Sample-based Search Methods for Bayes-Adaptive Planning

Arthur Guez

Dissertation submitted for the degree of

Doctor of Philosophy

of

University College London

Gatsby Computational Neuroscience Unit

University College London

2015

DECLARATION

I, Arthur Guez, declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

Arthur Guez
February 3, 2015

Abstract

A fundamental issue for control is acting in the face of uncertainty about the environment. Amongst other things, this induces a trade-off between exploration and exploitation. A model-based Bayesian agent optimizes its return by maintaining a posterior distribution over possible environments, and considering all possible future paths. This optimization is equivalent to solving a Markov Decision Process (MDP) whose hyperstate comprises the agent's beliefs about the environment, as well as its current state in that environment. This corresponding process is called a Bayes-Adaptive MDP (BAMDP). Even for MDPs with only a few states, it is generally intractable to solve the corresponding BAMDP exactly. Various heuristics have been devised, but those that are computationally tractable often perform indifferently, whereas those that perform well are typically so expensive as to be applicable only in small domains with limited structure.

Here, we develop new tractable methods for planning in BAMDPs based on recent advances in the solution to large MDPs and general partially observable MDPs. Our algorithms are sample-based, plan online in a way that is focused on the current belief, and, critically, avoid expensive belief updates during simulations. In discrete domains, we use Monte-Carlo tree search to search forward in an aggressive manner. The derived algorithm can scale to large MDPs and provably converges to the Bayes-optimal solution asymptotically. We then consider a more general class of simulation-based methods in which approximation methods can be employed to allow value function estimates to generalize between hyperstates during search. This allows us to tackle continuous domains. We validate our approach empirically in standard domains by comparison with existing approximations. Finally, we explore Bayes-adaptive planning in environments that are modelled by rich, non-parametric probabilistic models. We demonstrate that a fully Bayesian agent can be advantageous in the exploration of complex and even infinite, structured domains.

ACKNOWLEDGMENTS

I would like to express my gratitude to Peter Dayan for his exemplary and passionate supervision. Peter patiently listened to my incomplete ideas and tirelessly revised my drafts. Our stimulating discussions not only informed my research directions for this thesis, they planted seeds for countless years of future investigations. I want to thank equally David Silver, my secondary supervisor, for his guidance and support. His pragmatism helped me direct my efforts towards reachable objectives, and his knack for deconstructing complex ideas into its essential components was invaluable. I am grateful to all of the Gatsby Unit residents for thought-provoking discussions, feedback, and for providing such a great and friendly environment for research. A special thanks to my office mates for enduring me (and my noisy keyboard) daily. Most of all, I would like to thank my family and my wife, Andrée-Anne, for their continuing love and support.

Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

I

Introduction

A key challenge in sequential decision making is to understand how agents[1] can learn to collect rewards — and avoid costs — through interactions with the world. There are two underlying, but interlinked, problems: learning about the environment (*exploring*) and gathering rewards (*exploiting*). The latter problem is one of planning, given knowledge of the environment. This knowledge takes the form of a model which provides a way to access, or sample, the dynamics of the environment so that trajectories can be internally simulated to assist decision making. For instance, a model of a helicopter's dynamics reports how the helicopter's state (its position, roll, yaw, pitch, etc.) will be modified, perhaps stochastically, as a function of the controls and other external factors. Given a model, the agent can *plan* to select actions that optimize future rewards. Planning must account for the long-term consequences of actions, which entails the consideration of, and optimization over, many possible future trajectories; thus, planning using some assumed model can be rather computationally challenging (Bertsekas and Tsitsiklis, 1996).

The other problem is learning a model of the environment in the first place (Tol-

[1]This terminology is borrowed from artificial intelligence. An agent is an abstract autonomous entity that acts. It could be realized as an animal, a robot, or a computer program. The term controller is often used instead.

man, 1948). For instance, the model for the helicopter's dynamics can be learned from interaction data. This itself has easy and difficult facets. The easy facet is supervised learning: given a fixed set of example transitions, the learning task is to generalize so that the outcome of arbitrary actions in arbitrary situations can be predicted accurately — a problem that can be addressed with a variety of statistical methods. The difficult facet is that the data for this supervised learning depends on the agent's own choices in the online *reinforcement learning* (RL) setting (Sutton and Barto, 1998), which is characterized by a continual closed-loop interaction with an unknown environment. Furthermore, until it has learned, the agent will typically be at least somewhat ignorant about how to collect reward, and so will need to collect information to do this more proficiently.

Since, ultimately, the goal of the agent is to maximize the collection of rewards, this leads to a blurring between exploiting existing knowledge to collect reward, and exploring to collect knowledge so as to get future reward. Exploring is generally costly, at least because of the missed opportunity of collecting rewards using the readily available knowledge. On the other hand, exploiting can always be improved in the future with more exploration, especially early on when not much learning has taken place. As a result, the problem of maximizing reward under model uncertainty involves a fine balance between these two conflicting behaviors. Weighing the benefits of exploring against the benefits of exploiting known sources of rewards is generally referred to as the *exploration-exploitation (EE) trade-off*.

This problem is not a theoretical curiosity. An abundance of scenarios exist in medicine, industry, robotics, policy making, finance, and science that exhibit the same tension between taking some time to acquire more data, potentially with some risk and cost attached, and harnessing what is currently known for generally sub-optimal gains. Animals also face this trade-off when they look for sources of food, avoid predators, or decide on mating partners; thus, practical solutions to this EE trade-off, including idiosyncrasies of efficient approximation schemes, may additionally inform modeling of decision making in animals and humans (Acuña and Schrater, 2010; Dayan, 2013; Huys and Dayan, 2009).

The canonical and minimalist example of this trade-off is found in stationary multi-armed bandit problems (Robbins, 1952) which consist of multiple slot machines with unknown random payoffs. One machine is on average better than the others, and the agent can play each machine one at a time to figure this out. Since repeatedly playing a machine will reduce uncertainty about its mean payoff, it will reveal the true value of that machine. Playing all the machines enough times will ultimately unmask the best machine. But how long should one play a machine before switching to a more promising one? What is a good strategy to maximize the overall payoff? How can we incorporate prior beliefs about the payoff distribution for each arm in the agent's strategy?

Thanks to decades of work on this problem, we now have a good grasp of the solutions for bandit problems, at least in some of their simpler forms (Auer et al., 2002; Gittins and Jones, 1974; Lai and Robbins, 1985) — we will describe the bandit problem in more details, along with its solutions, in Chapter 2. Nevertheless, unlike more general problems, bandits are stateless (each machine's outcomes are independent draws from a fixed distribution). This considerably simplifies exploration; for example, although the agent could make an unfortunate choice of machine from which to sample, this choice has no ramifications on the subsequent behavior of the machines. In the more general setting the agent can be in different states, implying that not all learning data is easily available at the pull of an arm. Instead, the agent has to navigate states through a series of actions to reach the part of the state space it would want to explore. Due to these additional complexities, and in contrast to the bandit case, practical solutions are still lacking to solve the EE trade-off in the general RL setting.

In this thesis, we take a Bayesian viewpoint where agents can optimally explore and exploit by following their beliefs about the environment — a setting where planning in the face of uncertainty is notoriously taxing. We develop new efficient methods for planning that can scale to previously impractical domains. After describing more formally the problem of acting under uncertainty, in particular the Bayesian formulation that we address, we summarize our contributions in Section 1.2.

1.1 A Bayesian Formulation of Acting under Model Uncertainty

To be precise about the nature of the EE trade-off and the exact RL problem to be solved, formalization is needed. In Chapter 2, we will introduce the necessary notation, formalisms, and tools we employ in this thesis; they are only summarized below to motivate our work.

1.1.1 Formalization

A natural way to characterize the agent's interactions with its environment is as a Markov Decision Process (MDP). MDPs consist of a set of states, a set of possible actions, and a transition kernel that stochastically decides a successor state from a given state and action (Puterman, 1994). In addition, a reward or cost is associated with each state and action. MDPs are employed to describe both the true, objective, interactions in the environment and often also the subjective, simulated, interactions the agent considers when planning with a model of the environment. The problem for learning arises when some aspects of the transitions (or rewards) are unknown to the agent, implying uncertainty about the best fixed *policy* that selects an action in each state for gathering rewards and avoiding costs. In the RL problem, the agent cannot rely on a fixed policy because the data accumulates over time and may affect which action the agent would want to select in any given state. Instead, the learning agent's policy, or *EE policy*, depends on both the current state and all the past data — it appears non-stationary when looked only as function of the state. To solve meaningfully the problem of acting under model uncertainty, it is necessary to ask what exactly constitutes a good EE policy, and how we should judge one over another.

Although we note the existence of other objectives in Chapter 2, our contributions focus on the Bayesian setting where we assume that the MDP is drawn from some *known* distribution. This distribution may not be the true distribution over MDPs but it encodes the agent's prior belief about the possible environments he may be in, hence it is subjectively taken as true by the agent. In the light of this prior distribution over models, Bayesian decision theory prescribes maximizing the expected discounted sum of rewards, where the discount weighs early rewards more heavily than future ones, a problem known as *Bayesian stochastic adaptive control* (Bellman and Kalaba, 1959; Kumar, 1985). This is the problem we address in this thesis. Although the agent faces a learning task, finding the optimal learning policy is purely a computational task because of the Bayesian formulation of the problem, which allows the agent 1) to foresee the effect of a future observation on its posterior distribution before any observation takes place[2] and 2) to evaluate the (subjective) probability of a particular event occurring — such as the outcome of an action in a given state.

Planning in this Bayesian setting involves reasoning about future potential beliefs since future optimal decisions — necessary to determine the current optimal decision — are affected by future beliefs, just in the way that future optimal decisions are affected by future states in fully-observable domains. Therefore, the agent's belief is another form of state, an *information state* (Bellman and Kalaba, 1959), and its dynamics have to be taken into account in order to optimally learn.

As an example, the bandit problem can be cast in a Bayesian way by assuming a prior distribution on the payoffs for each arm. Deciding which arm to pull from any given prior distribution so as to maximize the sum of discounted future rewards can be done in principle by enumerating all possible future interactions with the arms. Each future interaction trajectory will have a known probability associated with it according to the agent's prior. Moreover, each point along a trajectory can be identified by the information the agent has obtained about the arms, the information state corresponding to a posterior distribution on the payoffs. Since everything is known about these trajectories, the agent can plan (or optimize) to select the course of action that leads to the best return on average. An illustration of these trajectories in a simple bandit example is presented in Figure 1.1. It is worth noting that exploration happens implicitly as a consequence

[2]This counterfactual reasoning is sometimes called *preposterior analysis* in statistics.

of maximizing rewards in this Bayesian setting. Without introducing any artificial incentive for exploration, the agent can perceive an uncertain arm as valuable since many likely trajectories that start by pulling that arm reach an information state where the arm is likely to be rewarding (for example state A in Figure 1.1). While there also exist likely trajectories that start by pulling the same arm which reach states where the arm does not seem rewarding (for example state B in Figure 1.1), the agent still has the option of exploiting other arms from these states, thereby mitigating the impact of these more negative trajectories. Combining that information, it may appear attractive to pull the uncertain arm purely from a reward-maximizing viewpoint; there does not need to be an additional external mechanism to encourage exploration.[3]

1.1.2 Finding the Optimal Learning Plan

There exists a special optimization procedure to maximize the sum of discounted rewards for the case of bandits — using Gittins indices (Gittins and Jones, 1974). For MDPs in general, one way to compute the optimal learning plan is by solving an MDP with augmented states, called the *Bayes-Adaptive MDP* (BAMDP), in which the corresponding augmented dynamics are known (Duff, 2002; Martin, 1967). The state augmentation is the posterior belief distribution over the dynamics, given the data so far observed. In other words, the augmented state contains both the regular MDP state and the information state. The dynamics of the BAMDP are known because information evolves according to known rules and regular states evolve according to this information; to obtain the probability of a BAMDP transition from a given augmented state, we integrate over all the possible transition probabilities according to the belief — itself derived from the information state.

Since everything is known about the BAMDP, it can in principle be solved to obtain its optimal policy, providing the optimal action for all possible states and beliefs. The agent starts in the augmented state corresponding to its prior and,

[3]For example, in Figure 1.1, the blue arm, with the lower prior mean, is actually the optimal arm to play for a discounting factor of 0.99.

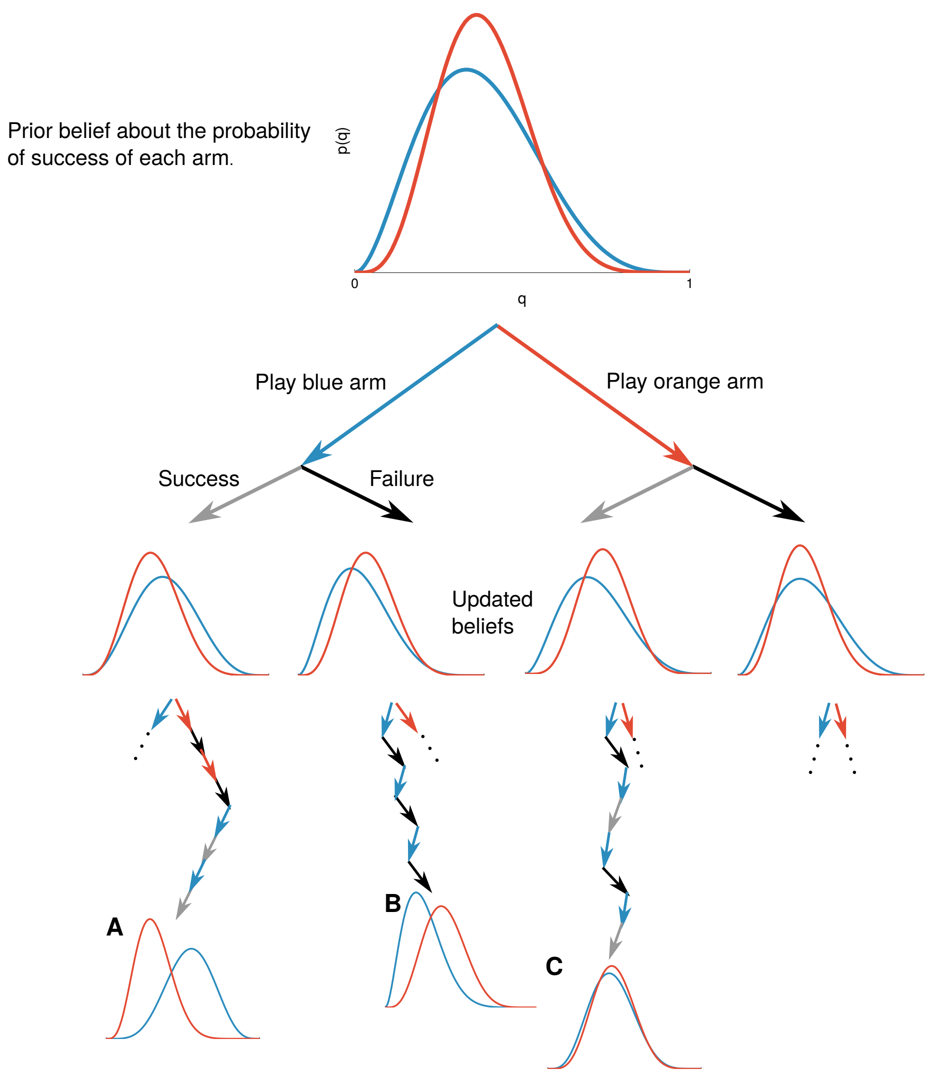


Figure 1.1: Example of Bayesian bandit problem with 2 possible arms (blue and orange), each giving a payoff of 1 (success) or 0 (failure) with some unknown probability. The initial belief about these probabilities is in the top plot, which indicates a bit more certainty about the probability of success of the orange arm. In the center of the figure are all the possible beliefs that could arise after seeing the outcome of a single action — notice that the belief about the blue arm is unchanged if the orange arm is played (and vice versa). Each of these outcomes happens with some known probability according to the belief. We can recursively consider all future events in the same way. We only show a selection of 3 possible partial trajectories from this point where: **A** the orange arm appears to be worse than the blue arm, **B** the blue arm is likely to be worse than the orange arm, **C** the beliefs for both arms are relatively similar. Given all these trajectories, the agent needs to find the best arm to pull at the top to maximize its average sum of discounted payoffs.

by executing the greedy policy in the BAMDP whilst updating its posterior, acts optimally in the environment *with respect to its beliefs*. The Bayes-optimal policy is the optimal policy of the BAMDP: it integrates exploration and exploitation in an ideal manner with respect to its prior knowledge so as to maximize the expected discounted sum of rewards. *Bayes-Adaptive planning* is the process of computing such an optimal policy.

One attractive feature of the Bayesian framework is that structured prior knowledge can be incorporated into the solution in a principled manner. When such prior knowledge is available, it is duly reflected in the Bayes-optimal policy and allows the agent to balance exploration and exploitation in a structured way justified by its beliefs. For example, such directed exploration may ignore parts of the environment where reward gains are likely to be low and information useful to exploiting more surely lucrative regions is unlikely to be obtained. If there are different ways to reach a particularly useful state of knowledge, the Bayes-optimal policy will select the least expensive way of reaching it, taking into account how knowledge (in the form of beliefs) evolves based on new observations. By carefully prioritizing what to do according to existing beliefs, the Bayesian framework provides the means to tackle, at least in theory, large and complex unknown environments in a principled way.

Unfortunately, the exact Bayes-adaptive (BA) solution is computationally intractable. Various algorithms have been devised to approximate optimal learning, but often at rather large cost. This computational barrier has restricted Bayesian adaptive control to small domains with simple priors, preventing its theoretical advantage to be realized in many potential application domains. In this thesis, as a step towards practical applications, we directly address the shortage of practical solutions by introducing new approximation methods for Bayesian adaptive control that can deal more efficiently with both large state spaces and complex priors.

1.2 Bayes-Adaptive Planning

Even if the true underlying MDP only contains a small number of states, the resulting BAMDP contains all possible beliefs over the MDP dynamics. The augmented MDP's state space is therefore in general infinite, even for the smallest problems. To tackle the problem of planning in this BAMDP, it is thus natural to rely on planning techniques that are adapted to large (or infinite) MDPs.

In known environments with available models, there are two main paradigms for planning. It can be done *offline* to find the optimal policy for all states; this requires some preprocessing computation but results in a policy that can then be executed online with little effort. Alternatively, planning can be carried out *online* separately for each visited state. The concept of online planning was inspired by early tree search techniques for games (Korf, 1990), a setting where the large number of states often prevents offline planning (for example in the game of chess).

In the context of MDPs, a powerful notion introduced by Kearns et al. (1999) is that the complexity of planning offline for all states (which grows linearly with the state space size) can be contained by planning online with a complexity independent of the size of the state space — albeit with an exponential cost in the planning horizon — using a technique called *sparse sampling*. Simply said, sparse sampling builds a look-ahead tree of future paths based on sampled transitions to optimize the policy. It therefore simulates possible futures from the current state. By sampling transitions rather than considering all possible transitions, the dependence on the size of the state space is avoided but the method still provably finds a (near-)optimal policy. Another useful aspect of sparse sampling is that it only requires a *generative model* of the transitions, as opposed to explicit transition probabilities, further widening the applicability of the method.

One issue with sparse sampling is that it expands the search tree uniformly. Since the tree grows in an exponential way, this restricts sparse sampling to small search depths in practice, preventing it from perceiving rewards past a short horizon. Rather than truncating search, the Monte-Carlo tree search (MCTS) algorithm (Kocsis and Szepesvári, 2006), an extension of sparse sampling, runs simulations from the tree leaves using a sub-optimal policy to obtain an estimate of the return for longer horizons. In addition, MCTS grows its search tree in a non-uniform way based on these returns to spend more resources searching in promising regions. By effectively ignoring entire subtrees, the tree can be extended much deeper along some trajectories and MCTS avoids wasting resources on unpromising tree branches. As a result, MCTS has been able to tackle large fully-observable problems where other approaches have failed (Gelly et al., 2012).

When planning under model uncertainty in a Bayesian setting, the same dichotomy between online and offline planning applies (Duff, 2002; Wang et al., 2005). Computing the Bayes-optimal policy offline requires considering all possible states and beliefs, while planning online seeks the best action only for the current state and belief. Therefore, given the infinite augmented state space containing both states and beliefs, online sample-based planning seems particularly well-suited for Bayes-adaptive planning. The same computational constraints apply in Partially Observable MDPs, a closely connected problem to Bayesian adaptive control where the state, and not the dynamics, is partially observed and where online sample-based planning has been considered with some success (Ross et al., 2008; Silver and Veness, 2010).

In this thesis, we present tractable approaches that exploit and extend recent advances in sample-based online planning in MDPs — such as MCTS — and Partially Observable MDPs (POMDP) for the problem of BA planning. A common theme in our approach is to exploit sampling to reduce computational complexity.

1.2.1 Contribution 1: The BAMCP Algorithm

Some existing methods for BA planning, which we will review in Chapter 2, already take advantage of sparse sampling (Asmuth and Littman, 2011; Ross and Pineau, 2008; Wang et al., 2005). However, they usually suffer from at least two kinds of intractabilities. First, the large cost of optimizing the search tree in sparse sampling prevents deep searches — an issue also found in the fully-observable setting which we highlighted above. Second, another aspect of intractability not addressed by previous work is the cost of using the generative model necessary to sample forward trajectories in the BAMDP when planning. Classically, this requires updating the posterior distribution and generating BAMDP transitions by integrating the corresponding posterior at each simulation step. Since these operations can only be carried out cheaply for simple prior distributions, this severely limits the applicability of online planning methods to Bayesian adaptive control.

In Chapter 3, we propose an algorithm that extends a version of MCTS for POMDPs (Silver and Veness, 2010) to the BA setting. Since the BAMDP corresponding to a Bayesian adaptive control problem can be viewed as a regular, albeit large or infinite, MDP, the MCTS algorithm can be applied to plan online. This leads to a non-uniform expansion of the search tree, using the sampled returns to guide this process. As in the fully-observable case, this enables more effective tree searches and addresses the first issue. However, we show that a naive application of MCTS to the BAMDP is not tractable in general, principally because it does not address the second issue. We propose a set of principled modifications to obtain a practical algorithm, which is called BAMCP for 'Bayes-Adaptive Monte-Carlo Planner'.

We directly address the second issue of intractable BAMDP transitions by relying on a different trajectory sampling scheme, one that avoids updating the posterior belief state when simulating future trajectories. Instead, BAMCP relies on a single MDP sample from the current posterior *for each simulation*, and leverages the sampled model to generate all the MDP transitions for that simulation; thus avoiding repeated applications of Bayes rule. This method was introduced by Silver and Veness (2010) in the context of POMDPs. We refer to it as *root sampling* because MDP samples are only generated at the root of the search tree; we also generalize the scope of this method in Chapter 5. To increase computational efficiency further, we introduce an additional innovation: a lazy sampling scheme that only samples the variables in the posterior distribution that are necessary for a given simulation.

Theoretically, we show that BAMCP converges to the Bayes-optimal solution, thereby establishing that these computational advantages have a principled foundation. Moreover, we show that this convergence result holds even when combined with some forms of approximate inference schemes.

Empirically, we show in Chapter 4 that BAMCP consistently and significantly outperforms existing Bayesian control methods, and also recent non-Bayesian approaches, on a representative sample of benchmark problems. We also show that BAMCP can tackle a domain with an infinite number of states and a structured prior over the dynamics, a challenging, if not radically intractable, task for existing approaches.

1.2.2 Contribution 2: Generalizing the BAMCP Algorithm

Tree-search methods optimise the policy by maintaining the expected return, or *value*, of the current policy at each tree node (each corresponding to a state, or an augmented state). One major limitation of MCTS, and other tree-search algorithms, is that they fail to generalize values between related states, as a separate value is stored for each distinct path of possible interactions. In the BA case, algorithms like BAMCP fail not only to generalize values between related paths, but also to fail to reflect the fact that different partial trajectories can correspond to the same belief about the environment — since data obtained in different ways may result in the same belief. As a result, the number of required simulations grows exponentially with search depth. Worse yet, except in very restricted scenarios, this lack of generalization renders Monte-Carlo search algorithms effectively inapplicable to BAMDPs with continuous state spaces. To address this problem, we propose in Chapter 5 a class of efficient simulation-based algorithms for BA planning which use function approximation to estimate the value of interaction histories during search. This enables *generalization* between different beliefs, states, and actions during planning, and therefore also works for continuous state spaces. These algorithms build on the BAMCP algorithm and exploit value function approximation for generalization across trajectories, similar to simulation-based search algorithms for MDPs (Silver et al., 2012). As a crucial step towards this end, we develop a suitable parametric form for the value function estimates that can generalize appropriately across trajectories, while remaining invariant to partial reorderings that do not modify beliefs.

Experimental results in Chapter 5 demonstrate the viability of the approach in continuous domains. They also show that value generalization can lead to more efficient planning even in discrete domains.

1.2.3 Contribution 3: BA Planning with Rich Models

Given all the computational intractibilities, it is not unfair to question whether the Bayes-adaptive approach actually has any advantage over simpler alternatives. Indeed, except for certain types of bandit problems, the literature does not contain any real application of such methods, and published examples have been restricted to domains where BA planning only leads to marginal improvements over approaches that plan myopically and more cheaply. We believe the lack of significant improvements arise partly as a result of fully Bayesian planning being traditionally coupled with simple priors. For it has not hitherto been possible to exploit more complex forms of prior knowledge to their full extent in a BA setting due to computational complexity.

Building on our algorithmic contributions, another objective of this thesis is to demonstrate the practical power of Bayes-adaptive planning in situations where rich and structured prior knowledge is available. Thus, in Chapter 6, we provide two sorts of evidence in its favour. First, we consider non-parametric contextual bandit tasks that contain repetitive structure and require careful exploration. We show that the Bayes-adaptive policy found by our BAMCP algorithm in these tasks performs dramatically better than myopic forms of planning and varies with the prior and the horizon to reflect the changing optimization objective. This case study illustrates the feasibility of propagating complex beliefs forward in an exploration-exploitation setting to determine an appropriate course of action, and how this results in superior performance compared to more naive or uninformed exploration strategies. Second, we show that the benefits of Bayesian inference with rich models can be squandered by more myopic forms of Bayesian planning (i.e., planning that does not reason about future beliefs). We illustrate these modes of failure in the tasks described above as well as in a series of counterexamples.

1.3 Summary of Contributions

In this thesis, we provide evidence that our contributed algorithms are particularly well suited to support BA planning in large domains. This allows us to test BA planning in complex settings (large number of states, continuous-state spaces, complex priors, relatively long horizons) and witness the significant advantage that it can have, in terms of maximizing rewards, against more naive exploration-exploitation approaches. Unsurprisingly, the advantages of BA planning come at a computational cost which threatens to hinder their applicability. Our general strategy to mitigate these costs is to rely on sampling in order to reduce the effective search dimensions and focus on those that matter.

In more general terms, our thesis is that by relying on sample-based reinforcement-learning methods, Bayes-adaptive planning can be scaled to realistically large problems and lead to significant improvement over more heuristic methods. Thus, while a popular strategy to handle the exploration-exploration trade-off has been to *approximate the problem* to be solved in order to obtain practical algorithms, we show that it is feasible, and desirable, to directly *approximate the solution* of the original, but intractable, problem. Chapter 7 discusses how the contributions in this thesis could be extended with further approximation strategies to deal with even larger, real-world, domains.

II

Existing Work

This chapter reviews existing work that addresses the problem of acting under model uncertainty in MDPs, with a particular emphasis on Bayesian methods. First, in Section 2.1 we present the relevant formalisms for decision making as a whole (bandit, MDP, POMDP, and BAMDP); then, in Section 2.2, we cover corresponding solution methods (e.g., Dynamic Programming, RL algorithms, planning algorithms). Finally, in Section 2.3, we review the existing work that addresses the Bayesian statistical modeling problem for MDPs.

2.1 Problem Types and Formalisms

Although multi-armed bandit problems are special cases of MDPs, presenting them first (Section 2.1.1) allows us to formalize the critical different notions of exploration without getting lost in notation. We then present the more general MDP formulation, along with corresponding objectives for EE in the MDP setting (Section 2.1.2). As previously mentioned, the Bayesian formulation of the EE problem corresponds to an augmented MDP, the BAMDP, which we describe in

Section 2.1.3. Finally, in Section 2.1.4, we draw the link between BAMDPs and Partially-Observable MDPs, the setting where the state is partially observed.

2.1.1 Multi-armed Bandits

A *multi-armed bandit* problem is composed of A different slot machines. Pulling the arm of the a-th machine results in a random reward drawn from a distribution $H(\theta_a)$ parametrized by θ_a and with mean $\mu_a(\theta_a)$ associated with that machine. The payoff sequence for a single arm is formed of identically distributed draws from $H(\theta_a)$, thus the arms are stateless. The agent chooses an arm at every step so, if the payoff parameters are known, the agent can act optimally by simply pulling the arm a^* with the highest expected payoff, $a^* = \operatorname{argmax}_{a \in A} \mu_a(\theta_a)$, at every step.[1] Furthermore, the payoff from arms that are not selected on a step are not revealed to the agent. For example, in Bernoulli bandits, H is simply the Bernoulli distribution, with its parameter $\theta_a = \mu_a$ being the probability of success for arm a. Success corresponds to a payoff of 1, and the payoff is 0 otherwise.

For the adaptive control problem, it is assumed that $\boldsymbol{\theta} = [\theta_1 \dots \theta_A]$ is a hidden, or unknown, variable. Bandit problems do not have the complexity of more general sequential decision tasks since decisions do not have long-term consequences (as the system is stateless, only the state of knowledge changes when pulling an arm). Despite their apparent simplicity, they present many of the complex issues of exploration-exploitation that appear in more general settings, and their study is still an active research area.

Thompson (1933) first introduced a bandit problem with two arms and Bernoulli distributions, motivating it as a decision model to select medical treatments. Robbins (1952) extended the model to more general payoff distributions and showed that a control law, or allocation rule, could be designed that is guaranteed (with probability 1) to find the best arm (highest mean) in the limit of infinitely many interactions.

Rather than only expecting asymptotically optimal behavior, Lai and Robbins

[1] To simplify notation, we assume there is a unique best arm.

(1985) considered minimizing the rate of growth of the *regret* (the difference between the expected total payoff obtained by a strategy and the best possible achievable total payoff if the identity of the best arm had been known). In that evaluation scheme, all rewards are accounted for, but the regret is usually an undiscounted notion (i.e., later losses count as much as early losses). To formalize this notion, let $\tilde{\Pi}$ be the set of possible adaptive strategies, where each such policy $\tilde{\pi} \in \tilde{\Pi}$ is a map from an observed history of past arm pulls and obtained rewards $h = a_1 r_1 a_2 r_2 \ldots a_{t-1} r_{t-1}$ (i.e., all the existing data) to the next action a_t, $\tilde{\pi} : \mathcal{H} \rightarrow [1, \ldots A]$, with $\mathcal{H}$ the set of possible observation sequences. Then, the expected regret for policy $\tilde{\pi}$ after n steps in the bandit problem described by $\boldsymbol{\theta}$ is:

$$\Delta(\boldsymbol{\theta}, \tilde{\pi}, n) := n\mu_{a^*} - \mathbb{E}\left[\sum_{m=0}^{n} R_m \mid \tilde{\pi}\right], \tag{2.1}$$

where R_m is the random reward obtained at the m-th trial. Lai and Robbins (1985, Theorem 1) proved that, for any configuration $\boldsymbol{\theta}$ of the arms, the regret of an adaptive policy $\tilde{\pi}$ (satisfying some minimum efficiency condition) in a bandit problem is lower-bounded asymptotically:

$$\lim_{n\rightarrow\infty} \inf \Delta(\boldsymbol{\theta}, \tilde{\pi}, n) \geq \log(n) \sum_{a \neq a^*} \frac{\mu_{a^*} - \mu_a}{\mathrm{KL}(H(\theta_a)||H(\theta_{a^*}))}, \tag{2.2}$$

where $\mathrm{KL}(H(\theta_1), H(\theta_2))$ denotes the Kullback-Leibler divergence between the reward distributions parametrized by θ_1 and θ_2. In regret-based methods, the agent competes against the strategy that always pulls the best arm, with an average payoff of μ_{a^*} per time-step. To achieve a sub-linear regret as a function of n, the agent must asymptotically only pull a^*. Finding the identity of a^* requires the agent to pull all arms (explore), but the agent must be careful to avoid frequently pulling arms that are almost certainly sub-optimal (i.e., different than a^*) since this increases the regret (exploit). The stochasticity of the feedback (success or failure for Bernoulli bandits) means the agent will never be completely certain of the identity of a^*; therefore it must continue to explore for all finite n. Exploration does not have to be uniform across arms, finding an efficient way to allocate exploration between arms is necessary to obtain a competitive regret rate.

Bellman (1956) adopted a Bayesian approach to the 2-armed Bernoulli bandit problem, leading to an adaptive control process that could be solved with dynamic programming. Here, the Bayesian agent assumes each θ_a is drawn from a Beta prior: $\theta_a \sim$ Beta(α_a, β_a) for each a, where α_a and β_a are hyperparameters. The arm that maximizes the expected sum of discounted future payoffs given the current posterior distribution $P(\{\theta_a\} \mid h)$ is selected at each step. This pushes the Bayesian agent to explore and exploit by following its prior. In doing this, it may ignore arms believed to be inferior, even if they turn out to be objectively superior. Gittins and Jones (1974) generalized this Bayesian bandit problem to cover a larger class of payoff distributions and essentially solved the Bayesian formulation of the bandit problem, as we will describe in the section on solution methods (Section 2.2).

More recently, bandit problems have been extended to allow for continuous selection of arms (Kleinberg, 2004), a problem related to stochastic optimization, or to incorporate a context that influences the return of the arms (Langford and Zhang, 2007). We will be covering a form of contextual bandit problem in a Bayesian setting in Chapter 6.

2.1.2 Markov Decision Processes

Markov Decision Processes (MDP) constitute a popular and mature formalism for sequential decision making. The Markov assumption underlying each process says that the outcome of an action in a state is independent of past states and actions. Formally, a discrete-time infinite-horizon Markov Decision Process is described as a 5-tuple $M = \langle S, A, \mathcal{P}, \mathcal{R}, \gamma \rangle$, where S is the set of discrete states, A is the finite set of actions, $\mathcal{P} : S \times A \times S \rightarrow [0, 1]$ is the state transition probability kernel, $\mathcal{R} : S \times A \rightarrow \mathbb{R}$ is a bounded reward function, and $\gamma < 1$ is the discount factor (Puterman, 1994).

A deterministic stationary MDP policy π is defined as a mapping $\pi : S \rightarrow A$ from states to actions. Although optimal policies in MDPs are generically deterministic, it is sometimes useful to consider more general stochastic policies $\pi : S \times A \to [0,1]$ — it should be clear from context whether we are referring to a deterministic or stochastic policy. The *value function* of a policy π at state $s \in S$ is its expected (discounted) return, defined as:

$$V^{\pi}(s) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, \pi\right], \tag{2.3}$$

where r_t is the random reward obtained at time t when following policy π from state s — $\mathbb{E}$ denotes the expectation operator that averages over all possible paths that policy π implies. V^{π} is the solution of an associated Bellman equation:

$$V^{\pi}(s) = \sum_{a \in A} \pi(s,a)\left[R(s,a) + \gamma \sum_{s' \in S} \mathcal{P}(s,a,s') V^{\pi}(s')\right] \quad \forall s \in S. \tag{2.4}$$

A related quantity is the *action-value function* of a policy π for executing a particular action $a \in A$ at state $s \in S$ before executing π:

$$\begin{aligned} Q^{\pi}(s,a) &:= \mathcal{R}(s,a) + \gamma \sum_{s' \in S} \mathcal{P}(s,a,s')\mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} r_t \mid s_1 = s', \pi\right] & (2.5) \\ &= \mathcal{R}(s,a) + \gamma \sum_{s' \in S} \mathcal{P}(s,a,s') V^{\pi}(s') \quad \forall s \in S, \forall a \in A, & (2.6) \end{aligned}$$

implying the relation $V^{\pi}(s) = \sum_a \pi(s,a) Q^{\pi}(s,a)$. Finding V^{π}, or Q^{π}, is referred to as the *policy evaluation*, or prediction, problem. A policy corresponding to a particular Q function, that is a policy that selects $\pi(s) = \operatorname{argmax}_{a \in A} Q(s,a)$, is called *greedy* with respect to Q.

The *optimal* action-value function, denoted Q^*, provides the maximum expected return $Q^*(s,a)$ that can be obtained after executing action a in state s. It satisfies the Bellman optimality equation:

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s' \in S} \mathcal{P}(s,a,s') \max_{b \in A} Q^*(s',b) \quad \forall s \in S, \forall a \in A. \tag{2.7}$$

The optimal value function, V^*, is similarly defined and is related to Q^* as $V^*(s) = \max_{a \in A} Q^*(s,a)$. An optimal policy π^* achieves the maximum expected return from all states, and can be obtained from Q^* as: $\pi^*(s) = \operatorname{argmax}_{a \in A} Q^*(s,a)$, breaking ties arbitrarily.

Undiscounted Formulations

We may also consider undiscounted MDPs ($\gamma = 1$) in a finite-horizon setting with few changes to the notation, since a finite-horizon MDP can be transformed into a infinite-horizon MDP with an additional state component for the current time and with a terminal state where the agent stays foreover without accruing any rewards.

The undiscounted setting combined with an infinite horizon needs to be treated differently simply because the expected return might turn into an infinite sum of rewards. One way around this issue is to look at the *average reward* per step (or gain) of a policy π from a state s:

$$\rho^\pi(s) := \lim_{T \to \infty} \frac{\mathbb{E}\left[\sum_{t=0}^{T} r_t \mid s_0 = s, \pi\right]}{T}. \tag{2.8}$$

In ergodic MDPs, there is a limiting state distribution independent of the start state which implies that the average reward of a policy will be the same for all states; we can then drop the dependence on s and write $\rho^\pi := \rho^\pi(s)$ for any s.[2] The optimal average reward is denoted $\rho^*(s) := \sup_\pi \rho^\pi(s)$; we only need the weaker condition that all pairs of states are reachable with some positive probability under some policy (i.e., we need the MDP to be communicating) to drop the dependence on s for the optimal reward rate.

Optimizing ρ^π leads to *gain-optimal* policies, but gain-optimality is not a satisfying notion of optimality in itself. Many policies can attain the optimal average reward rate ρ^*, yet they may take an arbitrarily long time to reach it since there is no pressure in this objective to quickly converge to the asymptotic rate (Howard, 1960; Schwartz, 1993). Therefore, it is desirable to also optimize the transient

[2]An MDP is ergodic if the Markov chain induced by any policy is ergodic. An ergodic Markov chain is one that is recurrent and aperiodic, meaning that every state will be visited infinitely often without any particular period.

part of the policy; and one option is to consider the differential value of a policy:

$$D^{\pi}(s) := \lim_{T\to\infty} \mathbb{E}\left[\sum_{t=0}^{T}(r_t - \rho^{\pi}) \mid s_0 = s, \pi\right], \tag{2.9}$$

the average-adjusted expected sum of rewards when running the policy from state s, which expresses the bias value of starting in s against other states (Mahadevan, 1996; Schwartz, 1993). A policy π^*_b is then *bias-optimal* if 1) $\rho^{\pi^*_b} = \rho^*$ (i.e., it is gain-optimal) and 2) $D^{\pi^*_b}(s) \geq D^{\pi}(s)$ for all $s \in S$ and all policies $\pi \in \Pi$. As for the discounted case, we can derive Bellman equations from the differential value definition. It is worth noting that, for ergodic MDPs, the discounted optimal policy converges to the bias-optimal policy as $\gamma \to 1$ (Tsitsiklis and Van Roy, 2002).

Continuous state spaces

MDPs can also be defined more generally with continuous state spaces S, such as $S \subseteq \mathbb{R}^n$. Summations over states in the equations above need to be replaced by integration over the state space, and probability mass functions need to be replaced with probability densities (under some technical conditions, see Puterman (1994, Section 2.3) for details).

2.1.2.1 Objectives for Exploration-Exploitation in MDPs

When the model of the dynamics ($\mathcal{P}$) or rewards ($\mathcal{R}$) is unknown or only partially known, the agent may still interact with the system and can learn to optimize its behavior over time. In the special case of bandit problems (Section 2.1.1), optimizing the behavior amounts to searching the most rewarding *arm*. In MDPs, optimizing the behavior requires looking for a rewarding *policy*. While an arm can be (stochastically) evaluated in bandits in a single step, policies require many steps of interaction to be evaluated in an MDP. A link to the bandit setting can be drawn if we consider testing a given policy as a macro-action. The macro-action is to commit to a given policy in Π for a full episode. By executing a selected policy $\pi \in \Pi$ and by adding all the rewards along the policy's trajectory, we can obtain an estimate of the mean return V^π — just like selecting an arm gets us an estimate of the mean payoff in bandits. The EE problem then becomes a bandit problem with each arm corresponding to a different policy (Deisenroth et al., 2013; Goschin et al., 2012); however, due to the huge number of possible policies, it is restrictive and can be costly to only rely on these macro-actions to act under model uncertainty.

Since the agent is continuously learning about the dynamics from its interaction with the MDP, we do not want its behavior policy to be stationary. In general, the learning policy at time step t depends, in addition to the current state, on the history $h_t = s_0 a_0 r_0 s_1 a_1 r_1 \ldots s_{t-1} a_{t-1} r_{t-1} s_t$ of past states, actions, and rewards, which contains all the information that the agent has observed about $\mathcal{P}$ and $\mathcal{R}$.[3] Therefore, we can write the policy of the agent during learning as $\tilde{\pi} : S \times \mathcal{H} \to A$ (or more generally $\tilde{\pi} : S \times \mathcal{H} \times A \to [0, 1]$ for stochastic policies), where $\mathcal{H}$ denotes the set of possible histories. $\tilde{\Pi}$ denotes the set of all history-dependent policies, we will sometimes refer to them as adaptive policies or exploration-exploitation (EE) policies.[4]

We would now like to quantify the policies in $\tilde{\Pi}$ in terms of their desirability for the adaptive control problem. As we discussed in more general terms in the introduction, there are different ways to proceed. We formalize different objectives in the context of MDPs in this section, including the Bayesian objective which is the focus of this thesis.

[3]The redundancy in the state-history notation throughout this thesis, namely that the current state could be extracted from the history, is only present to ensure clarity of exposition.

[4]Because of its history dependency, in general an EE policy is a non-stationary policy as a function of the state.

Non-Bayesian Objectives

— PAC-MDP

One possible demand we might have for an exploration policy is that it produces a near-optimal policy after a small number of steps in the environment, without accounting for rewards during the learning period. In other words, we want to minimize the number of steps from which the adaptive policy does not achieve a near-optimal return on average, a constraint which also limits the number of suboptimal actions. To formalize the notion of unaccounted learning steps, we define the *sample complexity* of an exploration policy in Definition 1.

Definition 1 *(Adapted from Kakade (2003)) Let* $h_t = s_0 a_0 r_0 s_1 a_1 r_1 \dots s_{t-1} a_{t-1} r_{t-1}$ *be a random path generated by executing a policy* $\tilde{\pi}$ *in an MDP* M*. For any* $\epsilon > 0$*, the* ***sample complexity (of exploration)*** *of* $\tilde{\pi}$ *is the number of timesteps* t *such that the return of the non-stationary policy* $\tilde{\pi}$ *from step* t *is not* ϵ*-optimal in* M *from the current state. Formally, let* $V_M(h_t; \tilde{\pi})$ *be the expected discounted return of policy* $\tilde{\pi}$ *in the true MDP from step* t *onwards (after having observed* h_t*), then the sample complexity is the number of time steps for which* π_t *satisfies* $V_M(h_t, \tilde{\pi}) < V^*_M(s_t) - \epsilon$*.*

One way to formalize the objective of low sample complexity is to require the sample complexity to be polynomial in the parameters of the problem with some probability. This is described formally in Definition 2.

