Complexity of stochastic branch and bound methods for belief tree search in Bayesian reinforcement learning

Christos Dimitrakakis
Intelligent Systems Laboratory Amsterdam,* University of Amsterdam,
Science Park 107, Amsterdam 1097 XG, The Netherlands
c.dimitrakakis@uva.nl

Abstract

There has been a lot of recent work on Bayesian methods for reinforcement learning exhibiting near-optimal online performance. The main obstacle facing such methods is that in most problems of interest, the optimal solution involves planning in an infinitely large tree. However, it is possible to obtain lower and stochastic upper bounds on the value of each tree node. This enables us to use stochastic branch and bound algorithms to search the tree efficiently. This paper examines the complexity of such algorithms.

1 Introduction

Bayesian methods for exploration in Markov decision processes (MDPs) and for solving known partially-observable Markov decision processes (POMDPs), as well as for exploration in the latter case, have been proposed many times previously [RPCd08, THS06, HDFJ08, PVHR06, Duf02, RCdP08, DB97]. However, such methods typically suffer from computational intractability problems.

The sources of intractability are two-fold. Firstly, there may be no compact representation of the current belief. This is especially true for POMDPs. Secondly, optimally behaving under uncertainty requires that we create an *augmented* MDP model in the form of a tree [Duf02], where the root node is the current belief-state pair and children are all possible subsequent belief-state pairs. This tree grows large very fast, and it is particularly problematic to grow in the case of continuous observations or actions. In this work, we concentrate on the second problem – and consider algorithms for expanding the tree.

Since the Bayesian exploration methods require a tree expansion to be performed, we can view the whole problem as that of *nested* exploration. For the simplest exploration-exploitation trade-off setting, bandit problems, there already exist nearly optimal, computationally simple methods [Aue02]. Such methods have recently been extended to tree search [KS06]. This work proposes to take advantage of the special structure of belief trees in order to design nearly-optimal algorithms for expansion of nodes. In a sense, by recognising that the tree expansion problem in Bayesian look-ahead exploration methods is also an optimal exploration problem, we develop tree algorithms that can solve this problem efficiently. Furthermore, we are able to derive interesting upper and lower bounds for the value of branches and leaf nodes which can help limit the amount of search. The ideas developed are tested in the multi-armed bandit setting for which nearly-optimal algorithms already exist.

1.1 Related work

Up to date, most work had only used full expansion of the belief tree up to a certain depth. A notable exception is [WLBS05], which uses Thompson sampling [Tho33] to expand the tree. In very recent work [RPPCd08], the importance of tree expansion in the closely related POMDP setting[1] has been recognised. Therein, the authors contrast and compare many different methods for tree expansion, including branch and bound methods and Monte Carlo sampling.

Monte Carlo sampling methods have also been recently explored in the upper confidence bounds on trees (UCT) algorithms, proposed in [GS07, KS06] in the context of planning in games. Recently, [HM08] also considered the application of UCT to planning in deterministic systems. Our case is similar, however we are now acting within stochastic trees whose nodes arise from Bayesian beliefs. Such trees are Markov decision process referred to as BAMDPs. In our case, we can take advantage of the special structure of the belief tree but stochasticity makes the problem harder.

The proposed methods are also related to the ones used in the discrete-state POMDP setting [RPPCd08]. However, our setting requires the evaluation of different bounds at leaf nodes. In particular, for each tree node we can obtain a lower bound and a stochastic upper bound on the value of the optimal policy. This is summarised in Appendix 2.

While the setting is essentially a form of multi-stage stochastic programming[SN05, KSHdM01, KWK94], sample complexity results of the type we present here have not been reported. In previous work, we had presented first results on bandit problems [Dim08] employing algorithms discussed in this paper, for which nearly-optimal distribution-free algorithms are known. The current paper's main contribution is complexity results on a number of tree search algo-

*This work was supported by the ICIS project. Many thanks to Zhou Fang and Gwenn Englebienne for useful discussions.

[1] The BAMDP setting is equivalent to a POMDP where the unobservable part of the state is stationary, but continuous (chap. 5 [Duf02])

rithms. This includes a variant of a Stochastic branch and bound algorithm first introduced in [NPR98], for which only an asymptotic convergence proof had existed. This paper provides a complexity analysis for this variant, under similar smoothness conditions. There is also a close relation to the bandit algorithm for smooth trees introduced by [CM07]. However, this, and other UCT variants algorithms are not directly applicable to this problem, fistly since the tree is stochastic and secondly since we cannot obtain unbiased estimates of the optimal value function, but rather an unbiased sample from an upper bound. This paper also presents a suitable tree search algorithm for belief trees.

The rest of this paper is organised as follows. The following section introduces belief-augmented MDPs and gives bounds on the value of nodes that arise in the constructed tree. Section **??** contains the paper's main contributions. Section **??** discusses potential improvements. Tedious proofs are given in Appendix **??**

2 BAMDPs[2]

We are interested in sequential decision problems where, at each time step t, the agent seeks to maximise the expected utility

$$\mathbf{E}[u_t \mid \cdot] \triangleq \sum_{k=1}^{\infty} \gamma^k \mathbf{E}[r_{t+k} \mid \cdot],$$

where r is a stochastic reward and u_t is simply the discounted sum of future rewards. We shall assume that the sequence of rewards arises from a Markov decision process, defined below.

Definition 1 (Markov decision process) *A Markov decision process (MDP) is defined as the tuple $\mu = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R})$ comprised of a set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, a transition distribution $\mathcal{T}$ conditioning the next state on the current state and action,*

$$\mathcal{T}(s'|s,a) \triangleq \mu(s_{t+1}{=}s'|s_t{=}s, a_t = a) \tag{2.1}$$

satisfying the Markov property $\mu(s_{t+1} \mid s_t, a_t) = \mu(s_{t+1} \mid s_t, a_t, s_{t-1}, a_{t-1}, \ldots)$, and a reward distribution $\mathcal{R}$ conditioned on states and actions:

$$\mathcal{R}(r|s,a) \triangleq \mu(r_{t+1}{=}r \mid s_t{=}s, a_t{=}a), \tag{2.2}$$

with $a \in \mathcal{A}$, $s, s' \in \mathcal{S}$, $r \in \mathbb{R}$. Finally,

$$\mu(r_{t+1}, s_{t+1}|s_t, a_t) = \mu(r_{t+1}|s_t, a_t)\mu(s_{t+1}|s_t, a_t). \tag{2.3}$$

We shall denote the set of all MDPs as $\mathcal{M}$. For any policy π that is an arbitrary distribution on actions, we can define a T-horizon value function for an MDP $\mu \in \mathcal{M}$ at time t as:

$$V^{\pi,\mu}_{t,T}(s,a) = \mathbf{E}[r_{t+1} \mid s_t{=}s, a_t{=}a, \mu] + \gamma \sum_{s'} \mu(s_{t+1}{=}s' \mid s_t{=}s, a_t{=}a) V^{\pi}_{\mu,t+1,T}(s').$$

Note that for the infinite-horizon case, $\lim_{T\to\infty} V^{\pi,\mu}_{t,T} = V^{\pi,\mu}$ for all t.

