         end if
15:         update belief $m^a_{xy} = [m^a_{xy} + 1]|C_\Omega$
16:         repeat
17:             choose i
18:             for each action b do
19:                 find $\vec{p}^{\,b}_{opt,i}$ using COMS algorithms described in Chapter 7.
20:                 update $\xi^b_i$ using Equation 4.3
21:             end for
22:         until ARTDP algorithm stops
23:     end loop
24: end procedure
```

Algorithm 9.1: The Main Loop of the **T**emplate based **O**ptimistic **M**odel **S**election (TOMS) Algorithm

The procedure takes as input a known specification of an MDP i.e. the states S, actions A and reward function R. Also passed as input to the procedure are the pro-forma prior distribution M^{ω} of the agent, the discount rate γ for infinite horizon tasks, α that signifies the level of confidence attached to probability intervals, transition constraints C_Ω and the historical information F captured in the templates. The algorithm extends the constrained OMS in lines 11-14 where information from the templates is incorporated into the learning steps, as described above. In line 13, the information from the templates are transferred at state x i.e. $\vec{m}_x^a \leftarrow \vec{m}_x^{\omega,a} + \vec{F}_x^a$. An alternative would be to look ahead, given observed transition $x \overset{a}{\rightsquigarrow} y$, and transfer at state y i.e. $\vec{m}_y^a \leftarrow \vec{m}_y^{\omega,a} + \vec{F}_y^a$.

9.5 Experiment VII: Transfer using Process Templates

In this section we describe the experiments we conducted to study the performance of the template-based transfer algorithm. The experiments were performed in a grid domain.

9.5.1 Experiment Setup

First, we carried out an initial experiment to acquire historical data for the templates. This is then followed by two further sets of experiments that we used to study learning performance in the presence of the acquired historical data and a transition constraint.

9.5.1.1 Template Provisioning

We use the baby maze described in Section 9.3.2 to generate the historical data. Learning proceeds in multiple episodes. Each episode positions the learning agent at the start and terminates when it reaches a goal state or exceeds a maximum of one hundred steps. As before, the objective is to maximize the expected discounted rewards. Each action succeeds in moving the agent in the chosen direction with a probability of 0.94 or fails and remains in its current state with probability 0.06. If the agent tries to take an action going through an obstacle, it stays at the current state. The reward function is specified as follows.

$$r_s = \begin{cases} +10.0 & \text{for goal state,} \\ +0.0 & \text{for accessible cells,} \\ -1.0 & \text{penalty for attempting to go through an obstacle.} \end{cases} \tag{9.1}$$

The Full OMS Algorithm (4.1 & 4.3) of Chapter four is used to learn the baby maze task with fifty trials of ten episodes each for the four instances of the maze shown in Figure 9.4.

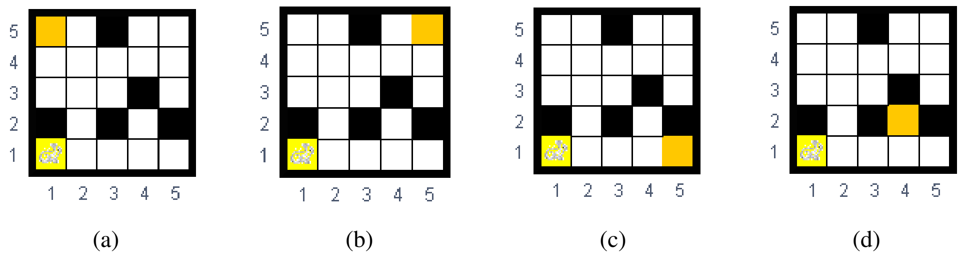


Figure 9.4: Instances of the baby maze task used for template provisioning. The baby maze instances differ only in their goal cells. The yellow cell (1,1) is the start state. The goal state is coloured orange.

9.5.1.2 Transfer to new Tasks

Following the acquisition of templates for the grid domain described in the previous section, the second part of Experiment VII involves transfer of information from the templates to two new tasks in the grid world. The new tasks are the six-room maze and the G-maze shown in Figures 9.5 and 9.6 respectively.

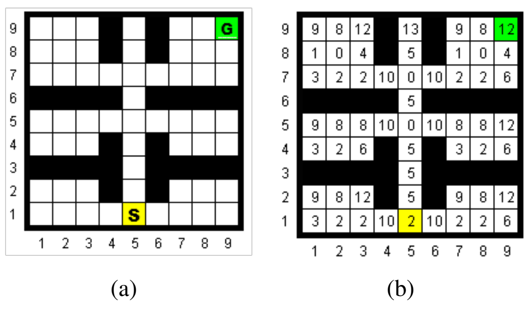


Figure 9.5: A Six Room Maze in a grid domain - a) The maze and b) associated templates

We carried out two sets of experiments. The experiments are labelled VII-A and VII-B for the six-room-maze and the G-maze respectively. The details of the experiments are summarised in Tables 9.2 and 9.3

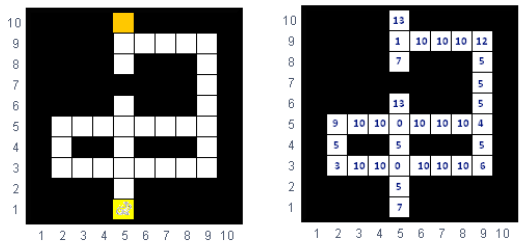

(a) (b)

Figure 9.6: A G-Maze in a grid domain - a) The maze and b) associated templates

	VII-A-1	VII-A-2	VII-A-3	VII-A-4
Domain	Grid World	Grid World	Grid World	Grid World
Task	Six-Room Maze	Six-Room Maze	Six-Room Maze	Six-Room Maze
	$\gamma = 0.95$	$\gamma = 0.95$	$\gamma = 0.95$	$\gamma = 0.95$
Reward	Equation 9.1	Equation 9.1	Equation 9.1	Equation 9.1
Learning Algorithm	Optimal Policy Known Model	OMS $\alpha = 95\%$	TOMS $\alpha = 95\%$ Simple Transfer	TOMS $\alpha = 95\%$ Simple Transfer
Pro-Forma Prior	-	Uniform(1)	Uniform(1)	Uniform(1)
Pre-Prior	- -	- -	Templates acquired in Section 9.5.1.1	Templates acquired in Section 9.5.1.1
Constraints	-	-	-	Equation 9.2

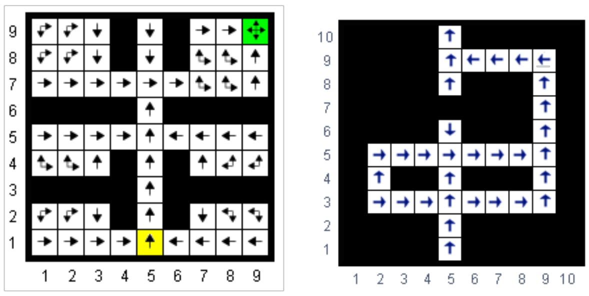
Table 9.2: Setup for Experiment VII-A in the template-based six-room maze

	VII-B-1	VII-B-2	VII-B-3	VII-B-4
Domain	Grid World	Grid World	Grid World	Grid World
Task	G Maze	G Maze	G Maze	G Maze
	$\gamma = 0.95$	$\gamma = 0.95$	$\gamma = 0.95$	$\gamma = 0.95$
Reward	Equation 9.1	Equation 9.1	Equation 9.1	Equation 9.1
Learning Algorithm	Optimal Policy Known Model	OMS $\alpha = 95\%$	TOMS $\alpha = 95\%$ Simple Transfer	TOMS $\alpha = 95\%$ Simple Transfer
Pro-Forma Prior	-	Uniform(1)	Uniform(1)	Uniform(1)
Pre-Prior	- -	- -	Templates acquired in Section 9.5.1.1	Templates acquired in Section 9.5.1.1
Constraints	-	-	-	Equation 9.2

Table 9.3: Setup for Experiment VII-B in the template-based G maze

The actual transition probabilities and reward function for the tasks are kept the same as we specified for the grid domain in Section 9.5.1.1. The discount factor is 0.95 throughout. The start cell configuration is $(5,1)$ in both the six-room and G mazes. The goal cell configurations are $(9,9)$ and $(5,10)$ in the six-room maze and the G maze respectively. Both the start and goal cell configurations are highlighted in Figures 9.5 and 9.6. The reward is same as in the baby maze. The objective is to maximize total discounted reward.