Definition 2 *(Adapted from Strehl et al. (2006)) An algorithm* $\mathcal{A}$ *(inducing an exploration policy* $\tilde{\pi}$*) is said to be* ***PAC-MDP*** *(Probably Approximately Correct in Markov Decision Processes) if, for any* $\epsilon > 0$ *and* $0 < \delta < 1$*, the sample complexity is less than some polynomial in the quantities* $\{|S|, |A|, 1/\epsilon, 1/\delta, 1/(1-\gamma)\}$ *with probability at least* $1 - \delta$*.*

This notion of PAC-MDP has to hold for every MDP the algorithm encounters, therefore algorithms that satisfy the PAC-MDP property usually rely on large deviation bounds (e.g., the Chernoff bound) to guarantee enough confidence in their estimates. Instead of expecting to perform well in every MDP equally, one can consider a prior over MDPs and tailor the PAC statement to that prior, so that bad performance in a rare MDP will be absorbed by the probabilistic nature of the performance statement (Mannor and Tsitsiklis, 2004). The PAC-MDP objective can also be defined for the undiscounted setting (Kakade, 2003). More details on PAC-MDP methods can be found in a recent review of PAC analysis in finite MDPs (Strehl et al., 2009).

Regret

Another way to rank policies in $\tilde{\Pi}$ is to consider their regret. The undiscounted sum of rewards gathered in T time steps by an exploration policy $\tilde{\pi}$ in an MDP M starting from state s is denoted by the random variable $X(M, \tilde{\pi}, s, T) = \sum_{t=1}^{T} r_t$, and the expected average reward of these T steps is simply $\mathbb{E}[X(M, \tilde{\pi}, s, T)]/T$. As T grows to infinity, we obtain the previously defined average reward (in the EE setting, we make explicit the dependence on the underlying MDP M) $\rho^{\tilde{\pi}}(M, s) = \lim_{T \to \infty} \mathbb{E}[X(M, \tilde{\pi}, s, T)]/T$ and optimal average reward $\rho^*(M) = \rho^*(M, s) = \max_{\tilde{\pi} \in \tilde{\Pi}} \rho^{\tilde{\pi}}(M, s)$. Using this notation, as for bandit problems, we can now define the total regret of an exploration policy in Definition 3.

Definition 3 *(Adapted from Jaksch et al. (2010)) The **total regret** of an exploration policy $\tilde{\pi}$ in an MDP M after T steps from state s is defined as $\Delta(M, \tilde{\pi}, s, T) = T\rho^*(M) - X(M, \tilde{\pi}, s, T)$.*[5]

The total regret provides an objective to minimize; the quality of a given algorithm (inducing an exploration policy) is usually presented as an upper bound on the expected regret rate. The expectation may be taken only over the stochasticity of the interaction, or may be amortized over a prior over MDPs (Osband et al., 2013). Sublinear growth in the regret implies that the adaptive control converges asymptotically to the performance of the optimal policy for the underlying MDP. As for the bandit setting, it is possible to prove lower bounds on the regret of any

[5] If we set $\tilde{\pi} = \pi^*$ (the optimal policy if we knew the MDP), then the limit of the expected total regret simply corresponds to the differential value of π^* as defined in Equation 2.9.

EE policy (Jaksch et al., 2010, Theorem 5), in the sense that for any EE policy there exists an MDP for which the expected regret of that policy is greater than that lower bound.

Bayesian Objectives

In the Bayesian formulation of optimal behavior in an uncertain MDP, the Bayesian agent starts with a prior belief over the dynamics $P(\mathcal{P}, \mathcal{R})$ of the MDP M and selects actions so as to maximize its expected discounted return for an infinite-horizon setting with respect to this prior. Since uncertainty about rewards can in most cases be transformed into uncertainty about dynamics — by adding states that are rewarded differently — we will assume $\mathcal{R}$ is known and so the prior belief is simply written as $P(\mathcal{P})$. Formally, given S, A, $\mathcal{R}$, γ, we define the expected discounted return $J(q, \tilde{\pi})$ starting from a start-state distribution $q : S \to [0, 1]$ when following an adaptive policy $\tilde{\pi} \in \tilde{\Pi}$ as:

$$J(q, \tilde{\pi}) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid q, \tilde{\pi}\right] \tag{2.10}$$

$$= \sum_{s \in S} q(s) \int_{\mathcal{P}} P(\mathcal{P})\, \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, \tilde{\pi}, M(\mathcal{P})\right] \mathrm{d}\mathcal{P}, \tag{2.11}$$

where $M(\mathcal{P})$ is the MDP $\langle S, A, \mathcal{P}, \mathcal{R}, \gamma\rangle$. The task is then to find $\tilde{\pi}$ that maximizes $J(q, \tilde{\pi})$ for a starting state distribution q. This is the Bayesian stochastic adaptive control problem formulated in the language of the MDP formalism.

If the state space is finite, a maximizing adaptive strategy exists, as is made clear in the following statement.

Definition 4 *Given S (finite), A, $\mathcal{R}$, γ, and a prior distribution $P(\mathcal{P})$ over the dynamics of the MDP M, define the optimal return as*

$$J^*(q) = \sup_{\tilde{\pi} \in \tilde{\Pi}} J(q, \tilde{\pi}). \tag{2.12}$$

Martin (1967, Thm. 3.2.1) shows that there exists an EE policy $\tilde{\pi}^* \in \tilde{\Pi}$ that *achieves that expected return (i.e., $J(q, \tilde{\pi}^*) = J^*(q)$) for every q. Any such EE policy $\tilde{\pi}^*$ is called a* ***Bayes-optimal policy****.*[6]

While there sometimes does not exist a policy that attains the optimal return (when the $\sup$ in Equation 2.12 is not attained in some infinite state spaces), for any $\epsilon > 0$ there always exists at least one near-Bayes-optimal policy $\tilde{\pi}^*_\epsilon$ such that $J(q, \tilde{\pi}^*_\epsilon) > J^*(q) - \epsilon$ for every q (Bertsekas and Shreve, 1978). For clarity and since this does not have an impact on our work, we will abuse the nomenclature and refer to any such near-Bayes-optimal policy as Bayes-optimal for a suitably small ϵ (e.g., numerical error), and drop the ϵ subscript in $\tilde{\pi}^*_\epsilon$.

The choice of the discount factor can greatly affect the optimal adaptive behavior since it plays the crucial role of arbitrating the relative importance of future rewards. In general, a low γ does not warrant much exploration because future exploitation will be heavily downweighted. The opposite is true as $\gamma \to 1$.

Since we are maximizing the expected discounted return in the Bayesian setting, it is by definition an optimization problem. Unlike the PAC-MDP or regret framework, the objective function prescribes a natural recipe — albeit intractable — to compute the desired optimal adaptive behavior. Indeed, as Bellman originally realized, one can consider a completely-observed surrogate MDP, now referred to as the Bayes-Adaptive MDP, to reframe this optimization problem; we detail its construction and some of its properties in the following section.

— Bayesian Average Case

It is also possible to consider an average, rather than discounted, reward objective in a Bayesian setting. One naive objective is to look for EE policies that optimize their average reward rate according to the following objective:

$$J_a(q, \tilde{\pi}) := \sum_s q(s) \rho^{\tilde{\pi}}(s), \tag{2.13}$$

[6] It is worth mentioning that the concept of the Bayes-optimal policy in statistics is the *Bayes procedure*, the strategy that minimizes the *Bayes risk* (the negative expected return of the strategy) and chooses *Bayes actions*. Usually the setting is more constrained than can be expressed with MDPs — it focuses on stopping-like problems.

where $\rho^{\tilde{\pi}}$ is defined as in Equation 2.8 — the expectation now also integrates over possible models according to the prior. However, as pointed out by Kumar (1985), any policy $\tilde{\pi}$ that eventually self-optimizes (i.e., ultimately achieves the optimal average reward rate for the true MDP) would be optimal. These self-optimizing policies (any EE policy achieving sublinear regret would suffice) would be optimal in this Bayesian setting irrespective of the prior; therefore the objective J_a does not appear to be desirable. As for MDPs, we can instead further constrain the EE policy to be bias-optimal through the following objective:

$$J_b(q, \tilde{\pi}) := \mathbb{E}\left[\sum_{t=0}^{\infty}(r_t - \rho^*) \mid q, \tilde{\pi}\right], \tag{2.14}$$

under some technical conditions to ensure the existence of this objective. This is essentially a Bayesian view on minimizing regret, where the regret minimization is informed by a prior, which could also be addressed with an augmented MDP. However, to the best of our knowledge, this objective has not been explicitly considered before in the literature, which has largely focused on discounted (or finite-horizon) problems. In this thesis, we will also focus on the discounted case, but we will discuss the average case again in the last chapter.

2.1.3 Bayes-Adaptive Markov Decision Processes

Given that the dynamics $\mathcal{P} \in \mathbb{P}$, where $\mathbb{P}$ is the set of all possible models, are only incompletely known, a Bayesian agent treats them as a latent random variable which follows a prior distribution $P(\mathcal{P})$. Observations about the dynamics contained in the history h_t (at time t) of actions and states: $h_t = s_1 a_1 s_2 a_2 \dots a_{t-1} s_t$,[7] duly lead to a posterior distribution over $\mathcal{P}$ via a likelihood. After observing history h_t from the MDP, the posterior belief over $\mathcal{P}$ is updated using Bayes' rule:

$$P(\mathcal{P} \mid h_t) \propto P(h_t \mid \mathcal{P}) P(\mathcal{P}), \tag{2.15}$$

or in recursive form $P(\mathcal{P} \mid h_t) \propto \mathcal{P}(s_{t-1}, a_{t-1}, s_t) P(\mathcal{P} \mid h_{t-1})$.

[7]Here, the reward function is assumed known so we can leave rewards out from the history.

The return of an EE policy $\tilde{\pi}$, $J(q, \tilde{\pi})$, depends implicitly on the prior distribution $P(\mathcal{P})$, which can be thought of as the generative model for the dynamics. The objective J only captures the return from the start-state distribution when no data has been observed. It is also useful to consider the return of $\tilde{\pi}$ for other distributions corresponding to posterior distributions $P(\mathcal{P} \mid h)$ and for a start-state distribution centered at some particular state s; we denote this quantity as $V^{\tilde{\pi}}(s, h)$:

$$V^{\tilde{\pi}}(s,h) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, h_0 = h, \tilde{\pi}\right] \tag{2.16}$$

$$= \int_{\mathcal{P}} P(\mathcal{P} \mid h)\, \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, h_0 = h, \tilde{\pi}, M(\mathcal{P})\right] \mathrm{d}\mathcal{P}, \tag{2.17}$$

As a special case, the return of policy $\tilde{\pi}$ from a degenerate start-state distribution q_{s_0} is $J(q_{s_0}, \tilde{\pi}) = V^{\tilde{\pi}}(s_0, \emptyset)$, where $q_{s_0}(s_0) = 1$ and $q_{s_0}(s) = 0$ for $s \neq s_0$.

The inner expectation in Equation 2.17 corresponds to the return of applying the EE policy $\tilde{\pi}$ (which depends internally on an evolving history starting from h) in a particular MDP $M(\mathcal{P})$ from state s. Since it will come in handy in the next derivation, we introduce the shorthand notation for this term:

$$W(s, h, \tilde{\pi}, \mathcal{P}) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, h_0 = h, \tilde{\pi}, M(\mathcal{P})\right]. \tag{2.18}$$

Notice that the time-indices in the summation need not correspond to absolute time, since the result of applying a particular policy in an MDP only depends on the state from the Markov property. Another useful relation to derive the Bayes-Adaptive MDP follows from Bayes' rule:

$$P(\mathcal{P}|has') = \frac{P(s'|ha,\mathcal{P})P(\mathcal{P}|h)}{\int_{\mathcal{P}} P(\mathcal{P}|h)P(s'|ha,\mathcal{P})\mathrm{d}\mathcal{P}} \tag{2.19}$$

$$= \frac{P(s'|s,a,\mathcal{P})P(\mathcal{P}|h)}{\int_{\mathcal{P}} P(\mathcal{P}|h)P(s'|s,a,\mathcal{P})\mathrm{d}\mathcal{P}} \tag{2.20}$$

$$= \frac{\mathcal{P}(s,a,s')P(\mathcal{P}|h)}{\int_{\mathcal{P}} P(\mathcal{P}|h)\,\mathcal{P}(s,a,s')\mathrm{d}\mathcal{P}} \tag{2.21}$$

$$= \frac{\mathcal{P}(s,a,s')P(\mathcal{P}|h)}{\bar{\mathcal{P}}(s,a,s',h)}, \tag{2.22}$$

where $\bar{\mathcal{P}}(s,a,s',h) \equiv \int_{\mathcal{P}} P(\mathcal{P}|h)\,\mathcal{P}(s,a,s')\mathrm{d}\mathcal{P}$ denotes the normalization constant, expressing the marginal probability of transitioning from state s to s' after executing a under a distribution of dynamics $P(\mathcal{P}|h)$. Note also that $\mathcal{P}(s,a,s') = P(s'|s,a,\mathcal{P})$ by definition of $\mathcal{P}$. Hence, we straightforwardly obtain the relation:

$$\mathcal{P}(s,a,s')P(\mathcal{P}|h) = \bar{\mathcal{P}}(s,a,s',h)P(\mathcal{P}|has'). \tag{2.23}$$

We are now in a position to describe a dynamic programming solution to the issue of optimizing $\tilde{\pi}$. The key observation is that we can expand and reformulate the expression for the return V to obtain the following recursive relation:

$$\begin{aligned}
V^{\tilde{\pi}}(s,h) &= \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t r_t \,|\, s_0 = s, h_0 = h, \tilde{\pi}\right] \\
&= \int_{\mathcal{P}} P(\mathcal{P}\,|\,h)W(s,h,\tilde{\pi},\mathcal{P})\mathrm{d}\mathcal{P} \\
&= \int_{\mathcal{P}} P(\mathcal{P}\,|\,h)\sum_{a\in A}\tilde{\pi}(s,h,a)\left(\mathcal{R}(s,a)+\gamma\sum_{s'\in S}\mathcal{P}(s,a,s')W(s',has',\tilde{\pi},\mathcal{P})\right)\mathrm{d}\mathcal{P} \\
&= \sum_{a\in A}\tilde{\pi}(s,h,a)\left(\mathcal{R}(s,a)+\gamma\sum_{s'\in S}\int_{\mathcal{P}} P(\mathcal{P}\,|\,h)\,\mathcal{P}(s,a,s')W(s',has',\tilde{\pi},\mathcal{P})\mathrm{d}\mathcal{P}\right) \\
&= \sum_{a\in A}\tilde{\pi}(s,h,a)\left(\mathcal{R}(s,a)+\gamma\sum_{s'\in S}\bar{\mathcal{P}}(s,a,s',h)\int_{\mathcal{P}} P(\mathcal{P}\,|\,has')W(s',has',\tilde{\pi},\mathcal{P})\mathrm{d}\mathcal{P}\right) \\
&= \sum_{a\in A}\tilde{\pi}(s,h,a)\left(\mathcal{R}(s,a)+\gamma\sum_{s'\in S}\bar{\mathcal{P}}(s,a,s',h)V^{\tilde{\pi}}(s',has')\right),
\end{aligned} \tag{2.24}$$

where we have used definitions and Equation 2.23. This is essentially a Bellman equation in which we know all of the components.

Equation 2.24 makes clear that the uncertainty about the dynamics of the model can be transformed into certainty about the current state inside an augmented state space S^+. These augmented states are also called *hyperstates*. For this hyperstate to be Markovian, we need it to contain information to compute $\bar{\mathcal{P}}(s,a,s',h)$. Namely, it needs to contain information about the current state in addition to sufficient statistics for the current posterior distribution. In particular, the full history h_t is not needed if the posterior distribution can be described fully using lower-dimensional sufficient statistics θ_t that can be updated at each time step (i.e., $P(\mathcal{P} \mid h_t) = P(\mathcal{P} \mid \theta_t)$). This means different histories h and h' may correspond to the same hyperstate if $P(\mathcal{P} \mid h) = P(\mathcal{P} \mid h')$. In order to keep the notation general, we will write the augmented space as $S^+ = S \times \mathcal{H}$, where S is the state space in the original problem and $\mathcal{H}$ is the set of possible histories. Nevertheless, the results in the rest of this section apply for any augmented state space where the history is compressed to sufficient statistics of the belief.

The dynamics associated with this augmented state space are described by

$$\mathcal{P}^+(\langle s, h\rangle, a, \langle s', h'\rangle) = \mathbb{1}[h' = has'] \int_{\mathcal{P}} \mathcal{P}(s, a, s') P(\mathcal{P} \mid h) \, \mathrm{d}\mathcal{P} \,. \tag{2.25}$$

The reward function is simply the projected reward function in the original MDP:

$$\mathcal{R}^+(\langle s, h\rangle, a) = \mathcal{R}(s, a). \tag{2.26}$$

Together, the 5-tuple $M^+ = \langle S^+, A, \mathcal{P}^+, \mathcal{R}^+, \gamma\rangle$ forms the Bayes-Adaptive MDP (BAMDP) for the MDP problem M (Duff, 2002). Denote by Π^+ the set of policies in M^+, in other words $\pi^+ : S^+ \times A \to [0, 1]$ for $\pi^+ \in \Pi^+$. In general, since the augmented state space may be smaller than the set of histories, we have $\Pi^+ \subseteq \tilde{\Pi}$.

Since the dynamics of the BAMDP are *known*, it can, in principle, be solved to obtain the optimal value function associated with each action:

$$Q^*(\langle s_t, h_t\rangle, a) = \max_{\pi^+ \in \Pi^+} \mathbb{E}\left[\sum_{t'=t}^{\infty} \gamma^{t'-t} r_{t'} \mid a_t = a, s_t^+ = \langle s_t, h_t\rangle, \pi^+, M^+\right] \tag{2.27}$$

from which the optimal action for each state can be readily derived. Optimal actions in the BAMDP are executed greedily in the real MDP M and constitute the best course of action for a Bayesian agent with respect to its prior belief over $\mathcal{P}$. The following statements clarify the relation between general history policies and BAMDP policies.

Proposition 1 (Bertsekas and Shreve, 1978; Martin, 1967) *Let* $S, A, \gamma, \mathcal{R}$ *and*

$P(\mathcal{P})$ define a Bayesian adaptive control problem.

i) For any policy $\tilde{\pi} \in \tilde{\Pi}$, there exists $\pi' \in \Pi^+$ such that $V^{\tilde{\pi}}(s,h) = V^{\pi'}(s^+)$, where s^+ is the hyperstate when h has been observed in state s ($s^+ = \langle s,h\rangle$ in our notation). In particular, π' satisfies $J(q,\tilde{\pi}) = J(q,\pi')$ for any q.

ii) A deterministic optimal policy of the BAMDP performs as well as the Bayes-optimal policy, as defined in Definition 4 (When the Bayes-optimal policy does not exist, then, for any $\epsilon > 0$, there exists a deterministic policy $\pi' \in \Pi^+$ such that $J(q,\pi') \geq J^*(q) - \epsilon$ for all starting state distributions q).

[see Bertsekas and Shreve (1978, Theorem 2.1) for technical assumptions and statements.]

This shows that we are not losing anything by only considering and optimizing EE policies in the BAMDP, i.e. EE policies that are a function of the hyperstate. So we will call the optimal policy in the BAMDP the Bayes-optimal policy, and we will not insist on the distinction between Π^+ and $\tilde{\Pi}$ — only $\tilde{\Pi}$ will be used.

It is obvious that the expected performance of the Bayes-optimal policy in the MDP M is bounded above by that of the optimal policy obtained with a fully-observable model, with equality occurring, for example, in the degenerate case in which the prior only has support on the true model. We can bound the value of $\tilde{\pi}^*$ more generally as lying inbetween the average value of an overall best state-policy applied to all MDPs and the average value of the best state-policy for each MDP (Van Hee, 1978):

$$\max_{\pi\in\Pi} \int P(\mathcal{P} \mid h) V^{\pi}_{\mathcal{P}}(s) \mathrm{d}\mathcal{P} \leq V^*(\langle s,h\rangle) \leq \int P(\mathcal{P} \mid h) V^{\pi^*(\mathcal{P})}_{\mathcal{P}}(s) \mathrm{d}\mathcal{P}, \tag{2.28}$$

for any s and h, where $V^{\pi}_{\mathcal{P}}$ is the value function of the policy $\pi \in \Pi$ in the MDP with dynamics $\mathcal{P}$ and $V^{\pi^*(\mathcal{P})}_{\mathcal{P}}$ is the value function of the optimal policy for the MDP with dynamics $\mathcal{P}$.

The Bayes-optimal policy is stationary as a function of the augmented state, but evolves over time when observed in the context of the original MDP — as a function of the state in S only. Since the uncertainty about the dynamics is taken into account in the optimization of the return, the Bayes-optimal policy integrates exploration and exploitation optimally. It is worth noting that even though we refer to exploration and exploitation, actions are rarely actually labeled with one or the other in this Bayesian setting, it is only an interpretation for actions whose consequences are more uncertain (explore) or more certainly valuable (exploit).

2.1.3.1 PAC-BAMDP

An approximation to Bayes-optimality introduced by Kolter and Ng (2009), and later refined by Araya-López et al. (2012) leverages the PAC-MDP formulation to express a near-Bayesian property after a finite time. Instead of comparing the performance of the EE policy $\tilde{\pi}$ against the optimal MDP policy π^* in a given environment, as in the PAC-MDP framework, $\tilde{\pi}$ is compared against the Bayes-optimal policy $\tilde{\pi}^*$ under a Bayesian evaluation. The resulting property for an exploration policy is called the PAC-BAMDP property, it is formally stated below.

Definition 5 *(Adapted from (Araya-López et al., 2012)) An algorithm $\mathcal{A}$ (inducing an exploration policy $\tilde{\pi}$) is said to be **PAC-BAMDP** (Probably Approximately Correct in Bayes-Adaptive Markov Decision Processes) if, for any $\epsilon > 0$ and $0 < \delta < 1$, we have that $V^{\tilde{\pi}}(s_t, h_t) > V^*(s_t, h_t) - \epsilon$ for all but a polynomial number of steps in the quantities $\{|S|, |A|, 1/\epsilon, 1/\delta, 1/(1-\gamma)\}$ with probability at least $1 - \delta$.*

PAC-BAMDP algorithms need not be PAC-MDP since they are evaluated as a function of the belief (Kolter and Ng, 2009). According to Asmuth (2013), the motivation behind the PAC-BAMDP objective is to provide a mechanism to trade-off computational complexity against sample complexity, namely by allowing the planning algorithm to take suboptimal decisions (reducing the computational demand) for a polynomial number of steps (increasing the sample complexity). However, there is no guarantee that a PAC-BAMDP algorithm produces an EE policy $\tilde{\pi}$ that is near-optimal with respect to our Bayesian objective $J(q, \tilde{\pi})$, since the sub-optimal decisions may occur for all the steps that matter under the horizon defined by γ.

We now turn to a setting that generalizes Bayesian adaptive control in MDPs, namely Partially Observable Markov Decision Processes.

2.1.4 Partially-Observable Markov Decision Processes

When the state of the MDP is not fully observed, but the MDP is known including the dynamics $\mathcal{P}$, we obtain a Partially Observable MDP (POMDP) (Astrom, 1965). In addition to the components of an MDP tuple $\langle S, A, \mathcal{P}, \mathcal{R}, \gamma \rangle$, a POMDP also contain a set of observations O and a known observation function $Z(o \mid s)$ capturing the probability of observing an observation $o \in O$ while in state s. Like the BAMDP for Bayes-adaptive control, a *belief-MDP* exists for POMDPs where the augmented state is the belief over states and whose optimal policy is the best course of action in the POMDP, it balances information-gathering actions to reduce state uncertainty with more exploitative actions and maximizes the expected discounted return.[8]

In the POMDP setting, at time t, the agent receives an observation $o_t \in O$ from the current state s_t according to Z. The agent maintains a belief about the current state $b_t(s|\bar{h}_t) = P(s_t = s|\bar{h}_t)$, where $\bar{h}_t$ is the history of actions and observations until time t, $h_t = a_1, o_1, \ldots a_t, o_t$. The agent may start from a fully observed initial state, but the uncertainty over the states can grow over time if observations are insufficiently informative to disambiguate the state. The agent must combine two sources of information to update its belief, the observation likelihood and the known dynamic model for states:

$$b_{t+1}(s'|\bar{h}_{t+1}) = P(s_{t+1} = s' \mid h_t, a_t, o_{t+1}) \propto \sum_s b_t(s|\bar{h}_t)\, \mathcal{P}(s, a, s') Z(o_{t+1} \mid s'). \tag{2.29}$$

Just as for the BAMDP (Equation 2.24), the Bellman equation for the belief-MDP

[8]Confusingly, the semantics of the formalism and corresponding augmented MDP is different for POMDPs: the POMDP is the formal setting and the belief-MDP is the name of the augmented MDP. In Bayes-adaptive control, it is the augmented MDP that is called BAMDP.

is:

$$V^{\bar{\pi}}(\bar{h}) = \sum_{a \in A} \bar{\pi}(\bar{h}, a) \sum_{s \in S} b(s|\bar{h}) \left(R(s,a) + \gamma \sum_{o \in O} \sum_{s' \in S} \mathcal{P}(s,a,s') Z(o \mid s') V^*(\bar{h}ao) \right), \tag{2.30}$$

where $\bar{\pi}$ is a policy mapping state-beliefs (parametrized here by $\bar{h}$) to actions. Here, the augmented state consists of the belief over states, or any sufficient statistics thereof. As for the BAMDP, nothing is lost by considering policies based on these augmented states rather than general history policies. In just the same way that we obtained $\mathcal{P}^+$ from Equation 2.24, the augmented dynamics can be obtained from Equation 2.30: the probability of going from $\bar{h}$ to $\bar{h}ao$ by performing action a is $\sum_s b(s|\bar{h}) \sum_{s'} \mathcal{P}(s,a,s') Z(o \mid s')$.

The similarities between the BAMDP and the POMDP's belief-MDP are not coincidental, the BAMDP is in fact a particular form of a POMDP's belief-MDP. The underlying state space of this POMDP is $S \times \mathbb{P}$ (recall that $\mathbb{P}$ is the set of all possible models). The second component of the state space is static and hidden, and partially observed through experienced transitions. Planning can be conducted in the belief space, allowing decisions to be taken in the light of their likely outcomes in gathering exploitable information about the hidden state. In the case of BAMDP, such actions gather information about the hidden model $\mathcal{P}$, and uncertainty can only decrease since the hidden model is assumed static. However, the POMDP is not a discrete POMDP since its state space is continuous (with discrete observations) — because the state contains transition probabilities. Equation 2.30 makes clear that if S (and O) are finite (i.e., a discrete POMDP), then computing transitions in the POMDP's belief-MDP only requires a summation rather than an integral in the case of the BAMDP (Equation 2.25). Continuous POMDPs are much less studied (Porta et al., 2006), therefore, as pointed out by Duff (2002), many classical solutions to POMDPs cannot be directly applied to the BAMDP.

Lastly, it is possible to combine the ideas behind POMDPs and BAMDPs to consider a setting where both the state and the dynamics are partially observed, giving rise to a Bayes-Adaptive Partially-Observed MDP (BAPOMDP) (Ross et al., 2011), a complex and poorly-explored problem.

2.2 Solution Methods

We now turn to solution methods for the different decision problems we presented. First, we review how to address bandit problems in Section 2.2.1. We treat bandits separately since there exists specific bandit strategies that reflect their special structure. We then look at the solution methods for more general MDPs in Section 2.2.2. This leads us to consider solutions to POMDPs and BAMDPs in Sections 2.2.3-2.2.4 to treat the partial observability in a Bayesian way. We will see that solutions for BAMDPs rely heavily on solution methods for MDPs, and sometimes borrow elements from strategies invented for bandits and POMDPs.

2.2.1 Bandits

We first review regret-minimizing strategies, followed by the bandit strategies for the Bayesian setting. A large class of solutions, called index-based allocation policies, assign an *index* to each arm and select the arm with the largest index to play at each turn. Choosing the index assigned to each arm depends on the nature of the bandit problem and the EE objective, and may not be straightforward to compute.

2.2.1.1 Regret-based Strategies

When introducing the concept of regret, Lai and Robbins (1985) provided index-based allocation policies that achieve the asymptotic regret lower bound for bandits (Equation 2.2). These policies use a particular form of index that represents an *upper confidence bound* on the mean payoff of each arm in the following sense: the index for arm a can only be smaller than the mean payoff μ_a with a probability that scales as $\frac{1}{n}$, and the index cannot decrease when the arm is not played. Agrawal (1995) later called policies that satisfies these conditions Upper Confidence Bound (UCB) policies and refined the results of Lai and Robbins (1985); in particular, providing policies that are easier to compute. Independently, Kaelbling (1993) introduced the Interval Estimation algorithm that constructs confidence intervals based on large deviation bounds, also resulting in an index policy where indices represent upper confidence bounds. The regret of this algorithm was only analyzed for Bernoulli bandits and was not shown to reach Lai's lower bound, but it was the source of inspiration for work on exploration-exploitation in reinforcement learning (Meuleau and Bourgine, 1999; Strehl and Littman, 2005).

The basic mechanism at play with policies that rely on upper confidence bounds is the optimism in the face of uncertainty principle. If it is possible (according to some confidence level) that an uncertain arm may be better than a known arm, then the uncertain arm will be selected to avoid missing the opportunity for a better payoff. But it is less pressing to resolve uncertainty if the upper confidence bound of an arm falls below the estimated mean of others, thus avoiding the unnecessary accumulation of regret by focusing on more promising arms.

The policies introduced by Lai and Robbins (1985) and Agrawal (1995) only achieve asymptotically optimal regret behavior. Auer et al. (2002) introduced the UCB1 and UCB2 policies that achieve a logarithmic growth of the regret uniformly over time and are easy to compute — these policies are improved further in the work of Auer and Ortner (2010). The UCB1 computes the index of an arm as:

$$\hat{\mu}_a + \sqrt{\frac{2\ln n}{n_a}}, \tag{2.31}$$

where $\hat{\mu}_a$ is the mean payoff estimate of the a-th arm after n_a pull, with n the total number of pulls.

Thompson Sampling (Thompson, 1933), the oldest strategy for bandit problems, assumes a prior distribution over the parameters $\{\mu_a\}$. Thompson Sampling is not an index-based strategy: at each step, it samples the arm parameters $\{\mu_a\}$ from the current posterior and chooses the arm to pull greedily based on the sampled means. Optimism is generated because posterior samples are likely to yield optimistic values where posterior entropy about the mean payoff is large and that will force the agent to try these arms. Despite its Bayesian flavor, Thompson Sampling was recently analysed in a frequentist setting and proven to reach theoretical finite-time regret lower-bounds for multi-armed bandits (Agrawal and Goyal, 2012).[9]

2.2.1.2 Bayesian Strategies

To find the optimal exploration policy that maximizes the expected return, we could solve the BAMDP corresponding to the bandit problem as described in Section 2.1.3, as Bellman proposed when introducing adaptive control processes (Bellman and Kalaba, 1959).[10] However, when faced with a bandit problem, a more ingenious path to the solution can be obtained using allocation indices called Gittins indices (Gittins et al., 1989). The main insight is that an allocation index can be computed independently for each arm that indicates how valuable it is to pull at any point, based on an arm's history of payoffs. These indices describe an ordering of the arms for every step based on the history, with the highest scoring arm corresponding to the Bayes-optimal action. The fact that the arms can be considered in isolation to solve the global exploration problem is remarkable in itself, and considerably simplifies the solution. However, computing the index for an arm is not trivial.

There exist many theoretical constructions that lead to the Gittins indices, and most provide a different proof for their optimality and suggest a different algorithm to compute them (Duff, 2002; Weber, 1992). The indices can also be found in table form for standard payoff distributions, for example in the book by Gittins et al. (1989). We detail a particular construction to provide some intuitive meaning to the indices, but we leave out any proof of optimality.

[9]The bound is only optimal in terms of the dependence on the number of time steps, see (Agrawal and Goyal, 2012) for details.

[10]This is a case where we have a prior on $\mathcal{R}$, and not $\mathcal{P}$. As previously mentioned, it is straightforward to convert a bandit problem into an equivalent MDP problem with uncertain dynamics by adding states that correspond to the different payoffs.

Consider a single arm a and its associated history of payoffs h_t^a. The value of playing *only* this arm according to the current belief corresponds to the mean estimate (scaled by $\frac{1}{1-\gamma}$), but it can also be written in this recursive form:

$$V_a(h_t^a) = \mathbb{E}\left[\sum_{t'=t}^{\infty} \gamma^{t'-t} R_{a,t}\right] \quad \left(= \frac{1}{1-\gamma}\hat{\mu}_a(h_t^a)\right) \tag{2.32}$$

$$= \int \left(P(R_{a,t} = r \mid h_t^a) r + \gamma V_a(h_t^a r)\right) \, \mathrm{d}r, \tag{2.33}$$

where $\hat{\mu}_a(h_t^a)$ is the mean of the posterior conditioned on payoff history h_t^a.

For example, in the case that the payoffs are Bernoulli distributed, R can only take two values and prior uncertainty about the payoff distribution can be represented using a Beta distribution, which implies a posterior Beta distribution after some interactions by conjugacy. Assume that the posterior distribution over μ_a after observing h_t^a is the distribution Beta(α_t, β_t), in which case α_t and β_t are sufficient statistics that we can substitute for the history. Then, dropping the time-index for clarity, Equation 2.32 can be rewritten recursively as:

$$V_a(\alpha, \beta) = P(R_{a,t} = 0 \mid \alpha, \beta) \left(0 + \gamma V_a(\alpha, \beta + 1)\right) \tag{2.34}$$

$$+ P(R_{a,t} = 1 \mid \alpha, \beta) \left(1 + \gamma V_a(\alpha + 1, \beta)\right) \tag{2.35}$$

$$= \frac{\beta\gamma}{\alpha + \beta} V_a(\alpha, \beta + 1) + \frac{\alpha}{\alpha + \beta}(1 + \gamma V_a(\alpha + 1, \beta)). \tag{2.36}$$

Now, for the purpose of constructing the index, let us introduce a retirement option when playing arm a. At any step t, the agent can decide to opt-out and get a guaranteed reward M; a form of stopping problem. We can then compute the value of the optimal policy in the retirement scheme as a function of M:

$$V_a^*(\alpha, \beta; M) = \max\{M, \frac{\beta\gamma}{\alpha + \beta} V_a^*(\alpha, \beta + 1; M) + \frac{\alpha}{\alpha + \beta}(1 + \gamma V_a^*(\alpha + 1, \beta; M))\}. \tag{2.37}$$

The value function depends in a complex (but convex) way on the parameter M except when M is small enough that $V_a^*(h^a; M) = V_a(h^a) = \frac{1}{1-\gamma}\hat{\mu}_a(h_a)$ or when M is big enough so that $V_a^*(h^a; M) = M$. It can be proven that the Gittins index for arm a, g_a, is linked to that construction in the following way:

$$g_a(h^a) = (1-\gamma)\inf\{M \mid V_a^*(h^a; M) = M\}. \tag{2.38}$$

In other words, the Gittins index can be understood as an indifference threshold: the least M such that the value of playing the arm is simply the value of retiring right away. Nontrivially, this threshold value (i.e., the Gittins index up to constant rescaling) provides a metric to compare different arms, and playing the arm with the highest index is exactly equivalent to the optimal exploration policy's choice.[11] This construction, and its proof of correctness, is due to Whittle (1980); it is not meant as a computational tool to obtain the Gittins indices. Efficient algorithms to compute them can be found in the work of Varaiya et al. (1985), Niño-Mora (2007), or Sonin (2008).

It is known that, for any $\gamma < 1$, the Bayes-optimal policy will settle on an arm after an initial period of exploration and only play that arm thereafter. The probability that the selected arm a_∞ is the optimal one, $P(\operatorname{argmax}_a \mu_a = a_\infty \mid \gamma)$ is less than 1 for all $\gamma < 1$:

$$P(\operatorname*{argmax}_a \mu_a = a_\infty \mid \gamma) < 1 \quad \forall \gamma < 1, \tag{2.39}$$

but $P(\operatorname{argmax}_a \mu_a = a_\infty \mid \gamma) \to 1$ almost surely as $\gamma \to 1$, at least for Bernoulli bandits (Kelly et al., 1981). This result nicely illustrates the impact of γ on the exploration-exploitation trade-off: it can be optimal given the uncertainty and some effective horizon described by γ to accept the loss of not exploiting the best possible action — since the loss of exploring to find this action is judged even greater.

Another observation is that the Gittins indices are at least equal to the posterior mean estimate of the payoff for each arm. This implies that most EE strategies for both the non-Bayesian and the Bayesian setting can be seen as expressing optimism over the payoffs of the arms: the policy does not judge an arm by its estimated mean, but by its mean with an added bonus. The bonus encodes the

[11] In fact, the result behind the Gittins indices holds for a wide range of dynamics of the arm in the augmented space, they need not represent the evolution of a posterior distribution (or its hyperparameters in a conjugate setting), they could be some other known Markov process.

fact that the estimated mean may grow with more information (the optimism), but the exact value of that bonus depends heavily on the objective, the horizon, and past observed data.

2.2.2 MDPs

We first present classical solutions when the MDP is known and tractable. We then review relevant reinforcement learning and planning algorithms. Finally, we present solution methods that tackle the exploration-exploitation problem explicitly in MDPs.

2.2.2.1 Dynamic Programming

When all the components of the MDP tuple are known — including the model $\mathcal{P}$ — standard dynamic programming algorithms can be used to estimate the optimal policy off-line, such as Value Iteration (Bellman, 1954) or Policy Iteration (Howard, 1960).

The Value Iteration (VI) algorithm directly exploits the Bellman optimality equation (Equation 2.7) in an iterative scheme to find the optimal value function. It starts with an initial value function V_0 and successively applies *full Bellman backups* to the estimate until convergence as follows:

$$V_{i+1}(s) \leftarrow \max_{a \in A} \{R(s,a) + \gamma \sum_{s' \in S} \mathcal{P}(s,a,s') V_i(s')\} \quad \forall s \in S. \tag{2.40}$$

The Policy Iteration (PI) algorithm combines a *policy evaluation* step, where the value function V^π (or Q^π) for the current policy π is computed (for example based on the Bellman equation), and a *policy improvement* step where π is updated using the computed V^π. A greedy improvement step takes the form:

$$\pi_{i+1}(s) \leftarrow \operatorname*{argmax}_{a \in A} Q^{\pi_i}(s,a). \tag{2.41}$$

More details on these methods can be found in reference texts by Ross (1983) and Puterman (1994).

Even though VI or PI are not always directly applicable, they are the basis for many algorithms in reinforcement learning that sample or approximate these steps in various ways. Generalized Policy Iteration (GPI) is a class of methods that roughly follow the policy iteration idea, but may take an improvement step before the policy evaluation step is completed (also called *optimistic*, or *modified*, PI) or only take a soft improvement step (this is one facet of policy-gradient algorithms (Bartlett and Baxter, 2011; Williams, 1992)). Though some of these GPI algorithms have weaker, fragile, or non-existent theoretical guarantees, their applicability and empirical performance is typically taken as justifying their use.

2.2.2.2 Approximate Dynamic Programming

If the state space is very large, or continuous, then it is not feasible to represent the value function exactly and VI and PI cannot be straightforwardly applied. A common class of solutions approximate the value function using some parametric form, $V^\pi(s;\mathbf{w}) = f_\mathbf{w}(\phi(s))$, where most commonly the state s is represented using a feature vector $\phi(s)$ and $\mathbf{w}$ is a vector of tunable parameters for the function f. In a linear architecture, this corresponds to:

$$V^\pi(s;\mathbf{w}) = \phi(s)^T\mathbf{w}. \tag{2.42}$$

The problem of evaluating or optimizing policies in this context is referred to as *Approximate Dynamic Programming* (Bertsekas, 2011a), approximate versions of VI and PI involve fitting value functions rather than computing them exactly.

2.2.2.3 Learning with Simulation-based Methods

Dynamic programming techniques require the specification of an explicit transition model, something which may be difficult to obtain in general. Simulation-methods, or incremental methods, instead rely on traces of experiences with the environment, or sample paths, to learn the value of a specific policy, or to optimize policies. Since these methods do not construct an explicit model of the transition, thus they are referred to as *model-free* reinforcement learning, or simulation-based, methods. There is a rich literature on these methods; we only review the algorithms related to the work in this thesis; a survey of the field can be found in the books of Sutton and Barto (1998) and Bertsekas and Tsitsiklis (1996) (see also (Szepesvári, 2010) for a more recent treatment).

The traces of experience exploited by model-free methods may not always come from interactions with some real environment. Even though the learning mechanism itself is model-free, the agent may in fact possess a black-box model from which it can generate interaction traces for learning, for example in the case of planning. We will come back to this point when discussing planning in Section 2.2.2.4.

Monte-Carlo Evaluation

On-policy Monte-Carlo (MC) policy evaluation solves the prediction problem of estimating V^π for a given policy π by sampling trajectories acquired *on policy* and averaging the returns obtained from each state:

$$\hat{V}^\pi(s) = \frac{1}{N(s)} \sum_{n=1}^{N(s)} R_n(s), \tag{2.43}$$

where $R_n(s)$ is the (discounted) return obtained from state s after the n-th visit, with $N(s)$ visits in total. As $N(s) \to \infty$, then $\hat{V}^\pi(s) \to V^\pi(s)$.

This Monte-Carlo estimate can also straightforwardly be obtained in an incremental way using a *MC backup* which updates $\hat{V}(s)$ and $N(s)$ as:

$$N(s) \leftarrow N(s) + 1 \tag{2.44}$$

$$\hat{V}^\pi(s) \leftarrow \hat{V}^\pi(s) + \frac{(R_{N(s)}(s) - \hat{V}^\pi(s))}{N(s)}, \tag{2.45}$$

with N and $\hat{V}$ initialized to 0.

Similarly, the MC backup to estimate the action-value function Q^π is

$$N(s,a) \leftarrow N(s,a) + 1 \tag{2.46}$$

$$\hat{Q}^\pi(s,a) \leftarrow \hat{Q}^\pi(s,a) + \frac{(R_{N(s,a)}(s,a) - \hat{Q}^\pi(s,a))}{N(s,a)}. \tag{2.47}$$

This is known as the *every-visit* version of MC evaluation, it provides biased estimates (for finite sample size) but is more sample efficient than the first-visit version where each state is updated only with the trajectory corresponding to the first visit to that state (Singh and Sutton, 1996). First-visit MC is clearly unbiased because it averages independent unbiased estimates of the return. From a single trajectory, every-visit MC can generate multiple updates for a single state, these updates are thus not independent and this causes the bias.

In infinite-horizon problems, Monte-Carlo evaluation technically requires waiting for an infinite trajectory before updating any value. This issue is bypassed either by stopping a trajectory at each step with probability γ (and backing up the undiscounted return), or setting some numerical accuracy to define an effective horizon.

