Thompson Sampling Based Monte-Carlo Planning in POMDPs

Aijun Bai
Univ. of Sci. & Tech. of China
baj@mail.ustc.edu.cn

Feng Wu
Univ. of Southampton
fw6e11@ecs.soton.ac.uk

Zongzhang Zhang
National Univ. of Singapore
zhangzz@comp.nus.edu.sg

Xiaoping Chen
Univ. of Sci. & Tech. of China
xpchen@ustc.edu.cn

Abstract

Monte-Carlo tree search (MCTS) has been drawing great interest in recent years for planning under uncertainty. One of the key challenges is the trade-off between exploration and exploitation. To address this, we introduce a novel online planning algorithm for large POMDPs using Thompson sampling based MCTS that balances between cumulative and simple regrets. The proposed algorithm — *Dirichlet-Dirichlet-NormalGamma based Partially Observable Monte-Carlo Planning* (D^2NG-POMCP) — treats the accumulated reward of performing an action from a belief state in the MCTS search tree as a random variable following an unknown distribution with hidden parameters. Bayesian method is used to model and infer the posterior distribution of these parameters by choosing the conjugate prior in the form of a combination of two Dirichlet and one NormalGamma distributions. Thompson sampling is exploited to guide the action selection in the search tree. Experimental results confirmed that our algorithm outperforms the state-of-the-art approaches on several common benchmark problems.

Introduction

While *Markov decision process* (MDP) provides a rich mathematical framework for planning under uncertainty, *partially observable Markov decision process* (POMDP) is a generalization of MDPs in partially observable domains (Kaelbling, Littman, and Cassandra 1998). Given only a black-box simulator of the targeting domains, we consider the problem of online planning in POMDPs without explicitly knowing the underlying transition and observation functions in advance. *Monte-Carlo tree search* (MCTS) finds near-optimal policies in these settings by building a best-first search tree (Browne et al. 2012). The key idea of MCTS is to evaluate each information state (i.e., system state in MDPs or belief state in POMDPs) in the search tree by the statistics of sampled simulations starting from that state. MCTS has shown to be computationally efficient, anytime and highly parallelisable. To date, great success has been achieved by MCTS in variety of domains, such as game play (Winands, Bjornsson, and Saito 2010; Gelly and Silver 2011), planning under uncertainty (Kocsis and Szepesvári 2006; Silver and Veness 2010; Wu, Zilberstein, and Chen 2011), and Bayesian reinforcement learning (Guez, Silver, and Dayan 2012; Asmuth and Littman 2011).

When applying MCTS to online planning in (PO)MDPs, one of the fundamental challenge is the well-known *exploration vs. exploitation* dilemma: an agent must not only exploit by selecting the action that currently seems best, but should also keep exploring for possible higher future payoffs (Browne et al. 2012). UCB1, originally introduced in *multi-armed bandit* problems (MABs), is probably the most successful and widely-used algorithm to address this dilemma and it has been proved to be asymptotically optimal for MABs (Auer, Cesa-Bianchi, and Fischer 2002). Specifically, UCB1 selects the action that maximizes the UCB1 heuristic that defines an *upper confidence bound* for the underlying action values. On the other hand, Thompson sampling is perhaps one of the earliest heuristics that tackles this problem in a Bayesian fashion according to the principle of *randomized probability matching* (Thompson 1933). It selects actions stochastically based on their posterior probabilities of being optimal. Comparing to UCB1, the main advantage of Thompson sampling is that it allows more robust behavior in terms of convergence under a wide range of problem settings, e.g., MABs with multi-modal reward distributions.

In MABs, *cumulative regret* reduction aims to find actions that minimize the sum of differences between the expected reward of the best action and the obtained rewards of all action pulls, while *simple regret* is the difference between the best expected reward and the expected reward of the arm with the greatest sample mean after a number of pure exploration of action pulls (Bubeck, Munos, and Stoltz 2011). Apart from the fact that Thompson sampling empirically converges faster in terms of cumulative regret than the UCB1 approach (Chapelle and Li 2011), it has recently been proved that Thompson sampling achieves logarithmic cumulative regret which is asymptotically optimal for MABs (Kaufmann, Korda, and Munos 2012). However, in Monte-Carlo planning, it is usually the final action pull that collects a reward. Therefore, it is more reasonable to minimize the simple regret instead of the cumulative regret (Feldman and Domshlak 2012). Although the reduction rate of simple regret for Thompson sampling remains an open question, it is our observation that Thompson sampling empiri-

cally appears to yield lower simple regret than the state-of-the-art, particularly if the action space is large. A recently growing understanding is that it is better to balance between cumulative and simple regrets in MCTS (Tolpin and Shimony 2012). This motivated us to try Thompson sampling on MCTS for POMDPs as it seems to be a promising approach in handling both cumulative and simple regrets.