[2]This section was originally published in [Dim08] and is reproduced here for completeness.

In the case where the MDP is unknown, it is possible to use a Bayesian framework to represent our uncertainty (c.f. [Duf02]). This essentially works by maintaining a belief $\xi_t \in \Xi$, about which MDP $\mu \in \mathcal{M}$ corresponds to reality. In a Bayesian setting, $\xi_t(\mu)$ is our subjective probability measure that μ is true.

In order to optimally select actions in this framework, we need to use the approach suggested originally in [BK59] under the name of Adaptive Control Processes. The approach was investigated more fully in [DB97, Duf02]. This creates an *augmented* MDP, with a state comprised of the original MDP's state s_t and our belief state ξ_t. We can then solve the exploration *in principle* via standard dynamic programming algorithms such as backwards induction. We shall call such models Belief-Augmented MDPs, analogously to the Bayes-Adaptive MDPs of [Duf02]. This is done by not only considering densities conditioned on the state-action pairs (s_t, a_t), i.e. $p(r_{t+1}, s_{t+1}|s_t, a_t)$, but taking into account the belief $\xi_t \in \Xi$, a probability space over possible MDPs, i.e. augmenting the state space from $\mathcal{S}$ to $\mathcal{S} \times \Xi$ and considering the following conditional density: $p(r_{t+1}, s_{t+1}, \xi_{t+1} \mid s_t, a_t, \xi_t)$. More formally, we may give the following definition:

Definition 2 (Belief-Augmented MDP) *A* Belief-Augmented MDP *ν (BAMPD) is an MDP $\nu = (\Omega, \mathcal{A}, \mathcal{T}', \mathcal{R}')$ where $\Omega = \mathcal{S} \times \Xi$, where Ξ is the set of probability measures on $\mathcal{M}$, and $\mathcal{T}', \mathcal{R}'$ are the transition and reward distributions conditioned jointly on the MDP state s_t, the belief state ξ_t, and the action a_t. Here $\xi_t(\xi_{t+1}|r_{t+1}, s_{t+1}, s_t, a_t)$ is singular, so that we can define the transition*

$$p(\omega_{t+1}|a_t, \omega_t) \equiv p(s_{t+1}, \xi_{t+1}|a_t, s_t, \xi_t).$$

It should be obvious that s_t, ξ_t jointly form a Markov state in this setting, called the *hyper-state*. In general, we shall denote the components of a future hyper-state ω^i_t as (s^i_t, ξ^i_t). However, in occasion we will abuse notation by referring to the components of some hyper-state ω as s_ω, ξ_ω. We shall use $\mathcal{M}_B$ to denote the set of BMDPs.

As in the MDP case, finite horizon problems only require sampling all future actions until the horizon T.

$$V^{\pi^*}_{t,T}(\omega_t, a_t) = \mathbf{E}[r_{t+1}|\omega_t, a_t] + \gamma \int_\Omega V^{\pi^*}_{t+1,T}(\omega_{t+1}) \nu(\omega_{t+1}|\omega_t, a_t)\, d\omega_{t+1}. \tag{2.4}$$

However, because the set of hyper-states available at each time-step is necessarily different from those at other time-steps, the value function cannot be easily calculated for the infinite horizon case.

In fact, the only clear solution is to continue expanding a belief tree until we are certain of the optimality of an action. As has previously been observed [DFR98, Dim06], this is possible since we can always obtain upper and lower bounds on the utility of any policy from the current hyper-state. We can apply such bounds on future hyper-states in order to efficiently expand the tree.

2.1 Belief tree expansion

Let the current belief be ξ_t and suppose we observe $x_t^i \triangleq (s_{t+1}^i, r_{t+1}^i, a_t^i)$. This observation defines a unique subsequent belief ξ_{t+1}^i. Together with the MDP state s, this creates a hyper-state transition from ω_t to ω_{t+1}^i. By recursively obtaining observations for future beliefs, we can obtain an unbalanced tree with nodes $\{\omega_{t+k}^i : k = 1, \dots, T; i = 1, \dots\}$. However, we cannot hope to be able to fully expand the tree. This is especially true in the case where observations (i.e. states, rewards, or actions) are continuous, where we cannot perform even a full single-step expansion. Even in the discrete case the problem is intractable for infinite horizons – and far too complex computationally for the finite horizon case. However, had there been efficient tree expansion methods, this problem would be largely alleviated.

All tree search methods require the expansion of leaf nodes. However, in general, a leaf node may have an infinite number of children. We thus need some strategies to limit the number of children. More formally, let us assume that we wish to expand in node $\omega_t^i = (\xi_t^i, s_t^i)$, with ξ_t^i defining a density over $\mathcal{M}$. For discrete state/action/reward spaces, we can simply enumerate all the possible outcomes $\{\omega_{t+1}^j\}_{j=1}^{|\mathcal{S}\times\mathcal{A}\times R|}$, where R is the set of possible reward outcomes. Note that if the reward is deterministic, there is only one possible outcome per state-action pair. The same holds if $\mathcal{T}$ is deterministic, in both cases making an enumeration possible. While in general this may not be the case, since rewards, states, or actions can be continuous, in this paper we shall only examine the discrete case.