Figure 9.7 shows the optimal policies we calculated for the mazes assuming that the actual transition probabilities are known. The optimal policies are used in Experiments VII-A-1 and VII-



(a) Optimal Policy for the six-room maze (b) Optimal policy for the G maze

Figure 9.7: Optimal policies for the Six Room Maze, G-Grid Maze and D-Grid Maze Tasks

B-1 to simulate rewards and calculate the average total discounted reward. The actual transition probabilities are considered unknown in the other experiments. We assume availability of a uniform pro-forma prior distribution whose parameters as set to 1 i.e. $m_{ij}^{a,\omega} = 1 \; \forall i, j \in S$. In Experiments VII-A-3, VII-A-4, VII-B-3, and VII-B-4 the agent is presented with pre-prior information from the templates we acquired in Section 9.5.1.1 and the agent uses the simple transfer method in line 13 of TOMS (Algorithm 9.1) to combine prior information. Finally, in addition to the template based pre-prior information, the following absolute constraints are made available to the agent in experiments VII-A-4 and VII-B-4.

$$p_{ij}^{a} = 0.94 \; \forall i \in S, a \in A \text{ and successful transition } i \overset{a}{\rightsquigarrow} j. \tag{9.2}$$

9.5.2 Results and Discussion

The results of the experiments for the six-room maze and the G maze tasks are shown in Figures 9.8 – 9.11. The agent is able to learn the tasks using TOMS algorithms with and without the transferable prior information, which are the templates and the transition constraints. The six-room maze is more challenging without the template which we attribute to the non-informative uniform pro-forma prior of the agent. The G maze is less challenging to learn for the agent without the template because it is smaller in comparison to the six-room maze. In both the six-room maze and the G maze, learning performance is improved with the use of templates and the improvement increases when the template is combined with the constraints.

The contribution to performance improvement of the transition constraints is significant in comparison to learning with templates and no constraints. This is attributed to the 'strength' of the constraints. Despite this, the learning performance of the agent with constraints is less effective than the optimal policy. This is due to a) the uncertainty about transition probabilities for states other than those captured in the constraint, and b) the effect of the non-informative pro-forma prior distribution of the agent.

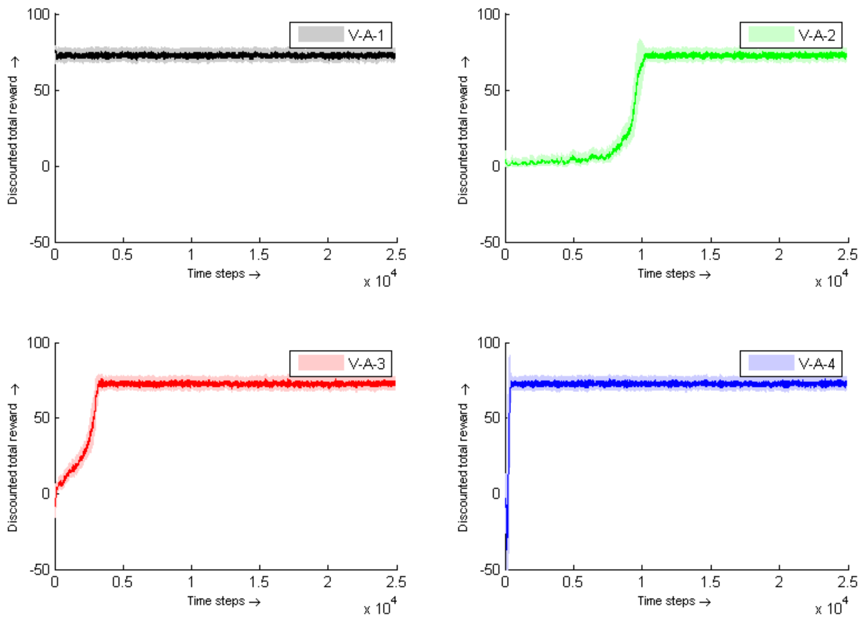


Figure 9.8: Results for the Six-Room Maze Task

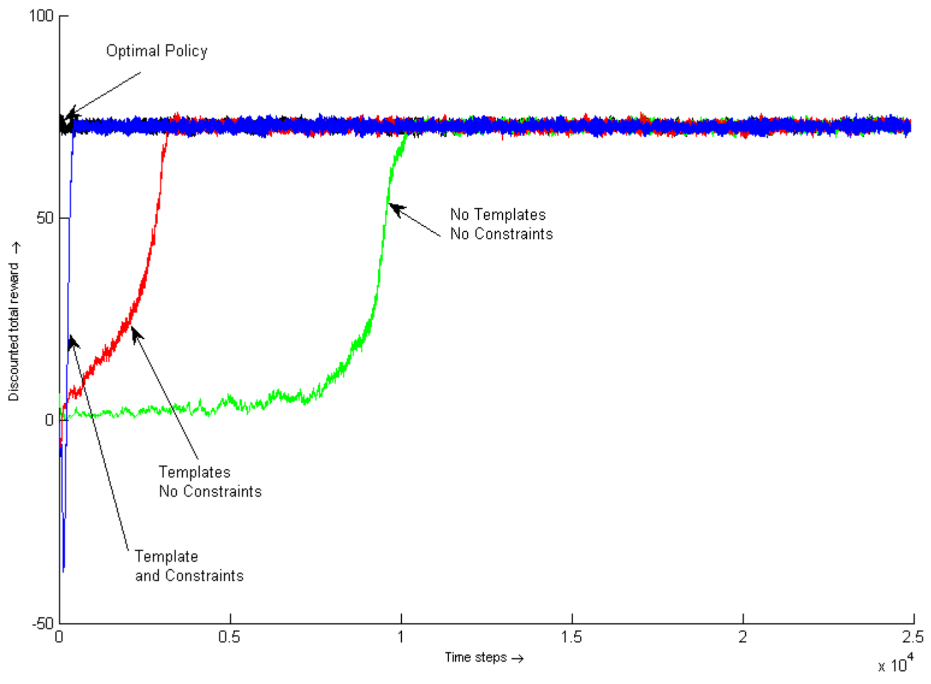


Figure 9.9: Comparison of Results for the Six-Room Maze Task

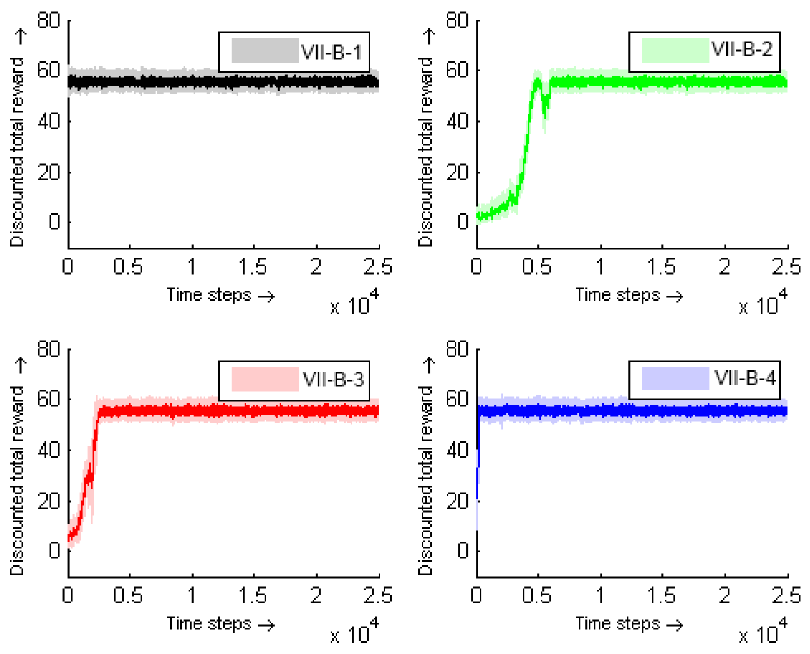


Figure 9.10: Results for the G-Grid Task

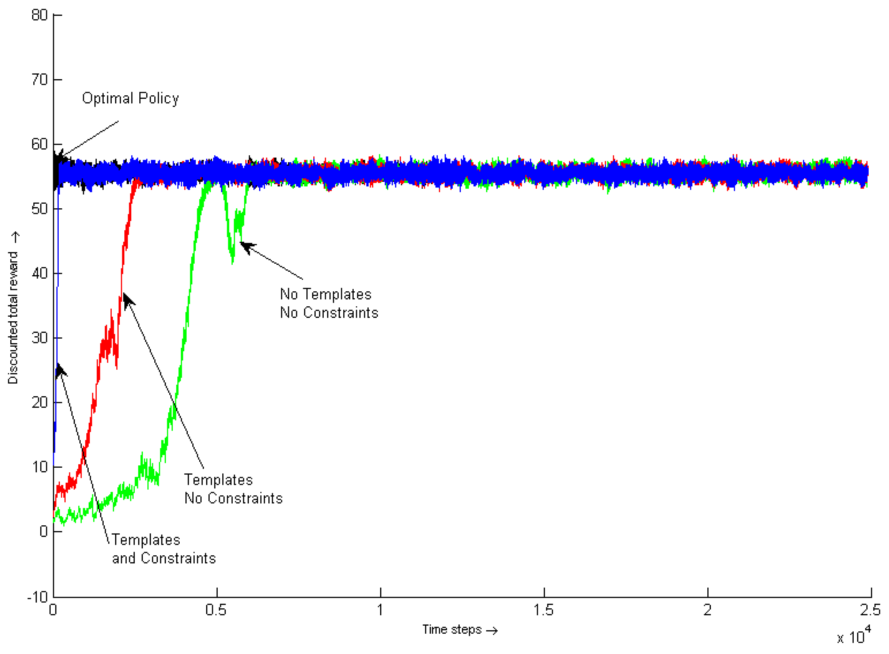


Figure 9.11: Comparison of Results for the G-Grid Task

9.6 Chapter Summary