In the function approximation setting, Monte-Carlo evaluation aims to minimize the weighted squared loss between the true value function and the estimate $\hat{V}^\pi$:

$$\begin{aligned} E(\mathbf{w};\pi) &= \frac{1}{2}\|V^\pi(s) - \hat{V}^\pi(s;\mathbf{w})\|^2_{\mathcal{D}^\pi} && (2.48)\\ &= \frac{1}{2}\sum_{s\in S}\mathcal{D}^\pi(s)\left(V^\pi(s) - \hat{V}^\pi(s;\mathbf{w})\right)^2, && (2.49)\end{aligned}$$

where $\mathcal{D}^\pi$ is the stationary state distribution of the policy π. Then on-policy Monte-Carlo evaluation becomes a stochastic gradient descent algorithm that finds a local minimum to that loss:

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha_t\left(R(s) - \hat{V}^\pi(s;\mathbf{w})\right)\nabla_{\mathbf{w}}\hat{V}^\pi(s;\mathbf{w}), \tag{2.50}$$

since $R(s)$ is an unbiased estimate of the value $V^\pi(s)$ and the states are sampled according to $\mathcal{D}^\pi$. Here α_t is an appropriately decreasing learning rate. The exact version of Monte-Carlo evaluation (Equation 2.45) can be recovered if one considers a linear architecture with features $\phi(s)_i = 1$ if $s = i$ (and 0 otherwise) and a learning rate specific for each state $\alpha_{N(s)} = \frac{1}{N(s)}$.

Monte-Carlo Control

On-policy *Monte-Carlo control* optimizes the policy by combining Monte-Carlo policy evaluation with some form of policy improvement (Sutton and Barto, 1998), a form of GPI. One standard version is to run Monte-Carlo evaluation under some policy π_i that guarantees enough exploration, such as an ϵ-greedy policy; that is a stochastic policy that chooses an action according to:

$$\pi_i(s,a) = \begin{cases} 1-\epsilon+\frac{\epsilon}{|A|} & \text{if } a = \operatorname{argmax}_{a'} \hat{Q}^{\pi_{i-1}}(s,a'), \\ \frac{\epsilon}{|A|} & \text{otherwise.} \end{cases} \tag{2.51}$$

After a full evaluation ($\hat{Q}^{\pi_i} = Q^{\pi_i}$), we update π_i by choosing the ϵ-greedy policy with respect to Q^{π_i}. The policy improvement theorem guarantees that we will find the optimal policy in the class of $\epsilon-$greedy policy (Sutton and Barto, 1998).

A more practical implementation of that method relies on an optimistic policy iteration scheme, where the policy is updated after each update in each state — rather than waiting for the evaluation to complete. It is still an open problem whether this method converges to the optimal policy, even in the tabular case (Tsitsiklis, 2003).

Monte-Carlo control can be combined with the function approximation variant of Monte-Carlo evaluation to deal with imperfect representation (Sutton and Barto, 1998).

Q-learning

Q-learning (Watkins, 1989) is an off-policy control algorithm. It optimizes a policy, based on the Bellman optimality equation, while following another. This algorithm relies on bootstrapping to learn: it updates its value based on previous estimates of the Q value function. The Q-learning update is:

$$\hat{Q}(s_t, a_t) \leftarrow \hat{Q}(s_t, a_t) + \alpha(r_t + \gamma \max_b \hat{Q}(s_{t+1}, b) - \hat{Q}(s_t, a_t)). \tag{2.52}$$

If the behavior policy explores sufficiently (it visits every state infinitely often), and under appropriate learning rate schedules, then $\hat{Q} \to Q^*$ with probability 1.

It is common practice to combine Q-learning with function approximation, even though it is known to be divergent in some problems (Baird et al., 1995). Recent development on gradient temporal-difference methods have started to address this by proposing a convergent Q-learning variant, but only under quite restrictive conditions (Maei et al., 2010).

2.2.2.4 Planning: Online Search

Even when the transition model is known, it is not always practical to find the optimal policy for all states in large MDPs in one fell swoop. Instead, there are methods that concentrate on *searching* online for the best action at just the current state s_t. This is particularly common for Bayes-Adaptive planning algorithms. We therefore introduce relevant existing online search methods for MDPs that are used as building blocks for Bayesian RL algorithms.

Online search methods evaluate a tree of possible future sequences. The root of the tree is the current state and the tree is composed of state and action nodes. Each state node, including the root, has as its children all the actions that are legal from that state. In turn, each action node has as its children all the successor states resulting from that action. This is illustrated in Figure 2.1. The goal of the forward search algorithm is recursively to estimate the value of each state and action node in the tree. Ultimately, the value of each possible action from the root is used to select the next action in the real environment, and the process repeats using the new state at the root.

While each state could be in principle uniquely identified with a state node in a search graph (there will be cycles in general so it is no longer a tree), in practice multiple state nodes will often correspond to the same state if each state node is identified by its path from the root, or by its depth in the tree. When referring to the value function computed for a state node for a state s at depth d, we will write $V_d(s)$ in that context.

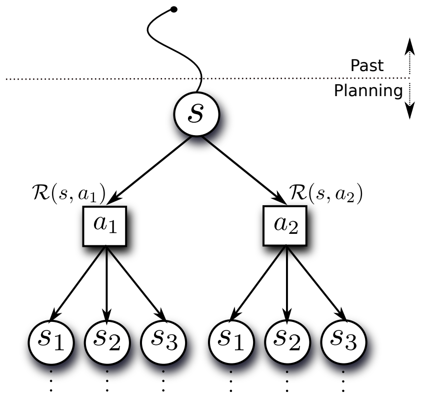


Figure 2.1: A part of a forward-search tree in an MDP with 3 states and 2 actions when the agent is in state s. State nodes are represented with squares, action nodes with circles. The top node is the root node, from where the agent plans to take the optimal action.

Online, tree-based, search methods may be categorised firstly by the *backup* method by which the value of each node is updated, and secondly by the order in which the nodes of the tree are traversed and backups are applied. Many planning methods derive from RL methods such as the ones in the previous sections but applied to the sub-MDP i) which has the current state as starting state and ii) which contains only the states reachable from the current state within the planning horizon.

Full-Width Search

Classical online search methods are based on *full-width* backups, which consider all legal actions and all possible successor states (or rather state nodes), for example using a Bellman backup,

$$V_d(s) \leftarrow \max_{a \in \mathcal{A}} \{\mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s,a,s') V_{d+1}(s')\}. \tag{2.53}$$

An illustration of this back-up is provided in Figure 2.2.

Search efficiency is then largely determined by the order in which nodes are traversed. One example is 'best-first', for which the current best is usually determined according to an optimistic criterion. This leads to an algorithm resembling A* (Hart et al., 1968), which applies in the deterministic case. The search tree may also be truncated, using knowledge of the most extreme reward and the discount factor to ensure that this is provably benign (Davies et al., 1998). If one is prepared to give up guarantees on optimality, an approximate value function (typically described in the online search literature as a heuristic function or evaluation function) can be applied at leaf nodes to substitute for the value of the truncated subtree.

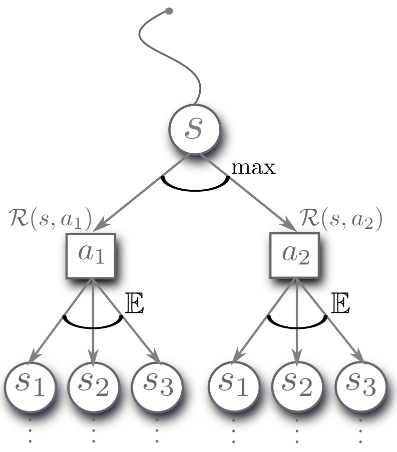


Figure 2.2: Applying full-width backups to the search tree of Figure 2.1. The value at a state node is obtained by applying a max operator on the value of its child action nodes. In turn, the value of an action node is determined by computing the expected value of its child state nodes, where the weight of each child is the probability of transition to that node.

Sample-Based Search

Rather than expanding every tree node completely, sample-based search methods overcome the curse of dimensionality by just sampling successor states from the transition distribution. These have the generic advantage over full-width search that they expend little effort on unlikely paths in the tree and their complexity is usually independent of the size of the state space.

More subtly, sample-based planning does not require an explicit transition model of the world. Rather, since trajectories are simulated internally through sampling, a *generative model* that provides the capacity to sample transitions is sufficient. Such generative models are often much easier to obtain than explicit transition models, for example for complex physical systems.

— Sparse Sampling

Sparse Sampling (Kearns et al., 1999) is a sample-based online search algorithm. The key idea is to sample C successor nodes from each action node, and apply a Bellman backup to these sampled transitions, so as to update the value of the parent state node from the values of the child nodes:

$$V_d(s) = \max_{a \in \mathcal{A}} \{\mathcal{R}(s,a) + \frac{\gamma}{C} \sum_{s' \in \text{Child}(s,a)} V_{d+1}(s') \text{Count}(s,a,s')\}, \tag{2.54}$$

where $\text{Child}(s,a)$ is the set of successor states sampled from C draws of $\mathcal{P}(s,a,\cdot)$, and $\text{Count}(s,a,s')$ is the number of times each set element was sampled; this is illustrated in Figure 2.3. The search tree is traversed in a depth-first manner, and an approximate value function is employed at truncated leaf nodes, after some pre-defined depth D. Sparse Sampling converges to a near-optimal policy given an appropriate choice of the parameters C and D.

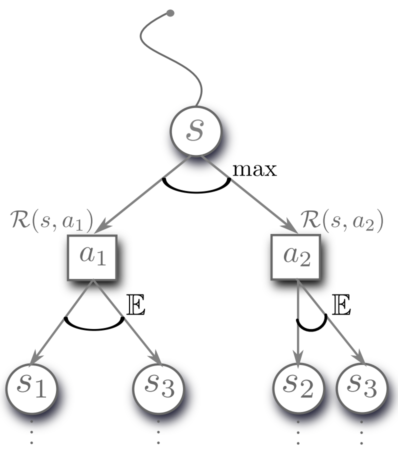


Figure 2.3: Applying sparse sampling in the scenario of Figure 2.1, with $C = 2$. The max operator is just like in Figure 2.2, the expectation operator weighs the value of child state nodes according to Count(s, a, s') (Equation 2.54).

FSSS

Although Sparse Sampling concentrates on likely transitions, it does not focus search on nodes that have relatively high values or returns. In the work of Walsh et al. (2010), Forward Search Sparse Sampling (FSSS) extends regular Sparse Sampling by maintaining both lower and upper bounds on the value of each node:

$$L_d(s,a) = \mathcal{R}(s,a) + \frac{\gamma}{C} \sum_{s' \in \text{Child}(s,a)} L_{d+1}(s') \text{Count}(s,a,s'), \tag{2.55}$$

$$U_d(s,a) = \mathcal{R}(s,a) + \frac{\gamma}{C} \sum_{s' \in \text{Child}(s,a)} U_{d+1}(s') \text{Count}(s,a,s'), \tag{2.56}$$

$$L_d(s) = \max_{a \in \mathcal{A}} L_d(s,a), \tag{2.57}$$

$$U_d(s) = \max_{a \in \mathcal{A}} U_d(s,a), \tag{2.58}$$

where Child(s,a) and Count(s,a,s') are defined as in the previous section. Whenever a node is created, the lower and upper bounds are initialized according to $L_d(s,a) = V_{\text{min}}$ and $U_d(s,a) = V_{\text{max}}$, i.e., the worst and best possible returns. The tree is traversed in a best-first manner according to these value bounds, starting from the root for each simulation through the tree. At each state node, a promising action is selected by maximising the upper bound on value. At each action (or chance) node, successor states are selected from a sampled set of C candidates by maximising the uncertainty (upper minus lower bound). This effectively prunes branches of the tree that have low upper bounds before they are exhaustively explored, while still maintaining the theoretical guarantees of Sparse Sampling.

— Monte-Carlo Tree Search

Despite their theoretical guarantees, in practice, sparse sampling and FSSS both suffer from the fact that they truncate the search tree at a particular depth, and so experience bias associated with the approximate value function they use at the leaves. Monte-Carlo Tree Search (MCTS) provides a way of reducing the bias by evaluating leaves exactly using the model, but employing a sub-optimal, rollout policy. More formally, in MCTS, states are evaluated by averaging over many simulations. Each simulation starts from the root and traverses the current tree until a leaf is reached, using a *tree policy* (e.g., greedy action selection) based on information that has so far been gathered about nodes in the tree. This results in a (locally) best-first tree traversal, where at each step the tree policy selects the best child (best according to some exploration criterion) given the current values in the tree. Rather than truncating the search and relying on a potentially biased value function at leaf nodes, a different policy, called a *rollout policy* (e.g., uniform random) is employed from the leaf node until termination or a search horizon. Each node traversed by the simulation is then updated by a Monte-Carlo backup, which simply evaluates that node by the mean outcome of all simulations that passed through that node. Specifically, the Monte-Carlo backups update the value of each action node as follows:

$$Q_d(s,a) \leftarrow Q_d(s,a) + {}^{(R - Q_d(s,a))}\!/_{N_d(s,a)}, \tag{2.59}$$

where R is the sampled discounted return obtained from the traversed action node s,a at depth d and $N_d(s,a)$ is the visitation count for the action node s,a

(i.e., the update computes the mean of the sampled returns obtained from that action node over the simulations).

A particular tree policy for MCTS that has received much attention, and indeed ultimately underlies our algorithm for the BAMDP in Chapter 3, is the UCT (Upper Confidence bounds applied to Trees) policy (Kocsis and Szepesvári, 2006). UCT employs the UCB1 (Upper Confidence Bounds) algorithm (Auer et al., 2002), designed for multi-armed bandit problems, to select adaptively between actions at every state node according to:

$$\operatorname*{argmax}_{a \in \mathcal{A}} \{Q_d(s,a) + c\sqrt{\log(N_d(s))/N_d(s,a)}\}, \tag{2.60}$$

where c is an exploration constant that needs to be set appropriately and $N_d(s)$ is the visitation count for the state node s. This tree policy treats the forward search as a meta-exploration problem, preferring to exploit regions of the tree that currently appear better than others, while continuing to explore unknown or less known parts of the tree. This leads to good empirical results even for small numbers of simulations, because effort is expended where search seems fruitful. Nevertheless all parts of the tree are eventually visited infinitely often, and therefore the algorithm can be shown to converge to the optimal policy in the very long run.

Despite some negative theoretical results showing that UCT can be slow to find optimal policies in carefully designed counterexample MDPs (Coquelin and Munos, 2007), UCT has been successful in many large MDP domains (Gelly et al., 2012).

Simulation-based Search

Instead of relying on a tree to perform search, Silver et al. (2012) consider more general simulation-based methods to search the optimal action from a given state. Simulation-based methods, such as MC-control or Q-learning, learn from traces of experiences generated from interaction. Most commonly, the interaction is with the real environment. Alternatively, the interaction can be simulated using a generative model to produce simulated experience. In simulation-based planning, the agent possesses such a generative model and can therefore simulate internally these traces of experience necessary for learning. Model-free RL algorithms can then be used to optimize (or evaluate) the policy given these interactions with the generative model, with the aim of finding the optimal action for the current state.

In the work of Silver et al. (2012), to be able to generalize the result of different simulated trajectories when planning, the value is represented using value function approximation rather than in a tree. Monte-Carlo Tree Search is then a special case for particular choice of state feature $\phi(s)$. This is equivalent to running an RL algorithm (in simulation) on the sub-MDP which has the current state as a starting state.

2.2.2.5 Exploration-Exploitation

So far, we have focused on ways to learn in MDPs without worrying too much about the data distribution: the distribution of states and actions encountered during learning. If that distribution does not cover some states which the optimal policy π^* visits, then the optimal policy cannot be learned. If we learn from interactions (either in the true MDP or in a simulated MDP when planning), then we have to explore in order to obtain the necessary interactions. Otherwise, acting greedily according to the current learned policy may confine the agent to a particular region of the state space, hindering learning in other regions. When discussing MC-control and Q-learning in Section 2.2.2.3, we mentioned exploration policies such as $\epsilon-$greedy policies that would visit every state infinitely often.[12] Similarly, the same exploration issue arises when searching inside a tree, we discussed the UCT policy in the context of the MCTS algorithm in Section 2.2.2.4 to address the exploration problem in that context. Using an $\epsilon-$greedy policy (or the UCT policy when searching) is a heuristic to ensure exploration of the state

[12] If the exploration policy is also greedy in the limit, it is sometimes called a Greedy in the Limit with Infinite Exploration (GLIE) policy (Singh et al., 2000).

space but there are more active ways of going about exploring the environment, in particular algorithms that address the exploration-exploitation objectives laid out in Section 2.1.2.1; we review these now.

PAC-MDP

Most PAC-MDP algorithms are *model based*, they construct an explicit model of the transitions to decide how to explore-exploit. Algorithms in this class include E^3 (Kearns and Singh, 2002), R-max (Brafman and Tennenholtz, 2003)[13], MBIE (Strehl and Littman, 2005), and MorMax (Szita and Szepesvári, 2010). For example, MBIE derives optimism for exploration by considering the transition model leading to the best value within some confidence interval around the mean. The learned model is usually a frequentist estimate for these algorithms, but the BOSS algorithm (Asmuth et al., 2009) maintains a Bayesian model of the transitions to induce optimism and still achieves the PAC-MDP property.

In addition, there is also at least one model-free PAC-MDP algorithm based on Q-learning, namely the Delayed Q-learning algorithm (Strehl et al., 2006), which works by following applying a form of Q-learning with greedy action selection and optimistic initialization for the values and a value bonus.[14]

Regret

The UCRL2 (Auer et al., 2009) is a similar construction to the MBIE algorithm (but with undiscounted reward and using different confidence intervals). It ensures that the expected total regret scales as $\tilde{O}(C|S|\sqrt{|A|T\log(T)})$ with high probability, where T is the horizon and C is the diameter of the MDP (the maximum average time it takes to go from any two states).

A more recent algorithm, PSRL, was proposed by Osband et al. (2013) for the

[13] The results for E^3 and R-max were stated a bit less generally in terms of mixing times and making ergodicity assumptions on the MDP. Kakade (2003) discusses the differences and links to the PAC-MDP framework.

[14] A chapter in the work of Li (2009) describes more model-free approaches that are PAC-MDP.

episodic setting. It is an extension of Thompson Sampling adapted for MDPs. One posterior MDP sample is solved at the start of each episode, and its optimal policy is applied greedily for a single episode. Since each possible MDP has a corresponding optimal policy, the algorithm is effectively applying policies according to the probability that they are optimal. This algorithm achieves an expected regret *under the prior distribution* of $O(\tau|S|\sqrt{|A|T\log(|SA|T)}$, with τ the length of the episode.

As in most other settings, the dependence of the regret bound on the size of the state space (and the hidden constants) means that a long exploration phase is necessary before the regret can be stabilized in practice; so far this has limited the applications of regret-based algorithms to small and finite MDPs.

Expected Return

This is the objective with which we are concerned in this thesis. As we described in Section 2.1.3, optimizing the expected return in a Bayesian framework is equivalent to solving a corresponding BAMDP. Since an BAMDP can be seen as the belief-MDP corresponding to a continuous POMDP, we first briefly review generic solution methods for POMDPs in the next section before presenting specific solution methods for BAMDPs.

2.2.3 POMDPs

Discrete POMDPs have the advantage that the value function of the belief is a piecewise-linear function, with finitely many hyperplanes for a finite horizon. This was recognized soon after POMDPs were introduced and led Cassandra et al. (1994); Monahan (1982); Smallwood and Sondik (1973) to propose solving methods based on computing the hyperplanes by propagating (and pruning) them using the Bellman equation. However, these methods suffered from an explosion in the number of hyperplanes for large state spaces or long horizons. Point-based algorithms were then proposed where the backups for hyperplanes are done at pre-selected belief-points only (Pineau et al., 2003; Spaan and Vlassis, 2005). One state-of-the-art POMDP algorithm, SARSOP (Kurniawati et al., 2008), combines this idea with online planning, it can therefore adapt its belief points based on the current situation.

While for finite-state POMDPs, the beliefs and hyperplanes can be described by finite vectors, and therefore the value can be computed using finite sums, this is no longer the case in continuous POMDPs (and therefore in BAMDPs), where the finite vectors representing hyperplanes (α-vectors) get replaced by functions of the state (α-functions) (Porta et al., 2006). A few researchers have considered tackling the computation of these α-functions by choosing appropriate representations (Duff, 2002; Porta et al., 2006; Poupart et al., 2006), but it still remains computationally challenging. One solution here is to use a particle-based representation of the belief to keep a finite-dimensional representation to be propagated (Thrun, 1999).

Other methods follow the path of forward-search sparse sampling to plan online (McAllester and Singh, 1999; Ross et al., 2008). The POMCP algorithm of Silver and Veness (2010) is one recent successful extension of that idea; it plans online using Monte-Carlo Tree Search in the belief space and avoids explicit belief-updates during search using a technique we refer to as *root sampling*. We tailor and extend this algorithm for BAMDPs in Chapter 3.

2.2.4 BAMDPs

From a practical perspective, solving the BAMDP exactly is computationally intractable, even for small state spaces. First, the augmented state space contains all possible beliefs and is therefore infinite. Second, the transitions of the BAMDP, described in Equation 2.25, require an integration of transition models over the posterior. Although this operation can be trivial for some simple probabilistic models (e.g., independent Dirichlet-Multinomial), it is intractable for most priors of interest. Calculating the posterior distribution itself presents computational problems; this was realized early on, so that researchers focused on conjugate priors to maintain a closed-form expression for the posterior in terms of the hyperparameters in a single family (Duff, 2002).[15]

However, certain special cases of the BAMDP are known to be somewhat more tractable. For example, the celebrated Gittins indices provide a shortcut solution for bandit problems (Gittins et al., 1989) which we discussed in Section 2.2.1, although calculating these indices remains a challenge in general. Further, the optimal solution to at least some finite-horizon linear-Gaussian control problems can be computed exactly (Tonk and Kappen, 2010). Nevertheless, it appears unlikely that there exists a tractable exact algorithm that can solve general BAMDPs, justifying a search for sound and efficient approximations.

Three coarse classes of approximation methods have been developed, which we now review. Note that all of them have analogues in solution methods for POMDPs.

First are offline methods that toil mightily to provide execution policies that can be used for any observed augmented state. Second and third are two sets of online methods that concentrate on just the current augmented state. One set of methods uses sparse sampling in the full tree of future states and actions associated with the BAMDP, starting from the current augmented state. The other samples and solves one or more MDPs from the current posterior over $\mathcal{P}$, possibly correcting for the bias towards exploitation to which this typically leads.

After describing these classes, we highlight what they currently lack, and so establish the basis for the central contribution in this thesis.

2.2.4.1 Offline Methods

One idea is to solve the entire BAMDP offline, for every state and belief (or history). This obviates the need for anything other than a simple value/policy lookup during execution. However, this avenue for approximation has not led to much practical success — presumably because of the difficulties associated with the

[15]In fact, the term conjugate prior was coined in this context of Bayesian decision theory (Raiffa and Schlaifer, 1961), to obtain an easy description of the posterior as it changes with new data.

size of the BAMDP, including the fact that gargantuan amounts of computation may be performed to find good policies in parts of the space of histories that are actually not sampled in practice.

Existing approaches in this class include an actor-critic algorithm (Duff, 2003), which does learning, and a point-based value iteration algorithm, called BEETLE (Bayesian Exploration Exploitation Tradeoff in LEarning) (Poupart et al., 2006). BEETLE builds an approximate policy off-line by exploiting facets of the structure of the value functions for BAMDPs, which they inherit from their broader, parent, class of POMDPs. More recently, Wang et al. (2012) propose to solve an offline POMDP in which they represent the latent dynamics as a discrete partially-observed state component, where the value of this state component corresponds to one of K possible models sampled from the prior. Their approach can fail if the true model is not well-represented in these K sampled models.

Offline methods are particularly poorly suited to problems with infinite state spaces.

2.2.4.2 Online Methods: Sparse Sampling

Online methods reduce the dependency on the size of the BAMDP by approximating the BAMDP solution around the current (augmented) state of the agent and running a planning algorithm at each step.

One idea is to perform forms of forward search from the current state. Although these methods concentrate on the current state, the search tree is still large and it can be expensive to evaluate a given path in the tree. In partial alleviation of this problem, most approaches rely on some form of sparse, non-uniform, tree exploration to minimize the search effort (but see also Fonteneau et al., 2013). While Section 2.2.2.4 described search algorithms for MDPs, here we present existing extensions to the BAMDP setting.

Wang et al. (2005) applied Sparse Sampling to search online in BAMDPs, expanding the tree non-uniformly according to sampled trajectories. At each state node, a promising action is selected via Thompson sampling (a model of the dynamics is drawn from the posterior distribution at the tree node, that sample is then solved to find the optimal action) to control the exploration of the tree. As in Sparse Sampling, this fails to exploit information about the real values of nodes in prioritizing the sampling process (since the action selection based on Thompson Sampling relies on a myopic value). At each chance (action) node, a successor belief-state is sampled from the transition dynamics of the BAMDP. Castro and Precup (2007) also applied Sparse Sampling to define a relevant region of the BAMDP for the current decision step. This leads to an optimization problem that is solved using Linear Programming. Ross and Pineau (2008) relied on a vanilla version of Sparse Sampling for Bayes-adaptive online planning.

Asmuth and Littman's BFS3 algorithm (Asmuth and Littman, 2011) adapts Forward Search Sparse Sampling (Walsh et al., 2010) to the BAMDP (treated as a particular MDP). Although BFS3 is described as Monte-Carlo tree search, it in fact uses a Bellman backup rather than Monte-Carlo evaluation. As in FSSS, each Bellman backup updates both lower and upper bounds on the value of each node.

Tree Exploration

As mentioned, some of these online methods do not expand the forward-search tree uniformly, they thus have to deal with a tree exploration problem to decide where to allocate search resources in order to optimize the tree policy. This is an internal meta-exploration problem which is treated differently from the main exploration problem the agent is facing againt the real environment. In particular, even though the outer EE problem is dealt with in a Bayesian way — which defines the planning problem to be solved, the meta-exploration problem for planning can be dealt with in frequentist terms if desired or with myopic strategies (Wang et al., 2005). We will come back to this meta-exploration problem in the final chapter.

2.2.4.3 Online Methods: Dual Optimism

Instead of applying sparse sampling methods in the tree of future states and actions, an alternative collection of methods derives one or more simpler MDPs from the posterior at a current augmented state, whose solution is often computationally straightforward. By itself, this leads to over-exploitation: corrections are thus necessary to generate sufficient exploration. Exploration can be seen as coming from optimism in the face of uncertainty – actions that have yet to be tried sufficiently must look more attractive than their current mean. Indeed, there are various heuristic forms of exploration bonus (Brafman and Tennenholtz, 2003; Dayan and Sejnowski, 1996; Kearns et al., 1999; Meuleau and Bourgine, 1999; Schmidhuber, 1991; Sutton, 1990) that generalize the optimism inherent in optimal solutions such as Gittins indices.

One such approximation was first derived in the work of Cozzolino et al. (1965), where the mean estimate of the transition probabilities (i.e., the mean of the posterior) was employed as a certainty equivalence approximation. Solving the corresponding mean MDP induces some form of optimism, but it is not always sufficient to drive exploration. This idea was revisited and linked to reinforcement learning formulations by Dayan and Sejnowski (1996).

Another way to induce optimism is to exploit the variance in the posterior when sampling MDPs at an augmented state. One of these approaches is the Bayesian DP algorithm (Strens, 2000). At each step (or after every couple of steps), a single model is sampled from the posterior distribution over transition models, and the action that is optimal in that model is executed. Although a popular approach in practice, no known theoretical guarantee relates it formally to the Bayes-optimal solution. In the Bandit case, this reduces to Thompson Sampling. Similar to Thompson Sampling in bandits, optimism is generated because solving posterior samples is likely to yield optimistic values in some unknown parts of the MDP (where posterior entropy is large) and that will force the agent to visit these regions. The PSRL algorithm (Osband et al., 2013) that addresses the regret objective in Section 2.2.2.5 is derived from the Bayesian DP algorithm, the difference is a formalized resampling criterion to obtain good regret guarantees. The Best Of Sampled Set (BOSS) algorithm generalizes this idea (Asmuth et al., 2009). BOSS samples a number of models from the posterior and combines them optimistically. This drives sufficient exploration to guarantee some finite-sample performance guarantees; however, again, these theoretical guarantees cannot be easily related to the Bayes-optimal solution. BOSS can be quite sensitive to its parameter that governs the sampling criterion, which can be difficult to select. Castro and Precup proposed a variant, referred to as SBOSS, which provides a more effective adaptive sampling criterion (Castro and Precup, 2010).

One can also see certain non-Bayesian methods in this light. For instance, Bayesian Exploration Bonus (BEB) solves the posterior mean MDP, but with an additional reward bonus that depends on visitation counts (Kolter and Ng, 2009). This bonus is tailored such that the method satisfies the PAC-BAMDP property presented in Section 2.1.3.1.[16] A more recent approach is the BOLT algorithm, which merges ideas from BEB and BOSS, enforces optimism in the transitions by (temporarily) adding fictitious evidence that currently poorly-known actions lead to currently poorly-known states (Araya-López et al., 2012). BOLT also has the PAC-BAMDP property.

2.2.4.4 Discussion of Existing Methods

Despite the recent progress in approximation algorithms, tackling large domains remains out of computational reach for existing Bayesian RL methods. This is especially true for domains where the posterior inference is not trivial, which is commonplace in large domains whose structure is only appropriately captured with rich priors. Unfortunately, it is exactly in these structured domains that Bayesian methods should shine, since they have the statistical capacity to take advantage of the priors — we will come back to this point in the section on Bayesian models (Section 2.3).

[16] A close variant of BEB is proven to be PAC-BAMDP in the work of Araya-López et al. (2012).

Indeed, methods that tackle the BAMDP directly such as forward-search methods can deal better with large state spaces but they suffer from the repeated computation of the BAMDP dynamics inside the search tree for most priors. As previously mentioned, to compute a single BAMDP transition in Equation 2.25, one needs to apply Bayes' rule and perform an integration over all possible models. This can be done cheaply for simple priors, but can be rather expensive for arbitrary priors.

On the other hand, the optimism-based methods of Section 2.2.4.3 are attractive because they appear more tractable — since they are dealing with smaller MDPs instead of tackling the BAMDP directly. However, it turns out to be hard to translate sophisticated prior knowledge into the form of a bonus — existing methods are only compatible with simple Dirichlet-Multinomial models. Moreover, the behavior in the early steps of exploration can be very sensitive to the precise parameter inducing the optimism.

2.3 Bayesian Models for MDPs

A key aspect of the exploration-exploitation problem is that agents can shape their uncertainty using prior knowledge about the class of environments they expect to encounter; in a Bayesian framework, this is naturally provided by a prior distribution over the transition model. Models embody inductive biases, allowing appropriately confident inferences to be drawn from limited observations. The structure present in the prior distribution is reflected in the optimal Bayesian learning strategy, it allows for, or justifies, complex, hyperopic exploration-exploitation strategies. Indeed, the confident inferences of a model in low-data regimes not only concern actually observed data, they also apply, counter-factually, to future data. With such models, a fully-Bayesian agent can be more selective about the data it wants to acquire. If it can foresee future trajectories that are likely to decrease the uncertainty about certain useful aspects of the dynamics quickly, then it can realize that these might have a dramatic advantage over others. Conversely, a fully-Bayesian agent only equipped with an uninformative prior might not find a justification to explore much, or only do so in some undirected way — since it is less clear to where any trajectory actually leads. Intuitively, we can expect an agent engaging in an environment that is likely under his prior to perform better than an uninformed agent and, conversely, a poor match between the prior and the environment is likely to result in low performance.[17]

From a purely Bayesian perspective, the prior distribution for an agent is selected since it exactly encodes a pre-existing subjective belief. However, when faced with the task of designing an agent for a class of environments, a trade-off arises between accuracy of modeling and computation. In practical terms, the designer's prior belief about the environment needs to be encoded in the agent in a way that correctly reflects the belief structure — requiring some form of compact description, and inference must be somewhat tractable in order to get a handle on the posterior distribution after seeing some data. Selecting more computationally convenient priors usually negatively affects the accuracy of the modeling, thus a pragmatic Bayesian balances the exactness and computational aspects of its prior distribution based on the agent's resource constraints and other practical matters.

Because of the immense impact that the prior distribution can have on the behavior in a Bayesian adaptive control scenario, finding informative and tractable prior distributions about a given class of environments to encode prior knowledge is a crucial part of a Bayes-adaptive solution to an exploration-exploitation problem. There is a huge range of possible models for MDPs. In fact, the entire Bayesian toolbox can be leveraged and specific choices will depend on the application domain. Thus, rather than trying to be comprehensive, this section presents some of the useful building blocks for designing MDP priors that we employ in this thesis. These can then be assembled in various ways to express different inductive biases.

[17]In fact, the consequences of maladaptive priors have been poorly investigated in the control setting, but they are at least explored in computational psychiatry (Huys and Dayan, 2009; Huys et al., 2014) as well as in the experimental results of this thesis.

2.3.1 Flat priors

Early work on Bayesian control of Markov chains focused on simple flat models, such as independent Dirichlet-Multinomial priors, where each state transition is modeled independently.

2.3.1.1 Dirichlet distribution

The Dirichlet distribution is a multi-dimensional generalization of the Beta distribution; it is the conjugate distribution of the multinomial distribution. As a distribution on the simplex, it is a natural choice to express beliefs over the probabilities of finitely many outcomes such as the transitions from one state to others for a given action.

The probability density function of the Dirichlet distribution (denoted Dir) with a N-dimensional parameter vector $\boldsymbol{\alpha}$, $\alpha_i > 0\ \forall i$, is:

$$P(\mathbf{p} \mid \boldsymbol{\alpha}) \propto \frac{\prod_{i=1}^{N} \Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^{N} \alpha_i)} \prod_{i=1}^{N} \mathbf{p}_i^{\alpha_i - 1}. \tag{2.61}$$

If $\boldsymbol{\alpha}$ is a vector composed of identical entries, then the distribution is symmetric and we can replace the parameter vector by a single scalar α; we will focus on the symmetric case from here on.

Given independent observations $(n_1, n_2, \ldots, n_N)$, as counts observed for each component, the posterior distribution on $\mathbf{p}$ is also a Dirichlet distribution:

$$\mathbf{p} \mid (n_1, n_2, \ldots, n_N), \alpha \sim \text{Dir}(\alpha + n_1, \alpha + n_2, \ldots, \alpha + n_N), \tag{2.62}$$

since

$$P(\mathbf{p} \mid (n_1, n_2, \ldots, n_N), \boldsymbol{\alpha}) \propto P((n_1, n_2, \ldots, n_N) \mid \mathbf{p}))P(\mathbf{p} \mid \alpha) \tag{2.63}$$

$$\propto \prod_{i=1}^{N} \mathbf{p}_i^{(\alpha + n_i) - 1} \tag{2.64}$$

Using the Dirichlet distribution, a simple prior for the MDP dynamics is the following generative model:

$$\mathcal{P}_s^a \sim \text{Dir}(\boldsymbol{\alpha}) \quad \forall s \in S, a \in A, \tag{2.65}$$

where $\boldsymbol{\alpha}$ is usually set to $\alpha_i = \frac{1}{|S|}$, and $N = |S|$. In this prior, transition probabilities are independent for each state and action. Samples from this prior distribution produce MDPs with little structure, except for what can be explicitly coded in the $\boldsymbol{\alpha}$ parameters. Nevertheless, this has been the staple prior in past work on Bayes-Adaptive planning, due to its simplicity and conjugacy to an obvious likelihood (Duff, 2002; Martin, 1967).

2.3.1.2 Sparse-Dirichlet distribution

In many MDPs, only a subset of states have a non-zero probability of being reached from a given state. A vanilla Dirichlet distribution does not encode that knowledge, so it can be an inappropriate prior when the number of states is large. The Sparse-Dirichlet distribution (Friedman and Singer, 1999) is a hierarchical prior that incorporates a sparseness assumption on the random probability vector.

In this construction the set V of non-zero components is chosen first. This is achieved by sampling a number of non-zero components k (e.g., from a geometric distribution) followed by choosing uniformly a set V of size k among the N indices. Then, a regular Dirichlet distribution restricted to set V is considered as the distribution over probability vectors — probabilities corresponding to elements not in V are set to zero. We refer to this construction as Sparse-Dir(α).

As for the Dirichlet distribution, we can straightforwardly define a prior over MDP transition probabilities by independently assigning a Sparse-Dirichlet prior to each state-action pair:

$$\mathcal{P}_s^a \sim \text{Sparse-Dir}(\alpha) \quad \forall s \in S, a \in A. \tag{2.66}$$

Posterior inference can be done in closed form (see (Friedman and Singer, 1999) for details), albeit at rather greater computational cost than for the vanilla Dirichlet distribution. This distribution was employed in the context of MDP exploration in the work of Strens (2000).

2.3.2 Structured Priors

Many researchers have considered powerful statistical models in the context of sequential decision making (Deisenroth and Rasmussen, 2011; Lazaric and Ghavamzadeh, 2010; Wingate et al., 2011), including in exploration-exploitation settings (Asmuth et al., 2009; Doshi-Velez et al., 2010; Huys and Dayan, 2009; Ross and Pineau, 2008; Tziortziotis et al., 2013), though rarely in combination with Bayes-Adaptive planning. Structured priors usually introduce latent variables that may represent some underlying relation between state variables and allow for generalization when learning the transition probabilities associated with different states and actions.

2.3.2.1 Finite mixtures

One powerful way of adding relations between states is to consider a finite mixture model. The generative model is simple: a random mixture proportion π decides to which component m a state s (or more generally a subset of states) belongs. Each component m is represented by a vector of parameters θ_m. The transition probabilities from any state s can then be obtained based on the value of θ_m, either deterministically or through another random process. This generative model can be written as:

$$\pi \sim \text{Dir}(\boldsymbol{\alpha}) \tag{2.67}$$

$$\boldsymbol{\theta}_m \sim H \qquad \forall m \in 1 \ldots M \tag{2.68}$$

$$z_{s,a} \sim \text{Categorical}(\pi) \qquad \forall s \in S, a \in A \tag{2.69}$$

$$\mathcal{P}_s^a \sim F(\boldsymbol{\theta}_{z_{s,a}}) \qquad \forall s \in S, a \in A, \tag{2.70}$$

where H is a base distribution on component parameters and F is a distribution which specifies how θ leads to low-level transition probabilities. This is illustrated in the plate diagram in Figure 2.4.

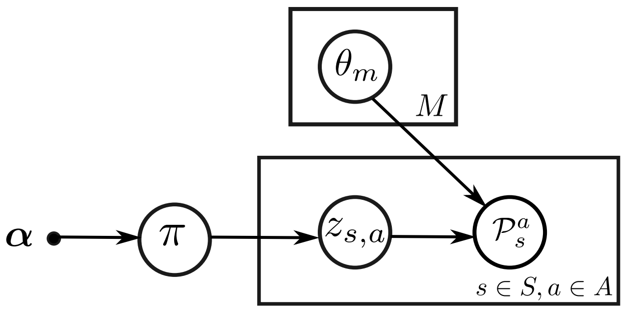


Figure 2.4: Plate diagram for a finite mixture of MDP dynamics. The transition probabilities for each state and action are obtained from a corresponding parameter vector $\boldsymbol{\theta}$. The mixture weights π that decide the assignment of state-action pairs to parameters is obtained from a Dirichlet distribution.

Here, posterior inference of $\mathcal{P}$ given the observed transition data is often intractable, and so is typically performed by a sample-based approximate Markov Chain Monte-Carlo (MCMC).

2.3.2.2 Bayesian Non-Parametric Models

Bayesian non-parametric models provide a more flexible form of prior knowledge (Orbanz and Teh, 2010). They carefully parameterize how structure is expected to repeat, and so allow the posterior to become more complex as evidence accumulates with extra observations. Non-parametric models have been considered in the context of control before (Asmuth et al., 2009; Asmuth, 2013; Deisenroth and Rasmussen, 2011; Doshi-Velez, 2009) but with an emphasis on modeling the data rather than planning. Here, we describe two such models that have been employed to define priors over MDPs. One process, the Dirichlet Process, provides a conjugate distribution over discrete distributions with an unbounded number of components. The other, the Gaussian Process, provides a distribution over real-valued functions. After presenting each process, we discuss how they can be employed for MDP modeling.

Dirichlet Process

The Dirichlet distribution can be extended to incorporate an unbounded number of components. The outcome is the Dirichlet process (DP), a distribution on random probability measures (Ferguson, 1973). A Dirichlet process is parametrized by a base measure H on some (measurable) space Θ, and a concentration parameter $\alpha > 0$, denoted DP(α, H). The concentration α encodes the relation between a measure sampled from DP(α, H) and the base measure H. While H can be an arbitrary continuous distribution, a sample $G \sim$ DP(α, H) is a discrete distribution — only countably many elements of Θ have positive support in G.

The Dirichlet process can be defined in relation to the Dirichlet distribution. In turn, this establishes the conjugacy of the Dirichlet process. If we observe a set $\{\theta_1, \ldots, \theta_N\}$ drawn independently from G, and $G \sim$ DP(α, H), then:

$$G \mid \{\theta_1, \ldots, \theta_N\} \sim \text{DP}\left(\alpha + N, \frac{1}{\alpha + N}\left(\alpha H + \sum_{n=1}^{N} \delta_{\theta_n}\right)\right), \tag{2.71}$$

where δ_x is a dirac delta function centered at x.

This above characterization of the DP is useful theoretically, however it is not constructive. In particular, it does not provide a sampling mechanism. The following Chinese Restaurant Process construction solves this issue through another more practical characterization of the DP.

— Chinese Restaurant Process

Taking the limiting case of the posterior DP as $N \to \infty$, it appears that all the probability mass gets gradually shifted to existing atoms (the observed θs), with the role of the base measure H vanishing. For this reason, it is not surprising that one can view a DP sample $G \sim DP(\alpha, H)$ as an infinite mixture of atoms sampled from the base measure:

$$G(\theta) = P(\theta \mid G) = \sum_{m=0}^{\infty} \pi_m \delta(\theta, \theta_m), \tag{2.72}$$

where $\{\pi_n\}$ are the mixture weights and $\{\theta_n\}$ are the corresponding atoms. These mixture weights can be sampled directly, through a *stick-breaking process*. Instead of sampling the mixture weights explicitly, we can directly obtain the observations arising from that infinite mixture through the Chinese Restaurant Process.

Formally, the Chinese Restaurant Process (CRP) is a distribution on partitions, it provides another way to sample observations from a DP (Aldous, 1985). If we have N observations $\theta_1, \dots, \theta_N$, composed of $M \leq N$ unique atoms $\{\theta_m\}$, then sampling a new observation amounts to picking the label z_{N+1} of the new observation between 1 and $M+1$, where the first M labels correspond to the observed unique atoms, and the last label correspond to an unseen atom.

In the CRP, this is viewed as customers arriving to sit in a restaurant. The first customer sits at the first table. The second customer has the choice between sitting at the first table ($z_2 = 1$), or sitting at a new table ($z_2 = 2$). In general, the N-th customer that comes in finds M occupied tables, and can sit on either one or choose to sit at a new empty table. The probability of the N-th customer choosing table m is:

$$P(z_N = m \mid z_1, \dots, z_{N-1}, \alpha) = \begin{cases} \frac{N_m}{\alpha+N} & m = 1, \dots, M \\ \frac{\alpha}{\alpha+N} & m = M+1 \end{cases} \tag{2.73}$$

where N_m is the number of customers already sitting at the m-th table and $\alpha > 0$ is the concentration parameter. We denote such process to obtain the z variables as CRP(α).

Each table in the CRP is like a cluster, or partition, each of which can be associated with an atom $\theta \sim H$ to obtain a DP. Indeed, if we sample the cluster assignment $\{z_n\}_{n=1}^{\infty} \sim$ CRP(α), and sample $\theta_m \sim H \; \forall m \in \mathbb{Z}^+$, then the random sequence $\theta_{z_1}, \theta_{z_2}, \dots$ can also be equivalently expressed as observations from $\theta \sim G$, with $G \sim$ DP(α, H).

CRP Mixture for MDP modeling

It is straightforward to define a mixture model from a DP:

$$G \sim \text{DP}(\alpha, H) \tag{2.74}$$

$$\theta_n \sim G \tag{2.75}$$

$$x_n \sim F(\theta_n), \tag{2.76}$$

where $F(\theta)$ is some distribution with parameter θ.