In this paper, we borrow the idea of Thompson sampling and propose *Dirichlet-Dirichlet-NormalGamma based Partially Observable Monte-Carlo Planning* (D^2NG-POMCP). In this algorithm, we represent the uncertainty of the immediate reward as a categorical distribution and the accumulated reward of performing a particular action from a belief state in the search tree as a convex combination of Normal mixtures. We perform statistical inference on the posterior distribution in Bayesian settings by choosing the conjugate prior in the form of a combination of two Dirichlet distributions and a NormalGamma distribution. At each decision node, Thompson sampling is used to select the action to be performed by Monte-Carlo simulation. We have tested D^2NG-POMCP in several instances of the common benchmark problems. Experimental results showed that our algorithm outperforms the state-of-the-art approaches for online planning in POMDPs. Furthermore, we show the convergence of our algorithm to confirm its technical soundness.

Background

In this section, we briefly review the MDP and POMDP models, the MAB problem, the MCTS framework, and the POMCP algorithm. Related work is also discussed.

MDPs, POMDPs and MABs

Formally, an MDP is defined as a 4-tuple $\langle S, A, T, R\rangle$, where S is the state space, A is the action space, $T(s'|s,a)$ is the probability of reaching state s' if action a is applied in state s, and $R(s,a)$ is the reward received. A *policy* is a mapping from states to actions, specifying which action should be taken in each state. For a given policy π, the expected total reward (also known as the value function) is defined as $V_\pi(s) = E[\sum_{t=0}^{\infty}\gamma^t R(s_t, \pi(s_t))]$, where $\gamma \in (0,1]$ is the discount factor, s_t is the state in time step t and $\pi(s_t)$ is the action selected by policy π in state s_t. The aim of solving an MDP is to find the *optimal* policy π^* that maximizes the value function for all states. The optimal value function, denoted by V^*, satisfies the famous Bellman equation (Bellman 1957):

$$V^*(s) = \max_{a\in A}\left\{R(s,a) + \gamma\sum_{s'}T(s'|s,a)V^*(s')\right\}.$$

When extending to partially observable environments, a POMDP is defined as a 6-tuple $\langle S, A, O, T, \Omega, R\rangle$, where S, A, T and R remain the same meanings as in MDPs, O is the observation space, and $\Omega(o|s,a)$ is the probability of observing o after having applied action a and reaching state s. A POMDP can be transformed into an MDP over *belief state* (or *belief* for short) space. A belief b is a sufficient statistic for the history of actions and observations, defined as a probability distribution over the state space, with $b(s)$ denoting the probability of being in state s. Given an initial belief b_0, a history of action-observation pairs $h = \{a_0, o_0, a_1, o_1, \dots\}$ uniquely determines the resulting belief, which can be obtained by recursively using a Bayesian filter $b' = \zeta(b,a,o)$:

$$b'(s') = \eta\Omega(o|s',a)\sum_{s\in S}T(s'|s,a)b(s),$$

where $\eta = 1/\Omega(o|b,a)$ is a normalizing constant with $\Omega(o|b,a) = \sum_{s'\in S}\Omega(o|s',a)\sum_{s\in S}T(s'|s,a)b(s)$. A policy is defined as a mapping from belief space (or history space) to actions. The goal is to find the optimal policy that maximizes the expected total reward. The Bellman equation for POMDP in terms of belief is:

$$V^*(b) = \max_{a\in A}\left\{r(b,a) + \gamma\sum_{o\in O}\Omega(o|b,a)V^*(\zeta(b,a,o))\right\},$$

where $r(b,a) = \sum_{s\in S}b(s)R(s,a)$ is the reward function in the transformed belief MDP. It is worth noticing that the transition function in the resulting MDP can be written as $T(b'|b,a) = \sum_{o\in O}\mathbf{1}[b' = \zeta(b,a,o)]\Omega(o|b,a)$, where $\mathbf{1}$ is the indicator function. Given an initial belief, the terms *belief* and *history* can be used interchangeably. In this paper, we present our main results in terms of belief for convenience, but implement our algorithm with respect to history instead.