The focus in this chapter has been on how to use reference patterns (templates) from donor models of source tasks in leveraging learning on a recipient model of a new target task. A template is conceptualised simply as a set of 'locally-defined' features of the donor models, each of which can be located independently of the rest by analysing small regions of the model space. We demonstrated the benefits of historical information and also show how to use process templates to transfer information from one environment to another with related local process dynamics.

10 Conclusions and Areas of Future Research

Optimal learning and Transfer of Learning are two important subfields of Artificial Intelligence. Optimal learning is an approach to maximising behavioural patterns that leads to the highest possible expected total reward over the entire duration of an agent's interaction with an uncertain task environment. Prior uncertainty and prior experience plays a key role. Prior uncertainty over possible environments adds complexity to the problem of optimal learning. Transfer of learning exploits prior experience on different but related tasks to improve performance while learning. This thesis is at the intersection of the two subfields. It focuses on how an agent could incorporate user knowledge about an environment or information acquired during previous learning in order to make future learning more effective. We presented new algorithms to facilitate the transfer and showed that the algorithms can lead to better performance during learning (the exploitation-exploration trade-off) and offer better solutions at the end of the learning period. In this final chapter of the thesis we present concluding remarks and suggest directions for future research.

10.1 Conclusions

We addressed the problem of learning transfer in an online setting where the learner interacts with an unknown or partially known environment to repeatedly perform actions, observes the consequences of its actions i.e. reward and state transitions, revises its belief about the environment and updates its value and policy functions based on its observations. We informally describe the problem addressed in this thesis as follows. We have knowledge of transitions for a set of related tasks. We have a new task and a new reward function but do not yet have a policy and a model for that task. The problem is to transfer explicitly available information to help improve performance on the new task in comparison to learning the new task from scratch. We consider two types of transferable information - transition constraints and historical data.

The following are the conclusions drawn from the thesis.

1. Transition constraints play an important role in optimal learning. The constraints restrict model choices helping to direct the learning agent towards plausible solutions. We demonstrated in this thesis (Chapter seven) that transition constraints can play two important roles in optimal learning - First in the revision of beliefs about the space of possible models and second in exploration control through model selection on a restricted space of models, with the restrictions dictated by the specified transition constraints.