Using the CRP, the same mixture model can be equivalently expressed as:

$$\{z_n\}_{n=1}^{\infty} \sim \text{CRP}(\alpha) \tag{2.77}$$

$$\theta_m \sim H \ \forall m \in \mathbb{Z}^+ \tag{2.78}$$

$$x_n \sim F(\theta_{z_n}), \tag{2.79}$$

Here, one can map each x_n (or θ_n) to some state transition probabilities $\mathcal{P}_s^a$, or we may use some other deterministic map from the $\{x_n\}$ set to $\mathcal{P}$ in order to encode our prior knowledge about the MDP dynamics. This mixture model is illustrated in a plate diagram in Figure 2.5.

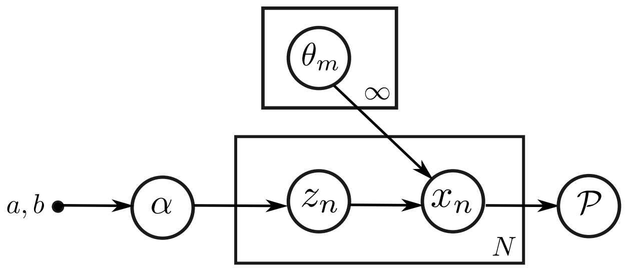


Figure 2.5: Plate diagram for a CRP mixture model for MDP dynamics, with a hyperprior on the concentration parameter, $\alpha \sim \text{Gamma}(a, b)$. In the pictured diagram, the set of x_n gives rise to the transition dynamics $\mathcal{P}$, the observations would then be obtained from $\mathcal{P}$; this is the setting in Chapter 6 for $N \to \infty$.

For inference, it is convenient to pick a distribution H that is conjugate to the likelihood distribution F. Gibbs sampling can then be employed to resample a cluster assignment based on others, and resample the cluster parameters based on observations and cluster assignments. More details can be found in the report by Griffiths and Ghahramani (2005). Since α may not be known, we can add an hyperprior on it, $\alpha \sim \text{Gamma}(a, b)$, and infer it from data.

In the work of Doshi-Velez (2009), a hierarchical Dirichlet Process is used to allow for an unbounded number of states in a POMDP and infer the size of the state space from data. This is referred as the iPOMDP model — this model is itself an extension of the infinite Hidden Markov Model (iHMM) model (Beal et al., 2001). The iPOMDP model is used in a online forward-search planning scheme, albeit of rather limited depth and tested on modestly-sized problems. In the work of Asmuth et al. (2009), a CRP mixture is employed to model state clustering in combination with the BOSS algorithm.

Gaussian Processes

Gaussian processes (GP) form a powerful family of priors on functions. As in the Dirichlet process, they are defined by considering finite aspects of what is, in this case, a distribution over uncountably many objects. A Gaussian process is a set of random variables, one for each element of the function domain X, for which any finite subset is distributed according to a multivariate normal distribution (Rasmussen, 2006).

A GP, denoted GP(m,$\mathcal{K}$), is specified by a mean function $m : X \to \mathbb{R}$ and a covariance function $\mathcal{K} : X \times X \to \mathbb{R}$. Let a function f be drawn from a GP, $f \sim \text{GP}(m, \mathcal{K})$. Then, given any vector $\mathbf{x}$ of points in X, the random function at these points $f(\mathbf{x})$ is distributed according to $f(\mathbf{x}) \sim \mathcal{N}(m(\mathbf{x}), \mathcal{K}(\mathbf{x}, \mathbf{x}))$; $\mathcal{K}(\mathbf{x}, \mathbf{x})$ is the *covariance kernel*: a matrix with entries corresponding to all the cross-terms $\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j)$.

A regression model with a GP prior can be seen as a Bayesian linear regression model, with a Gaussian prior on the weights, and (up to) an infinite number of features (from Mercer's Theorem, covariance kernels can be represented in terms of a dot product between feature vectors).

Posterior inference in GPs can be carried out in closed form (for a normally distributed noise model), though it requires a computationally expensive matrix inversion. Given a set of observations $f(\mathbf{x})$, we can obtain posterior predictions for $f(\mathbf{x}*)$ by first writing down the joint distribution:

$$\begin{bmatrix} f(\mathbf{x}) \\ f(\mathbf{x}*) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m(\mathbf{x}) \\ m(\mathbf{x}*) \end{bmatrix}, \begin{bmatrix} \mathcal{K}(\mathbf{x},\mathbf{x}) & \mathcal{K}(\mathbf{x},\mathbf{x}*) \\ \mathcal{K}(\mathbf{x}*,\mathbf{x}) & \mathcal{K}(\mathbf{x}*,\mathbf{x}*) \end{bmatrix} \right), \tag{2.80}$$

and then applying equations for conditioning a multivariate normal:

$$f(\mathbf{x}*) \mid f(\mathbf{x}) \sim \mathcal{N}\left(\mu^*, \Sigma^*\right), \tag{2.81}$$

where

$$\mu^* = m(\mathbf{x}*) + \mathcal{K}(\mathbf{x}*,\mathbf{x})\mathcal{K}(\mathbf{x},\mathbf{x})^{-1}(f(\mathbf{x}) - m(\mathbf{x})), \tag{2.82}$$

$$\Sigma^* = \mathcal{K}(\mathbf{x}^*,\mathbf{x}^*) - \mathcal{K}(x^*,x)\mathcal{K}(\mathbf{x},\mathbf{x})^{-1}\mathcal{K}(\mathbf{x},\mathbf{x}^*). \tag{2.83}$$

GPs for MDP modeling

One popular use of GPs in the context of MDP dynamics is to model continuous dynamical systems with independent GPs for each dimension and for each action (Deisenroth and Rasmussen, 2011; Deisenroth et al., 2009). The difference in each state dimension $s^i_{t+1} - s^i_t$ is modeled as GP(m^i_a,$\mathcal{K}^i_a$) for a given action a, where the domain of the random function is the state space S. Of course, GPs are not be restricted to fill this role in modeling MDP dynamics; they can also capture certain non-linear mappings between latent variables.

Deisenroth and Rasmussen (2011) employ Gaussian Processes to infer models of the dynamics from limited data, with excellent empirical performance. However, the uncertainty that the GP captures was not explicitly used for exploration-exploitation-sensitive planning. This is addressed by Jung and Stone (2010), but with heuristic planning based on uncertainty reduction.

2.3.2.3 MDP Models in this Thesis

In the experimental chapter for the BAMCP algorithm (Chapter 4), we use the Dirichlet and Sparse-Dirichlet priors for standard discrete domains as well as a custom prior for an infinite binary grid. In Chapter 5, we rely on GP priors to model uncertain continuous dynamics. Finally, in Chapter 6, we make use of a CRP mixture to model an infinite sequence of related contextual tasks.

2.4 Historical Notes on Bayesian Adaptive Control

The formulation of adaptive control has its roots in statistics in the context of experimental design. During the second world war, Wald and his colleagues developed the *sequential analysis* formalism for statistical testing in the context of industrial quality control (Wald, 1945). Rather than analysing a fixed-size sample of observations towards accepting or rejecting an hypothesis, Wald and his group recognized the advantage of deciding sequentially whether to acquire more observations, or stop and make as reliable a decision as possible based on the available data without the cost of additioning sampling.[18] They formalized a general procedure called the *Sequential Probability Ratio Test* (SPRT), which looks at the likelihood ratio between competing hypotheses and stops acquiring observations when this ratio crosses some boundary.[19] Arrow et al. (1949) focused on the Bayesian formulation of the sequential analysis problem initially posed by Wald. They relied on a backward induction argument to explicitly derive a procedure that depends on the posterior distribution to minimize the expected cost (called the average risk); and then observed that the resulting optimal sequential procedure (which Wald called Bayes solution) was in fact a SPRT for some carefully selected boundaries — they provide a computational procedure

[18]Fienberg (2006) reports that a navy officer made the observation that he could perceive the best of two firing procedures well before the end of a scheduled test, and asked Wald and his group whether this could be made into a formal procedure.

[19]In a frequentist setting, the boundaries can be chosen to correspond to the desired power for a hypothesis test, Wald and Wolfowitz (1948) proved that the SPRTs were the tests, among all tests, that required the fewest observations on average for the corresponding hypothesis testing task. Arrow et al. (1949) also proved the same result with a different approach.

to compute such boundaries in a simple hypothesis testing scenario involving two Bernoulli distributions.[20] At around the same time, Turing, with the assistance of Good, was using a similar sequential testing technique to help decipher enigma codes at Bletchey park (Good, 1979). Independently, Barnard also had developed a related sequential procedure for a different military task (Barnard, 1946). Although Bayesian statistics was still in its infancy at the time, their work had a Bayesian flavor, since prior information played a crucial role in the decisions (namely, a prior on the letter frequency in German for the case of Turing). Another early formalization of sequential Bayesian decision theory inspired by Wald's work is found in the work of Raiffa and Schlaifer (1961).

Following the formulation of *dynamic programming* in the fifties, Bellman considered the problem of optimal adaptive control in more general settings when the transition probabilities of a system are unknown, i.e., the *adaptive control process* (Bellman and Kalaba, 1959). He showed how it could be transformed into a dynamic programming problem with an augmented state, or *hyperstate*, that contains both the system's physical state and the current state of knowledge, just like the BAMDP of the previous section. This approach was formalized further and extended by Howard and his students in the years that followed (Cozzolino et al., 1965; Martin, 1967; Silver, 1963). In particular, the formalism of Markov chains was employed, so this field of study came to be known as *Bayesian control of Markov Chains*. The book by Martin (1967) summarizes these results and highlights the importance of conjugate distributions such as the Dirichlet distribution (then called "matrix beta"), since they give rise to closed-form belief updates with little computational overhead — a useful property when considering long sequences of belief updates.

At about the same time as Bellman, Fel'dbaum (1960) recognized the dual role

[20]It is interesting to note that Wald was careful to only present the use of prior distributions as a proof instrument for SPRTs ("We are aware of the fact that many statisticians believe that in most problems of practical importance either no a priori probability exists, or that even where it exists the statistical decision must be made in ignorance of it; in fact we share this view. Our introduction of the a priori probability distribution is a purely technical device for achieving the proof which has no bearing on statistical methodology [...]", excerpt from Wald and Wolfowitz (1948)) while Arrow et al. (1949) had a more Bayesian view of the sequential decision task in mind and said "It may be remarked that the problem of optimum sequential choice among several actions is closely allied to the economic problem of the rational behavior of an entrepreneur under conditions of uncertainty."

that control actions could have in a closed-loop adaptive setting, he highlighted the tension between "investigative" (probing) control and "directive" (cautious) control in what he called *dual control* systems, and also considered using dynamic programming to solve the problem. This started a body of work in the control literature to understand which systems have actions with this dual role and how to solve them, but also to try to categorize which systems are "neutral" (no active learning is needed, this is true for example of a known, partially observed, linear systems with quadratic costs, see for example the work of Bar-Shalom and Tse (1974)).

Kumar (1985) surveys the early work on stochastic adaptive control (pre-1985), separating Bayesian from non-Bayesian approaches to the problem. Despite the many developments in Bayesian adaptive control in these early years, researchers still could not solve problems with more than a few states, and hit a wall in terms of scalability. Meanwhile, efficient computational methods were being developed to solve Markov Decision Processes, in particular in the field of reinforcement learning. Duff saw an opportunity to leverage these methods for the Bayesian control of Markov chains. His thesis coined the term *Bayes-Adaptive Markov Decision Process* to refer to the extended dynamic programming problem in the language of MDPs and studied different reinforcement-learning algorithm applied to it (Duff, 2002).

III

Bayes-Adaptive Monte-Carlo Planning (BAMCP)

Outline

This chapter contains the description and analysis of the Bayes-Adaptive Monte-Carlo Planning (BAMCP) algorithm, an online sample-based algorithm for planning in BAMDPs with discrete MDP states. This includes a discussion of root sampling, and some simulations that illustrate BAMCP's convergence and its internal workings.

The goal of Bayes-adaptive planning method is to find, for each decision point $\langle s, h\rangle$ encountered, the action a that at least approximately maximizes the future expected return (i.e., find the Bayes-optimal EE policy $\tilde{\pi}^*(s, h)$). Our algorithm, Bayes-Adaptive Monte-Carlo Planning (**BAMCP**), does this online by performing a forward search in the space of possible future histories of the BAMDP using a tailored Monte-Carlo tree search.

We employ the UCT algorithm, as presented in Section 2.2.2.4, to allocate search effort to promising branches of the state-action tree, and use sample-based rollouts to provide value estimates at each node. For clarity, let us denote by Bayes-Adaptive UCT (BA-UCT) the algorithm that applies vanilla UCT to the BAMDP (i.e., the particular MDP with dynamics described in Equation 2.25). Sample-based search in the BAMDP using BA-UCT requires the generation of samples from $\mathcal{P}^+$ for every step of each simulation — an expensive procedure for all but the simplest generative models $P(\mathcal{P})$. We avoid this cost by only sampling a single transition model $\mathcal{P}^i$ from the posterior at the root of the search tree at the start of each simulation i, and using $\mathcal{P}^i$ to generate all the necessary samples during this simulation. Sample-based tree search then acts as a filter, ensuring that the correct distribution of state successors is obtained at each of the tree nodes, as if it was sampled from $\mathcal{P}^+$. This *root sampling* method was originally introduced in the POMCP algorithm (Silver and Veness, 2010), developed to solve discrete-state POMDPs.

Combining BA-UCT with a version of root sampling forms the basis of the proposed BAMCP algorithm; this is detailed in Section 3.1. In addition, BAMCP also takes advantage of lazy sampling to reduce sampling complexity at the root; this is detailed in Section 3.2. Finally, BAMCP integrates rollout learning to improve the rollouts online; this is detailed in Section 3.3. In Section 3.4, we show that BAMCP converges to the Bayes-optimal solution. In Section 3.5, we warn against using aspects of the sampling in the models to inform BAMCP's search, showing that planning building blocks cannot quite be mixed in arbitrary ways.

Following this chapter, we conduct an extensive comparative empirical analysis of BAMCP in Chapter 4.

3.1 BA-UCT with Root Sampling

The root node of the search tree at a decision point represents the current state of the BAMDP. The tree is composed of state nodes representing belief states $\langle s, h\rangle$ and action nodes representing the effect of particular actions from their parent state node. The visit counts: $N(\langle s, h\rangle)$ for state nodes, and $N(\langle s, h\rangle, a)$ for action nodes, are initialized to 0 and updated throughout search. A value, $Q(\langle s, h\rangle, a)$, which is initialized to 0, is also maintained for each action node.

Each simulation traverses the tree without backtracking by following the UCT policy at state nodes defined by $\text{argmax}_a\{Q(\langle s, h\rangle, a) + c\sqrt{\log(N(\langle s, h\rangle))/N(\langle s, h\rangle, a)}\}$, where c is an exploration constant that needs to be set appropriately. Given an action, the transition distribution $\mathcal{P}^i$ corresponding to the current simulation i is used to sample the next state. That is, at action node $(\langle s, h\rangle, a)$, s' is sampled from $\mathcal{P}^i(s, a, \cdot)$, and the new state node is set to $\langle s', has'\rangle$.

When a simulation reaches a leaf, the tree is expanded by attaching a new state node with its connected action nodes, and a rollout policy π_{ro} is used to control the MDP defined by the current $\mathcal{P}^i$. This policy is followed to some fixed total depth (determined using the discount factor). The rollout provides an estimate of the value $Q(\langle s, h\rangle, a)$ from the leaf action node. This estimate is then used to update the value of all action nodes traversed during the simulation: if R is the sampled discounted return obtained from a traversed action node $(\langle s, h\rangle, a)$ in a given simulation, then we update the value of each action node to $Q(\langle s, h\rangle, a) + (R - Q(\langle s, h\rangle, a))/N(\langle s, h\rangle, a)$ (i.e., the mean of the sampled returns obtained from that action node over the simulations).

A detailed description of the BAMCP algorithm is provided in Algorithm 1. A diagram example of BAMCP simulations is presented in Figure 3.1. In Section 3.4, we show BAMCP eventually converges to the Bayes-optimal policy.

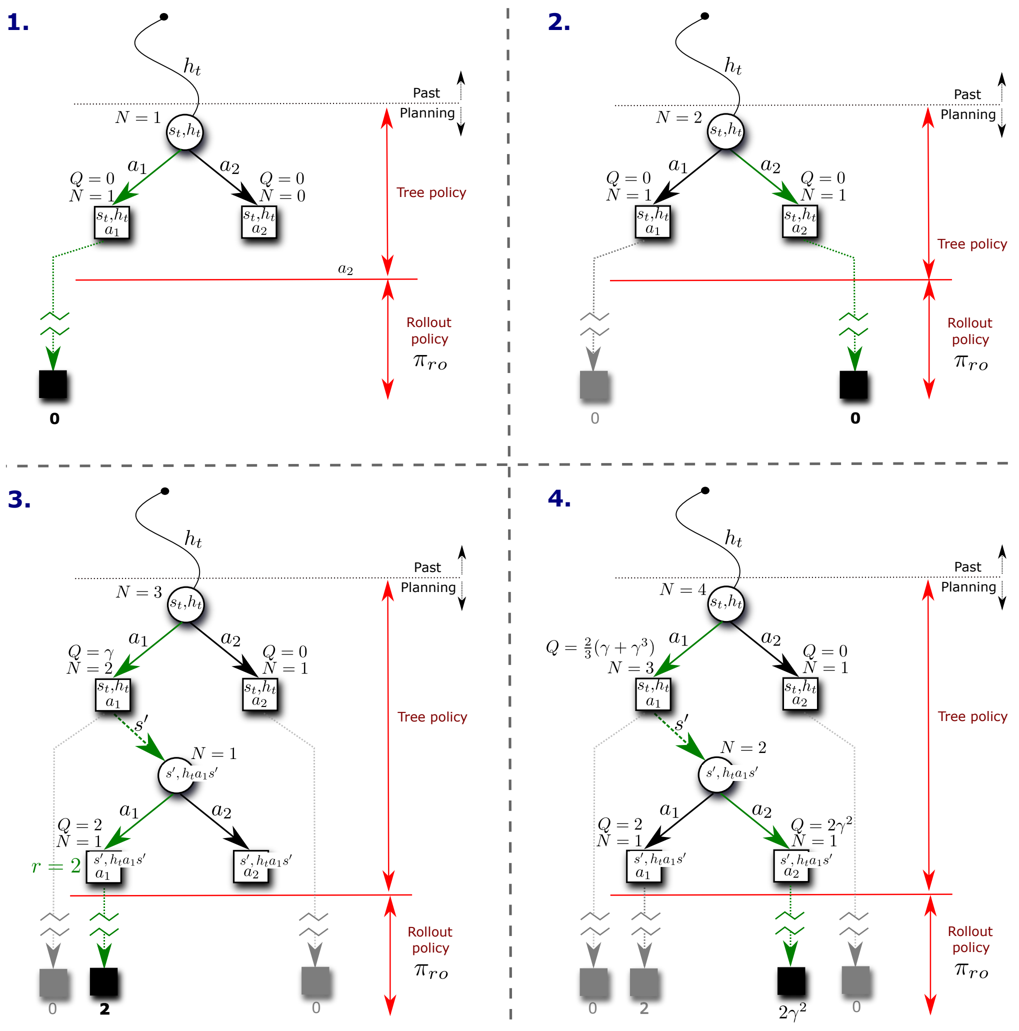


Figure 3.1: This diagram presents the first 4 simulations of BAMCP in an MDP with 2 actions from state $\langle s_t, h_t \rangle$. The rollout trajectories are represented with dotted lines (green for the current rollouts, and greyed out for past rollouts). **1.** The root node is expanded with two action nodes. Action a_1 is chosen at the root (random tie-breaking) and a rollout is executed in $\mathcal{P}^1$ with a resulting value estimate of 0. Counts $N(\langle s_t, h_t \rangle)$ and $N(\langle s_t, h_t \rangle, a_1)$, and value $Q(\langle s_t, h_t \rangle, a_1)$ get updated. **2.** Action a_2 is chosen at the root and a rollout is executed with value estimate 0. Counts and value get updated. **3.** Action a_1 is chosen (tie-breaking), then s' is sampled from $\mathcal{P}^3(s_t, a_1, \cdot)$. State node $\langle s', h_t a_1 s' \rangle$ gets expanded and action a_1 is selected, incurring a reward of 2, followed by a rollout. **4.** The UCB rule selects action a_1 at the top, the successor state s' is sampled from $\mathcal{P}^4(s_t, a_1, \cdot)$. Action a_2 is chosen from the internal node $\langle s', h_t a_1 s' \rangle$, followed by a rollout using $\mathcal{P}^4$ and π_{ro}. A reward of 2 is obtained after 2 steps from that tree node. Counts for the traversed nodes are updated and the MC backup updates $Q(\langle s', h_t a_1 s' \rangle, a_1)$ to $R = 0 + \gamma 0 + \gamma^2 2 + \gamma^3 0 + \cdots = 2\gamma^2$ and $Q(\langle s_t, h_t \rangle, a_1)$ to $\gamma + {}^{2\gamma^3 - \gamma}/_3 = \frac{2}{3}(\gamma + \gamma^3)$.

Algorithm 1: BAMCP

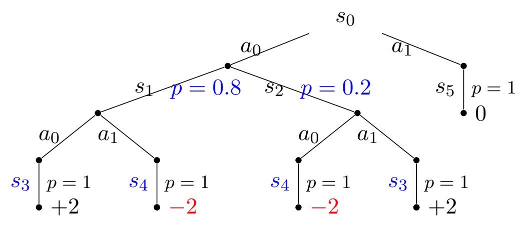
```
procedure Search( $\langle s, h\rangle$ )
  repeat
      $\mathcal{P} \sim P(\mathcal{P}|h)$
      Simulate($\langle s, h\rangle, \mathcal{P}, 0$)
  until Timeout()
  return $\operatorname{argmax}_a Q(\langle s, h\rangle, a)$
end procedure

procedure Rollout($\langle s, h\rangle, \mathcal{P}, d$ )
  if $\gamma^d R_{\max} < \epsilon$ then
      return $0$
  end
  $a \sim \pi_{ro}(\langle s, h\rangle, \cdot)$
  $s' \sim \mathcal{P}(s, a, \cdot)$
  $r \leftarrow \mathcal{R}(s, a)$
  return
  $r+\gamma$Rollout($\langle s', has'\rangle, \mathcal{P}, d+1$)
end procedure

procedure Simulate( $\langle s, h\rangle, \mathcal{P}, d$)
  if $\gamma^d R_{\max} < \epsilon$ then return $0$
  if $N(\langle s, h\rangle) = 0$ then
      for all $a \in A$ do
          $N(\langle s, h\rangle, a) \leftarrow 0,$
          $Q(\langle s, h\rangle, a)) \leftarrow 0$
      end
      $a \sim \pi_{ro}(\langle s, h\rangle, \cdot)$
      $s' \sim \mathcal{P}(s, a, \cdot)$
      $r \leftarrow \mathcal{R}(s, a)$
      $R \leftarrow r + \gamma$ Rollout($\langle s', has'\rangle, \mathcal{P}, d$)
      $N(\langle s, h\rangle) \leftarrow 1$, $N(\langle s, h\rangle, a) \leftarrow 1$
      $Q(\langle s, h\rangle, a) \leftarrow R$
      return $R$
  end
  $a \leftarrow \operatorname{argmax}_b Q(\langle s, h\rangle, b) + c\sqrt{\frac{\log(N(\langle s,h\rangle))}{N(\langle s,h\rangle,b)}}$
  $s' \sim \mathcal{P}(s, a, \cdot)$
  $r \leftarrow \mathcal{R}(s, a)$
  $R \leftarrow r + \gamma$
  Simulate($\langle s', has'\rangle, \mathcal{P}, d+1$)
  $N(\langle s, h\rangle) \leftarrow N(\langle s, h\rangle) + 1$
  $N(\langle s, h\rangle, a) \leftarrow N(\langle s, h\rangle, a) + 1$
  $Q(\langle s, h\rangle, a) \leftarrow Q(\langle s, h\rangle, a) + \frac{R-Q(\langle s,h\rangle,a)}{N(\langle s,h\rangle,a)}$
  return $R$
end procedure
```

Finally, note that the history of transitions h is generally not the most compact sufficient statistic of the belief in fully observable MDPs. It can, for instance, be replaced with unordered transition counts ψ, considerably reducing the number of states of the BAMDP and, potentially the complexity of planning. BAMCP can search in this reduced search space, which takes the form of an expanding lattice rather than a tree. We found this version of BAMCP to offer only a marginal improvement — though some domains may benefit from it more than others. This is a common finding for MCTS, stemming from its tendency to concentrate search effort on one of several equivalent paths (up to transposition), implying a limited effect on performance of reducing the number of those paths.

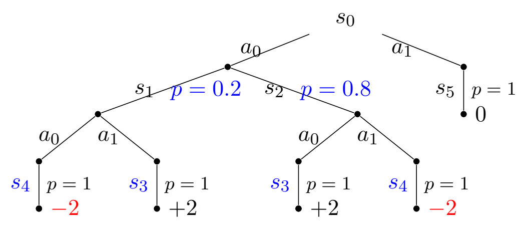
3.1.1 Root Sampling at Work in a Simple Example

We illustrate the workings of BAMCP, in particular root sampling, in a simulated example that showcases a crucial component of Bayes-adaptivity.

Consider a simple prior distribution on two MDPs ($\mathcal{P}_0$ and $\mathcal{P}_1$), illustrated in Figure 3.2, where $P(\mathcal{P} = \mathcal{P}_0) = P(\mathcal{P} = \mathcal{P}_1) = \frac{1}{2}$. The MDPs are episodic and stop at the leaves, and an episode starts in s_0. From state s_1 or s_2, any action has an expected reward of 0 *under the prior distribution over MDPs*. Nevertheless, the outcome of a transition from action a_0 in state s_0 carries information about the identity of the MDP, and allows a Bayes-adaptive agent to take an informed decision in state s_1 or s_2. Using Bayes-rule, we have that $P(\mathcal{P} = \mathcal{P}_0 | s_0 a_0 s_1) \propto P(s_1 | \mathcal{P} = \mathcal{P}_0, s_0 a_0) P(\mathcal{P} = \mathcal{P}_0) = 0.8$.



(a) $\mathcal{P} = \mathcal{P}_0$



(b) $\mathcal{P} = \mathcal{P}_1$

Figure 3.2: The two MDPs of Section 3.1.1, with prior probability $P(\mathcal{P} = \mathcal{P}_0) = P(\mathcal{P} = \mathcal{P}_1) = \frac{1}{2}$. Differences between the two MDPs are highlighted in blue.

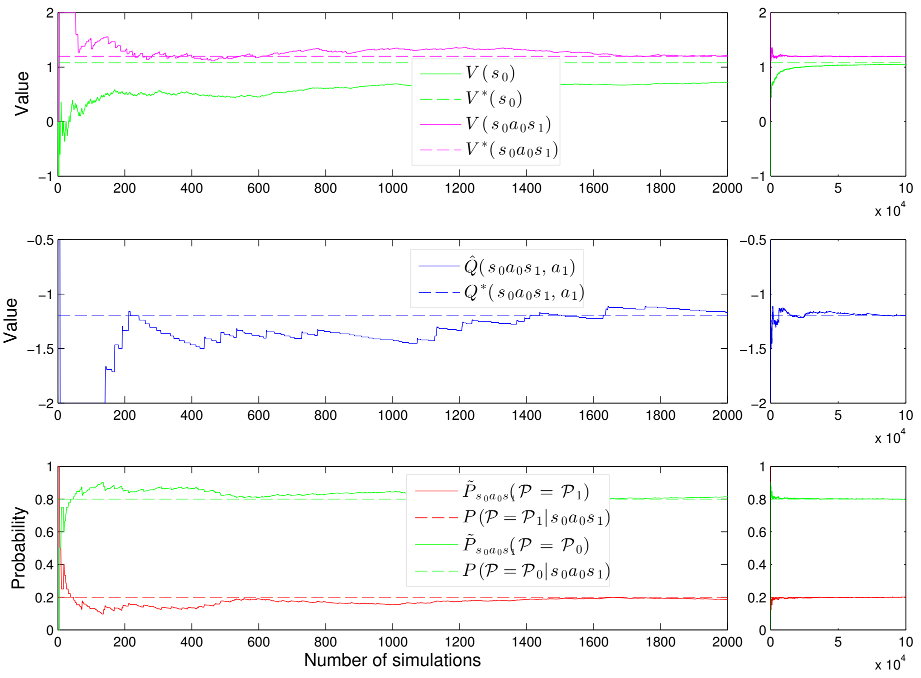


Figure 3.3: Tracking of different internal variables of BAMCP for the example of Section 3.1.1 with $\gamma = 0.9$. BAMCP is run at the starting state for a number of simulations (x-axis) and with $c = 20$. The first two rows show the evolution of values at tree nodes corresponding to different histories, along with target values as computed in Equation 3.2. The bottom row shows the evolution of $\tilde{P}_{s_0a_1s_1}(\mathcal{P} = \mathcal{P}_0) = 1 - \tilde{P}_{s_0a_1s_1}(\mathcal{P} = \mathcal{P}_1)$, the empirical distribution of MDPs seen going through tree node $\langle s_0a_1s_1\rangle$ (i.e., $\frac{1}{N(\langle s_0a_1s_1\rangle)}\sum_{i=0}^{N(\langle s_0a_1s_1\rangle)} \mathbf{1}[\mathcal{P}^i = \mathcal{P}_0]$). (Left) The first 2000 simulations (Right) Zoomed out view of 100,000 simulations, displaying empirical convergence to target values.

We can therefore compute the optimal values:[1]

$$V^*(h = s_0a_0s_1) = \max \begin{cases} 2P(\mathcal{P} = \mathcal{P}_0|h) - 2P(\mathcal{P} = \mathcal{P}_1|h) \\ 2P(\mathcal{P} = \mathcal{P}_1|h) - 2P(\mathcal{P} = \mathcal{P}_0|h) \end{cases} \tag{3.1}$$

$$= 2 \cdot 0.8 - 2 \cdot 0.2 = 1.2 \;\; (= V^*(h = s_0a_0s_2))$$

$$V^*(h = s_0) = \max\{0, 1.2\gamma\} = 1.2\gamma. \tag{3.2}$$

We now simulate BAMCP on this simple example for the first decision in state s_0. With root sampling, BAMCP only samples either $\mathcal{P}_0$ or $\mathcal{P}_1$ with equal probability

[1]For ease of notation, we sometimes refer to a node with its history only, as opposed to its state and history as in the rest of the thesis.

at the root of the tree, and does not perform any explicit posterior update inside the tree. Yet, as suggested by Lemma 1, we expect to find the correct distribution $P(\mathcal{P} = \mathcal{P}_0|s_0a_0s_1)$ of samples of $\mathcal{P}$ at the tree node $\langle s_0a_0s_1\rangle$. Moreover, BAMCP should converge to the optimal values V^* according to Theorem 1. This is what is observed empirically in Figure 3.3.

In the second row of Figure 3.3, we observe that $\hat{Q}(s_0a_0s_1, a_1)$ is slower to converge compared to other values. This is because time is ticking more slowly for this non-optimal node (i.e., a small fraction of simulations reach this node) so the value stays put for many simulations.

3.2 Lazy Sampling

In previous work on sample-based tree search, indeed including POMCP (Silver and Veness, 2010), a complete sample state is drawn from the posterior at the root of the search tree. However, this can be computationally very costly. Instead, we sample $\mathcal{P}$ lazily, generating only the particular transition probabilities that are required as the simulation traverses the tree, and also during the rollout.

Consider $\mathcal{P}(s, a, \cdot)$ to be parametrized by a latent variable $\theta_{s,a}$ for each state and action pair. These may depend on each other, as well as on an additional set of latent variables ϕ. The posterior over $\mathcal{P}$ can be written as $P(\Theta|h) = \int_\phi P(\Theta|\phi, h)P(\phi|h)$, where $\Theta = \{\theta_{s,a}|s \in S, a \in A\}$. Define $\Theta_t = \{\theta_{s_1,a_1}, \cdots, \theta_{s_t,a_t}\}$ as the (random) set of θ parameters required during the course of a BAMCP simulation that starts at time 1 and ends at time t. Using the chain rule, we can rewrite

$$P(\Theta|\phi, h) = P(\theta_{s_1,a_1}|\phi, h) \tag{3.3}$$

$$P(\theta_{s_2,a_2}|\Theta_1, \phi, h) \tag{3.4}$$

$$\vdots \tag{3.5}$$

$$P(\theta_{s_T,a_T}|\Theta_{T-1}, \phi, h) \tag{3.6}$$

$$P(\Theta \setminus \Theta_T|\Theta_T, \phi, h) \tag{3.7}$$

where T is the length of the simulation and $\Theta \setminus \Theta_T$ denotes the (random) set of parameters that are not required for a simulation. For each simulation i, we sample $P(\phi|h_t)$ at the root and then lazily sample the θ_{s_t,a_t} parameters as required, conditioned on ϕ and all Θ_{t-1} parameters sampled for the current simulation. This process is stopped at the end of the simulation, typically long before all θ parameters have been sampled. For example, if the transition parameters for different states and actions are independent, we can simply draw any necessary parameters individually for each state-action pair encountered during a simulation. In general, transition parameters are not independent for different states, but dependencies are likely to be structured. For example, the MDP dynamics could arise from a mixture model where ϕ denotes the mixture component and $P(\phi|h)$ specifies the posterior mixture proportion. Then, if the transition parameters θ are conditionally independent given the mixture component, sampling ϕ^i at the root for simulation i allows us to sample the required parameters $\theta_{s,a}$ independently from $P(\theta_{s,a}|\phi^i, h)$ just when they are required during the i-th simulation. This leads to substantial performance improvement, especially in large MDPs where a single simulation only requires a small subset of parameters (see for example the domain in Section 4.2 for a concrete illustration). This lazy sampling scheme is not limited to shallow latent variable models; in deeper models, we can also benefit from conditional independencies to save on sampling operations for each simulation by sampling only the necessary latent variables — as opposed to sampling all of ϕ.

3.3 Rollout Policy Learning

The choice of rollout policy π_{ro} is important if simulations are few, especially if the domain does not display substantial locality or if rewards require a carefully selected sequence of actions to be obtained. Otherwise, a simple uniform random policy can be chosen to provide noisy estimates. In this work, we learn Q_{ro}, the optimal Q-value in the real MDP, in a model-free manner, using Q-learning, from samples (s_t, a_t, r_t, s_{t+1}) obtained off-policy as a result of the interaction of the BAMCP agent with the MDP at time t. For each real transition (s_t, a_t, r_t, s_{t+1}) observed, we update

$$Q_{ro}(s_t, a_t) \leftarrow Q_{ro}(s_t, a_t) + \alpha(r_t + \gamma \max_a Q_{ro}(s_{t+1}, a) - Q_{ro}(s_t, a_t)), \tag{3.8}$$

where α is some learning rate parameter; this is the standard Q-learning rule (Watkins, 1989). Acting greedily according to Q_{ro} translates to pure exploitation of gathered knowledge. A rollout policy in BAMCP following Q_{ro} could therefore over-exploit. Instead, similar to the work of Gelly and Silver (2007), we select an ϵ-greedy policy with respect to Q_{ro} as our rollout policy π_{ro}. In other words, after t steps in the MDP, we have updated Q_{ro} t times and we use the following stochastic rollout policy for all MCTS simulations at the $t+1$ decision step:

$$\pi_{ro}(s, a) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|A|} & \text{if } a = \operatorname{argmax}_{a'} Q_{ro}(s, a') \\ \frac{\epsilon}{|A|} & \text{otherwise,} \end{cases} \tag{3.9}$$

where $\pi_{ro}(s, a)$ is the probability of selecting action a when in the MDP state s (i.e., history is ignored) during a rollout. This biases rollouts towards observed regions of high rewards. This method provides valuable direction for the rollout policy at negligible computational cost. More complex rollout policies can be considered, for example rollout policies that depend on the sampled model $\mathcal{P}^i$ or on the history h_t. However, these usually incur computational overhead, which may be less desired than running more simulations with worse estimates.

3.4 Theoretical Properties

In this section, we show that BAMCP converges to the Bayes-optimal policy. We first present theoretical results in the case that exact posterior inference can be conducted to obtain posterior samples of the dynamics (Section 3.4.1), we then extend the convergence guarantee to the case where approximate inference (MCMC-based) is necessary to produce posterior samples (Section 3.4.2).

3.4.1 Exact Inference Case

The main step is proving that root sampling does not alter the behavior of BA-UCT. Our proof is an adaptation of the POMCP proof by Silver and Veness (2010). We then provide some intuition and some empirical evidence of convergence on simple Bandit problems — where the Bayes-optimal solution is known.

Consider the BA-UCT algorithm: UCT applied to the Bayes-Adaptive MDP (its dynamics are described in Equation 2.25). Let $\mathcal{D}^{\tilde{\pi}}$ be the *rollout distribution* of BA-UCT: $\mathcal{D}^{\tilde{\pi}}(h_T)$ is the probability that history h_T is generated when running the BA-UCT search from $\langle s_t, h_t \rangle$, with h_t a prefix of h_T, $T-t$ the effective horizon in the search tree, and $\tilde{\pi}$ is an arbitrary EE policy. Similarly define the quantities $\tilde{\mathcal{D}}^{\tilde{\pi}}(h_T)$: the probability that history h_T is generated when running the BAMCP algorithm, and $\tilde{P}_h(\mathcal{P})$: the distribution of $\mathcal{P}$ at node h when running BAMCP. The following lemma shows that these rollout statistics are the same under BAMCP as BA-UCT.

Lemma 1 $\mathcal{D}^{\tilde{\pi}}(h_T) = \tilde{\mathcal{D}}^{\tilde{\pi}}(h_T)$ *for all EE policies* $\tilde{\pi} : \mathcal{H} \to A$*.*

Proof Let $\tilde{\pi}$ be arbitrary. We show by induction on the horizon that for all suffix histories h of h_t, (a) $\mathcal{D}^{\tilde{\pi}}(h) = \tilde{\mathcal{D}}^{\tilde{\pi}}(h)$; and (b) $P(\mathcal{P}|h) = \tilde{P}_h(\mathcal{P})$, where $P(\mathcal{P}|h)$ denotes (as before) the posterior distribution over the dynamics given h.

Base case: At the root ($h = h_t$, suffix history of size 0), it is clear that $\tilde{P}_{h_t}(\mathcal{P}) = P(\mathcal{P}|h_t)$ since we are sampling from the posterior at the root node and $\mathcal{D}^{\tilde{\pi}}(h_t) = \tilde{\mathcal{D}}^{\tilde{\pi}}(h_t) = 1$ since all simulations go through the root node.

Step case:

Assume proposition true for all suffices of size j. Consider any suffix has' of size $j+1$, where $a \in A$ and $s' \in S$ are arbitrary and h is an arbitrary suffix of size j

ending in s. The following relation holds:

$$\mathcal{D}^{\tilde{\pi}}(has') = \mathcal{D}^{\tilde{\pi}}(h)\tilde{\pi}(h,a)\int_{\mathcal{P}} \mathrm{d}\mathcal{P}\, P(\mathcal{P}|h)\,\mathcal{P}(s,a,s') \tag{3.10}$$

$$= \tilde{\mathcal{D}}^{\tilde{\pi}}(h)\tilde{\pi}(h,a)\int_{\mathcal{P}} \mathrm{d}\mathcal{P}\, \tilde{P}_h(\mathcal{P})\,\mathcal{P}(s,a,s') \tag{3.11}$$

$$= \tilde{\mathcal{D}}^{\tilde{\pi}}(has'), \tag{3.12}$$

where the second line is obtained using the induction hypothesis, and the rest from the definitions. In addition, we can match the distribution of the samples $\mathcal{P}$ at node has':

$$P(\mathcal{P}|has') = P(has'|\mathcal{P})P(\mathcal{P})/P(has') \tag{3.13}$$

$$= P(h|\mathcal{P})P(\mathcal{P})\,\mathcal{P}(s,a,s')/P(has') \tag{3.14}$$

$$= P(\mathcal{P}|h)P(h)\,\mathcal{P}(s,a,s')/P(has') \tag{3.15}$$

$$= ZP(\mathcal{P}|h)\,\mathcal{P}(s,a,s') \tag{3.16}$$

$$= Z\tilde{P}_h(\mathcal{P})\,\mathcal{P}(s,a,s') \tag{3.17}$$

$$= Z\tilde{P}_{ha}(\mathcal{P})\,\mathcal{P}(s,a,s') \tag{3.18}$$

$$= \tilde{P}_{has'}(\mathcal{P}), \tag{3.19}$$

where Equation 3.17 is obtained from the induction hypothesis, Equation 3.18 is obtained from the fact that the choice of action at each node is made independently of the samples $\mathcal{P}$. Finally, to obtain Equation 3.19 from Equation 3.18, consider the probability that a sample $\mathcal{P}$ arrives at node has', it first needs to traverse node ha (this occurs with probability $\tilde{P}_{ha}(\mathcal{P})$) and then, from node ha, the state s' needs to be sampled (this occurs with probability $\mathcal{P}(s,a,s')$); therefore, $\tilde{P}_{has'}(\mathcal{P}) \propto \tilde{P}_{ha}(\mathcal{P})\,\mathcal{P}(s,a,s')$. Z is the normalization constant: $Z = {}^{1}/(\int_{\mathcal{P}} d\mathcal{P}\,\mathcal{P}(s,a,s')P(\mathcal{P}|h)) = {}^{1}/(\int_{\mathcal{P}} d\mathcal{P}\,\mathcal{P}(s,a,s')\tilde{P}_h(\mathcal{P}))$. This completes the induction. □

The proof of Lemma 1 does not make explicit the use of lazy sampling, since this method for realizing the values of relevant random variables does not affect the rollout distribution and so does not affect what is being computed, only how.

Define $V(\langle s,h\rangle) = \max_{a \in A} Q(\langle s,h\rangle, a) \;\; \forall \langle s,h\rangle \in S \times \mathcal{H}$. We now show that BAMCP converges to the Bayes-optimal solution.

Theorem 1 *For all $\epsilon > 0$ (the numerical precision, see Algorithm 1) and a suitably chosen c (e.g. $c > \frac{R_{\max}}{1-\gamma}$), from state $\langle s_t, h_t\rangle$, BAMCP constructs a value function at the root node that converges in probability to an ϵ'-optimal value function, $V(\langle s_t, h_t\rangle) \xrightarrow{p} V^*_{\epsilon'}(\langle s_t, h_t\rangle)$, where $\epsilon' = \frac{\epsilon}{1-\gamma}$. Moreover, for large enough $N(\langle s_t, h_t\rangle)$, the bias of $V(\langle s_t, h_t\rangle)$ decreases as $O(\log(N(\langle s_t, h_t\rangle))/N(\langle s_t, h_t\rangle))$.*

Proof The UCT analysis by Kocsis and Szepesvári (2006) applies to the BA-UCT algorithm, since it is vanilla UCT applied to the BAMDP (a particular MDP). It also applies for arbitrary rollout policies, including the one developed in Section 3.3. By Lemma 1, BAMCP simulations are equivalent in distribution to BA-UCT simulations. The nodes in BAMCP are therefore evaluated exactly as in BA-UCT, providing the result. □

Lemma 1 provides some intuition for why belief updates are unnecessary in the search tree: the search tree filters the samples from the root node so that the distribution of samples at each node is equivalent to the distribution obtained when explicitly updating the belief. In particular, the root sampling in POMCP (Silver and Veness, 2010) and thus BAMCP is different from evaluating the tree using the posterior mean. This is illustrated empirically in Figures 3.4 and 3.5 in the case of simple Bandit problems.