2.2 Bounds on the optimal value function

At each point in the process, the next node ω_{t+k}^i to be expanded is the one maximising a utility $U(\omega_{t+k}^i)$. Let Ω_T be the set of leaf nodes. If their values were known, then we could easily perform the backwards induction procedure shown in Algorithm 1. The main problem is obtaining a good

Algorithm 1 Backwards induction action selection

procedure BACKWARDSINDUCTION(t, ν, Ω_T, V_T^*)
  for $n = T-1, T-2, \dots, t$ **do**
    for $\omega \in \Omega_n$ **do**

$$a_n^*(\omega) = \arg\max_a \nu(\omega'|\omega, a)[\mathbf{E}(r|\omega', \omega, \nu) + V_{n+1}^*(\omega')]$$

$$V_n(\omega)^* = \sum_{\omega' \in \Omega_{n+1}} \nu(\omega'|\omega, a_n^*)[\mathbf{E}(r|\omega', \omega, \nu) + V_{n+1}^*(\omega')]$$

    end for
  end for
  return a_t^*
end procedure

estimate for V_T^*, i.e. the value of leaf nodes. Let $\pi^*(\mu)$ denote the policy such that, for any π,

$$V_\mu^{\pi^*(\mu)}(s) \geq V_\mu^\pi(s) \qquad \forall s \in \mathcal{S}.$$

Furthermore, let the maximum probability MDP arising from the belief at hyper-state ω be $\hat{\mu}_\omega \triangleq \arg\max_\mu \hat{\mu}$. Similarly, we denote the mean MDP with $\hat{\mu}_\omega \triangleq \mathbf{E}[\mu|\xi_\omega]$.

Proposition 2.1 *The optimal value function at any leaf node ω is bounded by the following inequalities*

$$\int V_\mu^{\pi^*(\mu)}(s_\omega)\xi_\omega(\mu)d\mu \geq V^*(\omega) \geq \int V_\mu^{\pi^*(\hat{\mu}_\omega)}(s_\omega)\xi_\omega(\mu)\, d\mu. \tag{2.5}$$

Proof: By definition, $V^*(\omega) \geq V^\pi(\omega)$ for all ω, for any policy π. The lower bound follows trivially, since

$$V^{\pi^*(\hat{\mu}_\omega)}(\omega) \triangleq \int V_\mu^{\pi^*(\hat{\mu}_\omega)}(s_\omega)\xi_\omega(\mu)\, d\mu. \tag{2.6}$$

The upper bound is derived as follows. First note that for any function f, $\max_x \int f(x,u)\, du \leq \int \max_x f(x,u)\, du$. Then, we remark that:

$$V^*(\omega) = \max_\pi \int V_\mu^\pi(s_\omega)\xi_\omega(\mu)\, d\mu \tag{2.7a}$$

$$\leq \int \max_\pi V_\mu^\pi(s_\omega)\xi_\omega(\mu)\, d\mu \tag{2.7b}$$

$$= \int V_\mu^{\pi^*(\mu)}(s_\omega)\xi_\omega(\mu)\, d\mu. \tag{2.7c}$$

■

In POMDPs, a trivial lower bound can be obtained by calculating the value of the blind policy [Hau00, SS05], which always takes the same action. Our lower bound is in fact the BAMDP analogue of the value of the blind policy in POMDPs. This is because for any *fixed policy* π, it holds trivially that $V^\pi(\omega) \leq V^*(\omega)$. In our case, we have made this lower bound tighter by considering $\pi^*(\bar{\mu}_\omega)$, the policy that is greedy with respect to the current mean estimate.

The upper bound itself is analogous to the POMDP value function bound given in Theorem 9 of [Hau00]. However, while the lower bound is easy to compute in our case, the upper bound can only be approximated via Monte Carlo sampling with some probability, *unless $\mathcal{M}$ is finite*.

2.2.1 Leaf node lower bound

The lower bound can be calculated by performing value iteration in the mean MDP. This is because, for any policy π and belief ξ, $\int V_\mu^\pi(s)\xi(s)\, d\mu$ can be written as

$$\int \left\{ \mathbf{E}[r|s,\mu,\pi] + \gamma \sum_{s'} \mu(s'|s,\pi(s)) V_\mu^\pi(s') \right\} \xi(\mu) d\mu =$$

$$= \sum_a \pi(a|s) \left\{ \mathbf{E}[r|s,a,\bar{\mu}_\xi]\} + \gamma \sum_{s'} \bar{\mu}_\xi(s'|s,a) V_{\bar{\mu}_\xi}^\pi(s'), \right\}.$$

where $\bar{\mu}_\xi$ is the *mean MDP* for belief ξ. If the beliefs ξ can be expressed in closed form, it is easy to calculate the mean transition distribution and the mean reward from ξ. For discrete state spaces, transitions can be expressed as multinomial distributions, to which the Dirichlet density is a conjugate prior. In that case, for Dirichlet parameters $\{\psi_i^{j,a}(\xi) : i, j \in \mathcal{S}, a \in \mathcal{A}\}$, we have $\bar{\mu}_\xi(s'|s,a) = \psi_{s'}^{s,a}(\xi) / \sum_{i\in\mathcal{S}} \psi_i^{s,a}(\xi)$. Similarly, for Bernoulli rewards, the corresponding mean model arising from the beta prior with parameters $\{\alpha^{s,a}(\xi), \beta^{s,a}(\xi) : s \in \mathcal{S}, a \in \mathcal{A}\}$ is $\mathbf{E}[r|s,a,\bar{\mu}_\xi] = \alpha^{s,a}(\xi)/(\alpha^{s,a}(\xi)+\beta^{s,a}(\xi))$. Then the value function of the mean model, and consequently, a lower bound on the optimal value function, can be found with standard value iteration.

2.2.2 Leaf node upper bound with high probability

In general, (2.7b) cannot be expressed in closed form. However, the integral can be approximated via Monte Carlo sampling.

Let the leaf node which we wish to expand be ω. Then, we can obtain c MDP samples from the belief at ω: $\mu_1, \dots, \mu_c \sim \xi_\omega(\mu)$. For each μ_k we can derive the optimal policy $\pi^*(\mu_k)$ and estimate its value function $v_k^* \triangleq V_\mu^{\pi^*(\mu_k)} \equiv V_\mu^*$. We may then average these samples to obtain

$$\hat{v}_c^*(\omega) \triangleq \frac{1}{c} \sum_{k=1}^{c} \tilde{v}_k^*(s_\omega). \tag{2.8}$$

Let $\bar{v}^*(\omega) = \int_{\mathcal{M}} \xi_\omega(\mu) V_\mu^*(s_\omega)\, d\mu$. It holds that $\lim_{c\to\infty}[\hat{v}_c] = \bar{v}^*(\omega)$ and that $\mathbf{E}[\hat{v}_c] = \bar{v}^*(\omega)$. Due to the latter, we can apply a Hoeffding inequality

$$\mathbf{P}\left(|\hat{v}_c^*(\omega) - \bar{v}^*(\omega)| > \epsilon\right) < 2\exp\left(-\frac{2c\epsilon^2}{(V_{\max} - V_{\min})^2}\right), \tag{2.9}$$

thus bounding the error within which we estimate the upper bound. For $r_t \in [0,1]$ and discount factor γ, note that $V_{\max} - V_{\min} \leq 1/(1-\gamma)$.