2. We studied two main types of constraints: absolute and relative constraints. We also described specific instances of each type including equality constraints, significant inequalities in transition probabilities, and ordering constraints on transition probabilities. We demonstrated that these constraints can occur individually and can also manifest jointly (Chapter seven). The constraints differ not only in the way they can be reasoned with but may also contribute in different ways towards improvements in learning performance. To fully exploit transition constraints it is important that they are appropriately recognised and accounted for.

3. Reasoning with transition constraints when learning from rewards can be achieved in a variety of ways. We demonstrated in this thesis (Chapter seven) that inference can be done through either a) Monte Carlo Markov Chains, specifically via Gibbs Sampling from truncated Dirichlet distributions, or b) Specialised algorithms that exploit special structure exhibited by the constraints. A consequence of this flexibility is that the methodology with which an agent can reason with transition constraints can be seen as state contingent i.e. the choice of algorithm is dependent on the structure exhibited by the constraints applicable to a given state of the

decision process.

4. For agents using an optimistic model selection (OMS) approach to exploration control in Bayes-Adaptive Markov decision processes, pre-emptive goal program techniques present an elegant way of incorporating constraints in the model selection steps (Chapter four).

5. When an agent has at its disposal historical data from related task the agent should be able to appropriately incorporate the historical data into its learning procedure. Although historical data may come from a variety of sources, the most natural specification of historical data arises when the raw data from a similar previous activity is available. Historical data is conceptualised in this thesis as constituting pre-prior information which is made available to the agent before the commencement of learning on a new task. We showed in this thesis (Chapter six) that a way to incorporate historical data in optimal learning is via power prior Bayesian analysis. Power priors are constructed by raising the multinomial *likelihood function* of the historical data to a suitable power (i.e. the precision parameter) to discount the historical data relative to the pro-forma Dirichlet distribution of the learning agent. The precision parameter can be acquired by measuring the degree of match between the historical data and imaginary data drawn from the user specified constraints.

6. The benefits of historical data appear more visible and easier to conceptualise when transfer is based on process templates (Chapter nine).

10.2 Future Research

The research we reported in this thesis leads to several possible areas of future work that include the following:

- The consolidation of our research on the use of transferable prior information to optimise learning in enumerated state MDPs and template-based MDPs especially in relation to areas such as cross-domain transfer, historical data and constraints from multiple sources, considerations for other special types of absolute and relative transition constraints, and extensions to other information types such those relating to solutions i.e. value and / or policy functions.

- Extension to other formalisms such as Factored MDPs, Relational MDPs, Partially Observable MDPs, and continuous state and action spaces.
- Application to real world domains particularly where opportunity exists for exploiting historical data and/or transition constraints in sequential decision making that involves uncertainty in the associated task-environment models.

We made assumptions in the thesis about the nature of constraints. Specifically, we assumed that the transitions constraints are known and that they fit with the way the environment works. This may not always hold in practical applications. The data may provide strong evidence that the constraints are partially or completely untrue. We will require methods that appropriately reflect uncertainty about process constraints and allow estimates to contradict the constraints or to permit sensible compromise between unconstrained estimates and estimates based on the constraint.

References

Alagoz, O., Maillart, L. M., Schaefer, A. J. & Roberts, M. S. (2004), 'The Optimal Timing of Living-Donor Liver Transplantation', *Management Science* **50**(10), 1420–1430.

Asmuth, J. & Littman, M. L. (2011), Approaching Bayes-optimality using Monte-Carlo tree search, *in* 'Proceedings of the 21st International Conference on Automated Planning and Scheduling', Freiburg, Germany.

Atkeson, C. G. & Santamaria, J. C. (1997), A Comparison of Direct and Model-Based Reinforcement Learning, *in* 'IEEE International Conference on Robotics and Automation', Vol. 4, IEEE Press, pp. 3557–3564.

Ayer, M., Brunk, H. D., Ewing, G. M., Reid, W. T. & Silverman, E. (1955), 'An Empirical Distribution Function for Sampling with Incomplete Information', *The Annals of Mathematical Statistics* **26**(4), 641–647.

Barlow, R. E., Bartholomew, D. J., Bremner, J. M. & Brunk, H. D. (1972), *Statistical Inference Under Order Restrictions*, John Wiley & Sons, New York.

Barto, A. G., Sutton, R. S. & Anderson, C. W. (1983), 'Neuronlike adaptive elements that can solve difficult learning control problems', *IEEE Transactions on Systems, Man, and Cybernetics* **SMC-13**(5), 834–846.

Barto, A. G., Sutton, R. S. & Watkins, C. J. C. H. (1990), Learning and sequential decision making, *in* M. Gabriel & J. W. Moore, eds, 'Learning and Computational Neuroscience', MIT Press, Cambridge, MA.

Barzel, R., Hughes, J. & Wood, D. N. (1996), Plausible Motion Simulation for Computer Graphics Animation, *in* 'Proceedings of the Eurographics Workshop Computer Animation and Simulation', pp. 183–197.

Bellman, R. E. (1957), *Dynamic Programming*, Princeton University Press, Princeton, N.J., USA.

Bellman, R. E. (1961), *Adaptive control processes: A guided tour*, Rand Corporation Research Studies, Princeton University Press.

Bellman, R. E. & Kalaba, R. (1959*a*), 'A Mathematical Theory of Adaptive Control Processes', *Proceedings of the National Academy of Sciences of the United States of America* **45**(8), pp. 1288–1290.

Bellman, R. E. & Kalaba, R. (1959*b*), 'On Adaptive Control Processes', *IRE Transactions on Automatic Control* **4**(2), 1 – 9.

Berger, J. O. (1985), *Statistical Decision Theory and Bayesian Analysis*, Springer Series in Statistics, Springer-Verlag, New York.

Berger, J. O. (1990), 'Robust Bayesian analysis: sensitivity to the prior', *Journal of Statistical Planning and Inference* **25**(3), 303–328.

Berger, J. O. & Berliner, L. M. (1986), 'Robust Bayes and Empirical Bayes Analysis with $\varepsilon-$Contaminated Priors', *The Annals of Statistics* **14**(2), 461–486.

Bertsekas, D. P. (1987), *Dynamic Programming: Deterministic and Stochastic Models*, Prentice-Hall: Englewood Cliffs, NJ.

Bhattacharya, B. (2009), 'Optimal Use of Historical Information', *Journal of Statistical Planning and Inference* **139**(12), 4051 – 4063.

Bhattacharyya, A. (1943), 'On a measure of divergence between two statistical populations defined by their probability distributions', *Bulletin of the Calcutta Mathematical Society* **35**, 99–109.

Boutilier, C., Dean, T. & Hanks, S. (1999), 'Decision-theoretic planning: Structural assumptions and computational leverage', *Journal of Artificial Intelligence Research (JAIR)* **11**, 1–94.