MABs are usually seen as fundamental decision-making components of planning and learning problems. Intuitively, an MAB can be seen as an MDP with only one state s and a stochastic reward function $R(s,a) := X_a$, where X_a is a random variable following an unknown stationary distribution $f_{X_a}(x)$. At each time step t, one action a_t must be chosen and executed. A stochastic reward X_{a_t} is then observed. The goal of solving an MAB is usually to find a policy that minimizes the cumulative regret defined as $R_T = E[\sum_{t=1}^{T}(X_{a^*} - X_{a_t})]$, where a^* is the true best action. A simple regret defined for a pure exploration strategy after n times of action pulls is $r_n = E[X_{a^*} - X_{\bar{a}}]$, where $\bar{a} = \mathrm{argmax}_a\,\bar{X}_a$ is the action with maximal empirical mean of reward.

MCTS and POMCP

In the domain of online planning for (PO)MDPs, MCTS evaluates an information state by: 1) selecting an action according to an *action-selection* strategy; 2) performing the selected action by Monte-Carlo simulation; 3) recursively evaluating the resulting state if it is already in the search tree, or inserting it into the search tree and playing a *rollout policy* by simulations; and 4) updating the statistics of tree nodes by back-propagating the simulation results up to the root. MCTS iteratively repeats this process, and simultaneously builds an asymmetric best-first search tree. When interrupted, MCTS reports the best action based on the returned values of the tree search.

UCT is one of the most popular implementations of MCTS for MDPs, which uses the UCB1 heuristic to guide the selection of actions (Kocsis and Szepesvári 2006). POMCP is an extension of UCT to POMDPs, and has been

shown to work quite well in practice (Silver and Veness 2010). In POMCP, each simulation starts from a state sampled from the belief $b(h)$ for history h of the root node, and chooses the action that maximizes the UCB1 heuristic, formally defined as $\bar{Q}(h,a) + c\sqrt{\log N(h)/N(h,a)}$, where $\bar{Q}(h,a)$ is the mean outcome of action a applied in history node h in all previous simulations, $N(h,a)$ is the visitation count of action a following h, $N(h)$ is the overall count $N(h) = \sum_{a \in A} N(h,a)$, and c is a constant that determines the relative ratio of exploration to exploitation. POMCP uses a particle filter to approximate the belief state, by adapting a Monte-Carlo procedure to update particles based on sampled observations, rewards, and state transitions.

Related Work

The DNG-MCTS algorithm is our previous effort of applying Thompson sampling to MDPs (Bai, Wu, and Chen 2013). The basic assumptions of DNG-MCTS are made due to the fact that, given a policy, an MDP reduces to a Markov chain defined over the state space. Unfortunately, this cannot be straightforwardly extended to POMDPs because the Markov chain of a POMDP with given policy must be defined over the joint space of the state and belief space. Therefore, different assumptions are critically required to use Thompson sampling in POMDPs. In this paper, for each decision node, we assume categorical distribution for the immediate reward of POMDPs (known for MDPs), and use mixture of Normal distributions for the accumulated reward of following π (Normal distribution for MDPs), and do not explicitly model the accumulated reward of performing an action and then following π because only the expectation matters (mixture of Normal distributions for MDPs). We also present new algorithms and analysis for online planning in POMDPs with these assumptions, which are nontrivial with respect to the existing work in DNG-MCTS.

Wang et al. (2005) presented a sampling approach for approximately optimal decision-making in the context of Bayesian reinforcement learning. Specifically, the proposed algorithm maintains a posterior distribution over MDP models (i.e., the transition and reward functions), samples an MDP according to the belief of models, and solves the sampled MDP to select an action for current node. However, their method requires to repeatedly solve the sampled MDP for each action selected, which is very time consuming for large problems. In contrast, our method directly maintains posterior distributions of accumulated reward for actions taken in the search tree and selects an action based on its posterior probability of being optimal by using Thompson sampling, which is much more efficient for online planning in POMDPs.

The D^{2}NG-POMCP Algorithm

In this section, we first introduce our assumptions on online planning in POMDPs. Then, we propose our Bayesian modeling and inference methods and our action selection strategy given these assumptions. Finally, the main algorithm is presented and the convergence of our algorithm is discussed.

The Assumptions

Formally, we define $\mathcal{J} = S \times \mathcal{B}$ to be the joint state space of the state space S and the belief space $\mathcal{B}$. Given a policy π, a POMDP agent and the environment it is interacting with reduce to a Markov chain, $\{\langle s_t, b_t \rangle\}$, defined over the joint state space $\mathcal{J}$, where the transition function is $P(\langle s', b' \rangle | \langle s, b \rangle, a) = T(s'|s,a)T(b'|b,a)$.