2.3 Bounds on parent nodes

We can obtain upper and lower bounds on the value of every action $a \in \mathcal{A}$, at any part of the tree, by iterating over Ω_t, the set of possible outcomes following ω_t:

$$\bar{v}(\omega_t, a) = \sum_{i=1}^{|\Omega_t|} \mathbf{P}(\omega_t^i \mid \omega_t, a)\left[r_t^i + \gamma \bar{v}(\omega_t^i)\right] \tag{2.10}$$

$$\hat{v}^*(\omega_t, a) = \sum_{i=1}^{|\Omega_t|} \mathbf{P}(\omega_t^i \mid \omega_t, a)\left[r_t^i + \gamma \hat{v}^*(\omega_t^i)\right], \tag{2.11}$$

where the probabilities are implicitly conditional on the beliefs at each ω_t. For every node, we can calculate an upper and lower bound on the value of all actions. Obviously, if at the root node ω_t, there exists some $\hat{a}_t^*$ such that $\bar{v}(\omega_t, \hat{a}_t^*) \geq \hat{v}^* * (\omega_t, a)$ for all a, then $\hat{a}_t^*$ is unambiguously the optimal action.

3 PAC bounds for stochastic branch and bound methods

The search is on trees which arise in the context of planning under uncertainty in MDPs using the Bayesian BAMDP framework, which is described in Section 2. Because of this, the branches alternate between action selection and random outcomes. Each of the decision nodes has upper and lower bounds on the value function, which are described in Section 2.2. However, the upper bound must be estimated via Monte Carlo sampling, something which necessitates the use of stochastic branch and bound techniques for the seasrch.

We compare algorithms which utilise various combinations of stochastic and exact bounds for the value of each node. One of the main assumptions is that there is a uniform bound on the convergence speed. Indeed, if such a bound does not hold then we cannot hope to find the optimal branch in finite time. Given a measure of separation between the optimal branch and sub-optimal branches, we can then obtain complexity bounds.

3.1 Notation

We assume that the tree has a branching factor at most ϕ and unbounded depth. Each branch b defines a sequence of observations $\{x_i(b)\}_{i=1}^{\infty}$, and we use $S^b(k) \triangleq \{b' : x_i(b') = x_i(b)\ \forall i \leq k\}$ to denote the set of branches forming a particular continuation of a partial expansion of a branch b to a depth k. We furthermore define the following measure on branches $\|b\| \triangleq \sum_{t=1}^{\infty} \phi^{-t}\, \mathbb{I}\{x_t(b)\}$. Finally, the value of a branch b will be denoted by V^b and the value of the optimal branch b^* will be denoted by V^*. We shall denote the difference between the optimal branch and the highest suboptimal branch's lower and upper bounds with Δ_L, Δ_U, while the difference between their actual values will be denoted by Δ. A hat will signify estimated values.

3.2 Assumptions

We assume that we have upper and lower bounds $V_U^b(k), V_L^b(k)$ on the value of any branch such that

Assumption 3.1 (Bound monotonicity)

$$V_L^b(k) \leq V_L^b(k+1) \leq V^b, \quad V_U^b(k) \geq V_U^b(k+1) \geq V^b. \tag{3.1}$$

In order for any forward search algorithm to work asymptotically, it is necessary that the sequence of bounds converges to the true value in the limit.

Assumption 3.2 (Asymptotic convergence) *For any $\epsilon > 0$, $\exists k_0 : \forall k > k_0$*

$$V_L^b(k) \geq V^b - \epsilon, \qquad V_U^b(k) \leq V^b + \epsilon. \tag{3.2}$$

Remark 3.1 *For reinforcement learning we have $\epsilon_k \leq \gamma^k/(1-\gamma)$ thus, for any $\epsilon > 0$,*

$$k_0 = \log_\gamma \epsilon + \log_\gamma (1-\gamma).$$

However, we also need specific convergence rates in order to obtain complexity bounds, therefore we shall assume the following.

Assumption 3.3 (Uniform linear convergence) *There exists $\gamma \in (0,1)$ and $\beta > 0$ such that for any branch b, and any depth k,*

$$V^b - V_L^b(k) \leq \beta\gamma^k, \qquad V_U^b(k) - V^b \leq \beta\gamma^k. \tag{3.3}$$

It is easy to see that this assumption holds for reinforcement learning with $\beta = 1/(1-\gamma)$.

Remark 3.2 (Hölder continuity) *A direct result of Assumption 3.3 is that the value function of branches satisfies the following Hölder continuity condition:*

$$\|V^b - V^{b'}\| \leq 2\beta \|b - b'\|^{\log_\phi \gamma}. \tag{3.4}$$

Now we can proceed with the analysis. We shall initially compare two variants of the case where we expand the tree up to a fixed depth. We then compare two different stochastic branch and bound algorithms, which maintain unbalanced trees.

3.3 Flat oracle search

If we have exact lower bounds available, then one possible strategy is to expand all branches to a fixed depth and then select the branch with the highest lower bound. Algorithm 2 describes a method for achieving regret at most ϵ using this scheme.

Algorithm 2 Flat oracle search

Expand all branches until depth $k = \log_\gamma \epsilon/\beta$ or $\hat{\Delta}_L > \beta\gamma^k - \epsilon$.
Select the root branch $\hat{b}^* = \arg\max_b V_L^b(k)$.

Lemma 3 (Flat oracle complexity) *Running Algorithm 2 in a tree with branching factor ϕ, and $\gamma \in (0,1)$, has regret at most ϵ with complexity of order*

$$\mathcal{O}\left([\epsilon(1-\gamma)]^{\log_\gamma \phi}\right)$$

Note that the dependence on γ can make this bound grow very fast, since as soon as $\gamma \geq 1/\sqrt{\phi}$, the complexity becomes worse than ϵ^{-2}.

3.4 Flat stochastic search

Next, we tackle the case where we only have a stochastic lower bound on the value of each node. This may be the case when the lower bound is an integral which can only be calculated with a monte-carlo approximation for example. The following algorithm expands the tree to a fixed depth and then takes multiple samples from each leaf node. It is only useful when γ is known in advance, something which is true in reinforcement learning problems.

Algorithm 3 Flat stochastic search

1: Expand all branches until depth $k = \lceil \log_\gamma \epsilon/2\beta \rceil$.
2: **for** $b = 1, \ldots, \phi^k$ **do**
3: Calculate $\hat{V}_L^b = \frac{1}{m}\sum_{i=1}^m \tilde{v}_i^b$, with

$$m = 2\left\lceil \log_\gamma[\epsilon(1-\gamma)/2] \right\rceil \cdot \log \phi$$

4: **end for**
5: Select $b : \hat{V}_L^b > \hat{V}_L^{b'} \forall b' \neq b$.