Boutilier, C., Dearden, R. W. & Goldszmidt, M. (2000), 'Stochastic dynamic programming with factored representations', *Artificial Intelligence* **121**(1-2), 49–107.

Brafman, R. I. & Tennenholtz, M. (2002), 'R-MAX – A General Polynomial Time Algorithm for Near-Optimal Reinforcement Learning', *Journal of Machine Learning Research* **3**, 213–231.

Castro, P. (2011), On planning, prediction and knowledge transfer in Fully and Partially Observable Markov Decision Processes, PhD thesis, McGill University.

Castro, P. S. & Precup, D. (2007), Using Linear Programming for Bayesian Exploration in Markov Decision Processes, *in* 'Proceedings of the 20th International Joint Conference on Artifical Intelligence', IJCAI'07, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 2437–2442.

Chang, T. S. & Seborg, D. E. (1982), Process Control in the Presence of Constraints, *in* 'Proceedings of the 1982 American Control Conference', pp. 350 –357.

Chankong, V. & Haimes, Y. Y. (1983), *Multiobjective Decision Making Theory and Methodology*, Elsevier Science, New York.

Charnes, A. & Cooper, W. W. (1977), 'Goal programming and Multiple Objective Optimizations: Part 1', *European Journal of Operational Research* **1**(1), 39–54.

Charnes, A., Cooper, W. W. & Ferguson, R. O. (1955), 'Optimal Estimation of Executive Compensation by Linear Programming', *Management Science* **1**(2), 138–151.

Chen, M.-H., Ibrahim, J. G. & Shao, Q.-M. (2000), 'Power prior distributions for generalized linear models', *Journal of Statistical Planning and Inference* **84**(1), 121–137.

Chen, M.-H. & Shao, Q.-M. (1999), 'Monte Carlo Estimation of Bayesian Credible and HPD Intervals', *Journal of Computational and Graphical Statistics* **8**(1), pp. 69–92.

Chenney, S. & Forsyth, D. A. (2000), Sampling plausible solutions to multi-body constraint problems, *in* 'Proceedings of the 27th annual conference on computer graphics and interactive techniques', SIGGRAPH '00, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, pp. 219–228.

Coxeter, H. S. M. (1973), *Regular Polytopes*, Dover books on advanced mathematics, Dover Publications.

Cran, G. W. (1980), 'Algorithm AS 149: Amalgamation of Means in the Case of Simple Ordering', *Journal of the Royal Statistical Society. Series C (Applied Statistics)* **29**(2), pp. 209–211.

Ştefănescu, V. (1989), 'Generating Dirichlet random vectors using a rejection property', *Kybernetika* **25**(6), 467–475.

Dearden, R. W. (2000), *Learning and Planning in Structured World*, Ph.D.Thesis, Department of Computer Science, University of British Columbia, Canada.

Dixon, K., Malak, R. & Khosla, P. (2000), Incorporating Prior Knowledge and Previously Learned Information into Reinforcement Learning Agents, Technical report. Institute for Complex Engineered Systems, Carnegie Mellon University.

Dreyfus, S. (1962), 'Variational Problems with Inequality Constraints', *Journal of Mathematical Analysis and Applications* **4**, 297–308.

Drummond, C. (1997), Using A Case-Base of Surfaces to Speed-Up Reinforcement Learning, *in* 'Proceedings of the Second International Conference on Case-Based Reasoning', pp. 435–444.

Duan, Y. (2005), A Modified Bayesian Power Prior Approach with Applications in Water Quality Evaluation, PhD thesis, Faculty of Virginia Polytechnic Institute and State University, Blacksburg, Virginia.

Duan, Y., Ye, K. & Smith, E. P. (2006), 'Evaluating water quality using power priors to incorporate historical information', *Environmetrics* **17**(1), 95–106.

Duff, M. (2002), Optimal learning: Computational procedures for Bayes-adaptive Markov decision processes, PhD thesis, University of Massassachusetts Amherst.

Dzeroski, S., De Raedt, L. & Driessens, K. (2001), 'Relational Reinforcement Learning', *Machine Learning* **43**(1), 7–52.

Eddy, D. M. (1989), 'The Confidence Profile Method: A Bayesian Method for Assessing Health Technologies', *Operations Research* **37**(2), 210–228.

Eddy, D. M., Hasselblad, V. & Shachter, R. (1990*a*), 'A Bayesian method for synthesizing evidence: The Confidence Profile Method', *International Journal of Technology Assessment in Health Care* **6**(1), 31–35.

Eddy, D. M., Hasselblad, V. & Shachter, R. (1990*b*), 'An Introduction to a Bayesian Method for Meta-analysis: The Confidence Profile Method', *Medical Decision Making* **10**(1), 15–23.

Feinberg, E. & Shwartz, A. (2002), *Handbook of Markov Decision Processes: Methods and Applications*, International Series in Operations Research & Management Science, Kluwer Academic Publishers.

Fellows, M., Hartman, T., Hermelin, D., Landau, G., Rosamond, F. & Rozenberg, L. (2011), 'Haplotype Inference Constrained by Plausible Haplotype Data', *IEEE/ACM Transactions on Computational Biology and Bioinformatics,* **8**(6), 1692 –1699.

Frigyik, B. A., Kapila, A. & Gupta, M. R. (2010), Introduction to the Dirichlet Distribution and Related Processes, Technical Report UWEETR-2010-0006, Department of Electrical Engineering, University of Washington, Seattle, Washington, USA.

Gajewski, B. J. (2010), 'Comments on 'A note on the power prior' by Neuenschwander B, Branson M and Spiegelhalter DJ. Statistics in Medicine; DOI: 10.1002/sim.3722', *Statistics in Medicine* **29**(6), 708–709.

Gaskett, C. (2002), Q-learning for Robot Control, PhD thesis, Australian National University.

Gelfand, A. E., Smith, A. F. M. & Lee, T.-M. (1992), 'Bayesian Analysis of Constrained Parameter and Truncated Data Problems Using Gibbs Sampling', *Journal of the American Statistical Association* **87**(418), pp. 523–532.

Genest, C. & Zidek, J. V. (1986), ‘Combining Probability Distributions: A Critique and an Annotated Bibliography’, *Statistical Science* **1**(1), 114–135.

Gewali, L., Ntafos, S. & Singh, A. K. (2002), ‘Geometric approach for finding HPD-credible sets with applications’, *Applied Mathematics and Computation,* **125**(2), 195–207.

Geyer, C. J. (1991), Markov Chain Monte Carlo Maximium Likelihood, *in* E. Keramigas, ed., ‘Computing Science and Statistics: The 23^{rd} symposium on the interface’, Interface Foundation, FairFax.

Givan, R., Leach, S. & Dean, T. (2000), ‘Bounded-parameter Markov decision process’, *Artificial Intelligence* **122**(1-2), 71–109.