Let $X_{b,a}$ be a random variable denoting the immediate reward of performing action a in belief state b, let $X_{s,b,\pi}$ be a random variable that denotes the accumulated reward of following policy π from joint state $\langle s, b \rangle$, and let $X_{b,\pi}$ be a random variable denoting the accumulated reward of following policy π from belief state b. Our assumptions are: 1) $X_{b,a}$ follows a categorical distribution, 2) $X_{s,b,\pi}$ follows a Normal distribution, and 3) $X_{b,\pi}$ follows a mixture of Normal distributions. We argue that these assumptions are realistic with the following reasons.

We assume a discrete and finite set of possible immediate rewards in POMDP, $\mathcal{I} = \{r_1, r_2, \ldots, r_k\}$, where each $r_i = R(s,a)$ represents the immediate reward of taking some action a from some hidden state s. It is then easy to see that $X_{b,a}$ follows a categorical distribution, denoted by $Cat(p_1, p_2, \ldots, p_k)$, where $p_i = \sum_s \mathbf{1}[R(s,a) = r_i] b(s)$ is the probability of $X_{b,a} = r_i$ and $\sum_{i=1}^{k} p_i = 1$.

For POMDPs, the sub-space reachable from the initial belief state b_0 is countable, since each combination of historical action-observation pairs determines uniquely a resulting belief state, and the history space is naturally countable in lexicographic order. Therefore, the reachable sub-space from $\langle s_0, b_0 \rangle$ is also countable. Suppose that the resulting chain $\{\langle s_t, b_t \rangle\}$ is ergodic, i.e., it is possible to go from every belief state to every other belief state (not necessarily in one move). Let w denote the stationary distribution of $\{\langle s_t, b_t \rangle\}$. According to the central limit theorem for Markov chains (Jones 2004; DasGupta 2008), for any bounded function f defined on a countable sub-space of $\mathcal{J}$, we have:

$$\frac{1}{\sqrt{n}} \Big(\sum_{t=0}^{n} f(s_t, b_t) - n\mu \Big) \rightarrow N(0, \sigma^2) \text{ as } n \rightarrow \infty,$$

where $\langle s_t, b_t \rangle$ is the chain state at time step t, $\mu = E_w[f]$ and σ is a constant depending only on f and w. This implies that the sum of $f(s_t, b_t)$ follows $N(n\mu, n\sigma^2)$ as n grows to infinity. It is then natural to approximate the distribution of $\sum_{t=0}^{n} f(s_t, b_t)$ as a Normal distribution if n is large enough.

For finite horizon POMDPs with horizon H, if $\gamma = 1$, $X_{s_0,b_0,\pi} = \sum_{t=0}^{H} R(s_t, \pi(b_t))$ can be seen as a sum of $f(s_t, b_t) = R(s_t, \pi(b_t))$. We claim that $X_{s_0,b_0,\pi}$ is approximately normally distributed for each $\langle s_0, b_0 \rangle \in \mathcal{J}$, if H is sufficiently large. For the case $\gamma \neq 1$, $X_{s_0,b_0,\pi} = \sum_{t=0}^{H} \gamma^t R(s_t, \pi(b_t))$ can be expressed as a linear combination of $\sum_{t=0}^{n} f(s_t, b_t)$ for $n = 0$ to H: $X_{s_0,b_0,\pi} = (1-\gamma) \sum_{n=0}^{H-1} \gamma^n \sum_{t=0}^{n} f(s_t, b_t) + \gamma^H \sum_{t=0}^{H} f(s_t, b_t)$. Note that a linear combination of independent or correlated normally distributed random variables is still normally distributed, if H is sufficiently large and γ is close to 1. It is reasonable to approximate $X_{s_0,b_0,\pi}$ as a Normal distribution. Therefore, we assume $X_{s,b,\pi}$ is normally distributed in both cases.

The accumulated reward of following π from belief b is totally determined in the reduced Markov chain with stochastically sampled initial state s, i.e., $X_{b,\pi} = X_{s,b,\pi}$, where s is distributed according to b. Hence the probability density function (pdf) of $X_{b,\pi}$ can be expressed as a convex combination of pdfs of $X_{s,b,\pi}$ by definition: $f_{X_{b,\pi}}(x) = \sum_{s\in S} b(s) f_{X_{s,b,\pi}}(x)$. It is then straightforward to model the distribution of $X_{b,\pi}$ as a mixture of Normal distributions, if $X_{s,b,\pi}$ is assumed to be normally distributed for each $\langle s, b\rangle \in \mathcal{J}$.