After the tree is expanded to depth k, we taken m samples from each leaf node's bound estimate, with the guarantee that each sample is in the interval $[V^b - \beta\gamma^k, V^b]$.

Lemma 4 (Stochastic lower bound search complexity) *The number of samples that Algorithm 3 requires to bound the expected regret by ϵ is*

$$\mathcal{O}\left(\epsilon^{\log_\gamma\left(\phi\gamma^{-2}\right)}\left(\log_\gamma[\epsilon(1-\gamma)/2] \cdot \log \phi\right)\right)$$

3.5 Stage-wise stochastic branch and bound

The general idea for stochastic branch and bound is to expand only branches with an upper bound that is higher than the highest lower bound. Two main algorithms types are possible. The first one is stage-wise branch and bound, which at every stage discards a set of branches. The second one is continual branch and bound, which, while it never discards a branch, it expands the most promising branches first.

Algorithm 4 Stage-wise branch and bound

1: **for** $k = 1, 2, \ldots$ **do**
2: Take $m_{k,b}$ samples from each branch.
3: Select the root branch $\hat{b}^* = \arg\max_b V_L^b(k)$.
4: **end for**

In the stage-wise branch and bound search, we only keep one branch, $\hat{b}^* \triangleq \arg\max_b \hat{V}_L^b$ such that

$$\mathbf{P}\left(\exists b : \hat{V}_L^{\hat{b}^*} < \hat{V}_L^b - \epsilon/2\right) < \epsilon/2.$$

This ensures that the total regret is smaller than ϵ. A simple algorithm takes

$$\mathbf{P}\left(\hat{V} - V > \epsilon/2\right) = \mathbf{P}\left((\hat{V} - V)/\beta\gamma^k > \epsilon/2\beta\gamma^k\right) < \exp\left(-\frac{m\epsilon^2}{2\beta\gamma^{2k}}\right). \quad (3.5)$$

So if we take

$$m = \left\lceil \log(2\phi/\epsilon) 2\beta\gamma^{2k}\epsilon^2 \right\rceil$$

samples then

$$\mathbf{P}\left(\hat{V} - V > \epsilon/2\right) < \epsilon/2\phi$$

and since there are at most ϕ branches, by a union bound we obtain an error probability of $\epsilon/2$.

We can now extend this approach to a multi-stage algorithm that achieves regret $\epsilon_k \triangleq \lambda^k$ at each stage.

$$\sum_{k=1}^K \epsilon_k = \sum_{k=1}^K \lambda^k \leq \lambda/(1-\lambda),$$

so $\lambda \leq \epsilon/(1+\epsilon)$.

Lemma 5 *The total sample complexity of the staged algorithm is*

$$\tilde{\mathcal{O}}\left(\phi K + \beta\phi\left[1 + \epsilon^2 \log \frac{1}{\epsilon}\right]\right) \quad (3.6)$$

. [CD: TODO - this seems wrong!]

3.6 Stochastic branch and bound

The stochastic branch and bound algorithm [NPR98] is conceptually similar to BAST [CM07]. At each stage, this algorithm takes an additional sample at each leaf node, to improve their upper bound estimates. It then expands the node with the highest mean upper bound.

We need bounds for the number of samples required until we discover that a particular branch is sub-optimal. We first

Algorithm 5 stochastic-branch-and-bound

for $t = 1, 2, \ldots$ **do**
  Let $\mathcal{L}_t$ be the set of leaf nodes.
  for $b \in \mathcal{L}_t$ **do**
    $\hat{V}_U^b = \frac{1}{m_b} \sum_{i=1}^{m_b} \tilde{v}_i^b$
  end for
  $S_{k+1} = \mathbf{E}(\arg\max_A U(A))$
  Select $b : \hat{V}_L^b > \hat{V}_L^{b'} \; \forall b' \neq b$.
end for

need a lemma that bounds the number of times we shall sample a suboptimal node until we discover it is sub-optimal and similarly, the number of times we shall sample the optimal node until we discover it's not suboptimal. The two lemmata cover the case where we sample nodes in the optimal branch without expanding them and the case where we continuously expand nodes in a sub-optimal branch.

The following lemma can be used to bound the number of times that an optimal branch's leaf node will be sampled without being expanded.

Lemma 6 *If N is the (random) number of times we must sample a random variable $V \in [0, \beta]$ until its empirical mean $\hat{V}(j) < \bar{V} + \Delta$, then*

$$\mathbf{E}[N] \leq 1 + \beta^2 \Delta^{-2} \tag{3.7}$$

$$\mathbf{P}[N > n] \leq \exp\left(-2\beta^{-2} n^2 \Delta^2\right). \tag{3.8}$$

The same inequalities holds for the event $\hat{V}(j) > \bar{V} - \Delta$.

We now examine the converse: bounding the number of times that a suboptimal branch will be expanded.

Lemma 7 *If b is a branch with $V^b = V^* - \Delta$, then it will be expanded at least*

$$N > \phi^{\frac{\beta}{\Delta(1-\gamma)}} \tag{3.9}$$

times in the worst case. Subsequently,

$$\mathbf{P}(N > n) < \exp\left(\frac{-2\epsilon^2(1-\gamma^2)m^2}{\beta^2(1-\gamma^{2(m+1)})}\right). \tag{3.10}$$

3.7 Iterative stochastic branch and bound

The standard BAST algorithm [CM07] does not fit perfectly. It is a max search, where only one node is possible as a followup given an action. Let us consider applying it directly to observation sets (!).

So we also introduce a variant of BAST...

Plan: use the Δ-dependent bound of BAST?

4 Conclusion

As was mentioned in the previous section, there are some difficulties with planning Due to the fact that the Hoeff for the simple stochastic branch and bound algorithm degenerate results suggest that another algorithm, which would employ sparse sampling, may be more beneficial.

As stated in 2.2, the upper bound must be calculated via sampling unless $\mathcal{M}$ is finite. So, this would be easier for simpler problems.