Goncalves, A. P. C., Fioravanti, A. R. & Geromel, J. C. (2010), Filtering for discrete-time Markov jump systems with network transmitted mode, *in* ‘49th IEEE Conference on Decision and Control (CDC)’, pp. 924 –929.

Good, I. J. (1950), *Probability and the weighing of evidence*, Griffin, London.

Guestrin, C., Patrascu, R. & Schuurmans, D. (2002), Algorithm-Directed Exploration for Model-Based Reinforcement Learning in Factored MDPs, *in* ‘ICML-2002 The Nineteenth International Conference on Machine Learning, University of New South Wales, Sydney, Australia, July 8-12’, Morgan Kaufmann.

Gunn, L. H. & Dunson, D. B. (2005), ‘A tranformation approach for incorporating monotone or unimodal constraints’, *Biostatistics* **6**(3), 434–449.

Hernández-Lerma, O. & Lasserre, J. B. (1991), *Markov decision processes*, Vol. 28-29 of *Annals of Operations Research*, J. C. Baltzer AG.

Heyman, D. P. & Sobel, M. J. (2003), *Stochastic Models in Operations Research, Vol. II: Stochastic Optimization*, Vol. 2 of *Stochastic Models in Operations Research*, Dover Publications.

Ho, Y.-C. (1962), 'A Computational Technique for Optimal Control Problems with State Variable Constraint', *Journal of Mathematical Analysis and Applications* **5**(2), 216 – 224.

Howard, R. A. (1960), *Dynamic programming and Markov processes*, Cambridge, MA: MIT Press.

Hung, Y. C., Balakrishnan, N. & Cheng, C. W. (2011), 'Evaluation of algorithms for generating Dirichlet random vectors', *Journal of Statistical Computation and Simulation* **81**(4), 445–459.

Ibrahim, J. G. & Chen, M.-H. (2000), 'Power Prior Distributions for Regression Models', *Statistical Science* **15**(1), 46–60.

Ibrahim, J. G., Chen, M.-H. & Sinha, D. (2003), 'On Optimality Properties of the Power Prior', *Journal of the American Statistical Association* **98**(461), 204–213.

Ignizio, J. (1976), *Goal programming and extensions*, Lexington Books.

Jain, A., Zhong, Y. & Lakshmanan, S. (1996), 'Object matching using deformable templates', *IEEE Transactions on Pattern Analysis and Machine Intelligence,* **18**(3), 267–278.

Jewell, N. P. & Kalbfleisch, J. D. (2004), 'Maximum Likelihood Estimation of Ordered Multinomial Parameters', *Biostatistics* **5**(2), 291–306.

Kaelbling, L. P., Littman, M. L. & Moore, A. W. (1996), 'Reinforcement Learning: A Survey', *Journal of Artificial Intelligence Research* **4**, 237–285.

Kailath, T. (1967), 'The Divergence and the Bhattacharyya Distance Measures in Signal Selection', *IEEE Transactions on Communication Technology* **COM-15**(1), 52–60.

Kalyanasundaram, S., Chong, E. K. P. & Shroff, N. B. (2002), Markov decision processes with uncertain transition rates: sensitivity and robust control, *in* 'Proceedings of the 41st IEEE Conference on Decision and Control', Vol. 4, pp. 3799 – 3804.

Karasmaa, N. (2003), The Transferability of Travel Demand Models: An analysis of transfer methods, data quality and model estimation, PhD thesis, Department of Civil and Environmental Engineering, Helsinki University of Technology, Transportation Engineering Publication 106.

Karny, M. (1984), ‘Quantification of Prior Knowledge about Global Characteristics of Linear Normal Model’, *Kybernetika* **20**(5), 376—-385.

Kearns, M. J. & Singh, S. P. (2002), ‘Near-Optimal Reinforcement Learning in Polynomial Time’, *Machine Learning* **49**(2-3), 209–232.

Kearns, M. & Singh, S. (1999), Finite-sample convergence rates for Q-learning and indirect algorithms, *in* ‘Neural Information Processing Systems 12’, MIT Press, Cambridge, MA, USA, pp. 996–1002.

Konda, V. (2002), Actor-critic Algorithms, PhD thesis, Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science.

Konda, V. R. & Tsitsiklis, J. N. (2000), Actor-critic algorithms, *in* S. A. Solla, T. K. Leen & K.-R. Müller, eds, ‘Advances in Neural Information Processing Systems 12, [NIPS Conference, Denver, Colorado, USA, November 29 - December 4, 1999]’, The MIT Press, pp. 1008–1014.

Koppejan, R. & Whiteson, S. (2009), Neuroevolutionary Reinforcement Learning for Generalized Helicopter Control, *in* ‘GECCO 2009: Proceedings of the Genetic and Evolutionary Computation Conference’, pp. 145–152.

Koppelman, F. S., Kuah, G.-K. & Wilmot, C. G. (1985), ‘Transfer Model Updating with Disaggregate Data’, *Transportation Research Record N1037, Transportation Demand Analysis and Issues in Travel Behavior* **1037**, 102–107.

Lang, L., Chen, W.-S., Bakshi, B. R., Goel, P. K. & Ungarala, S. (2007), ‘Bayesian estimation via sequential Monte Carlo sampling - Constrained dynamic systems.’, *Automatica* **43**(9), 1615–1622.

Lazaric, A. (2008), Knowledge Transfer in Reinforcement Learning, PhD thesis, Politecnico di Milano.

Lazaric, A., Restelli, M. & Bonarini, A. (2008), Transfer of samples in batch reinforcement learning, *in* 'Proceedings of the 25th international conference on Machine learning', ICML '08, ACM, New York, NY, USA, pp. 544–551.

Lee, S. M. (1972), *Goal programming for decision analysis*, Management and communications series, Auerbach Publishers.

Lim, J., Wang, X. & Choi, W. (2009), 'Maximum likelihood estimation of ordered multinomial probabilities by geometric programming', *Computational Statistics & Data Analysis* **53**(4), 889 – 893.

Lin, Y.-P. & Li, X.-Y. (2003), 'Reinforcement learning based on local state feature learning and policy adjustment', *Information Sciences* **154**(1-2), 59 – 70. Introduction to Multimedia and Mobile Agents.

Liu, C. (2000), 'Estimation of Discrete Distributions with a Class of Simplex Constraints', *Journal of the American Statistical Association* **95**(449), pp. 109–120.

Magni, P., Quaglini, S., Marchetti, M. & Barosi, G. (2000), 'Deciding when to intervene: a Markov decision process approach', *International Journal of Medical Informatics* **60**(3), 237–253.

Marsaglia, G. & Tsang, W. W. (2000), 'A Simple Method for Generating Gamma Variables', *ACM Transactions on Mathematical Software (TOMS)* **26**(3), 363–372.