3.4.2 Approximate Inference Case

In Theorem 1, we made the implicit assumption that BAMCP is provided with true samples drawn iid from the posterior. However, most sophisticated priors will require some form of approximate sampling scheme (see, for example, the task in Section 4.2 and the domain in Chapter 6), such as Markov Chain Monte Carlo (MCMC), which generally deliver correlated posterior samples after the

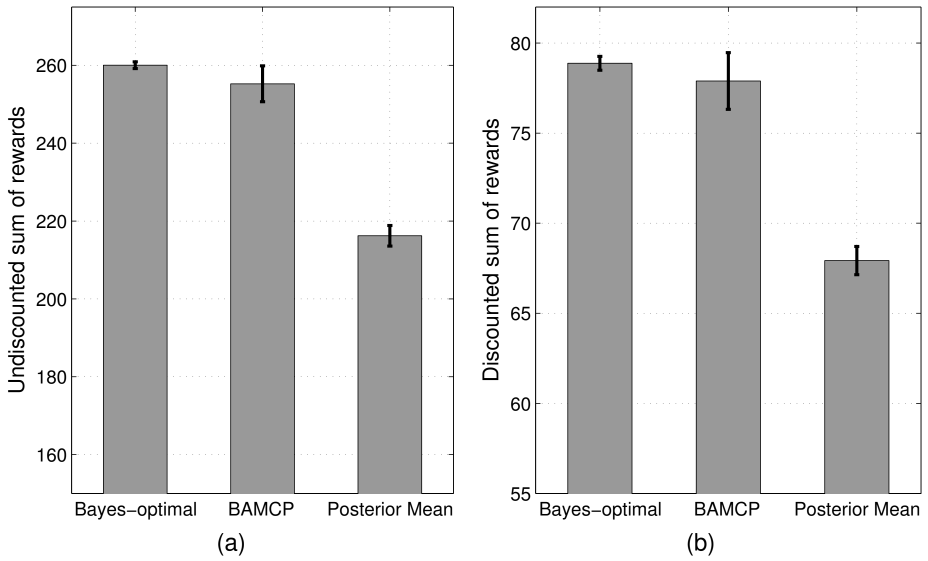


Figure 3.4: Performance comparison of BAMCP (50000 simulations, 100 runs) against the posterior mean decision on an 8-armed Bernoulli bandit with $\gamma = 0.99$ after 300 steps. The arms' success probabilities are all 0.6 except for one arm which has success probability 0.9. The Bayes-optimal result is obtained from 1000 runs with the Gittins indices (Gittins et al., 1989). **a.** Mean sum of rewards after 300 steps. **b.** Mean sum of discounted rewards after 300 steps.

chain converges to the stationary distribution (Neal, 1993). Thus, it is necessary to extend the proof of convergence of BAMCP to deal with samples of this nature.

Theorem 2 *Let $\epsilon > 0$. When using an approximate sampling procedure based on a MCMC chain with stationary distribution $P(\mathcal{P}|h_t)$ (e.g., Metropolis-Hastings or Gibbs sampling) to produce a sample sequence $\mathcal{P}^1, \mathcal{P}^2, \dots$ at the root node of BAMCP, the value $V(\langle s_t, h_t\rangle)$ found by BAMCP at the root node converges in probability to an ϵ-optimal value function. In other words, $V(\langle s_t, h_t\rangle) \xrightarrow{p} V^*_\epsilon(\langle s_t, h_t\rangle)$ where $|V^*_\epsilon(\langle s_t, h_t\rangle) - V^*(\langle s_t, h_t\rangle)| < \epsilon$.*

Proof Let $\epsilon > 0$ be the chosen numerical accuracy of the algorithm. We can choose a finite depth T for the search tree as a function of ϵ, $r_{\max}$, and γ that guarantees the total return after depth T amounts to less than ϵ. Now consider any leaf Q-node i of that tree, with mean value $\mu_{in} = \frac{1}{n}\sum_{m=1}^{n} r_m$ after n simulations, where r_m is the reward obtained from this node at the m-th simulation going through that node. Since UCB1 is used throughout the tree, exploration

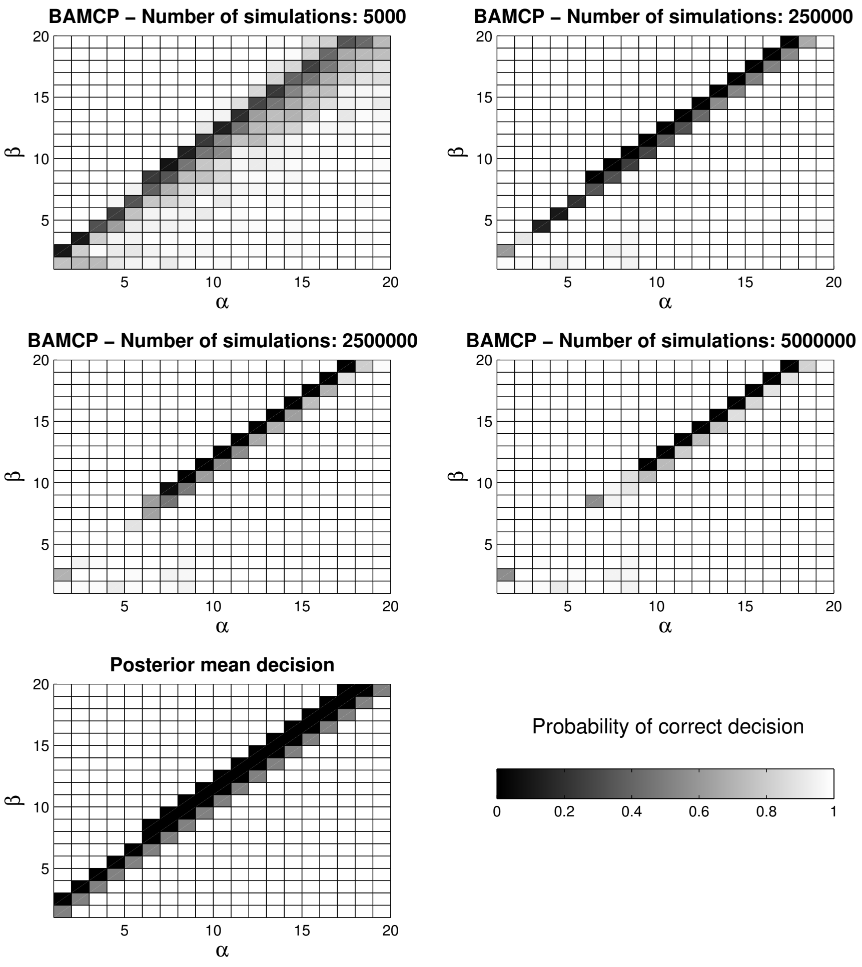


Figure 3.5: Evaluation of BAMCP against the Bayes-optimal policy, for the case $\gamma = 0.95$, when choosing between a deterministic arm with reward 0.5 and a stochastic arm with reward 1 with posterior probability $p \sim$ Beta(α, β). The result is tabulated for a range of values of α, β, each cell value corresponds to the probability of making the correct decision (computed over 50 runs) when compared to the Gittins indices (Gittins et al., 1989) for the corresponding posterior. The first four tables corresponds to different number of simulations for BAMCP and the last table shows the performance when acting according to the posterior mean. In this range of α, β values, the Gittins indices for the stochastic arm are larger than 0.5 (i.e., selecting the stochastic arm is optimal) for $\beta \leq \alpha + 1$ but also $\beta = \alpha + 2$ for $\alpha \geq 6$. Acting according to the posterior mean is different from the Bayes-optimal decision when $\beta > \alpha$ and the Gittins index is larger than 0.5. BAMCP is guaranteed to converge to the Bayes-optimal decision in all cases, but convergence is slow for the edge cases where the Gittins index is close to 0.5 (e.g., For $\alpha = 17, \beta = 19$, the Gittins index is 0.5044 which implies a value of at most $0.5044/(1-\gamma) = 10.088$ for the stochastic arm versus a value between 10 and $0.5 + \gamma \times 10.088 = 10.0836$ for the deterministic arm).

never ceases and this guarantees that $n \to \infty$ (see for example (Kocsis and Szepesvári, 2006, Thm. 3)).

Root sampling filtering (Lemma 1) still holds despite the approximate sampling at the root node; since it is a statement about the distribution of samples, not about the order in which these samples arrive. Therefore, the distribution of dynamics at node i converges to the right stationary distribution $P(\mathcal{P}|h_i)$, where h_i is the history corresponding to node i. Asymptotic results on Markov Chains (Law of large numbers for Markov Chains) guarantee us that $\mu_{in} \to \mu_i$ a.s., where μ_i is the true expected reward at leaf node i.

Given convergence at the leaves, we can work our way up the tree by backward induction to show that the values at each node converge to their (near-)optimal values. In particular the value at the root converges to an $\epsilon-$optimal value. □

3.5 Possible Misuse of Latent Variable Information: a Counter-Example

When planning in a BAMDP using a sample-based forward-search algorithm such as BAMCP, it could be tempting to use the knowledge available in the sampler when producing samples (such as the value of latent variables in the model) to take better planning decisions. For example, when generating a sample $\mathcal{P}^i$ of the dynamics according to a posterior distribution $P(\mathcal{P}|h)$ which can be written as $\int_\theta P(\mathcal{P}|\theta)P(\theta|h)$, $\mathcal{P}^i$ might have been generated by sampling θ^i from $P(\theta|h)$ before sampling $\mathcal{P}^i$ from $P(\mathcal{P}|\theta^i)$. Since the value of θ is available and contains high-level information, one natural question is to ask whether the search can be informed by the value of θ.

Here, we outline one incorrect way of using the latent variable value during search. Suppose we would want to split our search tree on the value of θ (this would occur implicitly if we were constructing history features based on the value of θ), we provide below a simple counter-example that shows that this is not a valid search approach.

Consider a simple prior distribution on two 5-state MDPs, illustrated in Figure 3.6, where $P(\theta = 0|h_0) = P(\theta = 1|h_0) = \frac{1}{2}$, and $P(\mathcal{P}|\theta)$ is a delta function on the illustrated MDP.

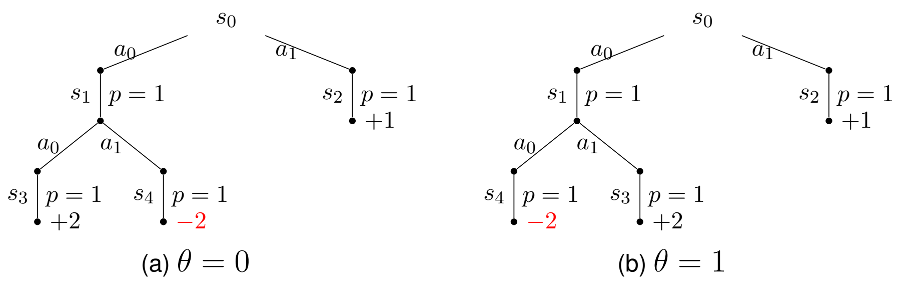


Figure 3.6: The two possible MDPs corresponding to the two settings of θ.

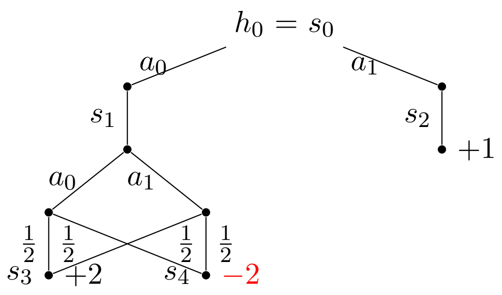


Figure 3.7: BAMDP, nodes correspond to belief(or history)-states.

There are 2 deterministic actions (a_0, a_1) in each MDP, the episode length is 1 or 2 steps. The only difference between the two MDPs is the outcome of taking action a_0 and a_1 in state s_1, as illustrated in Figure 3.6, so that a_0 is rewarding when $\theta = 0$ and costly when $\theta = 1$, and vice-versa for a_1. All the rewards are obtained from executing any action at any of the terminal states (s_2, s_3, s_4).

Observing the first transition is not informative, which implies that the posterior distribution is unchanged after the first transition: $P(\mathcal{P}|h_0) = P(\mathcal{P}|h_0 a_0 s_1) = P(\mathcal{P}|h_0 a_1 s_2)$. The BAMDP corresponding to this problem is illustrated in Figure 3.7.

At history-state $h_0 = s_0$, the Bayes-optimal Q values can easily be computed:

$$Q^*(h_0, a_1) = \gamma, \tag{3.20}$$

$$Q^*(h_0a_0s_1, a_0) = 0 + \gamma\left(2 \cdot P(s_3|h_0a_0s_1a_0) - 2 \cdot P(s_4|h_0a_0s_1a_0)\right) \tag{3.21}$$

$$= \gamma(1-1) = 0, \tag{3.22}$$

$$Q^*(h_0a_0s_1, a_1) = 0 + \gamma\left(2 \cdot P(s_3|h_0a_0s_1a_0) - 2 \cdot P(s_4|h_0a_0s_1a_0)\right) \tag{3.23}$$

$$= \gamma(1-1) = 0, \tag{3.24}$$

$$Q^*(h_0, a_0) = 0 + \gamma \max_a Q^*(h_0a_0s_1, a) = 0, \tag{3.25}$$

which implies that $a_1 = \pi^*(h_0)$ for any γ. We used the fact that $P(s_3|h_0a_0s_1a_0) = P(\theta = 0|h_0a_0s_1a_0) \cdot P(s_3|\theta = 0, s_1a_0) + P(\theta = 1|h_0a_0s_1a_0) \cdot P(s_3|\theta = 1, s_1a_0) = \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 0 = \frac{1}{2}$, and similarly for $P(s_4|h_0a_0s_1a_0)$.

Note that, since belief updates only occur at the terminal states, forward-search with or without root sampling will be equivalent. They both would construct a search tree as in Figure 3.7 and compute the right value and right decision.

The problem comes in if we decide to split our search tree at chance nodes based on the value of θ in the generated samples going down the tree. For example, after taking action a_0 in state s_0, we would be using either an MDP for which $\theta = 0$ w.p 0.5 or an MDP for which $\theta = 1$ w.p. 0.5. Since multiple values of θ go through the node h_0a_0, we would branch the tree as illustrated in Figure 3.8. This search tree is problematic because the value computed for $Q^*(h_0, a_0)$ becomes $2 \cdot \gamma^2$, which is larger than $Q^*(h_0, a_1) = \gamma$ for any $\gamma > 0.5$. Therefore, the policy that is computed at the root is no longer Bayes-optimal.

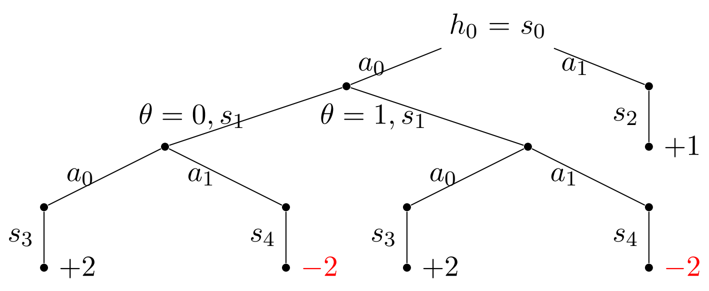


Figure 3.8: A problematic search tree.

By branching on the latent variable value, we are creating spurious observations: we are implying that the latent variable from the past will be observed in the future, which is not the case.

To summarize, the Bayes-adaptive policy to be optimized must be a function of future histories (i.e., things we will actually observe in the future), and cannot be a function of future unobserved latent variables. Ignoring this causes problems in simple domains such as the one illustrated above, but similar scenarios would occur in more complex latent variable models for the same reasons.

3.6 Conclusion

We have introduced a sample-based algorithm based on MCTS that performs Bayes-adaptive planning in discrete domains, which we called BAMCP. We started from the BA-UCT algorithm (i.e., UCT applied to the BAMDP) and, through a series of principled modifications, obtained an asymptotically Bayes-optimal planning algorithm that has the potential to scale to large domains. In the next chapter, we evaluate empirically the performance of BAMCP.

IV

BAMCP: Experimental Results

OUTLINE

This chapter contains an empirical evaluation of the BAMCP algorithm. We first present results of BAMCP on a set of standard problems with comparisons to other popular algorithms. We then showcase BAMCP's advantages in a large scale task: an infinite 2D grid with complex correlations between reward locations.

We applied BAMCP to a representative sample of benchmark problems and competitive algorithms from the literature. It consistently and significantly outperformed existing Bayesian RL methods, and also recent non-Bayesian approaches, thus achieving state-of-the-art performance.

Further, BAMCP is particularly well suited to support planning in large domains in which richly structured prior knowledge makes lazy sampling both possible and effective. This offers the prospect of applying Bayesian RL at a realistically complex scale. We illustrate this possibility by showing that BAMCP can tackle a domain with an infinite number of states and a structured prior over the dynamics, a challenging, if not radically intractable, task for existing approaches. This example exploits BAMCP's ability to use Markov chain Monte Carlo methods for inference associated with the posterior distribution over models.

4.1 Standard Domains

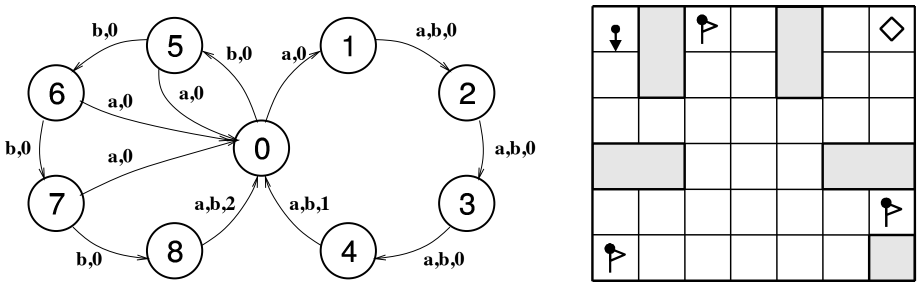
The following algorithms were run on the standard domains: BAMCP, SBOSS, BEB, BFS3. Details about their implementation and parametrization can be found in Section 4.1.3. In addition, we report results from the work by Strens (2000) for several other algorithms.

4.1.1 Description

For all the following domains, we fix $\gamma = 0.95$.

- The **Double-loop** domain is a 9-state deterministic MDP with 2 actions (Dearden et al., 1998), 1000 steps are executed in this domain. It is illustrated in Figure 4.1a.
- **Grid5** is a 5×5 grid with a reset state in one corner, and a single reward state diametrically opposite to the reset state. Actions in cardinal directions are executed with small probability of failure ($p_{\text{failure}} = 0.2$) for 1000 steps.

- **Grid10** is a 10×10 grid designed in the same way as Grid5. We collect 2000 steps in this domain.
- **Dearden's Maze** is a 264-states maze with 3 flags to collect Dearden et al. (1998). A special state provides reward equivalent to the number of flags collected since the last visit. 20000 steps are executed in this domain[1]. It is illustrated in Figure 4.1b.



(a) (b)

Figure 4.1: Two of the standard domains described in Section 4.1: a) The Double-loop domain, b) Dearden's maze. Figures from the work of Strens (2000).

To quantify the performance of each algorithm, we measured the total undiscounted reward over many steps. We chose this measure of performance to enable fair comparisons to be drawn with prior work. In fact, we are optimising a different criterion – the discounted reward from the start state – and so we might expect this evaluation to be unfavourable to our algorithm.

Although one major advantage of Bayesian RL is that one can specify priors about the dynamics, for these domains, we used rather generic priors to enable comparisons with previous work. For the Double-loop domain, the Bayesian RL algorithms were run with a simple Dirichlet-Multinomial model with symmetric Dirichlet parameter $\alpha = \frac{1}{|S|}$. For the grids and the maze domain, the algorithms were run with a sparse Dirichlet-Multinomial model, as described by Friedman and Singer (1999). For both these models, efficient collapsed sampling schemes are available; they are employed for the BA-UCT and BFS3 algorithms in our ex-

[1]The result reported for Dearden's maze with the Bayesian DP alg. by Strens (2000) is for a different version of the task in which the maze layout is given to the agent.

periments to compress the posterior parameter sampling and the transition sampling into a single transition sampling step. This considerably reduces the cost of belief updates inside the search tree when using these simple probabilistic models. Unfortunately, efficient collapsed sampling schemes are not available in general (see for example the model in Section 4.2).

4.1.2 Results

A summary of the results is presented in Table 4.1. Figures 4.2 and 4.3 report the planning time/performance trade-off for the different algorithms on the Grid5 and Maze domain.

	Double-loop	Grid5	Grid10	Dearden's Maze
BAMCP	**387.6** $\pm$ **1.5**	**72.9** $\pm$ **3**	**32.7** $\pm$ **3**	**965.2** $\pm$ **73**
BFS3 (Asmuth and Littman, 2011)	382.2 $\pm$ 1.5	66 $\pm$ 5	10.4 $\pm$ 2	240.9 $\pm$ 46
SBOSS (Castro and Precup, 2010)	371.5 $\pm$ 3	59.3 $\pm$ 4	21.8 $\pm$ 2	671.3 $\pm$ 126
BEB (Kolter and Ng, 2009)	386 $\pm$ 0	67.5 $\pm$ 3	10 $\pm$ 1	184.6 $\pm$ 35
Bayesian DP* (Strens, 2000)	377 $\pm$ 1	-	-	-
Bayes VPI+MIX* (Dearden et al., 1998)	326 $\pm$ 31	-	-	817.6 $\pm$ 29
IEQL+* (Meuleau and Bourgine, 1999)	264 $\pm$ 1	-	-	269.4 $\pm$ 1
QL Boltzmann*	186 $\pm$ 1	-	-	195.2 $\pm$ 20

Table 4.1: Experiment results summary. For each algorithm, we report the mean sum of rewards and confidence interval for the best performing parameter within a reasonable planning time limit (0.25 s/step for Double-loop, 1 s/step for Grid5 and Grid10, 1.5 s/step for the Maze). For BAMCP, this simply corresponds to the number of simulations that achieve a planning time just under the imposed limit. * Results by Strens (2000) reported without timing information.

On all the domains tested, BAMCP performed best. Other algorithms came close on some tasks, but only when their parameters were tuned to that specific domain. This is particularly evident for BEB, which required a different value of exploration bonus to achieve maximum performance in each domain. BAMCP's performance is stable with respect to the choice of its exploration constant ($c = 3$) and it did not require fine tuning to obtain the results.

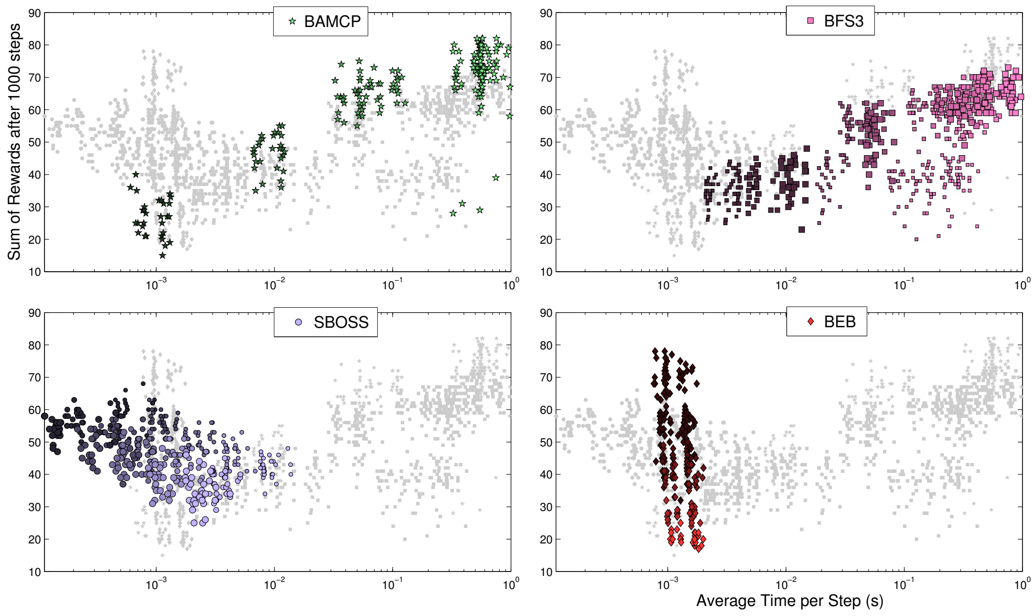


Figure 4.2: Performance of each algorithm on the Grid5 domain as a function of planning time. Each point corresponds to a single run of an algorithm with an associated setting of the parameters. Increasing brightness inside the points codes for an increasing value of a parameter (BAMCP and BFS3: number of simulations, BEB: bonus parameter β, SBOSS: number of samples K). A second dimension of variation is coded as the size of the points (BFS3: branching factor C, SBOSS: resampling parameter δ). The range of parameters is specified in Section 4.1.3.

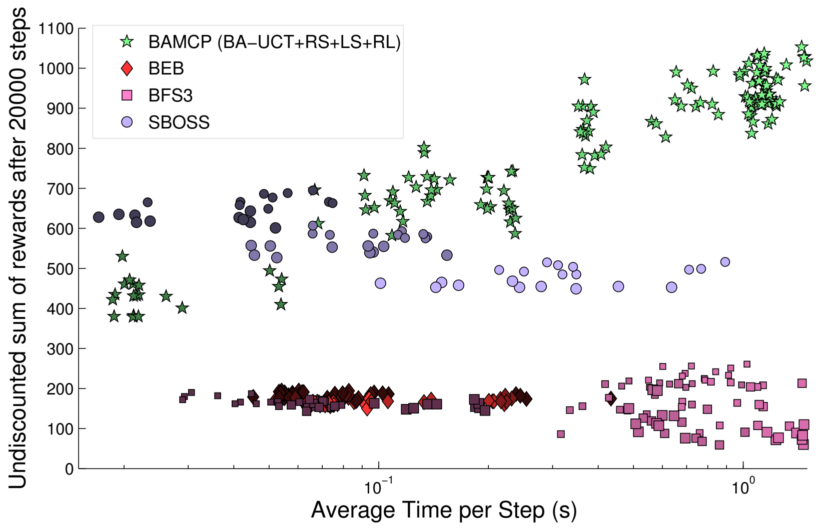


Figure 4.3: Performance of each algorithm, as in Figure 4.2 but on Dearden's Maze domain (RS = Root Sampling, LS = Lazy Sampling, RL = Rollout Learning).

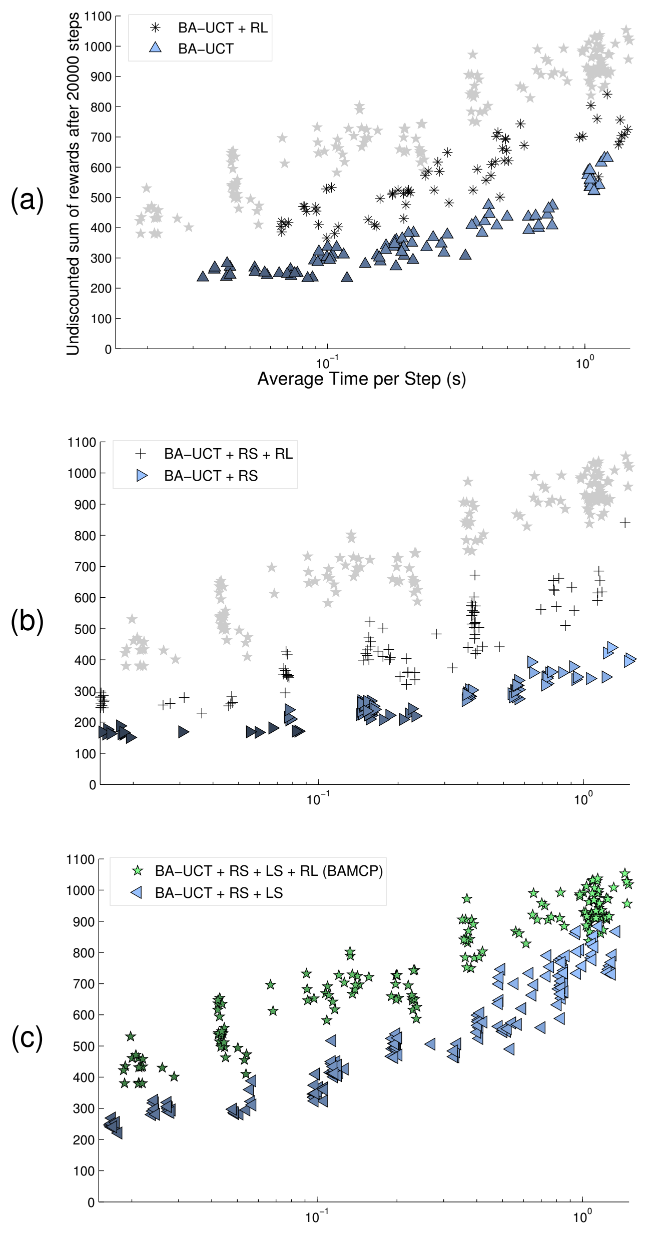


Figure 4.4: Evolution of performance from BA-UCT to BAMCP on Dearden's Maze domain. BAMCP is present on all plots for comparison, as also displayed in Figure 4.3. **a.** Performance of vanilla BA-UCT with and without rollout policy learning (RL) presented in Section 3.3. **b.** Performance of BA-UCT with Root Sampling (RS), as presented in Section 3.1, and with and without rollout learning. **c.** Performance of BA-UCT with Root Sampling and Lazy Sampling (LS), as presented in Section 3.2. With the addition of rollout policy learning, this is the BAMCP algorithm.

BAMCP's performance scaled well as a function of planning time, as is evident in Figures 4.2 and 4.3. In contrast, SBOSS follows the opposite trend. If more samples are employed to build the merged model, SBOSS actually becomes too optimistic and over-explores, degrading its performance. BEB cannot take advantage of prolonged planning time at all. The performance of BFS3 generally improves with more planning time, given an appropriate choice of parameters, but it is not obvious how to trade-off the branching factor, depth, and number of simulations in each domain. BAMCP greatly benefited from our lazy sampling scheme in the experiments, providing a $35\times$ speed improvement over the naive approach in the maze domain for example; this is illustrated in Figure 4.4.

Dearden's maze aptly illustrates a major drawback of forward search sparse sampling algorithms such as BFS3. Like many maze problems, all rewards are zero for at least k steps, where k is the solution length. Without prior knowledge of the optimal solution length, all upper bounds will be higher than the true optimal value until the tree has been fully expanded up to depth k – even if a simulation happens to solve the maze. In contrast, once BAMCP discovers a successful simulation, its Monte-Carlo evaluation will immediately bias the search tree towards the successful trajectory.

Figure 4.4 confirms that, even on a moderate-sized domain with a simple prior (Independent Sparse Dirichlet-Multinomial), BAMCP amply benefits from root sampling, lazy sampling, and rollout learning. For more complex priors, as in the following section, BA-UCT becomes computationally intractable. Root sampling and lazy sampling are then mandatory components.

4.1.3 Experimental Details

All algorithms below were implemented in C++ with code components shared across algorithms as much as possible:

- **BAMCP** - The algorithm presented in Section 3.1, implemented with root sampling, lazy sampling, and rollout learning. The algorithm was run for different number of simulations (10 to 10000) to span different planning times. In all experiments, we set π_{ro} to be an ϵ-greedy policy with $\epsilon = 0.5$. The UCT exploration constant was left unchanged for all experiments $(c = 3)$. We experimented with other values of $c \in \{0.5, 1, 5\}$ with similar results.

- **SBOSS** (Castro and Precup, 2010): for each domain, we varied the number of samples $K \in \{2, 4, 8, 16, 32\}$ and the resampling threshold parameter $\delta \in \{3, 5, 7\}$.
- **BEB** (Kolter and Ng, 2009): for each domain, we varied the bonus parameter $\beta \in \{0.5, 1, 1.5, 2, 2.5, 3, 5, 10, 15, 20\}$.
- **BFS3** (Asmuth and Littman, 2011) for each domain, we varied the branching factor $C \in \{2, 5, 10, 15\}$ and the number of simulations (10 to 2000). The depth of search was set to 15 in all domains except for the larger grid and maze domain where it was set to 50. We also tuned the V_{max} parameter for each domain — V_{min} was always set to 0.

Code for these experiments can be found online on the author's website, or directly by following this GitHub link `https://github.com/acguez/bamcp`.

4.2 Infinite 2D Grid Task

It is perhaps not unfair to characterize all the domains in the previous section as being of very limited scale. Indeed, this can be seen as a correct reflection of the state of the art of Bayesian RL. However, BAMCP, because of its root-based lazy sampling, can be applied to considerably larger and more challenging domains. We therefore designed a new problem that is well beyond the capabilities of prior algorithms since it has an infinite and combinatorially structured state space, and an even more challenging belief space. Although still abstract, this new task illustrates something of BAMCP's power.

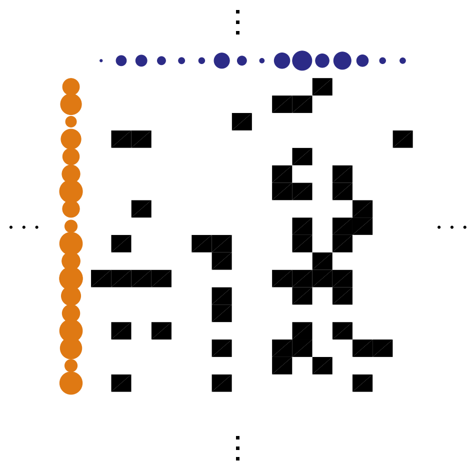


Figure 4.5: A portion of an infinite 2D grid task generated with Beta distribution parameters $\alpha_1 = 1, \beta_1 = 2$ (columns) and $\alpha_2 = 2, \beta_2 = 1$ (rows). Black squares at location (i,j) indicates a reward of 1, the circles represent the corresponding parameters p_i (blue) and q_j (orange) for each row and column (area of the circle is proportional to the parameter value). One way to interpret these parameters is that following column i implies a collection of $2p_i/3$ reward on average $(2/3$ is the mean of a Beta$(2, 1)$ distribution) whereas following any row j implies a collection of $q_j/3$ reward on average; but high values of parameters p_i are less likely than high values parameters q_j. These parameters are employed for the results presented in Figure 4.6-c).

4.2.1 Problem Description

The new problem is a class of complex MDPs over an infinite grid. In a draw of a particular MDP, each column i has an associated latent parameter $p_i \sim$ Beta(α_1, β_1) and each row j has an associated latent parameter $q_j \sim$ Beta(α_2, β_2). The probability of grid cell ij having a reward of 1 is $p_i q_j$, otherwise the reward is 0. The agent knows it is on a grid and is always free to move in any of the four cardinal directions. Rewards are consumed when visited; returning to the same location subsequently results in a reward of 0. As opposed to the independent Dirichlet priors employed in standard domains, here, dynamics are tightly correlated across states (i.e., observing a state transition provides information about other state transitions).

The domain is illustrated in Figure 4.5. Although the uncertainty appears to concern the reward function of the MDP rather than the dynamics, it can be viewed formally as uncertainty in the dynamics when the state is augmented with a binary variable that indicates whether a reward is present.[2]

Formally, since rewards disappear after one visit, the description of the state in the MDP needs to include information about the state of all the rewards (for example in the form of a set of grid locations previously visited) in addition to the position of the agent on the infinite grid. A state s is therefore the combination of the current agent's location (i, j), the unordered set of previously visited locations V, and the binary variable $R = r_{ij}$. The dynamics $\mathcal{P}$ then deterministically updates the position of the agent and the visited locations based on the agent's action, and updates R according to the reward map. The known reward function is then simply $\mathcal{R}(s, a) = s(R)$ for all a (i.e., as described before, the agent gets a reward in position ij if $r_{ij} = 1$).

4.2.2 Inference

Posterior inference (of the dynamics $\mathcal{P}$) in this model requires approximation because of the non-conjugate coupling of the variables. To see this, consider the posterior probability of a particular grid cell kl having a reward of 1 (denote this event $r_{kl} = 1$), then

$$P(r_{kl} = 1|O) = \int_{p_k, q_l} p_k q_l \; P(p_k, q_l|O) \; \mathrm{d}p_k \mathrm{d}q_l, \tag{4.1}$$

where $O = \{(i, j)\}$ is the set of observed reward locations, each associated with an observed reward $r_{ij} \in \{0, 1\}$. Sampling r_{kl} is straightforward given access to posterior samples of p_k and q_l. However, the posterior distribution on p_k and q_l,

[2]In fact, the BAMDP framework can be straightforwardly extended to deal with more general, partially-observed, reward functions (Duff, 2002).

$P(p_k, q_l|O)$, cannot be easily sampled from. It is given by:

$$P(p_k, q_l|O) \propto P(O|p_k, q_l)P(p_k)P(q_l) \tag{4.2}$$

$$= \int_{P_O \setminus p_k, Q_O \setminus q_l} P(O|P_O, Q_O) \prod_{p \in P_O} P(p) \prod_{q \in Q_O} P(q) \tag{4.3}$$

$$= \int_{P_O \setminus p_k, Q_O \setminus q_l} \prod_{(i,j) \in O} (p_i q_j)^{r_{ij}} (1 - p_i q_j)^{1 - r_{ij}}$$

$$\prod_{p \in P_O} \text{Beta}(p; \alpha_1, \beta_1) \prod_{q \in Q_O} \text{Beta}(q; \alpha_2, \beta_2), \tag{4.4}$$

where P_O denotes the set of parameters p_i for all observed columns i (columns where at least one observation exists) and similarly for Q_O with rows. This posterior suffers from non-conjugacy (because of the multiplicative interaction between the two Beta distribution) but also from a complicated dependence structure (p_k and q_l depend on observations outside of column k and row l). For these reasons, the inference is done approximately.

We construct a Markov Chain using the Metropolis-Hastings algorithm to sample from the posterior distribution of row and column parameters given observed transitions, following the notation introduced in Section 4.2. Let $O = \{(i, j)\}$ be the set of observed reward locations, each associated with an observed reward $r_{ij} \in \{0, 1\}$. The proposal distribution chooses a row-column pair (i_p, j_p) from O uniformly at random, and samples $\tilde{\mathrm{p}}_{i_p} \sim$ Beta$(\alpha_1 + m_1, \beta_1 + n_1)$ and $\tilde{\mathrm{q}}_{j_p} \sim$ Beta$(\alpha_2 + m_2, \beta_2 + n_2)$, where $m_1 = \sum_{(i,j) \in O} \mathbf{1}_{i = i_p} r_{ij}$ (i.e., the sum of rewards observed on that column) and $n_1 = (1 - {}^{\beta_2}\!/\!{}_{2(\alpha_2 + \beta_2)}) \sum_{(i,j) \in O} \mathbf{1}_{i = i_p} (1 - r_{ij})$, and similarly for m_2, n_2 (mutatis mutandis). The n_1 term for the proposed column parameter $\tilde{\mathrm{p}}_i$ has this rough correction term, based on the prior mean failure of the row parameters, to account for observed 0 rewards on the column due to potentially low row parameters. Since the proposal is biased with respect to the true conditional distribution (from which we cannot sample), we also prevent the proposal distribution from getting too peaked. Better proposals (e.g., taking into account the sampled row parameters) could be devised, but they would likely introduce additional computational cost and the proposal above generated large enough acceptance probabilities (generally above 0.5 for our experiments).

All other parameters p_i, q_j such that i or j is present in O are kept from the last accepted samples (i.e., $\tilde{\mathrm{p}}_i = p_i$ and $\tilde{\mathrm{q}}_j = p_j$ for these is and js), and all parameters p_i, q_j that are not linked to observations are (lazily) resampled from the prior — they do not influence the acceptance probability. We denote by $Q(\mathbf{p}, \mathbf{q} \to \tilde{\mathbf{p}}, \tilde{\mathbf{q}})$ the probability of proposing the set of parameters $\tilde{\mathbf{p}}$ and $\tilde{\mathbf{q}}$ from the last accepted sample of column/row parameters $\mathbf{p}$ and $\mathbf{q}$. The acceptance probability A can then be computed as $A = \min(1, A')$ where:

$$A' = \frac{P(\tilde{\mathbf{p}}, \tilde{\mathbf{q}}|h)Q(\tilde{\mathbf{p}}, \tilde{\mathbf{q}} \to \mathbf{p}, \mathbf{q})}{P(\mathbf{p}, \mathbf{q}|h)Q(\mathbf{p}, \mathbf{q} \to \tilde{\mathbf{p}}, \tilde{\mathbf{q}})} \tag{4.5}$$

$$= \frac{P(\tilde{\mathbf{p}}, \tilde{\mathbf{q}})Q(\tilde{\mathbf{p}}, \tilde{\mathbf{q}} \to \mathbf{p}, \mathbf{q})P(h|\tilde{\mathbf{p}}, \tilde{\mathbf{q}})}{P(\mathbf{p}, \mathbf{q})Q(\mathbf{p}, \mathbf{q} \to \tilde{\mathbf{p}}, \tilde{\mathbf{q}})P(h|\mathbf{p}, \mathbf{q})} \tag{4.6}$$

$$= \frac{p_{i_p}^{m_1}(1 - p_{i_p})^{n_1} q_{j_p}^{m_2}(1 - q_{j_p})^{n_2} P(h|\tilde{\mathbf{p}}, \tilde{\mathbf{q}})}{\tilde{\mathrm{p}}_{i_p}^{m_1}(1 - \tilde{\mathrm{p}}_{i_p})^{n_1} \tilde{\mathrm{q}}_{j_p}^{m_2}(1 - \tilde{\mathrm{q}}_{j_p})^{n_2} P(h|\mathbf{p}, \mathbf{q})}, \tag{4.7}$$

where

$$P(h|\tilde{\mathbf{p}}, \tilde{\mathbf{q}}) = \prod_{(i,j)\in O} \mathbb{1}[i = i_p \text{ or } j = j_p](\tilde{\mathrm{p}}_i \, \tilde{\mathrm{q}}_j)^{r_{ij}}(1 - \tilde{\mathrm{p}}_i \, \tilde{\mathrm{q}}_j)^{1-r_{ij}}, \tag{4.8}$$

$$P(h|\mathbf{p}, \mathbf{q}) = \prod_{(i,j)\in O} \mathbb{1}[i = i_p \text{ or } j = j_p](p_i q_j)^{r_{ij}}(1 - p_i q_j)^{1-r_{ij}}. \tag{4.9}$$

The last accepted sample is employed whenever a sample is rejected. Finally, reward values R_{ij} are resampled lazily based on the last accepted sample of the parameters p_i, q_j, when they have not been observed already. We omit the implicit deterministic mapping to obtain the dynamics $\mathcal{P}$ from these parameters.