In case if the policy π is not fixed and changes over time, e.g., the derived policy of an online algorithm before it converges, the real distribution of $X_{b,\pi}$ is actually unknown and could be some kind of very complex distribution. However, if the algorithm is guaranteed to converge at infinity, it is then convenient and reasonable to approximate $X_{b,\pi}$ as a mixture of Normal distributions.

The Bayesian Modeling and Inference Methods

In general Bayesian settings, the unknown distribution of a random variable X can be modeled as a parametric likelihood function $L(x|\theta)$ depending on some parameters θ. Given a prior distribution $P(\theta)$, and a set of past observations $Z = \{x_1, x_2, \dots\}$, the posterior distribution of θ can then be obtained by using Bayes' rules: $P(\theta|Z) \propto \prod_i L(x_i|\theta)P(\theta)$.

Assumption 1 implies that it suffices to model the distribution of $X_{b,a}$ as a categorical likelihood with unknown weights, i.e., $X_{b,a} \sim Cat(p_1, p_2, \dots, p_k)$. A natural representation on these unknown weights is via Dirichlet distributions, since Dirichlet distribution is the conjugate prior of a categorical distribution. For belief state b and action a, a Dirichlet distribution, denoted by $Dir(\boldsymbol{\psi}_{b,a})$, where $\boldsymbol{\psi}_{b,a} = (\psi_{b,a,r_1}, \psi_{b,a,r_2}, \dots, \psi_{b,a,r_k})$, gives the posterior distribution of p_i if the immediate reward of r_i has been observed $\psi_{b,a,r_i} - 1$ times. After observing an immediate reward r, the posterior distribution is also Dirichlet and can simply be updated as $\psi_{b,a,r} \leftarrow \psi_{b,a,r} + 1$.

Based on Assumption 2, we model the distribution of $X_{s,b,\pi}$ as a Normal likelihood $N(\mu_{s,b}, 1/\tau_{s,b})$ with unknown mean $\mu_{s,b}$ and precision $\tau_{s,b}$. The precision is defined as the reciprocal of the variance, $\tau = 1/\sigma^2$. This is chosen for mathematical convenience of introducing the NormalGamma distribution as a conjugate prior. Let us briefly review the NormalGamma distribution which is usually defined by the hyper-parameters $\langle \mu_0, \lambda, \alpha, \beta\rangle$ with $\lambda > 0$, $\alpha \geq 1$ and $\beta \geq 0$. It is said that (μ, τ) follows a NormalGamma distribution $NormalGamma(\mu_0, \lambda, \alpha, \beta)$ if the pdf of (μ, τ) has the form:

$$f(\mu, \tau|\mu_0, \lambda, \alpha, \beta) = \frac{\beta^\alpha \sqrt{\lambda}}{\Gamma(\alpha)\sqrt{2\pi}} \tau^{\alpha-\frac{1}{2}} e^{-\beta\tau} e^{-\frac{\lambda\tau(\mu-\mu_0)^2}{2}}.$$

By definition, the marginal distribution over τ is a Gamma distribution, $\tau \sim Gamma(\alpha, \beta)$, and the conditional distribution over μ given τ is a Normal distribution, $\mu \sim N(\mu_0, 1/(\lambda\tau))$. Suppose that X is normally distributed with unknown mean μ and precision τ, $x \sim N(\mu, 1/\tau)$, and that the prior distribution of (μ, τ) has a NormalGamma distribution, $(\mu, \tau) \sim NormalGamma(\mu_0, \lambda_0, \alpha_0, \beta_0)$. After observing n independent samples of X, denoted by $\{x_1, x_2, \dots, x_n\}$, according to the Bayes' theorem, the posterior distribution of (μ, τ) is also a NormalGamma distribution, $(\mu, \tau) \sim NormalGamma(\mu_n, \lambda_n, \alpha_n, \beta_n)$, where $\mu_n = (\lambda_0\mu_0 + n\bar{x})/(\lambda_0 + n)$, $\lambda_n = \lambda_0 + n$, $\alpha_n = \alpha_0 + n/2$ and $\beta_n = \beta_0 + (ns + \lambda_0 n(\bar{x} - \mu_0)^2/(\lambda_0 + n))/2$, where $\bar{x} = \sum_{i=1}^n x_i/n$ is the sample mean and $s = \sum_{i=1}^n (x_i - \bar{x})^2/n$ is the sample variance.