Algorithm 6 Iterative stochastic branch and bound

for $i \in \mathcal{L}_t$ **do**
  $\hat{V}_U(i) = \frac{1}{1+n(i)} \left[n(i)\hat{V}_U(i) + n(i)\tilde{v}(i) \right]$
end for
Use BACKWARDSINDUCTION$(t, \nu, \mathcal{L}_t, \hat{V}_U)$ to obtain $\hat{V}_U$ for all nodes.
Set i_0 to root.
for $d = 1, \ldots$ **do**
  Select action a or transition to go from i_{d-1} to i_d.
  $i_d = \arg\max_a \sum_{j \in \mathcal{N}_{d+1}} \omega_i(j|a) \hat{V}_U(j)$
end for

A Proofs

Proof:[Lemma 3] For any b' with $V_L^{b'} < V_L^b$, it holds that

$$V^{b'} \leq V_L^{b'} + \beta\gamma^k < V_L^b + \beta\gamma^k \leq V^b + \beta\gamma^k. \tag{A.1}$$

This holds for $b = \hat{b}^*$. Thus, in the worst case, the regret that we suffer if there exists some $b' : V^{b'} > V^{\hat{b}^*}$ is

$$\epsilon = V^{b'} - V^{\hat{b}^*} < \beta\gamma^k, \tag{A.2}$$

The number of expansions required to reach depth k in all branches is

$$n = \sum_{t=1}^{k} \phi^k < \phi^{k+1}. \tag{A.3}$$

Thus, $k = \frac{\log(\epsilon/\beta)}{\log\gamma}$ and

$$n <= \phi^{\frac{\log(\epsilon/\beta)}{\log\gamma}} \phi = \phi^{\log_\gamma \epsilon} \phi^{1 - log_\gamma \beta} \tag{A.4}$$

$$= \epsilon^{\log_\gamma \phi} \phi^{1 - log_\gamma \beta}. \tag{A.5}$$

■

Proof:[Lemma 4] The total number of samples is km, the number of leaf nodes times the number of samplesa t each leaf node. The search is until depth

$$k = \left\lceil \log_\gamma \epsilon/2\beta \right\rceil \leq 1 + \log_\gamma \epsilon/2\beta \tag{A.6}$$

and the number of samples is

$$m = 2\log_\gamma(\epsilon/2\beta)\log\phi. \tag{A.7}$$

The complexity follows trivially. Now we must prove that this bounds the expected regret with ϵ. Note that $\beta\gamma^k < \epsilon/2$, so for all branches b:

$$\hat{V}_L^b - V^b < \epsilon/2. \tag{A.8}$$

The expected regret can now be written as

$$\mathbf{E}\, R \leq \frac{\epsilon}{2} + \mathbf{E}[R|\hat{V}_L^{\hat{b}^*} < \hat{V}_L^{b^*} + \epsilon/4]\, \mathbf{P}(\hat{V}_L^{\hat{b}^*} < \hat{V}_L^{b^*} + \epsilon/4) \tag{A.9}$$

$$+ \mathbf{E}[R|\hat{V}_L^{\hat{b}^*} \geq \hat{V}_L^{b^*} + \epsilon/4]\, \mathbf{P}(\hat{V}_L^{\hat{b}^*} \geq \hat{V}_L^{b^*} + \epsilon/4). \tag{A.10}$$

From the Hoeffding bound (C.1)

$$\mathbf{P}(\hat{V}_L - V_L > \epsilon/4) < \exp\left(-\frac{1}{8} m \beta^{-2} \gamma^{-2k} \epsilon^2\right)$$

and with a union bound the total error probability is bounded by $\phi^k \exp\left(-\frac{1}{8} m\beta^{-2}\gamma^{-2k}\epsilon^2\right)$. If our estimates are within $\epsilon/4$ then the sample regret is bounded by $\epsilon/4$, while the other terms are trivially bounded by 1, to obtain

$$\mathbf{E}\, R \leq \frac{\epsilon}{2} + \left\{ \phi^k \exp\left(-\frac{1}{8} m\beta^{-2}\gamma^{-2k}\epsilon^2\right) + \frac{\epsilon}{4}. \right\} \quad \text{(A.11)}$$

Substituting m and k, we obtain the stated result. ■

Proof:[Lemma 5] The sample complexity of this algorithm is $m\phi \leq (1 + \log(2\phi/\epsilon_k)2\beta\gamma^{2k}\epsilon_k^2)\phi$ at each stage.

$$\begin{aligned}
&\phi \sum_{k=1}^{K} \left(1 + \log(2\phi\lambda^{-k})2\beta\gamma^{2k}\lambda^{2k}\right) \\
&= \phi K + \phi \sum_{k=1}^{K} (\log(2\phi) - k\log(\lambda))2\beta\gamma^{2k}\lambda^{2k} \\
&< \phi K + 2\beta\phi \sum_{k=1}^{K} \left[\log(2\phi) + k\log\left(\frac{1+\epsilon}{\epsilon}\right)\right] \left(\frac{\gamma\epsilon}{1+\epsilon}\right)^{2k} \\
&< \phi K + 2\beta\phi \frac{1}{1 - [\gamma\epsilon/(1+\epsilon)]^2} + 2\beta\phi \log\left(\frac{1+\epsilon}{\epsilon}\right) \frac{\lambda^2}{(1-\lambda^2)^2},
\end{aligned}$$

as

$$\sum_k k\lambda^{2k} = \lambda^2 \cdot \partial/\partial(\lambda^2) \sum_k \lambda^{2k} = \frac{\lambda^2}{(1-\lambda^2)^2}.$$

Substituting λ completes the proof.

$$\frac{\lambda}{1-\lambda^2} = \frac{\lambda}{(1-\lambda)(1+\lambda)} = \frac{\epsilon/(1+\epsilon)}{[1/(1+\epsilon)][(1+2\epsilon)/(1+\epsilon)]} \quad \text{(A.12)}$$

$$= \frac{\epsilon(1+\epsilon)}{1+2\epsilon} = \frac{\epsilon^2+\epsilon}{1+2\epsilon} \quad \text{(A.13)}$$

■

Proof:[Lemma 6]

$$\mathbf{E}[N] = \sum_{n=1}^{\infty} n \prod_{j=1}^{n-1} \mathbf{P}(\hat{V}(j) \geq V + \epsilon)\, \mathbf{P}(\hat{V}(n) < V + \epsilon) \quad \text{(A.14)}$$

$$\leq \sum_{n=1}^{\infty} n \prod_{j=1}^{n-1} \exp\left(-2j\beta^{-2}\epsilon^2\right) \cdot 1 \quad \text{(A.15)}$$

$$= \sum_{n=1}^{\infty} n \exp\left(-2\beta^{-2}\epsilon^2 \sum_{j=1}^{n-1} j\right) \quad \text{(A.16)}$$

$$= \sum_{n=1}^{\infty} n \exp\left(-\beta^{-2}\epsilon^2 n(n+1)\right) \quad \text{(A.17)}$$