4.2.3 Results

Planning algorithms that attempt to solve an MDP based on sample(s) (or the mean) of the posterior (e.g., BOSS, BEB, Bayesian DP) cannot directly handle this large combinatorial state space. Previous forward-search methods (e.g., BA-UCT, BFS3) can deal with the state space, but not the complex belief space: at every node of the search tree they must solve an approximate inference problem to estimate the posterior beliefs. By contrast, BAMCP limits the posterior

(a)
(b)
(c)

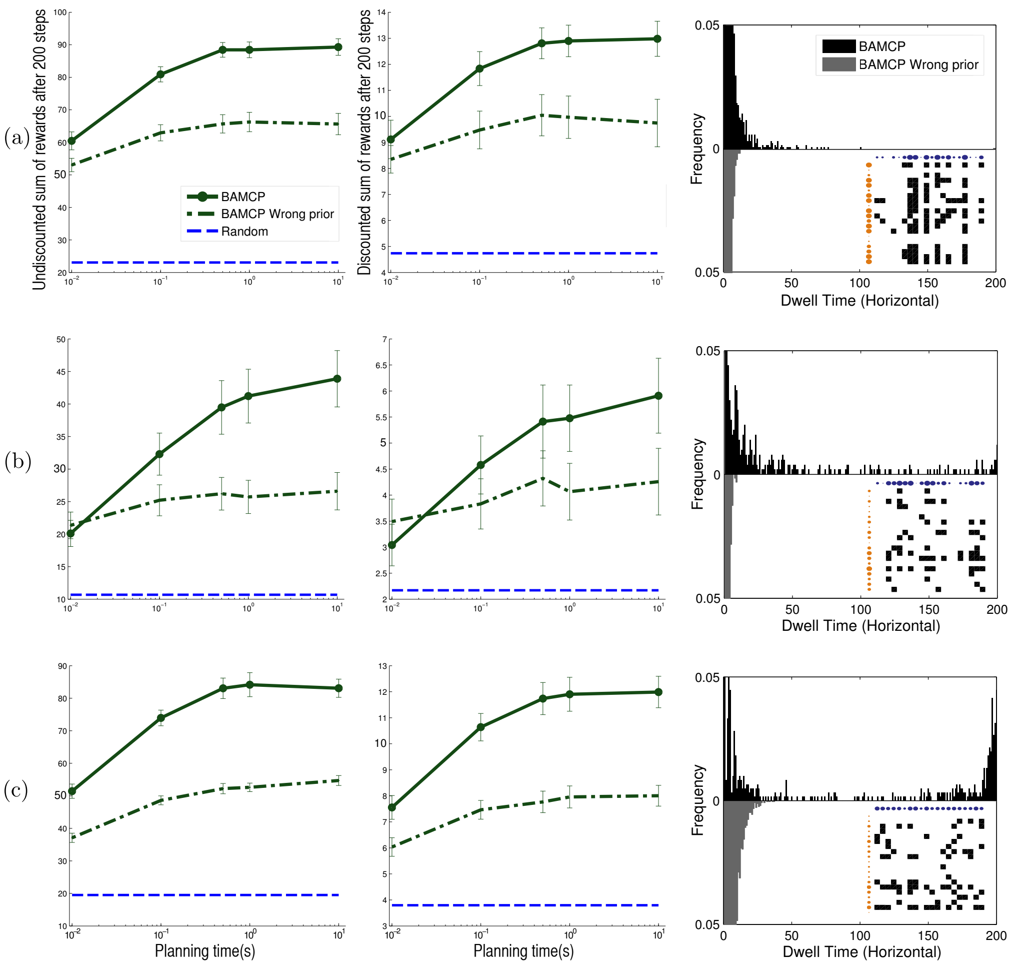


Figure 4.6: Performance of BAMCP as a function of planning time on the infinite 2D grid task, for $\gamma = 0.97$, where each row corresponds to a different set of parameters generating the grid. The performance during the first 200 steps in the environment is averaged over 50 sampled environments (5 runs for each sample) and is reported both in terms of undiscounted (left) and discounted (center) sum of rewards. BAMCP is run either with the correct generative model as prior (solid green) or with an incorrect prior (dotted green). The performance of a uniform random policy is also reported (blue). A small sample portion of a grid generated with these parameters is displayed on each row, presented as in Figure 4.5. The frequency histogram of dwell times — the number of consecutive steps the agent stays on a row before switching — is reported for each scenario. The grids are generated with Beta parameters **a)** $\alpha_1{=}0.5, \beta_1{=}0.5, \alpha_2{=}0.5, \beta_2{=}0.5$, **b)** $\alpha_1{=}0.5, \beta_1{=}0.5, \alpha_2{=}1, \beta_2{=}3$, and **c)** $\alpha_1{=}2, \beta_1{=}1, \alpha_2{=}1, \beta_2{=}2$. For the case of wrong priors (dot-dashed lines), BAMCP is given the parameters **a)** $\alpha_1{=}4, \beta_1{=}1, \alpha_2{=}0.5, \beta_2{=}0.5$, **b)** $\alpha_1{=}1, \beta_1{=}3, \alpha_2{=}0.5, \beta_2{=}0.5$, and **c)** $\alpha_1{=}1, \beta_1{=}2, \alpha_2{=}2, \beta_2{=}1$.

inference to the root of the search tree and is not directly affected by the size of the state space or belief space, which allows the algorithm to perform well even with a limited planning time. Note that lazy sampling is required in this setup since a full sample of the dynamics involves infinitely many parameters.

Figure 4.6 demonstrates the planning performance of BAMCP in this complex domain. Performance improves with additional planning time. The quality of the prior clearly affects the agent's performance, BAMCP can take advantage of correct prior information to gain more rewards. In addition, the behavior of the agent is qualitatively different depending on the prior parameters employed.

For example, for the case of Figure 4.6-a, rewards are often found in relatively dense blocks on the map and the agent exploits this fact when exploring; this explains the high frequency of short dwell times. For Figure 4.6-b, good reward rates can be obtained by following the rare rows that have high q_j parameters, but finding good rows can be expensive for at least two reasons: 1) good rows can be far from the agent's current position and 2) it takes longer to decide the value of a row if most observations lack rewards; this is because the entropy of the posterior is larger given observations of no rewards (which can be explained by either rows or columns being poor, or both at the same time) than given observations of rewards (which can be explained with high probability by both rows and columns being good, since $r_{ij} \sim$ Bernoulli$(p_i q_j)$). Hence, the agent might settle on sub-optimal rows for large periods of time, for example until it gains enough confidence that a better row is likely to be found nearby (as in Bandit problems where the Bayes-optimal agent might settle on a sub-optimal arm if it believes it likely is the best arm given past data). The heavier-tail distribution of dwell times for this scenario, in Figure 4.6-b, reflects this behavior.

The case of Figure 4.6-c consists of a mixture of rich and poor rows. The agent can determine moderately quickly if a row is not good enough, given what it expects to find, and then switches to a nearby row. Once a good enough row is found, the agent can stick to it for large periods of time. This is reflected in the bimodal nature of the distribution of dwell times in Figure 4.6-c. In many cases, the agent is satisfied with one of the first rows he visits, since it is likely that the agent starts on a good row. He then decides to stay on it for the entire duration of the episode, which explains the peak towards 200.

When BAMCP's prior belief about the dynamics is not the same as the generative model's distribution (*Wrong prior* dot-dashed lines in Figure 4.6), then maladaptive behavior can be observed. For instance, in Figure 4.6-a, the deluded agent expects most columns to be rich, and some rows to be rich and others to be poor. Hence, a good strategy given this prior belief is to find one of the good rows and exploit it by travelling horizontally. However, since a lot of columns are actually poor in this generative model, the agent never encounters the continuous sequence of rewards it expects to find on good rows. Given its wrong prior, even if on what is actually a good row, it explains the observation by the row being poor — rather than the column — and switches to a different row. This behavior is reflected in the shorter horizontal dwell times plotted in Figure 4.6-a. Similar effects can be observed in the *Wrong prior* cases of Figure 4.6-b,c.

It should be pointed out that the actual Bayes-optimal strategy in this domain is not known — the behavior of BAMCP for finite planning time might not qualitatively match the Bayes-optimal strategy. Nevertheless, we speculate that some of the behavior we observe with BAMCP, including the apparently maladaptive behaviors, would also be found in the Bayes-optimal solution.

4.3 Conclusion

We demonstrated the efficiency of BAMCP in standard discrete domains by comparing its empirical performance against existing approaches. Perhaps more importantly, we showed that BAMCP can approximate Bayes-adaptive planning in a domain with an infinite state space and a prior that requires approximate inference — a challenging task for existing planning algorithms. We now turn to a generalization of some of the ideas behind BAMCP that will allow us to consider continuous state spaces.

V

BAYES-ADAPTIVE SIMULATION-BASED SEARCH

OUTLINE

We generalize the BAMCP algorithm by incorporating function approximation to estimate the value of interaction histories during search. This enables generalization in the search tree and allow us to consider continuous state spaces. We compare this simulation-based search method to BAMCP and other approaches in three varied domains.

When performing online planning, tree-search algorithms like BAMCP treat each path of possible interaction separately. Each tree node represents a different history, with the corresponding values being stored separately. This ignores the fact that related belief-states will generally have similar values, and, worse, that different histories (for instance those containing the same observations in different orders) can correspond to the same belief. This implies that the most important method of addressing the catastrophic exponential growth of the search tree (as a function of horizon) is ignored, making tree-search algorithms inefficient in discrete state spaces, and essentially inapplicable to BAMDPs with continuous state or action spaces — except in very restricted scenarios.

In this chapter, we propose a class of efficient simulation-based algorithms for Bayes-adaptive online planning which use function approximation to estimate the value of interaction histories during search. This enables generalization between different beliefs, states, and actions during planning, and therefore also works for continuous state spaces. This result is a broadly applicable MC search algorithm for continuous BAMDPs.

Our algorithm builds on the BAMCP algorithm (described in Chapter 2) and exploits value function approximation for generalization across interaction histories, as has been proposed for simulation-based search in MDPs (Silver et al., 2012). As a crucial step towards this end, we develop a suitable parametric form for the value function estimates that can generalize appropriately across histories, using the importance sampling weights of posterior samples to compress such histories into a finite-dimensional feature vector.

As in BAMCP, we take advantage of root sampling to avoid expensive belief updates at every step of simulation. This makes the algorithm practical for a broad range of priors over environment dynamics. We also provide an interpretation of root sampling as an auxiliary variable sampling method. This leads to a new proof of its validity in general simulation-based settings, including BAMDPs with continuous state and action spaces, and a large class of algorithms that includes MC and TD upates.

Our simulation-based search algorithm for the Bayes-adaptive setting combines efficient MC search via root-sampling, and value function approximation. In Section 5.1, we first explain its underlying idea, assuming a suitable function approximator exists. In Section 5.2, we provide a novel proof justifying the use of root sampling that also applies in continuous state-action BAMDPs. Then, in Section 5.3, we explain how to model Q-values as a function of interaction histories. Finally, in Section 5.4, we investigate our approach empirically in a (discrete) bandit task and two continuous control tasks with a Gaussian process prior over the dynamics (Deisenroth and Rasmussen, 2011; Deisenroth et al., 2009). In the well-known pendulum swing-up task, our algorithm learns how to balance after just a few seconds of interaction.

5.1 Algorithm

As in other forward-search planning algorithms for Bayesian model-based RL (Asmuth and Littman, 2011; Guez et al., 2012; Ross and Pineau, 2008; Wang et al., 2005) (this includes the BAMCP algorithm presented in Chapter 2), at each step t, which is associated with the current history h_t (or belief) and state s_t, we plan online to find $\tilde{\pi}^*(\langle s_t, h_t\rangle)$ by constructing an action-value function $Q(\langle s, h\rangle, a)$. Such methods use simulation to build a search *tree* of belief states, each of whose nodes corresponds to a single (future) history, and estimate optimal values for these nodes. However, existing algorithms only update the nodes that are directly traversed in each simulation. This can be inefficient, as it fails to generalize across multiple histories corresponding either to *exactly* the same, or similar, beliefs. Instead, each such history must be traversed and updated separately.

Here, we use a more general simulation-based search that relies on function approximation, rather than a tree, to represent the values for possible simulated histories and states. This approach was originally suggested in the context of planning in large MDPs (Silver et al., 2012); we extend it to the case of Bayes-Adaptive planning. The Q-value of a particular history, state, and action is represented as $Q(\langle s, h\rangle, a; \mathbf{w})$, where $\mathbf{w}$ is a vector of learnable parameters. Fixed-length simulations are run from the current hyperstate $\langle s_t, h_t\rangle$, and the parameter $\mathbf{w}$ is updated online, during search, based on experience accumulated along these trajectories, using an incremental RL control algorithm (e.g., Monte-Carlo control, Q-learning). If the parametric form and features induce generalization between histories, then each forward simulation can affect the values of histories that are not directly experienced. This can considerably speed up planning, and enables continuous-state problems to be tackled. Note that a search tree would be a special case of the function approximation approach when the representation of states and histories is tabular.

Algorithm 2: Bayes-Adaptive simulation-based search with root sampling

```
procedure Search( $\langle s_t, h_t\rangle$ )
    Initialize $\mathbf{w}$
    repeat
        $\mathcal{P} \sim P(\mathcal{P}|h_t)$
        Simulate($s_t, h_t, \mathcal{P}, 0$)
    until Timeout()
    return $\mathrm{argmax}_a \; Q(\langle s_t, h_t\rangle, a; \mathbf{w})$
end procedure
procedure Simulate( $s, h, \mathcal{P}, t$)
    if $t > T$ then return $0$
    $a \leftarrow \tilde{\pi}_{\epsilon-\text{greedy}}(Q(\langle s, h\rangle, \cdot; \mathbf{w}))$
    $s' \sim \mathcal{P}(s, a, \cdot)$, $r \leftarrow \mathcal{R}(s, a)$
    $R \leftarrow r + \gamma$ Simulate($has', s', \mathcal{P}, t{+}1$)
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha \, (R - Q(\langle s, h\rangle, a; \mathbf{w})) \, \nabla_{\mathbf{w}} Q(\langle s, h\rangle, a; \mathbf{w})$
    return $R$
end procedure
```

In the context of Bayes-Adaptive planning, simulation-based search works by simulating a future trajectory $h_{t+T} = s_t a_t s_{t+1} \ldots a_{t+T-1} s_{t+T}$ of T transitions (the planning horizon) starting from the current belief-state. Actions are selected by following a fixed policy $\tilde{\pi}$, which is itself a function of the hyperstate,

$a \sim \tilde{\pi}(\langle s, h\rangle, \cdot)$. State transitions can be sampled according to the BAMDP dynamics, $s_{t'} \sim \mathcal{P}^+(\langle s_{t'-1}, h_{t'-1}\rangle, a_{t-1}, \langle \cdot, h_{t'-1}a_{t'-1}\cdot\rangle)$. However, this can be computationally expensive since belief updates must be applied at every step of the simulation. As an alternative, we use root sampling (as described in Section 3.1), which only samples the dynamics $\mathcal{P}^k \sim P(\mathcal{P}|h_t)$ once at the root for each simulation k and then samples transitions according to $s_{t'} \sim \mathcal{P}^k(s_{t'-1}, a_{t'-1}, \cdot)$; we provide justification for this approach in Section 5.2. After the trajectory h_T has been simulated on a step, the Q-value is modified by updating $\mathbf{w}$ based on the data in h_{t+T}. Any incremental algorithm could be used, including SARSA, Q-learning, or gradient TD (Sutton et al., 2009); we use a simple scheme to minimize an appropriately weighted squared loss $\mathbb{E}[(Q(\langle s_{t'}, h_{t'}\rangle, a_{t'}; \mathbf{w}) - R_{t'})^2]$:

$$\Delta\, \mathbf{w} = \alpha\ \left(R_{t'} - Q(\langle s_{t'}, h_{t'}\rangle, a_{t'}; \mathbf{w})\right) \nabla_{\mathbf{w}} Q(\langle s_{t'}, h_{t'}\rangle, a_{t'}; \mathbf{w}), \tag{5.1}$$

where α is the learning rate and $R_{t'}$ denotes the discounted return obtained from history $h_{t'}$.[1] In other words, we apply Monte-Carlo control to our search problem. Algorithm 2 provides pseudo-code for this scheme; here we suggest using as the fixed policy for simulation the ϵ−greedy policy $\tilde{\pi}_{\epsilon-\text{greedy}}$ based on some given Q value. Other policies could be considered (e.g., the UCT policy for search trees), but are not the main focus of this chapter. Note that $\mathbf{w}$ is only updated at the end of the recursion in Algorithm 2 using the data from a sampled trajectory, hence the policy is fixed during a given simulation.

5.2 Analysis

To exploit general results on the convergence of classical RL algorithms for our simulation-based search, it is necessary to show that, starting from the current history, root sampling produces the appropriate distribution of rollouts. For the purpose of this section, a simulation-based search algorithm includes Algorithm 2 (with Monte-Carlo backups) but also incremental variants, as discussed

[1]The squared loss is weighted according to the distribution of belief-states visited from the current state by executing the policy $\tilde{\pi}$.

above, or BAMCP.

Let $\mathcal{D}_t^{\tilde{\pi}}$ be the *rollout distribution* function of forward-simulations that explicitly updates the belief at each step (i.e., using $\mathcal{P}^+$): $\mathcal{D}_t^{\tilde{\pi}}(h_{t+T})$ is the probability density that history h_{t+T} is generated when running that simulation from $\langle s_t, h_t \rangle$, with T the horizon of the simulation, and $\tilde{\pi}$ an arbitrary history policy. Similarly define the quantity $\tilde{\mathcal{D}}_t^{\tilde{\pi}}(h_{t+T})$ as the probability density that history h_{t+T} is generated when running forward-simulations *with root sampling*, as in Algorithm 2. The following lemma, a generalization of Lemma 1 to a wider class of search algorithms, shows that these two rollout distributions are the same.

Lemma 1 *$\mathcal{D}_t^{\tilde{\pi}}(h_{t+T}) = \tilde{\mathcal{D}}_t^{\tilde{\pi}}(h_{t+T})$ for all policies $\tilde{\pi} : \mathcal{H} \times A \to [0,1]$ and for all $h_{t+T} \in \mathcal{H}$ of length $t+T$.*

Proof

A similar result has been obtained for discrete state-action spaces as Lemma 1 using an induction step on the history length. Here we provide a more intuitive interpretation of root sampling as an auxiliary variable sampling scheme which also applies directly to continuous spaces. We show the equivalence by rewriting the distribution of rollouts. The usual way of sampling histories in simulation-based search, with belief updates, is justified by factoring the density as follows:

$$\begin{aligned}
p(h_{t+T}|h_t, \tilde{\pi}) &= p(a_t s_{t+1} a_{t+1} s_{t+2} \dots s_{t+T}|h_t, \tilde{\pi}) && (5.2)\\
&= p(a_t|h_t, \tilde{\pi}) p(s_{t+1}|h_t, \tilde{\pi}, a_t)\\
&\quad p(a_{t+1}|h_{t+1}, \tilde{\pi}) \dots p(s_{t+T}|h_{t+T-1}, a_{t+T}, \tilde{\pi}) && (5.3)\\
&= \prod_{t \le t' < t+T} \tilde{\pi}(h_{t'}, a_{t'}) \prod_{t < t' \le t+T} p(s_{t'}|h_{t'-1}, \tilde{\pi}, a_{t'-1}) && (5.4)\\
&= \prod_{t \le t' < t+T} \tilde{\pi}(h_{t'}, a_{t'}) \prod_{t < t' \le t+T} \int_{\mathcal{P}} P(\mathcal{P}|h_{t'-1})\, \mathcal{P}(s_{t'-1}, a_{t'-1}, s_{t'})\, \mathrm{d}\mathcal{P}, && (5.5)
\end{aligned}$$

which makes clear how each simulation step involves a belief update in order to compute (or sample) the integrals. Instead, one may write the history density as the marginalization of the joint over history and the dynamics $\mathcal{P}$, and then notice that an history is generated in a Markovian way *if conditioned on the dynamics*:

$$p(h_{t+T}|h_t,\tilde{\pi}) = \int_{\mathcal{P}} p(h_{t+T}|\mathcal{P},h_t,\tilde{\pi})p(\mathcal{P}|h_t,\tilde{\pi})\,\mathrm{d}\mathcal{P} \tag{5.6}$$

$$= \int_{\mathcal{P}} p(h_{t+T}|\mathcal{P},\tilde{\pi})p(\mathcal{P}|h_t)\,\mathrm{d}\mathcal{P} \tag{5.7}$$

$$= \int_{\mathcal{P}} \prod_{t\le t'<t+T} \tilde{\pi}(h_{t'},a_{t'}) \prod_{t<t'\le t+T} \mathcal{P}(s_{t'-1},a_{t'-1},s_{t'})\ \ p(\mathcal{P}|h_t)\,\mathrm{d}\mathcal{P}, \tag{5.8}$$

where Equation (5.8) makes use of the Markov assumption in the MDP. This makes clear the validity of sampling only from $p(\mathcal{P}|h_t)$, as in root sampling. From these derivations, it is immediately clear that $\mathcal{D}_t^{\tilde{\pi}}(h_{t+T}) = \tilde{\mathcal{D}}_t^{\tilde{\pi}}(h_{t+T})$. □

The result in Lemma 1 does not depend on the way we update the value Q, or on its representation, since the policy is fixed for a given simulation. Furthermore, the result guarantees that simulation-based searches will be identical in distribution with and without root sampling. Thus, we have:

Corollary 1 *Define a Bayes-adaptive simulation-based planning algorithm as a procedure that repeatedly samples future trajectories $h_{t+T} \sim \mathcal{D}_t^{\tilde{\pi}}$ from the current history h_t (simulation phase), and updates the Q value after each simulation based on the experience h_{t+T} (special cases are Algorithms 1-2). Then such a simulation-based algorithm has the same distribution of parameter updates with or without root sampling. Asymptotically, this also implies that the two variants share the same fixed-points, since the updates match in distribution.*

For example, for a discrete environment we can choose a tabular representation of the value function in history space. Applying the MC updates in eq. (5.1) results in a MC control algorithm applied to the sub-BAMDP from the root state. This is exactly the (BA version of the) MC tree search algorithm BAMCP from Chapter 3. The same principle can also be applied to MC control with function approximation with convergence results under appropriate conditions (Bertsekas, 2011b), although it is useful to point out that these theoretical results are quite weak at the moment. Finally, more general updates such as gradient

Q-learning could be applied with corresponding convergence guarantees (Maei et al., 2010).

5.3 History Features and Parametric Form for the Q-value

The quality of a history policy obtained using simulation-based search with a parametric representation $Q(\langle s, h\rangle, a; \mathbf{w})$ crucially depends on the features associated with the arguments of Q, i.e., the history, state and action. These features should arrange for histories that lead to the same, or similar, beliefs to have the same, or similar, representations, to enable appropriate generalization. This is challenging since beliefs can be infinite-dimensional objects with non-compact sufficient statistics that are therefore hard to express or manipulate. Learning good representations from histories is also tough, for instance because of hidden symmetries (e.g., the irrelevance of the order of the experience tuples that lead to a particular belief).

We propose a parametric representation of the belief at a particular planning step based on *sampling*. That is, we draw a set of M independent MDP samples or particles $U = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_M\}$ from the current belief $b_t = P(\mathcal{P}|h_t)$, and associate each with a weight $z_m^U(h)$, such that the vector $z^U(h)$ is a finite-dimensional approximate representation of the belief based on the set U. We will also refer to z^U as a function $z^U : \mathcal{H} \to \mathbb{R}^M$ that maps histories to a *feature vector*.

There are various ways one could design the z^U function. It is computationally convenient to compute $z^U(h)$ recursively as importance weights, just as in a sequential importance sampling particle filter (Gordon et al., 1993); this only assumes we have access to the likelihood of the observations (i.e., state transitions). In other words, the weights are initialized as $z_m^U(h_t) = \frac{1}{M}\ \forall m$ and are then updated recursively using the likelihood of the dynamics model for that particle of observations as $z_m^U(has') \propto z_m^U(h)P(s'|a, s, \mathcal{P}_m) = z_m^U(h)\,\mathcal{P}_m(s, a, s')$.

One advantage of this definition is that it enforces a correspondence between the history and belief representations in the finite-dimensional space, in the sense that $z^U(h') = z^U(h)$ if belief$(h) =$ belief(h'). That is, we can work in history space *during planning*, alleviating the need for complete belief updates, but via a finite and well-behaved representation of the actual belief — since different histories corresponding to the same belief are mapped to the same representation.

This feature vector (i.e., $z^U(h)$) can be combined with any function approximator. In our experiments, we combine it with features of the current state and action, $\phi(s, a)$, in a simple bilinear form:

$$Q(\langle s, h \rangle, a; \mathbf{W}) = z^U(h)^T \, \mathbf{W} \, \phi(s, a), \tag{5.9}$$

where $\mathbf{W}$ is the matrix of learnable parameters adjusted during the search (eq. 5.1). Here $\phi(s, a)$ is a domain-dependent state-action feature vector as is standard in fully observable settings with function approximation. Special cases include tabular representations or forms of tile coding.

In the POMDP literature, as we mentioned in Chapter 2, a key idea to represent beliefs is to sample a finite set of (possibly approximate) *belief points* (Pineau et al., 2003; Thrun, 1999) from the set of possible beliefs in order to obtain a small number of (belief-)states for which to backup values offline or in a forward search setting (Kurniawati et al., 2008). In contrast, our sampling approach to belief representation does not restrict the number of (approximate) belief points since our belief features ($z(h)$) can take an infinite number of values, but it instead restricts the *dimension* since we are dealing with infinite-dimensional belief spaces. Our Monte-Carlo construction was also explored in the work of Wang et al. (2012) to transform the BAMDP into a discrete-state POMDP using a finite sample set from the prior distribution over dynamics. However, in that work, the POMDP was then solved offline with no (further) generalization between beliefs, and no opportunity to re-adjust the belief representation based on past experience. A function approximation scheme in the context of BA planning has been considered by Duff (2003), in an offline actor-critic paradigm. However, this was in a discrete setting where counts could be used as features for the belief.

5.4 Experimental Results

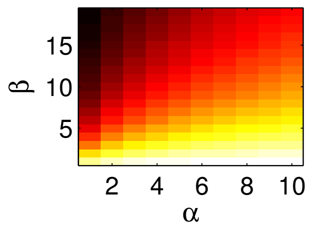
We investigate empirically in three varied domains the combination of the parametric form in the last section, simulation-based search and Monte-Carlo backups, collectively known as BAFA (for *B*ayes *A*daptive planning with *F*unction *A*pproximation).

The discrete *Bernoulli bandit* domain (Section 5.4.1) demonstrates dramatic efficiency gains due to generalization with convergence to a near Bayes-optimal solution. The *navigation task* (Section 5.4.2) and the *pendulum* (Section 5.4.3) demonstrate the ability of BAFA to handle non-trivial planning horizons for large BAMDPs with continuous states. We provide comparisons to our BA tree-search algorithm (BAMCP, Chapter 3), choosing a suitable discretization of the state space for the continuous problems. For the pendulum we also compare to two Bayesian, but not Bayes adaptive, approaches.

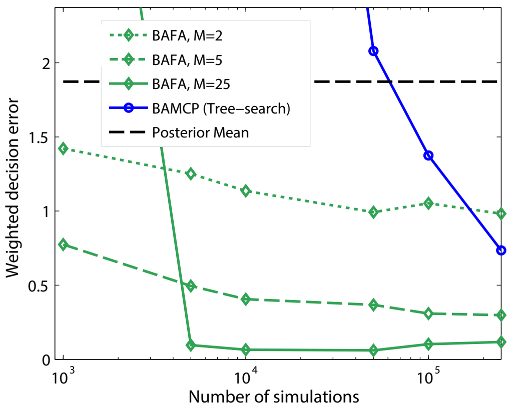
5.4.1 Bernoulli Bandit

Bandits have simple dynamics, yet they are still challenging for a generic Bayes-Adaptive planner. Importantly, ground truth is sometimes available (Gittins et al., 1989), so we can evaluate how far the approximations are from Bayes-optimality.

We consider a 2-armed Bernoulli bandit problem. We oppose an uncertain arm with prior success probability $p_1 \sim Beta(\alpha, \beta)$ against an arm with known success probability p_0. We consider the scenario $\gamma = 0.99, p_0 = 0.2$ for which the optimal decision, and the posterior mean decision, frequently differ. Decision errors for different values of α, β do not have the same consequence, so we weight each scenario according to the difference between their associated Gittins indices. Define the weight as $m_{\alpha,\beta} = |g_{\alpha,\beta} - p_0|$ where $g_{\alpha,\beta}$ is the Gittins index for α, β; this is an upper-bound (up to a scaling factor) on the difference between the value of the arms. The weights are shown in Figure 5.1-a.



(a) $m_{\alpha,\beta}$



(b)

Figure 5.1: (a) The weights $m_{\alpha,\beta}$. (b) Averaged (weighted) decision errors for the different methods as a function of the number of simulations.

We compute the weighted errors over 20 runs for a particular method as $E_{\alpha,\beta} = m_{\alpha,\beta} \cdot P(\text{Wrong decision for } (\alpha,\beta))$, and report the sum of these terms across the range $1 \leq \alpha \leq 10$ and $1 \leq \beta \leq 19$ in Figure 5.1-b as a function of the number of simulations.

Though this is a discrete problem, these results show that the value function approximation approach, even with a limited number of particles (M) for the history features, learns considerably more quickly than BAMCP. This is because BAFA generalizes between similar beliefs.

5.4.2 Height map navigation

We next consider a 2-D navigation problem on an unknown continuous height map. The agent's state is (x, y, z, θ); it moves on a bounded region of the $(x, y) \in 8 \times 8m$ plane according to (known) noisy dynamics. The agent chooses between 5 different actions. The dynamics for (x, y) are $(x_{t+1}, y_{t+1}) = (x_t, y_t) + l(\cos(\theta_a), \sin(\theta_a)) + \boldsymbol{\epsilon}$, where θ_a corresponds to the action from this set $\theta_a \in \theta + \{-\frac{\pi}{3}, -\frac{\pi}{6}, 0, \frac{\pi}{6}, \frac{\pi}{3}\}$, $\boldsymbol{\epsilon}$ is small isotropic Gaussian noise ($\sigma = 0.05$), and $l = \frac{1}{3}$m is the step size.

Within the bounded region, the reward function is the value of a latent height map $z = f(x, y)$ which is only observed at a single point by the agent. The height map is a draw from a Gaussian process (GP), $f \sim GP(0, \mathcal{K})$, using a multi-scale squared exponential kernel for the covariance matrix and zero mean. In order to test long-horizon planning, we downplay situations where the agents can simply follow the expected gradient locally to reach high reward regions by starting the agent on a small local maximum. To achieve this, we simply condition the GP draw on a few pseudo-observations with small negative z around the agent and a small positive z at the starting position, which creates a small bump (on average). The domain is illustrated in Figure 5.2-a with an example map.

We compare BAMCP against BAFA on this domain, planning over 75 steps with a discount of 0.98. Since BAMCP works with discrete state, we uniformly discretize the height observations. For the state-features in BAFA, we use a regular tile coding of the space; an RBF network leads to similar results. We use a common set of a 100 ground truth maps drawn from the prior for each algorithm/setting, and we average the discounted return over 200 runs (2 runs/map) and report that result in Figure 5.2-b as a function of the planning horizon (T). This result illustrates the ability of BAFA to cope with non-trivial planning horizons in belief space. Despite the discretization, BAMCP is very efficient with short planning horizons, but has trouble optimizing the history policy with long horizons because of the huge tree induced by the discretization of the observations.

5.4.3 Under-actuated Pendulum Swing-up

Finally, we consider the classic RL problem in which an agent must swing a pendulum from hanging vertically down to balancing vertically up, but given only limited torque. This requires the agent to build up momentum by swinging, before being able to balance. Note that although a wide variety of methods can successfully learn this task given enough experience, it is a challenging domain for Bayes-adaptive algorithms, which have duly not been tried.

We use conventional parameter settings for the pendulum (Deisenroth et al., 2009), a mass of 1kg, a length of 1m, a maximum torque of 5Nm, and coefficient of friction of 0.05 kg m^2 / s. The state of the pendulum is $s = (\theta, \dot{\theta})$. Each time-

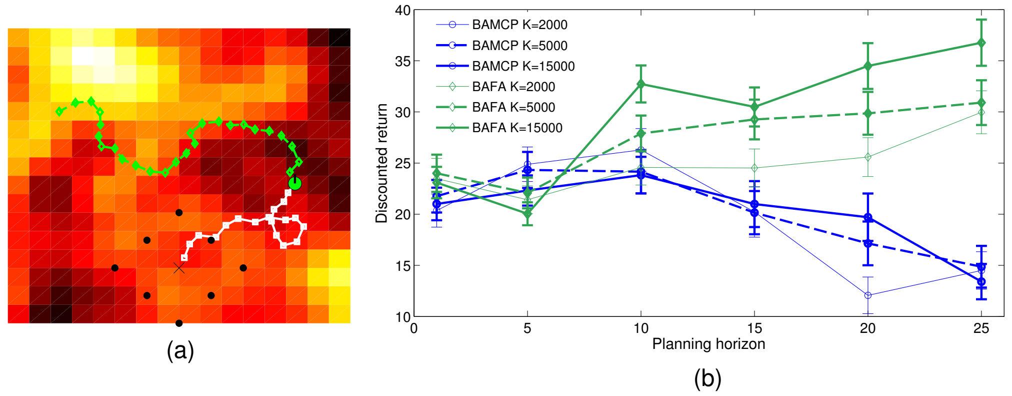


Figure 5.2: (a) Example map showing with the height color-coded from white (negative reward z) to black (positive reward z). The black dots denote the location of the initial pseudo-observations used to obtain the ground truth map. The white squares show the past trajectory of the agent, starting at the cross and ending at the current position in green. The green trajectory is one particular forward simulation of BAFA from that position. (b) Averaged discounted return (higher is better) in the navigation domain for discretized BAMCP and BAFA as a function of the number of simulations (K), and as function of the planning horizon (x-axis).

step corresponds to 0.05s, $\gamma = 0.98$, and the reward function is $\mathcal{R}(s) = \cos(\theta)$. In the initial state, the pendulum is pointing down with no velocity, $s_0 = (\pi, 0)$. Three actions are available to the agent, to apply a torque of either $\{-5, 0, 5\}$Nm.

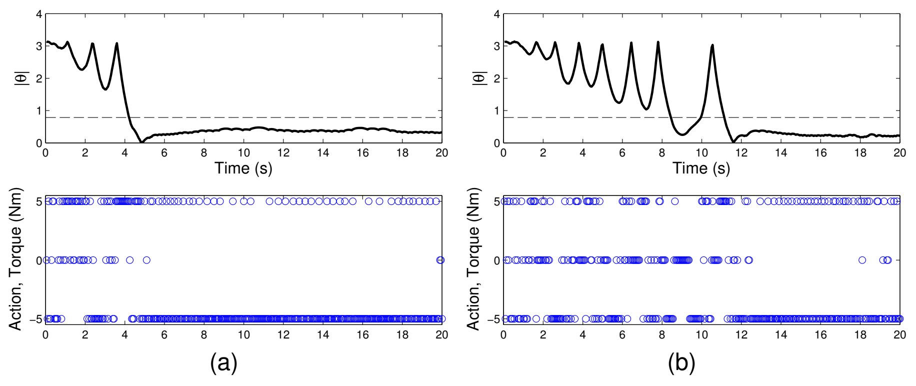


Figure 5.3: Two runs of BAFA on the pendulum domain. In each run this is the first few seconds of interaction of the agent with the domain (time refers to simulated time, not computation time). The runs are selected to illustrate a typical good run (a) and a typical slower run (b). Top row shows the absolute value of the pendulum angle θ. Bottom row shows the action selection. Dotted line marks the $\frac{\pi}{4}$ region for up-states.

The agent does not initially know the dynamics of the pendulum. As in Deisenroth et al. (2009), we assume it employs independent Gaussian processes to capture the state change in each dimension for a given action. That is, $s^i_{t+1} - s^i_t \sim GP(m^i_a, \mathcal{K}^i_a)$ for each state dimension i and each action a (where k^i_a are Squared Exponential kernels). Since there are 2 dimensions and 3 actions, we maintain 6 Gaussian processes, and plan in the joint space of $(\theta, \dot{\theta})$ together with the possible future GP posteriors to decide which action to take at any given step.

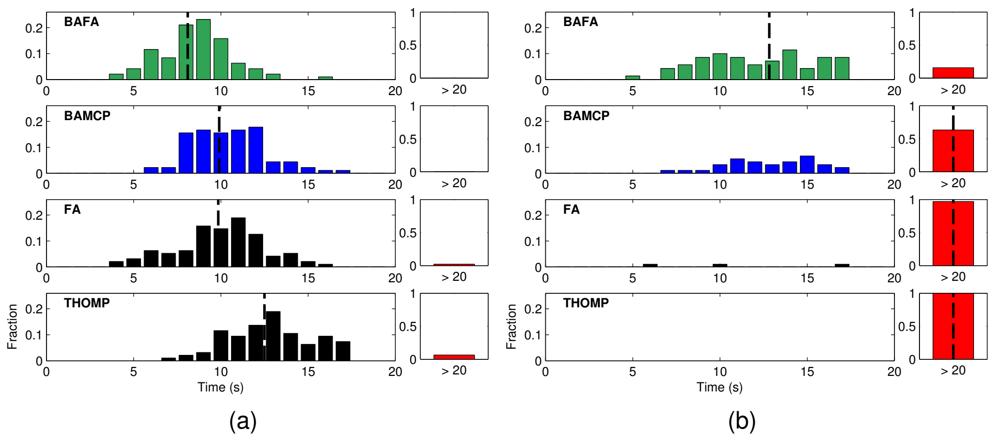


Figure 5.4: Histogram of delay until the agent reaches its first balance state ($|\theta| < \frac{\pi}{4}$ for ≥ 3s) for different methods in the pendulum domain. (a) A standard version of the pendulum problem with a cosine cost function. (b) A more difficult version of the problem with uncertain cost for balancing (see text). There is a 20s time limit, so all runs which do not achieve balancing within that time window are reported in the red bar. The histogram is computed with 100 runs with (a) $K = 10000$, or (b) $K = 15000$, simulations for each algorithm, horizon $T = 50$ and (for BAFA) $M = 50$ particles. The black dashed line represents the median of the distribution.

We compare four approaches on this problem to understand the contributions of both generalization and Bayes-Adaptive planning to the performance of the agent. BAFA includes both; we also consider two non-Bayes-adaptive variants using the same simulation-based approach with value generalization. In a Thompson Sampling variant (THOMP), we only consider a single posterior sample of the dynamics at each step and greedily solve using simulation-based search. In an exploit-only variant (FA), we run a simulation-based search that optimizes a *state-only* policy over the uncertainty in the dynamics, this is achieved by running BAFA with no history feature.[2] For BAFA, FA, and THOMP, we use the

[2]The approximate value function for FA and THOMP thus takes the form $Q(s, a) = \mathbf{w}^T \phi(s, a)$.

same RBF network for the state-action features, consisting of around 900 nodes. In addition, we also consider the BAMCP planner with a uniform discretization of the $\theta, \dot{\theta}$ space that worked best in a coarse initial search; this method performs Bayes-adaptive planning but with no value generalization.

We allow each algorithm a maximum of 20s of interaction with the pendulum (simulation time, and not computation time), and consider as up-state any configuration of the pendulum for which $|\theta| \leq \frac{\pi}{4}$ and we consider the pendulum balanced if it stays in an up-state for more than 3s. We report in Figure 5.4-a the time it takes for each method to reach for the first time a balanced state. We observe that Bayes-adaptive planning (BAFA or BAMCP) outperforms more heuristic exploration methods, with most runs balancing before 8.5s. Figure 5.3 shows traces of example runs. With the same parametrization of the pendulum, Deisenroth et al. reported balancing the pole after between 15 and 60 seconds of interaction when assuming access to a restart distribution (Deisenroth et al., 2009). More recently, Moldovan et al. reported balancing after 12-18s of interaction using a method tailored for locally linear dynamics (Moldovan et al., 2013).

However, the pendulum problem also illustrates that BA planning for this particular task is not hugely advantageous compared to more myopic approaches to exploration. We speculate that this is due to a lack of structure in the problem and test this with a more challenging, albeit artificial, version of the pendulum problem that requires non-myopic planning over longer horizons. In this modified version, balancing the pendulum (i.e., being in the region $|\theta| < \frac{\pi}{4}$) is either rewarding ($\mathcal{R}(s) = 1$) with probability 0.5, or costly ($\mathcal{R}(s) = -1$) with probability 0.5; all other states have an associated reward of 0. This can be modeled formally by introducing another binary latent variable in the model. These latent dynamics are observed with certainty if the pendulum reaches any state where $|\theta| \geq \frac{3\pi}{4}$. The rest of the problem is the same. To approximate correctly the Bayes-optimal solution in this setting, the planning algorithm must optimize the belief-state policy *after* it simulates observing whether balancing is rewarding or not. We run this version of the problem with the same algorithms as above and

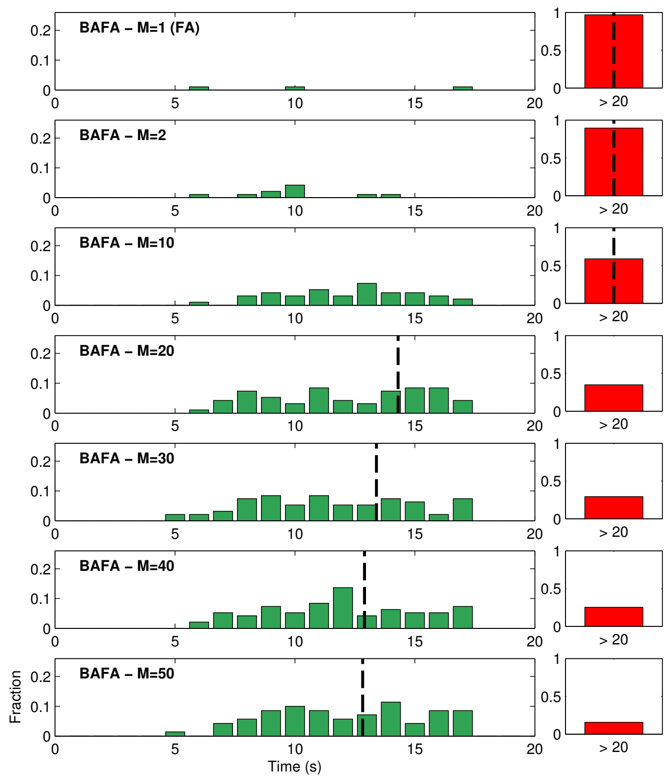


Figure 5.5: Histogram of delay until the agent reaches its first balance state ($|\theta| < \frac{\pi}{4}$ for ≥ 3s). The algorithm is BAFA, for different values of the number of particles in the belief representation (M), in the modified version of the pendulum problem with hidden costs. All the other parameters are as in Figure 3-b in the main text. We observe that around 20 particles are needed to obtain some reasonable performance in this domain. Increasing the number of particles past a certain point provides a diminishing return, since it requires more parameter to learn.

report the results in Figure 5.4-b. This hard planning problem highlights more clearly the benefits of Bayes-adaptive planning and value generalization. Our approach manages to balance the pendulum more than 80% of the time, compared to about 35% for BAMCP, while THOMP and FA fail to balance for almost all runs. Figure 5.5 illustrates the influence of the number of particles M on the performance of BAFA.

5.5 Representing the Value Function

It is known that the value function for the BAMDP is convex as a function of the belief for a particular state (Duff, 2002; Porta et al., 2006); it is piecewise linear if the horizon is finite and the state and action spaces are discrete. Suppose, for simplicity, that states and beliefs are represented exactly (i.e., for example assuming discrete states and $z^U(h) = b(h)$), then the bilinear form we introduced in Section 3.3 to represent the value function approximates the true convex value function (for a given state as a function of the belief) with a single linear function: $Q(h, s, a; \{\mathbf{w}_s\}) = \langle b(h), \mathbf{w}_s \rangle$. In general, this is not enough to represent exactly the true value function, but our experiments suggest that it is enough to reason approximately about the consequences of future beliefs.

We have also experimented with an alternative parametric form, an approximately piecewise linear form that combines multiple hyperplanes via a softmax:

$$Q(h, s, a; \{\mathbf{W}_i\}) = \sqrt[k]{\sum_i^I \left(z^U(h)^T \, \mathbf{W}_i \, \phi(s, a)\right)^k}, \tag{5.10}$$

inspired by the work of Parr and Russell in the context of POMDPs (Parr and Russell, 1995). The constants k and I are fixed parameters that trade-off computation and accuracy against the number of learnable parameters (the bilinear form is recovered from the soft-max form using $k = I = 1$). Given sufficient components, this form should be able to represesent the true value function arbitrarily closely. However, in our experiments with this more general form, this advantage was outweighed by its computational complexity, and it performed poorly in practice.