As explained in Assumption 3, $X_{b,\pi}$ follows a mixture of Normal distributions, which can be easily modeled as a convex combination of $b(s)$ and $X_{s,b,\pi}$ for $s \in S$.

Now consider the accumulated reward of first performing action a in belief b and then following policy π thereafter, denoted by $X_{b,a,\pi}$. According to the definition, $X_{b,a,\pi} = X_{b,a} + \gamma X_{b',\pi}$, where b' is the next belief distributed according to $T(b'|b, a)$. It is difficult to explicitly describe the distribution of $X_{b,a,\pi}$. However, it is rather easy to compute the expectation, expressed as $E[X_{b,a,\pi}] = E[X_{b,a}] + \gamma \sum_{b'\in\mathcal{B}} E[X_{b',\pi}]T(b'|b, a) = E[X_{b,a}] + \gamma \sum_{o\in O, b'=\zeta(b,a,o)} E[X_{b',\pi}]\Omega(o|b, a)$ (actually $E[X_{b,a,\pi}]$ is usually defined as the Q value with $Q^\pi(b, a) = r(b, a) + \gamma \sum_{o\in O} \Omega(o|b, a)V^\pi(\zeta(b, a, o))$ if the underlying transition and observation functions are known). Recall that in our assumptions, $X_{b,a}$ follows a categorical distribution, and $X_{b',\pi}$ follows a mixture of Normal distributions. The question turns to be how to model the previously unknown observation model–$\Omega(\cdot|b, a)$. Fortunately $\Omega(\cdot|b, a)$ can be easily inferred in Bayesian settings by introducing a Dirichlet distribution as the conjugate prior, denoted by $Dir(\boldsymbol{\rho}_{b,a})$, where $\boldsymbol{\rho}_{b,a} = (\rho_{b,a,o_1}, \rho_{b,a,o_2}, \dots)$ are the hyper-parameters. After having observed a transition $(b, a) \to o$, the posterior distribution of $\Omega(\cdot|b, a)$ can simply be updated as $\rho_{b,a,o} \leftarrow \rho_{b,a,o} + 1$.

Therefore, to compute the expectation of $X_{b,a,\pi}$ in Bayesian settings, we only need to maintain a set of hyper-parameters $\langle \mu_{s,b,0}, \lambda_{s,b}, \alpha_{s,b}, \beta_{s,b}\rangle$, $\boldsymbol{\psi}_{b,a}$ and $\boldsymbol{\rho}_{b,a}$ for each state s, belief state b and action a encountered in the MCTS search tree, and update them by using Bayesian rules.

Now we turn to the question of how to choose the priors by initializing hyper-parameters. While the impact of the prior tends to be negligible in the limit, its choice is important especially when only a small amount of data has been observed. In general, priors should reflect available knowledge of the hidden model.

In the absence of any knowledge, *uninformative priors* may be preferred. According to the principle of indifference, uninformative priors assign equal probabilities to all possibilities. For NormalGamma priors, we hope that the sampled distribution of μ given τ, i.e., $N(\mu_0, 1/(\lambda\tau))$, is as flat as possible. This implies an infinite variance $1/(\lambda\tau) \to \infty$, so that $\lambda\tau \to 0$. Recall that τ follows a Gamma distribution $Gamma(\alpha, \beta)$ with expectation $E[\tau] = \alpha/\beta$, so we have in expectation $\lambda\alpha/\beta \to 0$. Taking into consideration the parameter space ($\lambda > 0, \alpha \geq 1, \beta \geq 0$), we can choose λ small enough, $\alpha = 1$ and β sufficiently large to approximate this condition. Second, we hope the sampled distribution is

Agent (b_0 : *initial belief*)
Initialize $H \leftarrow$ maximal planning horizon
Initialize $\mathcal{I} \leftarrow \{$possible immediate rewards$\}$
Initialize $h \leftarrow \varnothing$
Initialize $\mathcal{P}(h) \leftarrow b_0$
repeat
  $a \leftarrow$ **OnlinePlanning** $(h, \varnothing)$
  Execute a and get observation o
  $h \leftarrow hao$
  $\mathcal{P}(h) \leftarrow$ **ParticleFilter** $(\mathcal{P}(h), a, o)$
until *terminating conditions*