Let us now set $\rho = \exp(-\beta^{-2}\epsilon^2)$. Observe that $n\rho^{n(n+1)} < n\rho^{n^2}$, since $\rho < 1$. Then, note that $\int n\rho^{n^2}\, dn = \mathcal{O}\left(\frac{\rho^{n^2}}{2\log\rho}\right)$. So we can bound the sum by

$$\sum_{n=1}^{\infty} n\rho^{n(n+1)} < 1 + \left[\frac{\rho^{n^2}}{2\log\rho}\right]_1^{\infty} \quad \text{(A.18)}$$

$$= 1 + \frac{\exp(-\beta^{-2}\epsilon^2)}{2\beta^{-2}\epsilon^2} < 1 + \left(\frac{\beta}{\epsilon}\right)^2. \quad \text{(A.19)}$$

The second inequality can be proven as follows:

$$\mathbf{P}(N > n) = \mathbf{P}\left(\bigwedge_{k=1}^{n} \hat{V}(k) > V + \epsilon\right) \quad \text{(A.20)}$$

$$< \prod_{k=1}^{n} \exp\left(-2k\beta^{-2}\epsilon^2\right) = \exp\left(-\beta^{-2}\epsilon^2 n(n+1)\right) \quad \text{(A.21)}$$

$$< \exp\left(-n^2\beta^{-2}\epsilon^2\right). \quad \text{(A.22)}$$

This completes the proof for the first case. The second case is symmetrical. ■

Proof:[Lemma 7] The proof also relies on Hoeffding bounds. It is important to note that there is no guarantee that the probability of reaching any leaf node is not aribtrarily close to 0. Thus, it is possible that there is only one sub-branch with non-zero probability. Thus, it is essential to re-use the samples that were obtained at previous expansions.

Definition 8 *The upper bound of a branch is*

$$V_U(\omega) = \sum_{\omega'} \mathbf{P}(\omega_{t+1} = \omega' | \omega_t = \omega) \left[R(\omega', \omega) + V_U(\omega')\right]$$

Definition 9 *The upper bound of a branch obtained by looking at nodes up to depth k is*

$$V_U^{b(\omega)}(k) = \sum_{\omega'} \mathbf{P}(\omega'|\omega) \left[R(\omega', \omega) + V_U^{b(\omega')}(k-1)\right].$$

Let us denote the corresponding empirical upper bound obtained by sampling the leaf nodes with $\hat{V}_U^{b(\omega)}(k)$. If we wish to bound the probability that the branches following ω will not be expanded more than a finite number of times, then we must derive a concentration inequality.

Remark A.1 *The worst case is that all the expanded branches, apart from a single branch, will have zero probability, in which case the Hoeffding bound that applies is simply* (C.4).

The proof is now easy to construct. Since $V_U^b(k) \geq V^b + \beta\gamma^k$, we can write the following for the upper bound averaged over stages $1, \ldots, m$:

$$\frac{1}{m} \sum_{k=1}^{m} V_U^b(k) \leq V^b + \frac{\beta(1-\gamma^{m+1})}{m(1-\gamma)}.$$

It immediately follows that:

$$\mathbf{P}\left(\hat{V}^b(k) > V^b + \Delta\right) < \mathbf{P}\left(\frac{1}{m}\sum_{k=1}^{m} \hat{V}_U^b(k) > V_U^b(k) + \Delta - h\right)$$

where $h \triangleq \frac{\beta(1-\gamma^{m+1})}{m(1-\gamma)}$. Now, Let $\epsilon \triangleq \Delta - h$. If $\Delta > h$, then $\epsilon > 0$ and we can use a Hoeffding bound to obtain

$$\mathbf{P}\left(\frac{1}{m}\sum_{k=1}^{m} V_U^b(k) > V^b + \Delta\right) < \exp\left(-2\frac{m^2\epsilon^2}{\sum_k(\beta\gamma^k)^2}\right).$$

We can now obtain bounds in both expectation and high probability for the number of expansions, similarly to Lemma 6. Firstly, observe that

$$\mathbf{P}(N > n) < \prod_{m=1}^{n} \exp\left(-2\frac{m^2\epsilon^2}{\sum_k(\beta\gamma^k)^2}\right) \quad \text{(A.23)}$$

$$= \exp\left(\frac{-2\epsilon^2(1-\gamma^2)}{\beta^2} \cdot \frac{m^2}{1-\gamma^{2(m+1)}}\right) \quad \text{(A.24)}$$

If $\Delta \leq h$, then the probability that the branch won't be expanded cannot be bounded. Thus, the branch must be expanded up to approximately $\frac{\beta}{\Delta(1-\gamma)}$. Thus, the number of expansions performed on a suboptimal branch with $V^b = V^* - \Delta$ is bounded from below by $n = \phi^{\frac{\beta}{\Delta(1-\gamma)}}$. ■

B Miscellaneous remarks

Remark B.1 *If we expand the tree T times and at each expansion we take a sample from every node, then the total number of samples taken is bounded by*

$$T(T+1)(\phi-1)/2. \quad \text{(B.1)}$$

Proof: At the k-th expansion, we create at most ϕ leaf nodes and remove one. Thus, the number of nodes t time $k+1$ is $n_{k+1} = n_k + \phi - 1$. If $n_0 = 1$, then it is easy to see that

$$n_k = k(\phi - 1).$$

At each step we take n_k samples, which means that the total number of samples is

$$\sum_{k=1}^{T} n_k = \sum_{k=1}^{T} k(\phi-1) = T(T+1)(\phi-1)/2.$$

■

Remark B.2 *If the oldest leaf node is always expanded, then all leaf nodes will be at depth* $\log_\phi(T)$ *or* $\log_\phi(T) - 1$. *[CD: the ϕ base is a rough guess.]*

Remark B.3 *Since there are n_k leaf nodes at stage k, the oldest leaf node will have at least $n_k - 1$ accumulated samples.*

C Hoeffding bounds for weighted averages

Standard Hoeffding bounds can of course be derived also for weighted averages. Let us first remember the standard Hoeffding inequality.