Martin, J. J. (1967), *Bayesian decision problems and Markov chains*, Publications in Operations Research, Wiley, New York.

McDannald, M. A., Takahashi, Y. K., Lopatina, N., Pietras, B. W., Jones, J. L. & Schoenbaum, G. (2012), 'Model-based learning and the contribution of the orbitofrontal cortex to the model-free world', *European Journal of Neuroscience* **35**(7), 991–996.

Meuleau, N. & Bourgine, P. (1999), 'Exploration of Multi-State Environments: Local Measures and Back-Propagation of Uncertainty', *Machine Learning* **35**(2), 117–154.

Moustafa, M. S., Maksoud, E. Y. A. & Sadek, S. (2004), 'Optimal major and minimal maintenance policies for deteriorating systems', *Reliability Engineering & System Safety* **83**(3), 363 – 368.

Narayanan, A. (1990), 'Computer Generation of Dirichlet Random Vectors', *Journal of Statistical Computation and Simulation* **36**(1), 19–30.

Narendra, K. S. & Thathachar, M. A. L. (1989), *Learning automata: an introduction*, Prentice Hall.

Neal, R. M. (2001), Transferring Prior Information Between Models Using Imaginary Data, Technical report, University of Toronto, Toronto, Ontario, Canada.

Neuenschwander, B., Branson, M. & Spiegelhalter, D. J. (2009), 'A note on the power prior', *Statistics in Medicine* **28**(28), 3562–3566.

Ng, K. W., Tian, G.-L. & Tang, M.-L. (2011), *Dirichlet and Related Distributions: Theory, Methods and Applications*, Volume 888 of Wiley Series in Probability and Statistics, John Wiley & Sons.

Nicholson, B. J. (1985), 'On the F-Distribution for Calculating Bayes Credible Intervals for Fraction Nonconforming', *IEEE Transactions on Reliability* **R-34**(3), 227 –228.

Pan, S. J. & Yang, Q. (2010), 'A survey on transfer learning', *IEEE Transactions on Knowledge and Data Engineering* **22**(10), 1345–1359.

Pintrich, P. R., Marx, R. W. & Boyle, R. A. (1993), 'Beyond Cold Conceptual Change: The Role of Motivational Beliefs and Classroom Contextual Factors in the Process of Conceptual Change', *Review of Educational Research* **63**(2), 167–199.

Poole, D. & Raftery, A. E. (2000), 'Inference for Deterministic Simulation Models: The Bayesian Melding Approach', *Journal of the American Statistical Association* **95**(452), 1244–1255.

Poupart, P., Vlassis, N. A., Hoey, J. & Regan, K. (2006), An Analytic Solution to Discrete Bayesian Reinforcement Learning, *in* 'Proceedings of the 23rd International Conference on Machine Learning, Pittsburgh, PA', pp. 697–704.

Price, R. R. (2003), Accelerating Reinforcement Learning with Imitation, PhD thesis, Department of Computer Science, University of British Columbia.

Proper, S. & Tadepalli, P. (2009), Multiagent Transfer Learning via Assignment-Based Decomposition, *in* 'International Conference on Machine Learning and Applications - ICMLA', pp. 345 –350.

Proper, S. & Tadepalli, P. (2010), Transfer Learning via Relational Templates, *in* L. Raedt, ed., 'Inductive Logic Programming', Vol. 5989 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 186–193.

Puterman, M. (2005), *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, Wiley Series in Probability and Statistics, Wiley-Interscience.

Raftery, A. E., Givens, G. H. & Zeh, J. E. (1995), 'Inference from a deterministic population dynamics model for bowhead whales', *Journal of The American Statistical Association* **90**(430), 402–416.

Rauber, T., Braun, T. & Berns, K. (2008), 'Probabilistic distance measures of the Dirichlet and Beta distributions', *Pattern Recognition* **41**(2), 637 – 645.

Roman, A., Jolad, S. & Shastry, M. C. (2012), 'Bounded divergence measures based on Bhattacharyya coefficient', *CoRR* **abs/1201.0418**.

Rummery, G. A. & Niranjan, M. (1994), On-line Q-learning using connectionist systems, Technical Report CUED/F-INFENG/TR 166, Cambridge University Engineering Department.

Russell, S. J. & Norvig, P. (2003), *Artificial Intelligence: A Modern Approach*, Prentice Hall Series in Artificial Intelligence, 2nd edn, Prentice-Hall, Englewood Cliffs, NJ.

Satia, J. K. & Lave, R. E. (1973), 'Markovian Decision Processes with Uncertain Transition Probabilities', *Operations Research* **21**(3), 728–740.

Schaible, B. R. (1999), Fuzzy Logic Based Regression Models of Flip-Chip Bonding Processes, PhD thesis, Department of Mechanical Engineering, University of Colorado.

Sedransk, J., Monahan, J. & Chiu, H. Y. (1985), 'Bayesian Estimation of Finite Population Parameters in Categorical Data Models Incorporating Order Restrictions', *Journal of the Royal Statistical Society. Series B (Methodological)* **47**(3), 519–527.

Sennott, L. I. (1999), *Stochastic Dynamic Programming and the Control of Queueing Systems*, Wiley series in probability and statistics: Applied probability and statistics, Wiley.

Shannon, T. T., Santiago, R. A. & George, G. L. (2003), Accelerating critic learning in approximate dynamic programming via value templates and perceptual learning, *in* 'Proceedings of the International Joint Conference on Neural Networks', Vol. 4, pp. 2922 – 2927.

Shechter, S. M., Bailey, M. D., Schaefer, A. J. & Roberts, M. S. (2008), 'The Optimal Time to Initiate HIV Therapy Under Ordered Health States', *Operations Research* **56**(1), 20–33.

Silver, D., Sutton, R. & Müller, M. (2007), Reinforcement Learning of Local Shape in the Game of Go, *in* 'Proceedings of the International Joint Conference on Artifical Intelligence, IJCAI', pp. 1053–1058.

Singh, S. P. (1991), Transfer of Learning Across Compositions of Sequential Tasks, *in* 'Machine Learning: Proceedings of the Eighth International Workshop', Morgan Kaufmann, pp. 348–352.

Singh, S. P. & Sutton, R. S. (1996), 'Reinforcement Learning with Replacing Eligibility Traces', *Machine Learning* **22**, 123–158.

Stolle, M. & Atkeson, C. G. (2007), Knowledge Transfer using Local Features, *in* 'Proceedings of the IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning', pp. 26 – 31.

Strens, M. J. A. (2000), A Bayesian Framework for Reinforcement Learning, *in* 'Proceedings of the Seventeenth International Conference on Machine Learning (ICML)', pp. 943–950.

Strens, M. J. A. (2003), Evolutionary MCMC Sampling and Optimization in Discrete Spaces, *in* 'Proceedings of the Twentieth International Conference on Machine Learning (ICML)', pp. 736–743.