D^{2}NG-POMCP (s : *state*, h : *history*, T : *tree*, d : *depth*)
if $d \geq H$ **or** s *is terminal* **then**
  return 0
else if *node* $\langle h \rangle$ *is not in tree* T **then**
  Initialize $(\mu_{s,h,0}, \lambda_{s,h}, \alpha_{s,h}, \beta_{s,h})$ for $s \in S$
  Initialize $\boldsymbol{\rho}_{h,a}$ and $\boldsymbol{\psi}_{h,a}$ for $a \in A$
  Add node $\langle h \rangle$ to T
  Play rollout policy by simulation for $H - d$ steps
  Get the accumulated reward r
  return r
else
  $a \leftarrow$ **ThompsonSampling** $(h, d, True)$
  Execute a by simulation
  Get next state s', observation o and reward i
  $h' \leftarrow hao$
  $\mathcal{P}(h') \leftarrow \mathcal{P}(h') \cup s'$
  $r \leftarrow i + \gamma$**D^{2}NG-POMCP** $(s', h', T, d+1)$
  $\alpha_{s,h} \leftarrow \alpha_{s,h} + 0.5$
  $\beta_{s,h} \leftarrow \beta_{s,h} + (\lambda_{s,h}(r - \mu_{s,h,0})^2/(\lambda_{s,h}+1))/2$
  $\mu_{s,h,0} \leftarrow (\lambda_{s,h}\mu_{s,h,0} + r)/(\lambda_{s,h}+1)$
  $\lambda_{s,h} \leftarrow \lambda_{s,h} + 1$
  $\rho_{h,a,o} \leftarrow \rho_{h,a,o} + 1$
  $\psi_{h,a,i} \leftarrow \psi_{h,a,i} + 1$
  return r

ThompsonSampling (h : *history*, d : *depth*, *sampling* : *boolean*)
foreach $a \in A$ **do**
  $q_a \leftarrow$ **QValue** $(h, a, d, sampling)$
return $\operatorname{argmax}_a q_a$

OnlinePlanning (h : *history*, T : *tree*)
repeat
  Sample s according to $\mathcal{P}(h)$
  D^{2}NG-POMCP $(s, h, T, 0)$
until *resource budgets reached*
return **ThompsonSampling** $(h, 0, False)$

QValue (h : *history*, a : *action*, d : *depth*, *sampling* : *boolean*)
$r \leftarrow 0$
foreach $o \in O$ **do**
  if $sampling = True$ **then**
    Sample w_o according to $Dir(\boldsymbol{\rho}_{h,a})$
  else
    $w_o \leftarrow \rho_{h,a,o} / \sum_{o' \in O} \rho_{h,a,o'}$
  $r \leftarrow r + w_o$**Value** $(hao, d+1, sampling)$
$r \leftarrow \gamma r$
foreach $i \in \mathcal{I}$ **do**
  if $sampling = True$ **then**
    Sample w_i according to $Dir(\boldsymbol{\psi}_{h,a})$
  else
    $w_i \leftarrow \psi_{h,a,i} / \sum_{i' \in \mathcal{I}} \psi_{h,a,i'}$
  $r \leftarrow r + w_i i$
return r

Value (h : *history*, d : *depth*, *sampling* : *boolean*)
if $d = H$ **then**
  return 0
else
  if $sampling = True$ **then**
    Sample (μ_s, τ_s) according to
    $NormalGamma(\mu_{s,h,0}, \lambda_{s,h}, \alpha_{s,h}, \beta_{s,h})$
    for $s \in \mathcal{P}(h)$
    return $\frac{1}{|\mathcal{P}(h)|} \sum_{s \in \mathcal{P}(h)} \mu_s$
  else
    return $\frac{1}{|\mathcal{P}(h)|} \sum_{s \in \mathcal{P}(h)} \mu_{s,h,0}$

Figure 1: Dirichlet-Dirichlet-NormalGamma based partially observable Monte-Carlo planing (D^{2}NG-POMCP)

in the middle of the axis, so $\mu_0 = 0$ seems to be a good selection. It is worth noting that intuitively β should not be set too large, or the convergence process may be very slow. For Dirichlet priors, it is common to set $\psi_{b,a,r}$ and $\rho_{b,a,o}$ be the same small enough positives for each encountered instance in the search tree to have uninformative priors.

On the other hand, if some domain knowledge is available, *informative priors* may be preferred. By exploiting domain knowledge, a decision node can be initialized with informative priors indicating its priority over other nodes. In D^{2}NG-POMCP, this is done by setting the hyper-parameters based on subjective estimation for state-belief pairs. Take the NormalGamma priors as an example. According to the interpretation of hyper-parameters of NormalGamma distribution in terms of pseudo-observations, if one has a prior mean of μ_0 from λ samples and a prior precision of α/β from 2α samples, the prior distribution over μ and τ is $NormalGamma(\mu_0, \lambda, \alpha, \beta)$, providing a straightforward way to initialize the hyper-parameters if some prior knowledge (such as historical data of past observations) is available. Specifying detailed priors based on prior knowledge for particular domains is beyond the scope of this paper. The

ability to include prior information provides important flexibility and can be considered an advantage of the approach.