5.6 Conclusion

We have introduced a tractable approach to Bayes-adaptive planning in large or continuous state spaces. Our method is quite general, subsuming Monte Carlo tree search methods, while allowing for arbitrary generalizations over interaction histories using value function approximation. Our general framework can be applied with more powerful methods for learning the parameters of the value function approximation, and it can also be adapted to be used with continuous actions. We expect that further gains will be possible, e.g. from the use of bootstrapping in the weight updates, alternative rollout policies, and reusing values and policies between (real) steps. We will discuss some of these future directions in more details in the final chapter.

VI

Bayes-Adaptive Planning with Rich Statistical Models

Outline

The motivation for this chapter is to demonstrate the practical power of Bayes-adaptive planning. We show that, despite the arduous optimization problem, sample-based planning approximations can excel with rich models in realistic settings. Conversely, we show that more myopic forms of planning, when equipped with the same probabilistic models, can perform dramatically worse.

In Chapters 3-5, we developed sample-based algorithms for Bayes-adaptive planning. These approximate the Bayes-optimal solution directly at each step. This involves running many forward simulations to integrate over possible futures and optimize the policy, a costly process in spite of all of our approximation schemes. Given that Bayes-adaptive planning is computationally demanding, particularly for complex models, it leaves open the possibility that it might not be justified compared to heuristic approaches such as Thompson Sampling (TS) that may perform very similarly at a much reduced computational cost. Indeed, many Bayesian exploration-exploitation approaches that use complex priors side-step the planning problem by sampling from the posterior but only planning myopically (Asmuth et al., 2009; Doshi-Velez et al., 2010; Tziortziotis et al., 2013). This chapter has two goals: first (Section 6.1) explaining through explicit counter-examples the perils of myopia; second (Section 6.2) showing relevant circumstances in which Bayes adaptivity is demonstrably worthwhile.

Skepticism about BA planning is rife, because of the apparently mediocre gains it offers. We argue that this is because past work has largely been confined to domains which lack substantial structure. The computational effort in planning properly is only likely to pay off when experience can be shared extensively — something that requires richly structured priors. In Section 6.2, we consider a class of non-parametric priors based on the CRP that models shared structure across sequences of tasks. We show that, despite the optimization cost, sample-based Bayes-adaptive planning approximations can excel with such priors in realistic settings — here a challenging exploration-exploitation task that uses real data coming from a popular supervised learning problem (the UCI 'mushroom' task) along with simulated extensions. The problems highlighted by the counter-examples described above are real: the benefits of Bayesian inference can be squandered by more myopic forms of planning — such as the provably over-optimistic Thompson Sampling — which fails to account for risk in these tasks and performs poorly. The experimental results highlight the fact that the Bayes-optimal behavior adapts its exploration strategy as a function of the cost, the horizon, and the uncertainty in a non-trivial but powerful way.

6.1 Issues with Myopic Forms of Planning

Myopic forms of planning ignore the evolution of future beliefs when taking decisions. They must then rely on other mechanisms to generate optimism so as to to induce sufficient exploration. As we discussed in Section 2.2.4.3, there exist different such mechanisms to derive optimism cheaply from the posterior (certainty equivalence, explicit reward bonuses, optimism from sampling). However, not all of them can deal with the rich statistical models that we want to consider. Certainty equivalence, which takes the posterior mean model as the true model, is one such option. Its main flaw has already been largely documented (Dayan and Sejnowski, 1996; Duff, 2002; Kumar, 1985): it does not generate enough optimism — we exhibited examples of this in past chapters and provide another example in the next section. Another option, generating optimism myopically via posterior sampling, can also scale to large domains, but its flaws have not been so well documented. We therefore consider it here.

Perhaps the most popular algorithm that generates optimism from sampling is Thompson Sampling. Though heuristically myopic from the perspective of Bayes-adaptivity, TS is computationally cheap, and has been proven to perform well in various domains. Because it relies on the posterior distribution without the burden of propagating (or filtering) beliefs when planning, it fits well with complex, e.g., Bayesian non-parametric, models that in any case are handled via MCMC sampling (Doshi-Velez et al., 2010). Further, as we described in Chapter 2, the idea behind TS can be extended to combine multiple posterior samples and to stick to a particular plan for multiple time steps. Though this way of deriving optimism may sometimes be appropriate when structure is lacking, or when only studying the long-term behavior of the agent, the optimism generated by these algorithms may not always be appropriate given the actual beliefs of the agent. Given some prior and some horizon, the Bayes-optimal policy carefully chooses what is worth exploring and in what order. The precise degree of justifiable optimism is a complex function of the belief and other parameters. By contrast, blind optimism is dangerous: failing to integrate over the posterior risks over-optimism from ignoring the potentially disastrous effect of some actions; being myopic risks under-optimism from ignoring quick improvements to the state of knowledge that some actions might deliver. Although these flaws may not be clearly apparent when there is too much uncertainty, they can be quite pronounced in scenarios in which some exploration-exploitation strategies are clearly better than others according to particular beliefs. To better understand these modes of failures, we consider four simple, and yet particularly pernicious, classes of counter-example; other failure modes are illustrated in the results section below in the context of larger domains.

Example 1

Consider an MDP consisting of a linear chain of $2x+1$ states. Each interior state admits 2 deterministic actions: going left or right. The only source of reward ($r = 1$) is at either one or other end. The agent starts in the middle (state $x+1$), and knows everything except the end which delivers the reward; each of the two MDPs $\mathcal{P}$ has prior probability $P(\mathcal{P}) = \frac{1}{2}$. The episode terminates after the reward is obtained. See Figure 6.1 for an illustration. Critically, the only transition that changes this belief is at an end. At each step, TS samples one of the chains, and so heads for the end which that sample suggests is rewarding. Since this depends on an unbiased coin flip, TS is effectively performing a random walk with probability $\frac{1}{2}$ of moving in either direction, and so takes $O(x^2)$ time to reach an end (Moon, 1973). This is much worse than the linear time of the Bayes-optimal policy which commits to a given direction by tie-breaking in the first step and then maintains this direction to the end of the chain.[1]

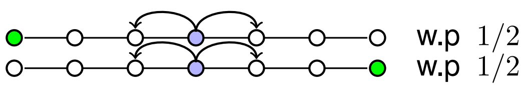


Figure 6.1: Illustration of Example 1. The two possible chains with $x = 3$, with green dots representing the reward states and the blue dot representing the start state.

[1]We note that a similar example has been independently put forward by Ortega and Braun (2010).

One might ascribe this failure to the fact that TS was developed for multi-armed bandits, which lack temporally extended structure. TS has duly been adapted to the MDP setting with the goal of controlling the expected regret. For instance, the PSRL algorithm (Osband et al., 2013), which was inspired by Bayesian DP (Strens, 2000), samples an MDP from the current posterior and executes its optimal policy for several steps (or an entire episode). This way of exploring an MDP bypasses the TS's lack of commitment in Example 1, but can still be problematic for discounted objectives, as illustrated in Example 2.

Example 2

Consider a slight modification of Example 1 where the start state is the second state on the chain, so that with probability $\frac{1}{2}$ the agent is only 1 step away from the reward source (See Figure 6.2). Clearly, the Bayes-optimal solution is to go left towards the nearest end first. The value of the policy at the start state is $V^* = \frac{1}{2}(\gamma + \gamma^{2x+1})$. On the other hand, an algorithm that commits to a particular policy based on a posterior sample will aim for the left or right end of the chain equally often (since they are equally likely). The resulting value of such a strategy is $V = \frac{1}{4}(\gamma + \gamma^{2x-2}(1+\gamma^3) + \gamma^{4x-1})$, which gives $V = \frac{1}{2}V^*$ as $x \to \infty$.

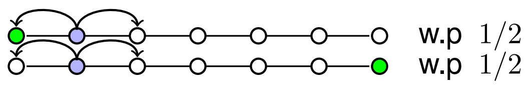


Figure 6.2: Illustration of Example 2.

The BOSS (Asmuth et al., 2009) algorithm is a more complicated construction that combines multiple posterior samples, Examples 3-4 illustrate a similar issue with the kind of optimism it generates for exploration.

Example 3

Consider a single-step decision between two actions, a_1 and a_2, with uncertainty in the payoff as follows. With probability p (case 1), action a_1 leads to reward

$c_1 < 0$ and a_2 leads to reward 0. With probability $1 - p$ (case 2), action a_1 leads to reward 1 and a_2 leads to reward 0. This is illustrated in Figure 6.3.

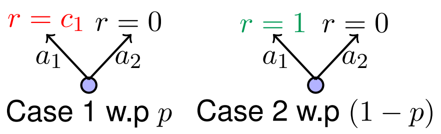


Figure 6.3: The two possible payoff structures of Example 3.

Conventional TS in this example involves sampling one of the transitions according to the prior and taking the corresponding optimal action. This results in the following expected reward: $V_{TS} = E[r] = p(p \cdot 0 + (1 - p) \cdot 0) + (1 - p)(p \cdot c_1 + (1 - p) \cdot 1) = (1 - p)(p \cdot c_1 + (1 - p))$. If c_1 is arbitrarily large and negative, V_{TS} can be made arbitrarily bad. The Bayes-optimal policy integrates over the possible outcomes, therefore it performs at least as well as always choosing action a_2 with an expected reward of 0. This implies $V^* \geq 0$. The BOSS algorithm (Asmuth et al., 2009) constructs an optimistic MDP based on K posterior samples, so that the best action across all K samples is taken. In this example, it is enough for a single sample of case 2 to be present in these K samples to decide to take action a_1 (since $r_2 > r_1$), resulting in the following value for BOSS (denoting X to be the number of samples in the set of K samples of case 2): $V_{BOSS}^{ex3} = P(X \geq 1)(p \cdot c_1 + (1 - p)) = (1 - p^K)(p \cdot c_1 + (1 - p)) := z(K)$, which is a decreasing function of K (since $c_1 < 0$), showing the cost of this added optimism.

Of course, we usually think of BOSS as being applied to MDPs with sequential decisions, but one can readily transform Example 3 in an MDP by putting these 1-step decisions one after the other. We provide details of the construction in Example 4.

Example 4

Consider linking together different instances of Example 3. The agent starts in s_0, and chooses between a_1 and a_2 with payoff described in Example 3. After *executing either action, the agent makes a transition to state s_1, where the process repeats until state s_n, which itself transits back to s_0. The outcome of a_1 and a_2 (determined by whether s_i is of case 1 or 2) is independent across states.*

BOSS has a parameter B that decides the number of steps between posterior resampling operations. However, B has no effect in this example for the first n steps. To compute the value of BOSS's policy from the initial belief state, leveraging the independence assumption, we can employ the value analysis of Example 3 for the first visit of every state. After every state gets visited once, the transition in some states (where action a_1 was chosen) will be uniquely known and BOSS can perfectly exploit the MDP in these states. For simplicity, we bound the value of the policy by assuming perfect knowledge of the MDP after the first n states: $V^{ex4}_{BOSS} < \sum_{t=0}^{n} \gamma^t z(K) + \sum_{t=n+1}^{\infty} \gamma^t (1-p) = z(K)\frac{1-\gamma^n}{1-\gamma} + \frac{\gamma^{n+1}(1-p)}{1-\gamma}$, where $z(K) = (1-p^K)(p \cdot c_1 + (1-p))$ (from Example 3). We can choose n large to make the second term arbitrarily small, whereas the first term again depends on $c_1 < 0$, which we can make arbitrarily bad. Again, the value of the Bayes-optimal policy in this example can easily be lower-bounded as $V^* \geq 0$ since the Bayes-optimal policy can at least choose action a_2 at all times (no exploration).

We stress that PSRL and BOSS do enjoy strong theoretical guarantees for different objectives (expected regret and PAC-MDP); our goal of Bayes adaptivity is a more severe objective because discounting exerts pressure to perform well within a relatively shorter time horizon.

6.2 Non-parametric Contextual Tasks

We consider three exploration-exploitation domains that are composed of an infinite sequence of related subtasks, whose associations are signalled by accompanying context vectors. In each domain, we employ a non-parametric prior to model statistical sharing between the parameters of each subtask.

First, in Section 6.2.1, we examine a realistic domain based on a popular supervised learning task where data were (likely) not generated from the prior.

This domain, despite its simple description, presents a challenging exploration-exploitation trade-off, because many actions are significantly costly. Careful evaluation of the benefits of future exploitability is necessary in the light of the discounted horizon.

Although (or perhaps because) the mushroom task is derived from real data, we cannot be sure that the examples follow the prior. This motivates a richer, artificial, domain derived from generalizing the probabilistic model. In this second domain, investigated in Section 6.2.2, we generate task instances directly from the generative model and not from data. Since we have control over the parameters of the task generation, it is more revealing about the differences between different methods of planning.

The internal dynamics in the subtasks are not sequential in the first two domains. While this can already model many real settings, we consider in Section 6.2.3 an extension of the model to a case of more general subtasks that are themselves small MDPs.

6.2.1 The Mushroom Exploration Task

The Mushroom Dataset from the UCI repository (Bache and Lichman, 2013) contains 8124 instances of gilled mushrooms from 23 different species in the Agaricus and Lepiota family, each of which is described by discrete attributes (e.g., color, odor, ring type) and whether the mushroom is poisonous or edible (51.8% of all instances are edible). We build an MDP based on the data as follows: at each point in time the agent is faced with the attributes of a random mushroom from the dataset, and has to choose whether to eat or ignore it. Ignoring a mushroom has no consequence; eating an edible mushroom is rewarding; but eating a poisonous mushroom incurs a large cost. This is illustrated in Figure 6.4. The agent may be provided with some initial 'free' observations of the attributes and edibility of a set of mushrooms.

This problem is conventionally thought of in terms of supervised learning. However, since the agent is allowed to ignore a mushroom, it is actually more akin to

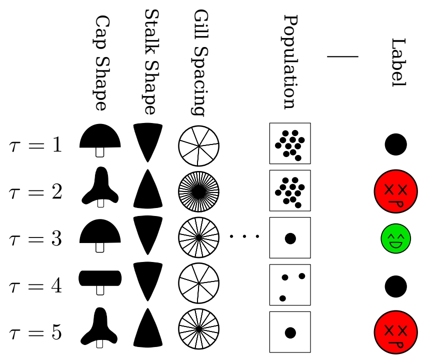


Figure 6.4: Illustration of the mushroom exploration domain, the rows represent the sequence of mushrooms, with the value of different attributes being displayed on each column. The agent may eat the mushroom to obtain a reward/cost (green/red circle) or choose to ignore it (black circle).

a contextual bandit task (Langford and Zhang, 2007). As briefly mentioned before, a contextual bandit problem is an extension of a multi-armed bandit problem where a context vector is available at each round and which relates to the payoffs for each arm. However in our case, unlike most past work on contextual bandits, early rewards are more valuable than later ones, characterized by a discount factor γ. This is a critical difference from exploration objectives based on regret that are dominated by the long-term behavior of the agent, and so fail adequately to reward fast learning. In addition, unlike our approach, existing work on contextual bandit rarely exploits the unsupervised learning that the context affords even when no label is obtained.

More formally, the mushroom MDP consists of a sequence of mushroom tasks parametrized by $\boldsymbol{x}_\tau$ where $\tau = 1, 2, \ldots$. Each parameter vector $\boldsymbol{x}_\tau$ contains $C = 22$ scalar parameters $x_\tau^1, \ldots, x_\tau^C$ to generate context (the mushroom attributes), and a single scalar parameter x_τ^{C+1} to generate the subtask dynamic (the outcome of eating the mushroom). Denoting $n = C + 1$, we have $\boldsymbol{x}_\tau = (x_\tau^1, x_\tau^2, \ldots, x_\tau^n)$. The MDP dynamics $\mathcal{P}$ can be described as follows. Let S be the set of states, each of the form $s = (\tau, x_\tau^1, \ldots, x_\tau^C, o_\tau)$, where $o_\tau = \square$ (meaning unobserved) if the mushroom was not eaten in task τ and $o_\tau = x_\tau^{C+1}$ otherwise. Choosing the exit action a_{exit} increments the first state component and updates the context and observation components. Choosing the eat action a_{eat} updates $o_\tau = x_\tau^{C+1}$ and delivers a reward or cost. If the mushroom is edible, the reward is $r = 5$. If the mushroom is poisonous, the reward is $r = -15$.

6.2.1.1 A Simple Statistical Model

The key aspect of the mushroom task is the joint statistics over the characteristics and danger of the mushrooms. The truth of the matter for the UCI data is actually unclear; it is therefore a stringent test of a planning algorithm whether it is possible to perform at all well based on what can only be a vague, and likely inaccurate model. Our contribution here is to assume a general non-parametric model that allows for substantial underlying complexity in the true model, but adapts its ongoing characterization as a function of the evidence in the data that has so far been observed. We employ one particularly popular non-parametric model called the Chinese Restaurant Process (introduced in Section 2.3.2.2), which postulates that the mushrooms come from a possibly infinite number of mixture components.

The generative model of the mushroom statistics is formally described as follows:

$$\begin{aligned}
\alpha &\sim \text{Gamma}(a, b), \\
\{z_\tau\}_{\tau=1}^{\infty} &\sim \text{CRP}(\alpha) \\
\boldsymbol{\theta}_k^i &\sim \text{Dirichlet}(\frac{\beta}{D_i}),\ \forall i \in \{1, \dots, n\}, \forall k \in \mathbb{Z}^+, \\
x_\tau^i &\sim \text{Categorical}(\boldsymbol{\theta}_{z_\tau}^i)\ \forall i \in \{1, \dots, n\}, \forall \tau \in \mathbb{Z}^+,
\end{aligned}$$

where α is the concentration parameter of the CRP, the z random variables are the cluster assignments. The base measure of this Dirichlet process is assumed to be a symmetric Dirichlet prior with hyperparameter $\beta = 1$ (D_i is the dimension of $\boldsymbol{\theta}^i$, the number of possible observations for x^i), which together with the conjugate observation model, allows for relatively straightforward inference schemes (see the appendix of this chapter for details). The collection $\{\boldsymbol{\theta}_k^i | i \in \{1, \dots, n\}\}$ of vectors contains the parameters corresponding to each mixture component k. The task parameters $\boldsymbol{x}_\tau$ for a particular τ are drawn by first choosing a mixture component z_τ according to the CRP and then using the corresponding $\boldsymbol{\theta}_{z_\tau}$

parameters to sample each component of $\boldsymbol{x}_\tau$. The infinite-state, infinite-horizon MDP is derived from this generative process by sampling an infinite sequence of tasks ($\tau \to \infty$) and patching them together.

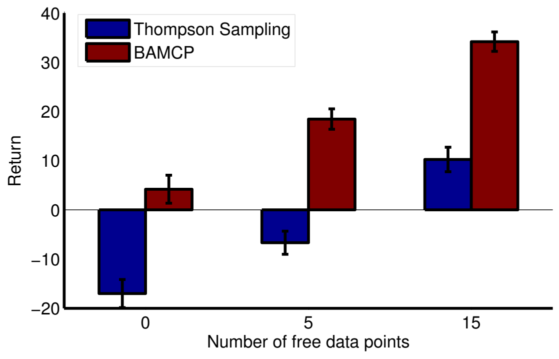
The data at time t, h_t, consist of all mushroom attributes and labels observed, including the current mushroom subtask (and any initial 'free' examples). The posterior distribution over the dynamics $P(\mathcal{P}|h_t)$ is then obtained straightforwardly from the posterior over all past and future $\boldsymbol{x}_\tau$ (denoted $\boldsymbol{x}_{1:\infty}$), $P(x_{1:\infty}|h_t)$, since $\mathcal{P}$ is uniquely characterized by $\boldsymbol{x}_{1:\infty}$.

For the mushroom data, we set $D_i = 12$ for each context dimension i — the maximum number of values for any of the 22 attribute dimensions in the data. This implies $2{\cdot}12^{22} \geq 10^{24}$ possible configurations of mushrooms assumed by the model. Since α is not known, we set a generic hyperprior on $\alpha \sim$ Gamma$(.5, .5)$.

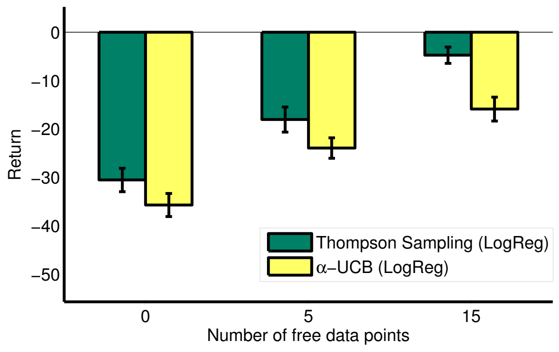
6.2.1.2 Results in the Mushroom Task

We stress that the mushroom data were not really generated by the process assumed in the previous section – this is what makes the task interesting as a test case. Indeed, when the agent lacks prior data, maximizing the return is highly challenging. Randomly eating mushrooms to sample the dataset is a particularly bad strategy because of the cost asymmetry between edible and poisonous mushrooms. A natural point of comparison is the policy of ignoring all mushrooms, which leads to a neutral return of 0.

We ran the Bayes-adaptive agent (BAMCP) and TS using this statistical model on the mushroom task. Since the concentration parameter is unknown, it is inferred from data, *both influencing, and being influenced by, the exploration*. Results are reported in Figure 6.5a for three different numbers of 'free' examples. A surprising result is that the Bayes-adaptive agent manages to obtain a positive return when starting with no data, despite the mismatch between true data and generative model. This demonstrates that abstract prior information about structure can guide exploration successfully. Given exactly the same statistical model, TS fails to match this performance. We speculate that this is due to over-



(a)



(b)

Figure 6.5: Exploration-Exploitation results on the mushroom dataset, after 150 steps with $\gamma = 0.97$. (a) Discounted return for BAMCP and TS with the CRP model of Section 6.2.1, including hyperparameter inference. Either starting from scratch from the prior (0 'free' data points), or from the prior plus an initial random (labeled) data set of size 5 or 15. At most 75+$\{0, 5, 15\}$ datapoints can be observed during these 150 steps. (b) Discounted return for TS and $\alpha-$UCB (with $\alpha = 1$ and upper confidence approximation U0 as defined by Li et al. (2012)) when using the Bayesian Logistic Regression model, and the same task setting as (a). Averaged over 50 runs.

optimism — we provide some evidence for this in Figure 6.6 and investigate this aspect further in Section 6.2.2. When initial data (incl. labels) is provided for free to reduce the prior uncertainty, TS can improve its performance by a large margin but its return remains inferior to a Bayes-adaptive agent in the same conditions.[2]

For the purposes of comparison, we also considered a simpler discriminative statistical model, namely Bayesian Logistic Regression, which Li et al. (2012) suggested for use in contextual bandits. In this model, the reward (between 0

[2]We also tested the PSRL version of TS (Osband et al., 2013), which commits to a policy for $\frac{1}{1-\gamma}$ steps. Performance was worse than for regular TS, an expected outcome, since PSRL takes more time to integrate and react to new observations.

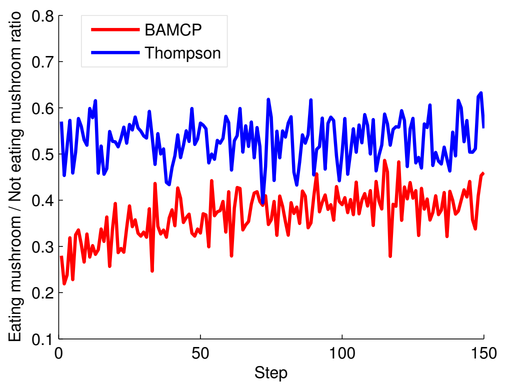


Figure 6.6: Average rate of exploitation of mushrooms over time for BAMCP and TS for the domain described in Section 6.2.1, when starting with no labelled observations (0 free data condition). For a given step, the reported value is the fraction of time the agent chose to eat the current mushroom (when it had the option to), rather than to skip it. Throughout the duration of a run, and especially early on, TS is more willing to try mushrooms than BAMCP.

and 1 before rescaling) for an action a in context $\mathbf{x}$, denoted $r(\mathbf{x}, \mathbf{w}_a)$, is assumed to be $(1+\exp(-\mathbf{x}^T\mathbf{w}_a))^{-1}$, with $\mathbf{w}_a \sim \mathcal{N}(0, I)$ the parameter vector for each arm. TS and $\alpha-$UCB (Li et al., 2012) are two ways to derive actions from the output of the logistic regression model. TS samples the $\mathbf{w}_a$ vectors from the posterior and takes the greedy action. Algorithm $\alpha-$UCB, following the idea of UCB for multi-armed bandits, selects the action that maximizes $E[r(\mathbf{x}, \mathbf{w}_a)] + \alpha\sqrt{\text{Var}[r(\mathbf{x}, \mathbf{w}_a)]}$ according to the posterior distribution on the weight vectors. Figure 6.5 shows the results of applying TS and $\alpha-$UCB with the logistic regression model in the context of our exploration task. TS does worse with the logistic regression model than with the CRP-based model; this demonstrates the added benefits of a prior that captures many aspects of the data with only a few datapoints. The $\alpha-$UCB algorithm, despite good performance in the long-run on large datasets, is too optimistic to perform well with discounted objectives.

6.2.2 Non-Parametric Contextual Bandit Sequence Model

The mushroom task can be seen as a sequence of subtasks that share structure, but whose order the agent cannot control. Other such domains are adaptive medical treatments where each patient can be understood as the subtask, handling customer interactions, or making decisions to drill for oil at different geological locations. In this section, we consider a generalized version of domains with this characteristic form of shared structure. Further, by addressing environments that were actually drawn from the model, we study planning in the absence of model mis-match.

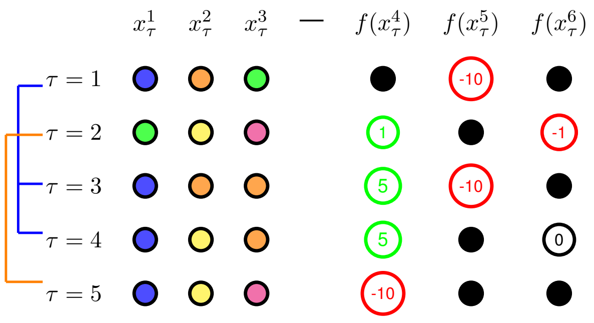


Figure 6.7: Example of a sample sequence of tasks from Section 6.2.2 with $C = 3$, $A = 3$. The left columns indicate the context variables (each color represents a possible value), and the right columns indicate the rewards that have been observed in each task (black when the arm was not pulled). On the left, the true clustering of the tasks (shared mixture component) is displayed. The dictionary of context variables is composed of $D = 5$ different colors in this case.

6.2.2.1 Model

The key generalization is to allow multiple arms in each subtask (rather than a single eat/exit decision). Using the same notation as Section 6.2.1, each parameter vector $\boldsymbol{x}_\tau$ now contains C scalar parameters $x^1_\tau, \dots, x^C_\tau$ to generate context, and Y scalar parameters $x^{C+1}_\tau, \dots, x^{C+Y}_\tau$ to generate the actual task dynamics (i.e., denoting $n = C + Y$, we have $\boldsymbol{x}_\tau = (x^1_\tau, x^2_\tau, \dots, x^n_\tau)$). The generative model is identical otherwise, but now the choices of the agent in any particular task τ are to either: 1) leave the subtask for the next; or 2) pull any of the Y arms that has not been previously pulled. The MDP states are now of the form $s = (\tau, x^1_\tau, \dots, x^C_\tau, o^1_\tau, \dots, o^Y_\tau)$, where $o^a_\tau = \square$ if arm a has not been pulled in task τ and $o^a_\tau = x^{C+a}_\tau$ otherwise. Figure 6.7 shows the first part of a draw of the generative process including a hypothetical agent trajectory.

The exact setting for the experiments is as portrayed in Figure 6.7: with $C = 3$

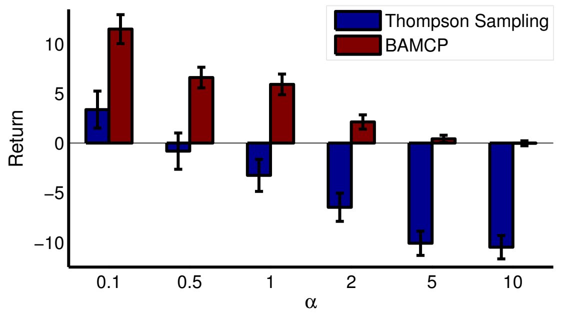
context cues, $A = 3$ arms in each task, and $D = 5$ possible values of x (i.e., the dimension $D_i = 5$ for each $\boldsymbol{\theta}$). The function f (that maps values of x^a to rewards) is 1-1 with the domain: $\{5, 2, 0, -1, -10\}$. We drew MDPs with different values of the concentration parameter $\alpha \in \{0.1, 0.5, 1, 2, 5, 10\}$. The agent was assumed to know the generative structure of the MDP; but we considered both cases when it knew the true value of α or just had a generic hyperprior on $\alpha \sim$ Gamma$(.5, .5)$, and had to learn.

This can be seen again as a contextual bandit task with shared structure modeled by a CRP. The difference in this model is that we give the option of playing multiple arms for the same context (subtask). Many extensions are possible, including more complex intra-task dynamics (we explore this avenue in Section 6.2.3) and more general forms of shared structure; however we focus here on planning rather than modeling.

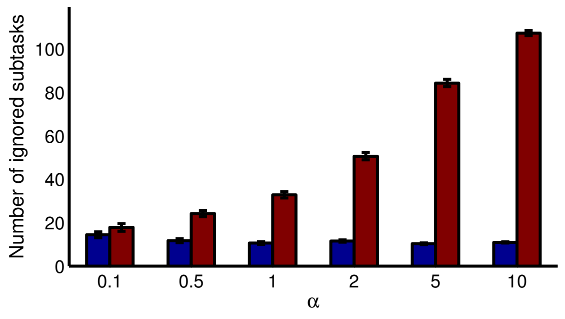
6.2.2.2 Results on synthetic data

We investigate the behaviour and performance of Bayesian agents acting in tasks sampled from the non-parametric model above. The reward mapping implies that $E[f(x^a)] = -0.8 < 0$ for all arms a and for all τ, since all values of x^a are equally likely *a priori*. Thus, again, the strategy π^{exit} of always exiting subtasks (without pulling any arms) is a fair comparison, with value 0 – a myopic planner based on the posterior mean only should never explore an arm, gaining this value of 0. Any useful adaptive strategy should be able to obtain a mean return of at least 0.

We concentrate on two metrics computed during the first 120 steps of the agent in the environment: the discounted sum of rewards (the formal target for optimization; Figure 6.8a), and the number of times the agent decides to skip a subtask before trying any of its arms (Figure 6.8b). The second metric relates to the safe exploration aspect of this task; sometimes optimism is unwarranted because it is more likely to lead to negative outcomes, even when taking into account the long-term consequences of the potential information gain.



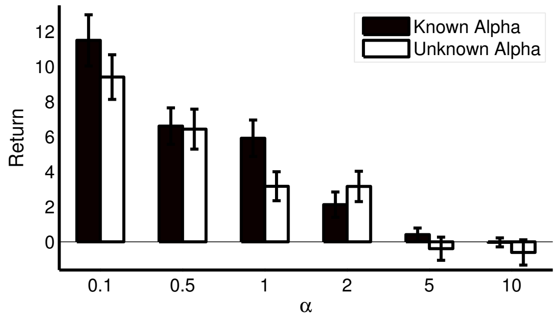
(a)



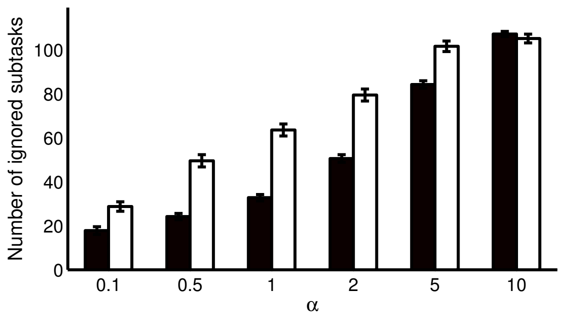
(b)

Figure 6.8: The performance of BAMCP and Thompson sampling on the non-parametric bandit task ($\gamma = 0.96$) in terms of discounted return (a). The concentration parameter α is known to the algorithm and varies on the x-axis. In (b), the average number of subtasks ignored by each algorithm in the different environmental conditions. Each plotted value is averaged over 200 runs of 120 steps (with 60K forward simulations per step).

We ran BAMCP and TS on the task. Figure 6.8 shows the performance as a function of α, when this concentration parameter is known. When the concentration parameter α is small, there will only be a few different mixture components, making for an easy case with little uncertainty about the identity of the mixture components after a few observations, and therefore little uncertainty about the outcome of an arm pull. In the limit of $\alpha \to 0$, only one cluster will exist and the domain essentially degenerates to a form of multinomial multi-armed bandit problem. As α grows, the identity of a given task's cluster becomes more uncertain and aliasing grows, so safe exploration becomes more challenging. Learning is slower in that regime too, simply because there are more parameter values to acquire. As $\alpha \to \infty$, every cluster will be different; this would prevent any kind of generalization and the Bayes-optimal policy will be to skip every subtask τ (since the *a priori* expected values of the arms in any given subtask is negative).



(a)



(b)

Figure 6.9: Performance of BAMCP for various values of the concentration parameter α, with and without hyperparameter inference. The learned Bayes-adaptive policy avoids more subtasks (b) but manages to maintain a similar level of performance (a) despite the uncertainty over α. Averaged over 200 runs with 60K simulations for each time step. The discount parameter is $\gamma = 0.96$.

Figure 6.8 shows that BAMCP adapts its exploration-exploitation strategy according to the structure in the environment; small values of α justify the risk of exploring and incurring costs but this optimism progressively disappears as α gets larger. This translates into positive return when generalization is feasible, despite the marginal negative expected cost for each arm, and a return close to 0 when costs cannot be avoided. On the other hand, TS suffers from over-optimism across the board, leading to poor discounted returns, especially when the number of mixture components is large. Intuition for TS's poor performance comes from considering an extreme case in which all or most subtasks are sampled from a different cluster. Here, past experience provides little information about the value of the arms for the current cluster; thus, discovering these val-

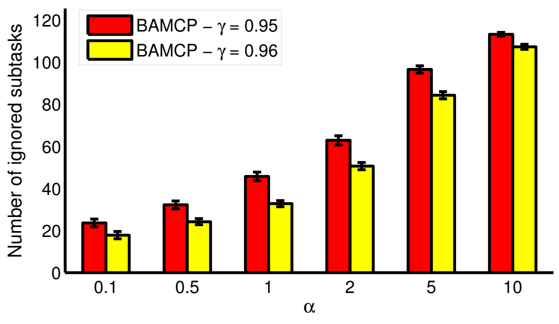


Figure 6.10: In the domain described in Section 6.2.2.2, the average number of ignored subtasks over 120 steps when using BAMCP with either $\gamma = 0.95$ or $\gamma = 0.96$ (Known α scenario). We see that a larger γ induces more exploration and risk-taking (fewer subtasks are ignored), showing the sensitivity of Bayes-Adaptive planning to the horizon.

ues (which, on average, is expensive) is not likely to help in the future. However, TS samples a single configuration of the arm, mostly informed by the prior in this situation, which likely results in at least *one of the arms* as having a positive outcome (for the prior, we repeat 3 times a draw having $\frac{2}{5}$ probability of success, so $p = 0.784$). TS then, incorrectly, picks this putatively positive arm rather than exiting. Other myopic sample-based exploration strategies, such as Bayesian DP (Strens, 2000) or BOSS (Asmuth et al., 2009), would suffer from similar forms of unwarranted optimism — since they also rely on one or more posterior samples according to which they act greedily (see Examples 3-4 in Section 6.1).

A yet more challenging scenario arises when α is not known to the agent. A Bayesian agent starts with a uninformative (hyper-)prior on α in order to infer the value from data, it also takes into account during planning how its belief about this hyperparameter changes over time. In Figure 6.9, we observe that the Bayes-adaptive agent explores more conservatively when α is unknown. As expected, this results in lower returns (compared to when α is known). However, robustness to increased uncertainty is shown by the modest difference.

In Figure 6.10, we also show that BAMCP is sensitive to the discount factor, highlighting the dependence of the exploration-exploitation strategy to the horizon.

6.2.3 CRP mixture of MDPs

To illustrate that our methodology is not restricted to tasks resembling contextual bandits, we consider an extension of the problem in Section 6.2.2 in which each subtask is a random MDP of a particular class, with contextual information being informative about the class. This extension is loosely motivated by the following oil exploration problem. Possible drilling sites are considered in a sequence; each site comes with some particular known contextual information (e.g., geological features). One can ignore a drilling site, or (buy acreage and) run one of two types of terrain preparation strategies. The outcomes of these two strategies is the potential for either natural gas or crude oil. Finally, one has to make a final choice about how to exploit (e.g., the type of extraction process), which results in a stochastic payoff for this drilling site: either dry hole with many expenses or a profitable exploitation.

6.2.3.1 Model

We model this task as in Section 6.2.2, with each drilling site corresponding to a cluster/subtask τ. To establish the transitions between states, additional x_τ variables are necessary, in addition to the variables already modeling context and rewards.

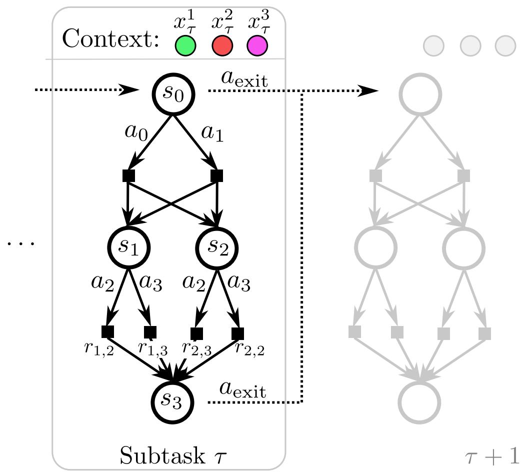
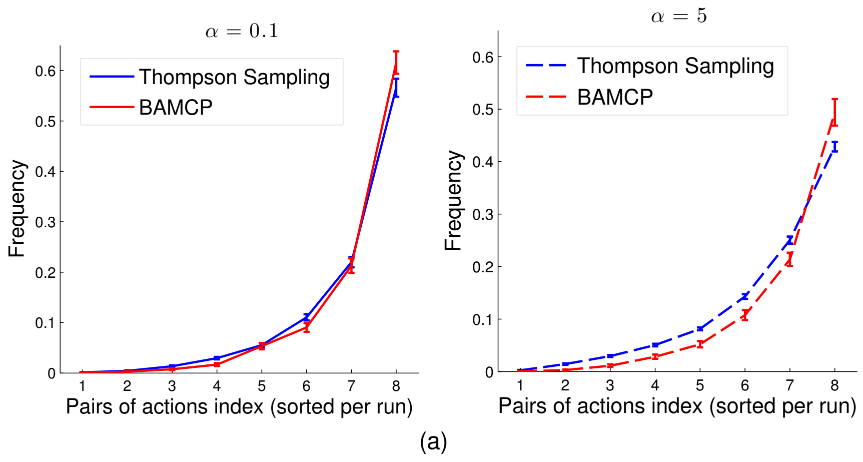


Figure 6.11: The extension of the CRP mixture model to MDPs, modelling an intermediate decision within each subtask before getting a payoff.



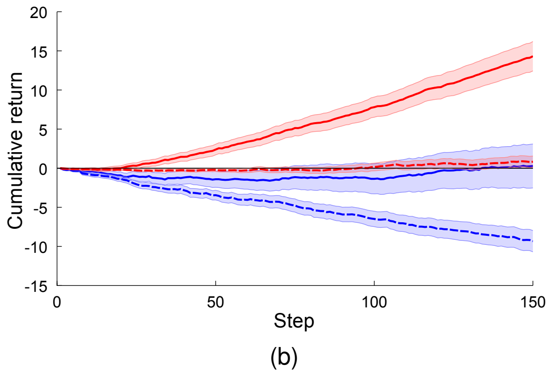


Figure 6.12: (a) Intratask exploration-exploitation statistics for BAMCP and TS in the drilling problem for $\alpha = 0.1$ (left) and $\alpha = 5$ (right). The distribution of action pairs (sorted in each run) executed *inside* the MDP subtasks that were explored by the agent. Mean over 100 runs. (b) Comparison of the cumulative return as a function of the time step for BAMCP (red) and TS (blue). Solid lines: $\alpha = 0.1$, dotted line: $\alpha = 5$.

From the starting state of a subtask, s_0, there are three possible actions: to ignore a drilling site (a_{exit}), or the two terrain preparation strategies (a_0 and a_1). After taking a_0 or a_1, the agent moves to either s_1 or s_2 based on the value of these additional x_τ variables. From these states, the choice of extraction strategy corresponds to actions a_2 and a_3.

The payoff $r_{1,3}$ from executing action a_3 in s_1 in subtask τ is determined by the binary variable $x_\tau^{1,3}$ as $r_{1,3} = f(x_\tau^{1,3})$, where $f(0) = -1.5$ (dry hole) and $f(1) = 1$ (profitable exploitation). In the generative model, $x_\tau^{1,3}$ (and other $x_\tau^{i,j}$) is determined like the other x_τ variables in Section 6.2.2. The model is illustrated in Figure 6.11.

6.2.3.2 Results

The performance of BAMCP and TS on simulated data closely resemble that in the contextual bandits of Section 6.2.2. BAMCP adaptively ignores, explores, or exploits drilling sites depending on the environmental statistics. As in the other examples, with the same statistical model, TS acts too optimistically to do well in terms of discounted return and is less inclined to ignore subtasks. Figure 6.12-a shows that BAMCP is also more conservative when acting within each MDP when α is large, compared to TS. The dynamics of the cumulative return as a function of the steps is presented in Figure 6.12-b.

6.3 Conclusion

We provided empirical evidence that agents can greatly benefit from non-myopic Bayes-adaptive planning in tasks with sufficient structured prior knowledge — even if that prior only approximately captures the true dynamics of the environment, as in the mushroom-exploration task. While this conclusion has previously been reached on intuitive grounds, an explicit demonstration of Bayes-adaptive planning using rich models had not previously been possible because of the algorithmic obstacles that our sample-based planning methods are designed to vault. Here, we showed these obstacles can be overcome using the sample-based methods developed in this thesis. Furthermore, our results indicate that the cheaper, myopic, planning methods relied on in past work cannot be as effective at selecting appropriate exploration-exploitation strategies in the Bayesian setting.

Appendix: Inference Details for the CRP-based Models

For all the CRP-based sequence models, we use a (Rao-Blackwellized) Gibbs sampler, as described, for example, in Sudderth (2006), to sample the assignment of subtasks to clusters (z), the cluster parameters (θ). We use an auxiliary variable trick from Escobar and West (1995) for tractable inference of the concentration parameter α. A couple of Gibbs sweeps are performed between every BAMCP simulation. This thinning is not necessary for convergence but it helps when running BAMCP with a finite number of simulations. For Thompson Sampling, we burn in the chain with 500 Gibbs sweep before selecting the sample used for a particular step; the performance of Thompson Sampling was not affected by the length of the burn in period past some minimum threshold.

Given a setting of the cluster assignments and cluster parameters for the observed subtasks (obtained from the Gibbs sampler), the future subtasks $x_{t:\infty}$ are sampled by running the generative model forward conditioned on the inferred variables. To improve the planning speed, posterior samples can be memoized at the root of the search tree. The simulations pick uniformly from a fixed-size pool of previously generated samples, a pool that gets slowly regenerated with new posterior samples.

VII

Discussion

Our thesis is that sample-based, online, reinforcement-learning methods can be employed to realize the benefits of Bayes-adaptive planning in large domains with complex priors. We first summarize our main contributions, and then discuss the wider context of our work, some relevant open questions, and important avenues for future investigation.