Sunmola, F. T. & Wyatt, J. L. (2006), Model Transfer for Markov Decision Tasks via Parameter Matching, *in* 'Workshop of the UK Planning and Scheduling Special Interest Group'.

Sutton, R. S. (1988), 'Learning to Predict by the Methods of Temporal Differences', *Machine Learning* **3**, 9–44.

Sutton, R. S. (1996), Generalization in Reinforcement Learning: Successful Examples using Sparse Coarse Coding, *in* 'Advances in Neural Information Processing Systems 8', Vol. 8, pp. 1038–1044.

Sutton, R. S. & Barto, A. G. (1998), *Reinforcement Learning: An Introduction*, Adaptive Computation and Machine Learning, MIT Press.

Sutton, R. S., McAllester, D. A., Singh, S. P. & Mansour, Y. (1999), Policy Gradient Methods for Reinforcement Learning with Function Approximation, *in* S. A. Solla, T. K. Leen & K.-R. Müller, eds, 'Advances in Neural Information Processing Systems 12 (Proceedings of the 1999 conference)', MIT Press, pp. 1057–1063.

Szepesvári, C. (2010), *Algorithms for Reinforcement Learning*, Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool Publishers.

Szita, I. & Lőrincz, A. (2008), The many faces of optimism: a unifying approach, *in* 'Proceedings of the 25th international conference on Machine learning', ICML '08, pp. 1048–1055.

Tanaka, F. & Yamamura, M. (2003), Multitask Reinforcement Learning on the Distribution of MDPs, *in* 'IEEE International Symposium on Computational Intelligence in Robotics and Automation', Vol. 3, pp. 1108 – 1113.

Taylor, M. E. (2008), Autonomous Inter-Task Transfer in Reinforcement Learning Domains, PhD thesis, Technical Report UT-AI-TR-08-5, Department of Computer Sciences, The University of Texas at Austin.

Taylor, M. E., Jong, N. K. & Stone, P. (2008), Transferring Instances for Model-Based Reinforcement Learning, *in* 'Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML / PKDD)', pp. 488–505.

Taylor, M. E. & Stone, P. (2009), 'Transfer Learning for Reinforcement Learning Domains: A Survey', *Journal of Machine Learning Research* **10**(1), 1633–1685.

Tesař, L. (1996), Processing of Prior Information as Function of Data, *in* L. Berec, J. Rojíček, M. Kárný & K. Warwick, eds, 'Preprints of the European IEEE Workshop CMP'96', ÚTIA AV ČR, Prague, pp. 145–149.

Torrey, L. (2009), Relational Transfer in Reinforcement Learning, PhD thesis, University of Winsconsin-Madison.

Wang, T., Lizotte, D., Bowling, M. & Schuurmans, D. (2005), Bayesian Sparse Sampling for On-line Reward Optimization, *in* 'Proceedings of the Twenty-Second International Conference on Machine Learning (ICML)', pp. 961–968.

Watkins, C. J. C. H. (1989), Learning from Delayed Rewards, PhD thesis, King's College, University of Cambridge, UK.

Watkins, C. J. C. H. & Dayan, P. (1992), 'Q-learning', *Machine Learning* **8**(3-4), 279–292.

White, C. C. (1986), ‘A Posteriori Representations Based on Linear Inequality Descriptions of a Priori and Conditional Probabilities’, *IEEE Transactions on Systems, Man and Cybernetics* **16**(4), 570 –573.

White, C. C. & Eldeib, H. K. (1994), ‘Markov Decision Processes with Imprecise Transition Probabilities’, *Operations Research* **42**(4), 739–749.

White, D. J. (1985), ‘Real Applications of Markov Decision Processes’, *Interfaces* **15**(6), 73–83.

White, D. J. (1988), ‘Further Real Applications of Markov Decision Processes’, *Interfaces* **18**(5), 55–61.

White, D. J. (1993), ‘A Survey of Applications of Markov Decision Processes’, *Journal of the Operational Research Society* **44**(11), 1073–1096.

Wiering, W. & Schmidhuber, J. (1998), Efficient Model-Based Exploration, *in* ‘Proceedings of the Fifth International Conference on Simulation of Adaptive Behaviour: From Animals to Animats 5’, MIT Press/Bradford Books, pp. 223–228.

Wilson, A., Fern, A., Ray, S. & Tadepalli, P. (2007), Multi-Task Reinforcement Learning: A Hierarchical Bayesian Approach, *in* ‘Proceedings of the 24th International Conference on Machine Learning, Corvallis, OR’.

Winston, P. H. (1984), *Artificial Intelligence*, 2nd edn, Addison Wesley.

Wolpert, D. H. & Macready, W. G. (1997), ‘No free lunch theorems for optimization’, *IEEE Transactions on Evolutionary Computation* **1**(1), 67–82.

Wolpert, R. L. & Mengersen, K. L. (2004), ‘Adjusted likelihoods for synthesizing empirical evidence from studies that differ in quality and design: Effects of environmental tobacco smoke’, *Statistical Science* **19**(3), 450–471.

Wyatt, J. (1997), Exploration and Inference in Learning from Reinforcement, PhD thesis, Department of Artificial Intelligence, University of Edinburgh.

Wyatt, J. (2001), Exploration Control in Reinforcement Learning Using Optimistic Model Selection, *in* A. Danyluk & C. Brodley, eds, 'Proceedings of the Eighteenth International Conference on Machine Learning'.

Wyatt, J. (2002), Reinforcement Learning: A brief overview, *in* 'Perspectives on Adaptivity and Learning', Springer, pp. 243–264.

Wyatt, J., Hayes, G. & Hallam, J. (1999), Investigating the Behaviour of Q(λ), *in* 'Colloquium on Self-Learning Robots', London.

Xu, M. & Golay, M. (2006), 'Data-guided model combination by decomposition and aggregation', *Machine Learning* **63**, 43–67.

Zellner, A. (1988), 'Optimal Information Processing and Bayes's Theorem', *The American Statistician* **42**(4), 278–294, with invited discussion and the author's reply.

Zellner, A. (1997), *Bayesian analysis in econometrics and statistics: the Zellner view and papers*, Economists of the twentieth century, Edward Elgar, Cheltenham, UK.

Zellner, A. (2002), 'Information processing and Bayesian analysis', *Journal of Econometrics* **107**(1-2), 41–50.

Zhang, L. & Lam, J. (2010), 'Necessary and Sufficient Conditions for Analysis and Synthesis of Markov Jump Linear Systems With Incomplete Transition Descriptions', *IEEE Transactions on Automatic Control* **55**(7), 1695 –1701.

Zhong, Y., Jain, A. K. & Dubuisson-Jolly, M.-P. (2000), 'Object Tracking Using Deformable Templates', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22**(5), 544–549.