Lemma 10 (Hoeffding inequality) *Let $\hat{x}_n \triangleq \frac{1}{n}\sum_{i=1}^{n} x_i$, with $x_i \in [b_i, b_i + h_i]$ drawn from some arbitrary distribution f_i. Then, if $\bar{x} \triangleq \mathbf{E}[x]$, it holds for all $\epsilon \geq 0$ that*

$$\mathbf{P}\left(\hat{x}_n \geq \bar{x} + \epsilon\right) \leq \exp\left(-\frac{2n^2\epsilon^2}{\sum_{i=1}^{n} h_i^2}\right). \quad \text{(C.1)}$$

In our case, we do not have a simple average, but a weighted sum over our samples,

$$\hat{x}'_n \triangleq \sum_{i=1}^{n} w_i x'_i, \quad \sum_{i=1}^{n} w_i = 1. \quad \text{(C.2)}$$

If we set $v_i \triangleq n w_i$, then we can write the above as $\frac{1}{n}\sum_{i=1}^{n} v_i x'_i$. So, if we let $x_i = v_i x'_i$ and assume that $x'_i \in [b, b+h]$, then $x_i \in [v_i b + v_i(b+h)]$. Substituting into (C.1) results in

$$\mathbf{P}\left(\hat{x}_n \geq \bar{x} + \epsilon\right) \leq \exp\left(-\frac{2\epsilon^2}{h^2\sum_{i=1}^{n} w_i^2}\right). \quad \text{(C.3)}$$

Furthermore, note that

$$\mathbf{P}\left(\hat{x}_n \geq \bar{x} + \epsilon\right) < \exp\left(-\frac{2\epsilon^2}{h^2}\right), \quad \text{(C.4)}$$

since $w_i^2 \leq w_i$ for all i, as $w_i \in [0,1]$. Thus $\sum_i w_i^2 \leq \sum_i w_i = 1$. Note that $\sum_i w_i^2 = 1$ iff $w_j = 1$ for some j.

References

[Aue02] P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *J. Machine Learning Research*, 3(Nov):397–422, 2002. A preliminary version has appeared in *Proc. of the 41th Annual Symposium on Foundations of Computer Science*.

[BK59] Richard Bellman and Robert Kalaba. A mathematical theory of adaptive control processes. *Proceedings of the National Academy of Sciences of the United States of America*, 45(8):1288–1290, 1959.

[CM07] Pierre-Arnaud Coquelin and Rémi Munos. Bandit algorithms for tree search. In *UAI '07, Proceedings of the 23rd Conference in Uncertainty in Artificial Intelligence, Vancouver, BC Canada*, 2007.

[DB97] Michael O. Duff and Andrew G. Barto. Local bandit approximation for optimal learning problems. In Michael C. Mozer, Michael I. Jordan, and Thomas Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, page 1019. The MIT Press, 1997.

[DFR98] Richard Dearden, Nir Friedman, and Stuart J. Russell. Bayesian Q-learning. In *AAAI/IAAI*, pages 761–768, 1998.

[Dim06] Christos Dimitrakakis. Nearly optimal exploration-exploitation decision thresholds. In *Int. Conf. on Artificial Neural Networks (ICANN)*, 2006. IDIAP-RR 06-12.

[Dim08] Christos Dimitrakakis. Tree exploration for Bayesian RL exploration. In *Proceedings of*

the international conference on computational intelligence for modelling, control and automation, (CIMCA 08), 2008.

[Duf02] Michael O'Gordon Duff. *Optimal Learning Computational Procedures for Bayes-adaptive Markov Decision Processes*. PhD thesis, University of Massachusetts at Amherst, 2002.

[GS07] Sylvain Gelly and David Silver. Combining online and offline knowledge in UCT. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 273–280, New York, NY, USA, 2007. ACM Press.

[Hau00] Milos Hauskrecht. Value-function approximations for partially observable markov decision processes. *Journal of Artificial Intelligence Resesarch*, pages 33–94, Aug 2000.

[HDFJ08] Matthew Hoffman, Arnaud Doucet, Nando De Freitas, and Ajay Jasra. Bayesian policy learning with trans-dimensional mcmc. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*. MIT Press, Cambridge, MA, 2008.

[HM08] Jean-François Hren and Rémi Munos. Optimistic planning of deterministic systems. In Sertan Girgin, Manuel Loth, Rémi Munos, Philippe Preux, and Daniil Ryabko, editors, *EWRL*, volume 5323 of *Lecture Notes in Computer Science*, pages 151–164. Springer, 2008.

[KS06] Levente Kocsis and Csaba Szepesvári. Bandit based Monte-Carlo planning. In *Proceedings of ECML-2006*, 2006.

[KSHdM01] A.J. Kleywegt, A. Shapiro, and T. Homem-de Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12(2):479–502, 2001.

[KWK94] P. Kall, S.W. Wallace, and P. Kall. *Stochastic programming*. Wiley New York, 1994.

[NPR98] Vladimir I. Norkin, Georg Ch. Pflug, and Andrzej Ruszczyski. A branch and bound method for stochastic global optimizatio n. *Mathematical Programming*, 83(1):425–450, January 1998.

[PVHR06] P. Poupart, N. Vlassis, J. Hoey, and K. Regan. An analytic solution to discrete Bayesian reinforcement learning. *Proceedings of the 23rd international conference on Machine learning*, pages 697–704, 2006.

[RCdP08] Stephane Ross, Brahim Chaib-draa, and Joelle Pineau. Bayes-adaptive POMDPs. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, Cambridge, MA, 2008. MIT Press.

[RPCd08] Stephane Ross, Joelle Pineau, and Brahim Chaib-draa. Theoretical analysis of heuristic search methods for online POMDPs. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*. MIT Press, Cambridge, MA, 2008.

[RPPCd08] Stéphane Ross, Joelle Pineau, Sébastien Paquet, and Brahim Chaib-draa. Online planning algorithms for POMDPs. *Journal of Artificial Intelligence Resesarch*, 32:663–704, July 2008.

[SN05] A. Shapiro and A. Nemirovski. On Complexity of Stochastic Programming Problems. *APPLIED OPTIMIZATION*, 99:111, 2005.

[SS05] T. Smith and R. Simmons. Point-based POMDP algorithms: Improved analysis and implementation. In *Proceedigns of the 21st Conference on Uncertainty in Artificial Intelligence (UAI-05)*, pages 542–547, 2005.

[Tho33] W.R. Thompson. On the Likelihood that One Unknown Probability Exceeds Another in View of the Evidence of two Samples. *Biometrika*, 25(3-4):285–294, 1933.

[THS06] Marc Toussaint, Stefan Harmelign, and Amos Storkey. Probabilistic inference for solving (PO)MDPs, 2006.

[WLBS05] Tao Wang, Daniel Lizotte, Michael Bowling, and Dale Schuurmans. Bayesian sparse sampling for on-line reward optimization. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 956–963, New York, NY, USA, 2005. ACM.