7.1 The BAMCP Algorithm

In Chapter 3, we suggested a sample-based algorithm for Bayesian RL called BAMCP. The main idea is to employ Monte-Carlo tree search to explore the augmented Bayes-adaptive search space efficiently. The naive implementation of that idea is an algorithm that we called BA-UCT. However, BA-UCT cannot cope with most priors because it employs expensive belief updates inside the search tree. We therefore introduced three modifications to obtain a computationally tractable sample-based algorithm: root sampling, which only requires beliefs to be sampled at the start of each simulation (based on the work of Silver and Veness (2010)); a model-free RL algorithm that learns a rollout policy; and a lazy sampling scheme that enables the posterior beliefs to be sampled cheaply. Furthermore, we proved that BAMCP converges to the Bayes-optimal solution, even when MCMC-based posterior sampling is employed.

In Chapter 4, we showed that BAMCP significantly outperformed many existing Bayesian RL algorithms, as well non-Bayesian approaches, on several well-known benchmark problems. While it is important to run comparisons on these standard domains, the domains are quite small and are associated with unstructured priors. Therefore, they are not representative of the sort of domains for which we expect Bayes-adaptive planning to be specially advantageous. BAMCP is particularly well suited to support planning in large domains in which richly structured prior knowledge makes lazy sampling both possible and effective. This offers the prospect of applying Bayesian RL at a realistically complex scale. We illustrated this possibility by showing that BAMCP can tackle a domain with an infinite number of states and a structured prior over the dynamics, a challenging, if not radically intractable, task for existing approaches. This example highlights BAMCP's ability to use Markov chain Monte Carlo methods for inference associated with the posterior distribution over models.

7.2 Bayes-Adaptive Planning with Function Approximation

In Chapter 5, we addressed generalization in Bayes-Adaptive planning, proposing a generic simulation-based search method that can operate with reduced representations of the history of the interaction between agent and environment. Our method, BAFA, subsumes various existing tree-based approaches such as BAMCP. It allows the values accorded to nearby belief-states to be shared – each simulation is no longer an isolated path in an exponentially growing tree, but can impact many non-visited beliefs and states. We proposed a parametric form for the action-value function that relies on a generic sampled representation for the history based on a Monte-Carlo approximation of the belief. To reduce the computational complexity of each simulation, we exploited root sampling, showing that it is valid for a large class of planning methods that includes BAFA and BAMCP.

7.3 Bayes-Adaptive Planning with Rich Statistical models

We argued in Chapter 4 that the prime motivation for paying the price of Bayes adaptivity is its power in structured domains. In Chapter 6, we considered some such tasks that are richer than the product model of Chapter 4. To create a more realistic problem, we derived a contextual bandit problem from the UCI mushroom dataset, through the medium of a Bayesian non-parametric model. In this challenging exploration-exploitation domain, we demonstrated the feasibility and advantages of using a Bayes-Adaptive, or fully Bayesian, agent. Furthermore, we showed that alternative, over-optimistic, myopic planning methods such as Thompson Sampling can run into severe problems that BAMCP avoids through explicit lookahead computations.

7.4 Discussion and Future Work

We consider four overlapping facets of the thesis. The first topic addresses the problem definition itself, namely by questioning the objective employed in our work on BA control (Section 7.4.1). Second, we discuss the impact and nature of the priors that play a large role in the behavior of Bayes-adaptive agents (Section 7.4.2). Turning more directly to the focus of this thesis, we discuss the algorithmic issues of online BA planning and how our contributions could be extended to deal with larger problems while minimizing computation (Section 7.4.3). Finally, in a section on modeling animal learning, we provide a brief perspective on the existence and implementation of Bayesian adaptive control mechanisms in biological systems (Section 7.4.4).

7.4.1 About the Objective

Bayesian adaptive control has been studied typically in the context of a geometrically-discounted infinite-horizon objective — or an undiscounted objective in the finite-horizon setting. Discounting makes future rewards worth less and has a huge impact on behavior. It is pervasive in RL, and more generally in control, and can be justified in different ways (Schwartz, 1993).

Normatively, discounting can reflect some general uncertainty about being able to enjoy rewards in the future. For the geometrically discounted case, it corresponds to a constant probability of death, γ, at every step. While convenient, the agent may have some other ways of valuing future rewards that would induce different kinds of discounting, for example a variable discount depending on the state. Even if largely unexplored, these other forms of discounting could be incorporated in the Bayes-Adaptive framework if the application domain justifies it.

A more pragmatic reason to rely on discounting is to avoid dealing with an undiscounted sum of rewards that can be infinite. An alternative is to define objectives in terms of average rewards. There are multiple notions of optimality for the average reward setting. We mentioned gain and bias optimality in Chapter 2, but more stringent objectives exist. One of particular importance is called Blackwell optimality; a notion which encompasses bias-optimality but that also relates to the limiting case of the discounted objective — this can be seen as another justification for discounting. Even in the fully-observable case, average reward objectives are much less studied than discounted objectives. In partially-observed settings, the average setting is barely mentioned. It is an open question as to whether these objectives are always well-defined when acting under uncertainty in the infinite or non-ergodic domains derived from belief-space dynamics. More importantly, even if the objective is well-defined, algorithms to optimize a Bayes-optimal policy in the average reward setting have not been developed yet.

In undiscounted problems, an alternative to Bayesian average reward objectives is to consider minimizing the regret (Section 2.1.2.1). The regret formulation also accounts for the performance during the learning period. Nevertheless, it is typically more forgiving of early transgressions. A Bayesian agent can be more conservative or aggressive in its exploration in a way justified by its prior in order to maximize every last bit of expected reward. Algorithms that optimize regret need to be conservative in their learning since missing an opportunity for exploitation in *any* environment will have dramatic consequences on its regret rate bounds. These algorithms are therefore more willing to accept the short term costs induced by over-exploration. By focusing on a learning mechanism to reduce the regret rate in all cases, rather than on optimizing the sequence of steps from any given state, learning can be done with much less computation.

More generally, we can ask whether these maximization objectives for adaptive control are sufficient to elicit complex behavior such as curiosity and skill-learning (Schmidhuber, 1991). We believe many of these behaviors can emerge from simple reward maximization principles if the agent has appropriately structured priors and if there is enough justification for generalization (Dayan, 2013). However, this has not been tested thoroughly. One reason is that it stretches the capacity of existing planning algorithms because such behavior requires a long, coherent, exploration plan. We can envision that complex exploratory behaviors, if they recur across environments, may be compressed into pre-compiled policies, forming a learning curriculum, that are much less computationally demanding to execute but almost as effective, albeit at the cost of flexibility.

A related open problem is to understand which classes of domains will truly benefit from the computations of Bayes-Adaptive planning (such as the ones explored in Chapter 6), and which will be adequately served with a simpler exploration-exploitation approach. Indeed, one can come up with examples for which additional computation barely matters, in that the gains are vanishingly small (e.g., the Gittins indices for a multi-armed bandit problem with $\gamma \approx 1$ are hard to obtain, but many myopic policies would do well in that scenario, see for example Ortega and Braun (2010)).

7.4.2 About Priors

In past work, the priors employed in combination with Bayesian stochastic control have frequently been selected for computational convenience rather than modeling accuracy. In supervised learning, the data distribution is usually fixed, so the prior then principally informs inference in low-data regimes. By contrast, the closed-loop interaction in RL causes the data distribution to be shaped according to the agent's original beliefs. This implies that a prior can have a long-standing effect on learning. Priors that are misaligned with the true distribution of the environment can be deleterious for a control setting and introduce maladaptive behavior — we showed examples of this in the infinite grid task in Chapter 4. For this reason, it is essential to look beyond computational convenience and incorporate in the prior the environment structure that we want to model. In recent years, there has been a slow appearance of richer prior distributions in model-based RL that actually reflect credible beliefs about the structure of the environment.

While the Bayesian toolbox is constantly growing with more sophisticated priors and better inference tools, it still remains challenging to build statistical models for large, complex, domains. Of the various options, non-parametric priors seem particularly well-suited to capture the complexity of the environment at a level supported by the data. Combining these priors in a hierarchical manner has the potential to deliver flexible and structured models for RL. For scalability, it may help to combine them with energy-models for unsupervised learning to perform feature extraction from low-level representations (Salakhutdinov et al., 2013). Another flexible tool is probabilistic programming, a general framework for representing complex generative models in a natural way (Milch et al., 2007). This leads to easier modeling, especially when causal prior knowledge is available. Though efficient inference schemes are still lacking, we see it as a promising tool to be combined with Bayesian model-based RL. To reduce the reliance on human intervention in designing agents, other promising approaches for large-scale modeling may take advantage of architectures that can learn a generative model from past experience (Dayan et al., 1995).

Priors license different sorts of generalization. When modeling domains, although some of that generalization will emerge from domain-specific features, it may be essential to consider more abstract properties of environments that will enable generalization across many domains. An example of such property is the notion of controllability, which describes whether an agent can reliably achieve certain outcomes in the environment (Huys and Dayan, 2009). Without specifying precise beliefs on the nature of low-level dynamics, priors about such abstract properties, combined with some learning, can exert substantial influence on the behavior of the agent and improve its fitness.

In this thesis, we focused on priors about dynamics — both transition and reward dynamics, since it is the dynamics that are uncertain. However, agents are ultimately interested only in optimal policies. One could derive the prior over the optimal policy implicitly implied by the prior over the dynamics. However, it might be more direct to put a prior on the optimal policy itself (Doshi-Velez et al., 2010). Agents might even have priors on both optimal policies *and* dynamics, and whereas certain combinations may result in incoherent beliefs (priors on the environment implicitly imply priors on policies through a complex mapping), it would be interesting to consider using the beliefs on policies to refine the search process during online planning.

Finally, in the same way that the true reward function is not necessarily the reward function that can best be employed by a learning system with bounded resources (Sorg et al., 2010), we speculate that there might be ways to alter the prior a constrained agent employs for planning to improve behavior. In particular, it may not be optimal to rely on the true belief to optimize the objective under that same belief. For example, an architecture that relies on Monte-Carlo methods may need to overestimate rare but significant events under the prior in order to account for them in the planning.

7.4.3 About Online Planning

One may be dissatisfied with the avalanche of approximations that results from the proposed approaches for adaptive control based on planning with beliefs. While we are optimistic that some special cases will be solved in more satisfying ways (e.g., through reductions, just like the Gittins indices for bandits), the general problem has long resisted complete solutions. If the problem is truly intractable, lacking general shortcuts, we face a classic dilemma between approximating the problem (e.g., by changing the objective) or the solution. The work in this thesis has tried to argue that directly approximating the solution to this intractable problem can be worth the effort, and at the same time can be done in principled ways. In this section, we continue this approach and discuss three ways in which Bayes-Adaptive planning could be further scaled up to larger domains.

7.4.3.1 Temporal Abstractions

The discount factor implies that there is an effective horizon for forward-planning. Since search grows exponentially with the horizon, high discount factors can be computationally prohibitive for planning. In fully-observable domains, a powerful idea is to rely on temporal abstractions (Sutton et al., 1999) to select time-scales at which planning is easier to perform. Instead of selecting primitive actions during search, one can employ macro-actions — abstractions for policy fragments that may take several time-steps to execute. By taking bigger jumps with each decision, planning can be done over longer time-scales with less effort.

Investigation of these ideas has begun for the case of general POMDPs (Lim et al., 2011), but has yet been applied to the special case of BA planning. Although attractive, there are some obstacles to planning under model uncertainty with such temporal abstractions. First, the macro-actions need to be defined. Even in fully-observable domains, there does not exist any generic way of deriving macro-actions that can help with planning. Typically, these macro-actions are hand-tuned based on the known solution to a domain. In a BA setting, it is even less clear which macro-actions should be employed, since the value of a macro-action for planning is likely to depend on the agent's belief. It might be that the structure in the prior distribution could be leveraged to generate good temporally extended actions. Second, although planning with macro-actions reduces the number of action nodes during forward search, the branching to (augmented) state nodes is large because, done naively, the entire sub-history seen during the macro-action matters in taking the next decision — since nodes should be labeled by their history or belief. To minimize these branching issues, it would be useful to consider function approximation, as we did with BAFA in Chapter 5, to generalize the value between different augmented states.

7.4.3.2 Meta-Control

In the framework we studied, the agent's own decision-making processes are assumed to be cost-free (and the environment remains in stasis until the decision is registered). In real environments, planning consumes internal resources. Moreover, time spent planning is time that is not spent on exploring and exploiting external resources. In some situations, additional planning time may translate into significantly improved behavior. But the same amount of planning time and resource may not always improve the performance of the agent at all. To take into account these planning costs and trade-offs dynamically, it is useful to consider internal states, actions and costs (Hay et al., 2012). The internal state gathers all the internal variables that are maintained as part of the planning process. This involves quantities like the estimated value for future states, the time spent on planning, or various visitation statistics. Internal actions have no (direct) effect on the external world, but they make explicit all the internal decisions based on internal states that constitute a planning algorithm. One of these decisions is to stop planning and execute a real action; other internal actions may decide to search some promising part of the state space based on the current internal state.

Because Bayes-adaptive planning is so computationally demanding and may not always yield a high return on computational investment, we see meta-level control as a necessary component to allocate planning resources in an intelligent and dynamic way. One particularly attractive idea is to leverage information encoded in the prior, such as controllability, to determine whether planning is worth the computational effort (Lieder et al., 2013). In a way, the UCT policy in BAMCP is a primitive example of such resource allocation mechanisms, but there are many more variables to control and many more ways to control them.

7.4.3.3 Long Term Memories

We introduced planning methods as if it there is a clear dichotomy between online and offline planning. Given the difficulty posed by scaling offline approximations for Bayes-adaptive planning, we chose to focus on online planning mechanisms. However, online and offline planning need not be treated as incompatible paradigms.

In its purest form, online planning optimizes the current action from scratch at every step. Instead, online planning can incorporate information from past searches, but also from some offline computations. In our BAMCP algorithm, the tree computed at the previous planning step can be used to bootstrap planning for the current time-step by starting from the corresponding computed subtree. Similarly, in the BAFA algorithm, the parameter values computed in previous time-steps can be used to warm-start the learning as long as the same particles are employed across steps. Another form of long-term memory is the rollout learning mechanism in BAMCP that learns a state-value function to help in all planning steps.

More generally, as in the Dyna 2 architecture (Silver et al., 2012), we could learn a representation of the value for all augmented states offline, or across steps. This value would then be combined with a local value representation that is learned at each time step during online planning. If appropriate representations can be found, then online planning could be greatly accelerated from these global representations. This also resonates with the meta-control idea in the last section; if there exists enough certainty about the value, or superiority, of an action in a particular belief-state according to some learned global representation, then it is not necessary to perform some expensive planning operations.

7.4.4 About Modeling

As we mentioned in the first chapter, we expect that animals also face adaptive control problems, since they must act in the face of substantial uncertainty about their environment. There is solid evidence that animals such as rodents learn models of the environment (Tolman, 1948) and use them to make decisions (Dickinson and Balleine, 2002), including recent suggestions that forms of sample-based forward-planning may be involved (Pfeiffer and Foster, 2013). There is also evidence that some aspects of perception in animals can be well explained by modeling it as Bayesian inference influenced by prior distributions about natural (or learned) statistics. However, there is little data about animals (or humans) combining Bayesian inference and planning to perform some kind of Bayesian adaptive planning (Acuña and Schrater, 2010; Daw et al., 2006; Huys and Dayan, 2009).

At the same time, there are suggestions (Daw et al., 2005; Keramati et al., 2011; Pezzulo et al., 2013) that forms of meta-control are present that can arbitrate between different types of controllers present in the brain and can adaptively allocate resources for planning based on internal states. We have described scenarios, including ethologically relevant ones such as the mushroom exploration task in Chapter 6, where Bayes-adaptive planning can have a substantial advantage using a moderate amount of computation. We expect similar scenarios to arise across animal niches, making it plausible that some animal brains would benefit from a mechanism to plan in belief-space in order to achieve these gains. We speculate that, if these systems are present, they would learn to rely on expensive Bayes-adaptive computations only if this is likely to be beneficial (at least sufficiently beneficial as to compensate the costs of computation).

To uncover whether such a mechanism is present, we need to find instrumental tasks for which the (presumed) meta-controller would judge it worthwhile to allocate a substantial amount of planning resources, if that decision process is conducted in a model-based way. An existing, or easily learned, representation for beliefs about the task dynamics should also be ideally present to optimize some belief-policy incrementally, or to help during planning. Even if such planning takes place, it is unclear whether the necessary approximations implemented in neurological tissues, with heavy constraints on resources, would have anything to do with the forms of sample-based planning methods we proposed; this would have to be tested by looking at various behavioral signatures in the data. An alternative is that animals and humans simply employ finely-tuned heuristics, akin to reward bonuses, to provide a cheap approximation to optimal exploration-exploitation. While such Pavlovian biases may be effective in natural historical environments, this should lead to inapppropriate behavior when making important decisions in uncertain settings not well captured by the animal's evolutionary history.

7.5 Final Words

In sum, properly integrating exploration and exploitation has been a central concern since the earliest days of normative approaches to sequential decision making. However, despite its theoretical elegance, the empirical impact of these concerns has been diminished by two linked factors: integration is most valuable in the context of richly structured models of uncertainty that specify broad patterns of generalization; but such structured models have hitherto posed insurmountable computational challenges. Here, we have shown how to address these challenges by exploiting modern RL techniques, providing scalability to large, and even formally infinite, domains, and thus further closing the gap to fully-Bayesian agents in real-world applications.

Bibliography

Acuña, D. E. and Schrater, P. (2010). Structure learning in human sequential decision-making. *PLoS computational biology*, 6(12). (pages 13 and 170)

Agrawal, R. (1995). Sample mean based index policies with O(log n) regret for the multi-armed bandit problem. *Advances in Applied Probability*, pages 1054–1078. (page 47)

Agrawal, S. and Goyal, N. (2012). Analysis of Thompson sampling for the multi-armed bandit problem. In *Proceedings of the 25th Annual Conference on Learning Theory (COLT)*. Arxiv preprint. (page 48)

Aldous, D. J. (1985). *Exchangeability and related topics*. Springer. (page 79)

Araya-López, M., Buffet, O., and Thomas, V. (2012). Near-optimal BRL using optimistic local transitions. In *Proceedings of the 29th International Conference on Machine Learning*. (pages 43 and 71)

Arrow, K. J., Blackwell, D., and Girshick, M. A. (1949). Bayes and minimax solutions of sequential decision problems. *Econometrica, Journal of the Econometric Society*, pages 213–244. (pages 83 and 84)

Asmuth, J., Li, L., Littman, M., Nouri, A., and Wingate, D. (2009). A Bayesian sampling approach to exploration in reinforcement learning. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 19–26. (pages 64, 71, 76, 77, 81, 139, 142, 143, and 155)

Asmuth, J. and Littman, M. (2011). Approaching Bayes-optimality using Monte-Carlo tree search. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence*, pages 19–26. (pages 21, 69, 108, 112, and 122)

Asmuth, J. T. (2013). *Model-based Bayesian reinforcement learning with generalized priors*. PhD thesis, Rutgers University-Graduate School-New Brunswick. (pages 43 and 77)

Astrom, K. J. (1965). Optimal control of markov decision processes with incomplete state estimation. *Journal of Mathematical Analysis and Applications*, 10(1):174–205. (page 44)

Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2):235–256. (pages 14, 47, and 62)

Auer, P., Jaksch, T., and Ortner, R. (2009). Near-optimal regret bounds for reinforcement learning. *Advances in neural information processing systems*, 21:89–96. (page 64)

Auer, P. and Ortner, R. (2010). UCB revisited: Improved regret bounds for the stochastic multi-armed bandit problem. *Periodica Mathematica Hungarica*, 61(1):55–65. (page 47)

Bache, K. and Lichman, M. (2013). UCI machine learning repository, mushroom dataset. `http://archive.ics.uci.edu/ml/datasets/Mushroom`. (page 145)

Baird, L. et al. (1995). Residual algorithms: Reinforcement learning with function approximation. In *Proceedings of the International Conference on Machine Learning*, pages 30–37. (page 56)

Bar-Shalom, Y. and Tse, E. (1974). Dual effect, certainty equivalence, and separation in stochastic control. *Automatic Control, IEEE Transactions on*, 19(5):494–500. (page 85)

Barnard, G. A. (1946). Sequential tests in industrial statistics. *Supplement to the Journal of the Royal Statistical Society*, pages 1–26. (page 84)

Bartlett, P. L. and Baxter, J. (2011). Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:319–350. (page 52)

Beal, M. J., Ghahramani, Z., and Rasmussen, C. E. (2001). The infinite hidden Markov model. In *Advances in Neural Information Processing Systems*, pages 577–584. (page 81)

Bellman, R. (1954). The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 60(6):503–515. (page 51)

Bellman, R. (1956). A problem in the sequential design of experiments. *Sankhyā: The Indian Journal of Statistics*, pages 221–229. (page 29)

Bellman, R. and Kalaba, R. (1959). On adaptive control processes. *Automatic Control, IRE Transactions on*, 4(2):1–9. (pages 16, 48, and 84)

Bertsekas, D. and Tsitsiklis, J. (1996). Neuro-dynamic programming. *Athena Scientific*. (pages 12 and 53)

Bertsekas, D. P. (2011a). Approximate dynamic programming. (page 52)

Bertsekas, D. P. (2011b). Approximate policy iteration: A survey and some new methods. *Journal of Control Theory and Applications*, 9(3):310–335. (page 126)

Bertsekas, D. P. and Shreve, S. E. (1978). *Stochastic optimal control: The discrete time case*, volume 139. Academic Press New York. (pages 37, 41, and 42)

Brafman, R. and Tennenholtz, M. (2003). R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *The Journal of Machine Learning Research*, 3:213–231. (pages 64 and 70)

Cassandra, A. R., Kaelbling, L. P., and Littman, M. L. (1994). Acting optimally in partially observable stochastic domains. In *Association for the Advancement of Artificial Intelligence*, volume 94, pages 1023–1028. (page 65)

Castro, P. and Precup, D. (2007). Using linear programming for Bayesian exploration in Markov decision processes. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 2437–2442. (page 69)

Castro, P. and Precup, D. (2010). Smarter sampling in model-based Bayesian reinforcement learning. In *Machine Learning and Knowledge Discovery in Databases*, pages 200–214. Springer. (pages 71, 108, and 112)

Coquelin, P. and Munos, R. (2007). Bandit algorithms for tree search. In *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence*, pages 67–74. (page 62)

Cozzolino, J., Gonzalez-Zubieta, R., and Miller, R. (1965). Markov decision processes with uncertain transition probabilities. Technical report, 11, Operations Research Center, MIT. (pages 70 and 84)

Davies, S., Ng, A., and Moore, A. (1998). Applying online search techniques to reinforcement learning. In *Proceedings of the National Conference on Artificial Intelligence*, pages 753–760. (page 58)

Daw, N., Niv, Y., and Dayan, P. (2005). Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nature Neuroscience*, 8(12):1704–1711. (page 170)

Daw, N. D., O'Doherty, J. P., Dayan, P., Seymour, B., and Dolan, R. J. (2006). Cortical substrates for exploratory decisions in humans. *Nature*, 441(7095):876–879. (page 170)

Dayan, P. (2013). Exploration from generalization mediated by multiple controllers. In *Intrinsically Motivated Learning in Natural and Artificial Systems*, pages 73–91. Springer. (pages 13 and 164)

Dayan, P., Hinton, G. E., Neal, R. M., and Zemel, R. S. (1995). The Helmholtz machine. *Neural computation*, 7(5):889–904. (page 166)

Dayan, P. and Sejnowski, T. (1996). Exploration bonuses and dual control. *Machine Learning*, 25(1):5–22. (pages 70 and 140)

Dearden, R., Friedman, N., and Russell, S. (1998). Bayesian Q-learning. In *Proceedings of the National Conference on Artificial Intelligence*, pages 761–768. (pages 106, 107, and 108)

Deisenroth, M. and Rasmussen, C. (2011). PILCO: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on Machine Learning*, pages 465–473. (pages 76, 77, 82, and 122)

Deisenroth, M. P., Neumann, G., and Peters, J. (2013). A survey on policy search for robotics. *Foundations and Trends in Robotics*, 2(1-2):1–142. (page 33)

Deisenroth, M. P., Rasmussen, C. E., and Peters, J. (2009). Gaussian process dynamic programming. *Neurocomputing*, 72(7):1508–1524. (pages 82, 122, 131, 132, and 134)

Dickinson, A. and Balleine, B. (2002). The role of learning in the operation of motivational systems. *Stevens' handbook of experimental psychology*. (page 170)

Doshi-Velez, F. (2009). The infinite partially observable Markov decision process. In *Advances in Neural Information Processing Systems*, volume 22, pages 477–485. (pages 77 and 81)

Doshi-Velez, F., Wingate, D., Roy, N., and Tenenbaum, J. (2010). Nonparametric Bayesian policy priors for reinforcement learning. In *Advances in Neural Information Processing Systems*. (pages 76, 139, 140, and 166)

Duff, M. (2002). *Optimal Learning: Computational Procedures For Bayes-Adaptive Markov Decision Processes*. PhD thesis, University of Massachusetts Amherst. (pages 17, 21, 41, 45, 48, 66, 67, 75, 85, 114, 135, and 140)

Duff, M. (2003). Design for an optimal probe. In *Proceedings of the 20th International Conference on Machine Learning*, pages 131–138. (pages 68 and 128)

Escobar, M. D. and West, M. (1995). Bayesian density estimation and inference using mixtures. *Journal of the american statistical association*, 90(430):577–588. (page 159)

Fel'dbaum, A. (1960). Dual control theory. *Automation and Remote Control*, 21(9):874–1039. (page 84)

Ferguson, T. S. (1973). A Bayesian analysis of some nonparametric problems. *The annals of statistics*, pages 209–230. (page 78)

Fienberg, S. E. (2006). When did Bayesian inference become ”Bayesian”? *Bayesian analysis*, 1(1):1–40. (page 83)

Fonteneau, R., Busoniu, L., and Munos, R. (2013). Optimistic planning for belief-augmented Markov decision processes. In *IEEE International Symposium on Adaptive Dynamic Programming and reinforcement Learning (ADPRL 2013)*. (page 68)

Friedman, N. and Singer, Y. (1999). Efficient Bayesian parameter estimation in large discrete domains. In *Advances in Neural Information Processing Systems*, pages 417–423. (pages 75, 76, and 107)

Gelly, S., Kocsis, L., Schoenauer, M., Sebag, M., Silver, D., Szepesvári, C., and Teytaud, O. (2012). The grand challenge of computer Go: Monte Carlo tree search and extensions. *Communications of the ACM*, 55(3):106–113. (pages 21 and 62)

Gelly, S. and Silver, D. (2007). Combining online and offline knowledge in UCT. In *Proceedings of the 24th International Conference on Machine learning*, pages 273–280. (page 95)

Gittins, J. and Jones, D. (1974). A dynamic allocation index for the sequential design of experiments. In Gani, J., editor, *Progress in Statistics*, pages 241–266. North-Holland, Amsterdam, NL. (pages 14, 17, and 29)

Gittins, J., Weber, R., and Glazebrook, K. (1989). *Multi-armed bandit allocation indices*. Wiley Online Library. (pages 48, 67, 99, 100, and 129)

Good, I. J. (1979). Studies in the history of probability and statistics. XXXVII AM Turing's statistical work in World War II. *Biometrika*, 66(2):393–396. (page 84)

Gordon, N. J., Salmond, D. J., and Smith, A. F. (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In *IEE Proceedings F (Radar and Signal Processing)*, volume 140, pages 107–113. (page 127)

Goschin, S., Weinstein, A., Littman, M. L., and Chastain, E. (2012). Planning in reward-rich domains via PAC bandits. In *European Workshop on Reinforcement Learning*, pages 25–42. (page 33)

Griffiths, T. and Ghahramani, Z. (2005). Infinite latent feature models and the Indian buffet process. (page 81)

Guez, A., Silver, D., and Dayan, P. (2012). Efficient Bayes-adaptive reinforcement learning using sample-based search. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1034–1042. (page 122)

Hart, P., Nilsson, N., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107. (page 58)

Hay, N., Russell, S., Tolpin, D., and Shimony, S. E. (2012). Selecting computations: Theory and applications. *arXiv preprint arXiv:1207.5879*. (page 168)

Howard, R. A. (1960). *Dynamic Programming and Markov Processes.* (pages 31 and 51)

Huys, Q. and Dayan, P. (2009). A Bayesian formulation of behavioral control. *Cognition*, 113(3):314–328. (pages 13, 73, 76, 166, and 170)

Huys, Q. J., Guitart-Masip, M., Dolan, R. J., and Dayan, P. (2014). Decision-theoretic psychiatry. *Clinical Psychological Science*. (page 73)

Jaksch, T., Ortner, R., and Auer, P. (2010). Near-optimal regret bounds for reinforcement learning. *The Journal of Machine Learning Research*, 99:1563–1600. (pages 35 and 36)

Jung, T. and Stone, P. (2010). Gaussian processes for sample efficient reinforcement learning with rmax-like exploration. In *Machine Learning and Knowledge Discovery in Databases*, pages 601–616. Springer. (page 82)

Kaelbling, L. (1993). *Learning in embedded systems*. The MIT Press. (page 47)

Kakade, S. (2003). *On the sample complexity of reinforcement learning*. PhD thesis, University College London. (pages 34, 35, and 64)

Kearns, M., Mansour, Y., and Ng, A. (1999). A sparse sampling algorithm for near-optimal planning in large Markov decision processes. In *Proceedings of the 16th international joint conference on Artificial intelligence-Volume 2*, pages 1324–1331. (pages 20, 59, and 70)

Kearns, M. and Singh, S. (2002). Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2):209–232. (page 64)

Kelly, F. et al. (1981). Multi-armed bandits with discount factor near one: The Bernoulli case. *The Annals of Statistics*, 9(5):987–1001. (page 50)

Keramati, M., Dezfouli, A., and Piray, P. (2011). Speed/accuracy trade-off between the habitual and the goal-directed processes. *PLoS computational biology*, 7(5). (page 170)

Kleinberg, R. D. (2004). Nearly tight bounds for the continuum-armed bandit problem. In *Advances in Neural Information Processing Systems*, pages 697–704. (page 29)

Kocsis, L. and Szepesvári, C. (2006). Bandit based Monte-Carlo planning. In *Machine Learning: ECML*, pages 282–293. Springer. (pages 21, 62, 98, and 101)

Kolter, J. and Ng, A. (2009). Near-Bayesian exploration in polynomial time. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 513–520. (pages 43, 71, 108, and 112)

Korf, R. E. (1990). Real-time heuristic search. *Artificial intelligence*, 42(2):189–211. (page 20)

Kumar, P. (1985). A survey of some results in stochastic adaptive control. *SIAM Journal on Control and Optimization*, 23(3):329–380. (pages 16, 38, 85, and 140)

Kurniawati, H., Hsu, D., and Lee, W. S. (2008). SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Robotics: Science and Systems*, pages 65–72. (pages 66 and 128)

Lai, T. and Robbins, H. (1985). Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22. (pages 14, 27, 28, 46, and 47)

Langford, J. and Zhang, T. (2007). The epoch-greedy algorithm for contextual multi-armed bandits. *Advances in Neural Information Processing Systems*, 20:1096–1103. (pages 29 and 146)

Lazaric, A. and Ghavamzadeh, M. (2010). Bayesian multi-task reinforcement learning. In *Proceedings of the International Conference on Machine Learning*. Citeseer. (page 76)

Li, L. (2009). *A unifying framework for computational reinforcement learning theory*. PhD thesis, Rutgers, The State University of New Jersey. (page 64)

Li, L., Chu, W., Langford, J., Moon, T., and Wang, X. (2012). An unbiased offline evaluation of contextual bandit algorithms with generalized linear models. In *JMLR Workshop: On-line Trading of Exploration and Exploitation 2*. (pages 149 and 150)

Lieder, F., Goodman, N. D., and Huys, Q. J. (2013). Controllability and resource-rational planning. In *Computational and Systems Neuroscience Conference*. (page 169)

Lim, Z., Sun, L., and Hsu, D. J. (2011). Monte carlo value iteration with macro-actions. In *Advances in Neural Information Processing Systems*, pages 1287–1295. (page 167)

Maei, H., Szepesvári, C., Bhatnagar, S., and Sutton, R. (2010). Toward off-policy learning control with function approximation. In *Proceedings of the International Conference on Machine Learning*, pages 719–726. Citeseer. (pages 56 and 127)

Mahadevan, S. (1996). Average reward reinforcement learning: Foundations, algorithms, and empirical results. *Machine learning*, 22(1-3):159–195. (page 32)

Mannor, S. and Tsitsiklis, J. (2004). The sample complexity of exploration in the multi-armed bandit problem. *The Journal of Machine Learning Research*, 5:623–648. (page 35)

Martin, J. (1967). *Bayesian decision problems and Markov chains*. Wiley. (pages 17, 36, 41, 75, and 84)

McAllester, D. A. and Singh, S. (1999). Approximate planning for factored pomdps using belief state simplification. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 409–416. (page 66)

Meuleau, N. and Bourgine, P. (1999). Exploration of multi-state environments: Local measures and back-propagation of uncertainty. *Machine Learning*, 35(2):117–154. (pages 47, 70, and 108)

Milch, B., Marthi, B., Russell, S., Sontag, D., Ong, D. L., and Kolobov, A. (2007). Blog: Probabilistic models with unknown objects. *Statistical relational learning*, page 373. (page 165)

Moldovan, T. M., Jordan, M. I., and Abbeel, P. (2013). Dirichlet process reinforcement learning. In *Reinforcement Learning and Decision Making Meeting*. (page 134)

Monahan, G. E. (1982). State of the art—a survey of partially observable markov decision processes: Theory, models, and algorithms. *Management Science*, 28(1):1–16. (page 65)

Moon, J. (1973). Random walks on random trees. *Journal of the Australian Mathematical Society*, 15(01):42–53. (page 141)

Neal, R. M. (1993). Probabilistic inference using markov chain monte carlo methods. Technical report, University of Toronto. (page 99)

Niño-Mora, J. (2007). A (2/3) n^3 fast-pivoting algorithm for the gittins index and optimal stopping of a markov chain. *INFORMS J. Comput*, 19(4):596–606. (page 50)

Orbanz, P. and Teh, Y. W. (2010). Bayesian nonparametric models. In *Encyclopedia of Machine Learning*. Springer. (page 77)

Ortega, P. A. and Braun, D. A. (2010). A minimum relative entropy principle for learning and acting. *Journal of Artificial Intelligence Research*, 38(1):475–511. (pages 141 and 164)

Osband, I., Russo, D., and Van Roy, B. (2013). (more) efficient reinforcement learning via posterior sampling. In *Advances in Neural Information Processing Systems (NIPS)*. (pages 35, 64, 70, and 142)

Parr, R. and Russell, S. (1995). Approximating optimal policies for partially observable stochastic domains. In *International Joint Conference on Artificial Intelligence*, volume 95, pages 1088–1094. (page 136)

Pezzulo, G., Rigoli, F., and Chersi, F. (2013). The mixed instrumental controller: using value of information to combine habitual choice and mental simulation. *Frontiers in psychology*, 4. (page 170)

Pfeiffer, B. E. and Foster, D. J. (2013). Hippocampal place-cell sequences depict future paths to remembered goals. *Nature*, 497(7447):74–79. (page 170)

Pineau, J., Gordon, G., and Thrun, S. (2003). Point-based value iteration: An anytime algorithm for POMDPs. In *International Joint Conference on Artificial Intelligence*, volume 18, pages 1025–1032. (pages 66 and 128)

Porta, J., Vlassis, N., Spaan, M., and Poupart, P. (2006). Point-based value iteration for continuous POMDPs. *The Journal of Machine Learning Research*, 7:2329–2367. (pages 45, 66, and 135)

Poupart, P., Vlassis, N., Hoey, J., and Regan, K. (2006). An analytic solution to discrete Bayesian reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 697–704. (pages 66 and 68)

Puterman, M. (1994). Markov decision processes. *John Wiely and Sons, New York*. (pages 15, 29, 32, and 52)

Raiffa, H. and Schlaifer, R. (1961). *Applied statistical decision theory*. (pages 67 and 84)

Rasmussen, C. E. (2006). *Gaussian processes for machine learning*. (page 81)

Robbins, H. (1952). Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58(5):527–535. (pages 14 and 27)

Ross, S. (1983). *Introduction to stochastic dynamic programming: Probability and mathematical*. Academic Press, Inc. (page 51)

Ross, S. and Pineau, J. (2008). Model-based bayesian reinforcement learning in large structured domains. In *Proceedings of the 24th Conference in Uncertainty in Artificial Intelligence*, pages 476–483. (pages 21, 69, 76, and 122)

Ross, S., Pineau, J., Chaib-draa, B., and Kreitmann, P. (2011). A Bayesian approach for learning and planning in Partially Observable Markov Decision Processes. *Journal of Machine Learning Research*, 12:1729–1770. (page 45)

Ross, S., Pineau, J., Paquet, S., and Chaib-Draa, B. (2008). Online planning algorithms for POMDPs. *Journal of Artificial Intelligence Research*, 32(1):663–704. (pages 21 and 66)

Salakhutdinov, R., Tenenbaum, J. B., and Torralba, A. (2013). Learning with hierarchical-deep models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1958–1971. (page 165)

Schmidhuber, J. (1991). Curious model-building control systems. In *IEEE International Joint Conference on Neural Networks*, pages 1458–1463. (pages 70 and 164)

Schwartz, A. (1993). A reinforcement learning method for maximizing undiscounted rewards. In *Proceedings of the International Conference on Machine Learning*, volume 93, pages 298–305. (pages 31, 32, and 163)

Silver, D., Sutton, R. S., and Müller, M. (2012). Temporal-difference search in computer Go. *Machine learning*, 87(2):183–219. (pages 24, 62, 63, 121, 122, and 169)

Silver, D. and Veness, J. (2010). Monte-Carlo planning in large POMDPs. In *Advances in Neural Information Processing Systems*, pages 2164–2172. (pages 21, 22, 66, 87, 93, 96, 98, and 160)

Silver, E. (1963). Markovian decision processes with uncertain transition probabilities or rewards. Technical report, DTIC Document. (page 84)

Singh, S., Jaakkola, T., Littman, M. L., and Szepesvári, C. (2000). Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine Learning*, 38(3):287–308. (page 63)

Singh, S. P. and Sutton, R. S. (1996). Reinforcement learning with replacing eligibility traces. *Machine learning*, 22(1-3):123–158. (page 54)

Smallwood, R. and Sondik, E. (1973). The optimal control of partially observable markov processes over a finite horizon. *Operations Research*, pages 1071–1088. (page 65)

Sonin, I. (2008). A generalized gittins index for a markov chain and its recursive calculation. *Statistics and Probability Letters*, 78(12):1526–1533. (page 50)

Sorg, J., Singh, S., and Lewis, R. (2010). Internal rewards mitigate agent boundedness. In *Proceedings of the 27th International Conference on Machine Learning*. (page 166)

Spaan, M. and Vlassis, N. (2005). Perseus: Randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research*, 24(1):195–220. (page 66)

Strehl, A., Li, L., and Littman, M. (2009). Reinforcement learning in finite MDPs: PAC analysis. *The Journal of Machine Learning Research*, 10:2413–2444. (page 35)

Strehl, A., Li, L., Wiewiora, E., Langford, J., and Littman, M. (2006). PAC model-free reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 881–888. (pages 34 and 64)

Strehl, A. L. and Littman, M. L. (2005). A theoretical analysis of model-based interval estimation. In *Proceedings of the 22nd international conference on Machine learning*, pages 856–863. (pages 47 and 64)

Strens, M. (2000). A Bayesian framework for reinforcement learning. In *Proceedings of the 17th International Conference on Machine Learning*, pages 943–950. (pages 70, 76, 106, 107, 108, 142, and 155)

Sudderth, E. B. (2006). *Graphical models for visual object recognition and tracking*. PhD thesis, Massachusetts Institute of Technology. (page 159)

Sutton, R. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the Seventh International Conference on Machine Learning*, volume 216, page 224. Citeseer. (page 70)

Sutton, R. and Barto, A. (1998). *Reinforcement learning*. MIT Press. (pages 13, 53, and 55)

Sutton, R., Maei, H., Precup, D., Bhatnagar, S., Silver, D., Szepesvári, C., and Wiewiora, E. (2009). Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 993–1000. ACM. (page 124)

Sutton, R. S., Precup, D., and Singh, S. (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1):181–211. (page 167)

Szepesvári, C. (2010). *Algorithms for reinforcement learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers. (page 53)

Szita, I. and Szepesvári, C. (2010). Model-based reinforcement learning with nearly tight exploration complexity bounds. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 1031–1038. (page 64)

Thompson, W. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294. (pages 27 and 47)

Thrun, S. (1999). Monte carlo POMDPs. In *Advances in Neural Information Processing Systems*, volume 12, pages 1064–1070. (pages 66 and 128)

Tolman, E. C. (1948). Cognitive maps in rats and men. *Psychological review*, 55(4):189. (pages 12 and 170)

Tonk, S. and Kappen, H. (2010). Optimal exploration as a symmetry breaking phenomenon. Technical report, Radboud University Nijmegen. (page 67)

Tsitsiklis, J. N. (2003). On the convergence of optimistic policy iteration. *The Journal of Machine Learning Research*, 3:59–72. (page 55)

Tsitsiklis, J. N. and Van Roy, B. (2002). On average versus discounted reward temporal-difference learning. *Machine Learning*, 49(2-3):179–191. (page 32)

Tziortziotis, N., Dimitrakakis, C., and Blekas, K. (2013). Cover tree Bayesian reinforcement learning. *arXiv preprint arXiv:1305.1809*. (pages 76 and 139)

Van Hee, K. M. (1978). *Bayesian control of Markov chains*, volume 95. Mathematisch centrum. (page 42)

Varaiya, P., Walrand, J., and Buyukkoc, C. (1985). Extensions of the multiarmed bandit problem: the discounted case. *Automatic Control, IEEE Transactions on*, 30(5):426–439. (page 50)

Wald, A. (1945). Sequential tests of statistical hypotheses. *The Annals of Mathematical Statistics*, 16(2):117–186. (page 83)

Wald, A. and Wolfowitz, J. (1948). Optimum character of the sequential probability ratio test. *The Annals of Mathematical Statistics*, pages 326–339.
(pages 83 and 84)

Walsh, T., Goschin, S., and Littman, M. (2010). Integrating sample-based planning and model-based reinforcement learning. In *Proceedings of the 24th Conference on Artificial Intelligence (AAAI)*. (pages 60 and 69)

Wang, T., Lizotte, D., Bowling, M., and Schuurmans, D. (2005). Bayesian sparse sampling for on-line reward optimization. In *Proceedings of the 22nd International Conference on Machine learning*, pages 956–963.
(pages 21, 68, 69, and 122)

Wang, Y., Won, K., Hsu, D., and Lee, W. (2012). Monte Carlo Bayesian reinforcement learning. In *Proceedings of the 29th International Conference on Machine Learning*. (pages 68 and 128)

Watkins, C. (1989). *Learning from delayed rewards*. PhD thesis, Cambridge.
(pages 55 and 95)

Weber, R. (1992). On the Gittins index for multiarmed bandits. *The Annals of Applied Probability*, pages 1024–1033. (page 48)

Whittle, P. (1980). Multi-armed bandits and the Gittins index. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 143–149. (page 50)

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.
(page 52)

Wingate, D., Goodman, N., Roy, D., Kaelbling, L., and Tenenbaum, J. (2011). Bayesian policy search with policy priors. In *Proceedings of the International Joint Conferences on Artificial Intelligence*. (page 76)