The Action Selection Strategy

Thompson sampling stochastically selects an action based on its probability of being optimal. Specifically, in Bayesian settings, action a is chosen with probability:

$$P(a) = \int \mathbf{1}[a = \operatorname*{argmax}_{a'} E[X_{a'}|\theta_{a'}]] \prod_{a'} P_{a'}(\theta_{a'}|Z)\,\mathrm{d}\boldsymbol{\theta},$$

where θ_a is the hidden parameter prescribing the underlying distribution of reward by applying a, $E[X_a|\theta_a] = \int x L_a(x|\theta_a)\,\mathrm{d}x$ is the expectation of X_a given θ_a, and $\boldsymbol{\theta} = (\theta_{a_1}, \theta_{a_2}, \dots)$ is the vector of parameters for all actions. Fortunately, this can efficiently be approached by a sampling method. To this end, a set of parameters θ_a is sampled according to the posterior distributions $P_a(\theta_a|Z)$ for each $a \in A$, and the action with highest expectation is selected, namely, $a^* = \operatorname{argmax}_a E[X_a|\theta_a]$.

In D^{2}NG-POMCP, at each decision node b of the search tree, we sample $w_{b,a,o}$ for each $o \in O$ according to $Dir(\boldsymbol{\rho}_{b,a})$, sample $w_{b,a,r}$ for each $r \in \mathcal{I}$ according to $Dir(\boldsymbol{\psi}_{b,a})$, and sample $\mu_{s',b'}$ for each $\langle s, b'\rangle \in \mathcal{J}$ according to $NormalGamma(\mu_{s',b',0}, \lambda_{s',b'}, \alpha_{s',b'}, \beta_{s',b'})$. The expectation of $X_{b,a,\pi}$ can then be computed as $\sum_{r\in\mathcal{I}} w_{b,a,r} r + \gamma \sum_{o\in O, b'=\zeta(b,a,o)} \sum_{s'\in S} \mu_{s',b'} b'(s') w_{b,a,o}$. Finally, the action with highest expectation is selected to be performed by Monte-Carlo simulation.

The Main Algorithm

The main process of D^{2}NG-POMCP is outlined in Figure 1. As aforementioned, our algorithm is implemented based on histories instead of explicit beliefs. Hence each node of the search tree is represented by a history h. Specifically, we use particles, denoted by $\mathcal{P}(h)$, to represent the respective belief state of history h, which are updated using particle filters. Here, the computation of $E[X_{b,a,\pi}]$ is also implemented to work with particles.

In more detail, the **`ThompsonSampling`** function has a boolean parameter $sampling$. If $sampling$ is true, Thompson sampling method is used to stochastically select an action, otherwise a greedy action is returned with respect to current mean of $E[X_{s,b,\pi}] = \mu_{s,b,0}$, $E[w_{b,a,r}] = \psi_{b,a,r} / \sum_{i\in\mathcal{I}} \psi_{b,a,i}$, and $E[w_{b,a,o}] = \rho_{b,a,o} / \sum_{x\in O} \rho_{b,a,x}$.

At each iteration, the **`D`^{2}`NG-POMCP`** function applies Thompson sampling to recursively select actions to be executed by simulation from the root node to leaf nodes through the existing search tree T. It inserts each newly visited node into the tree, plays a default rollout policy from the new node, and propagates the simulated outcome to update the hyper-parameters for visited histories, states and actions. Note that the rollout policy is only played once for each new node at each iteration, the set of past observations Z in the algorithm has size $n = 1$.

The **`OnlinePlanning`** function implements the anytime characteristic of D^{2}NG-POMCP. It is called with current history h and a search tree T initially empty. It repeatedly calls the **`D`^{2}`NG-POMCP`** function until some resource budgets are reached (e.g., the computation times out or the maximal number of iterations is reached). Then a greedy action to be performed in the environment is returned to the agent.

The **`Agent`** function is the overall procedure interacting with the real environment. It calls **`OnlinePlanning`** to select the planned best action, execute the action, get an observation, and update particles repeatedly until some terminating conditions are satisfied (e.g., the problem is solved or the maximal running time is reached).

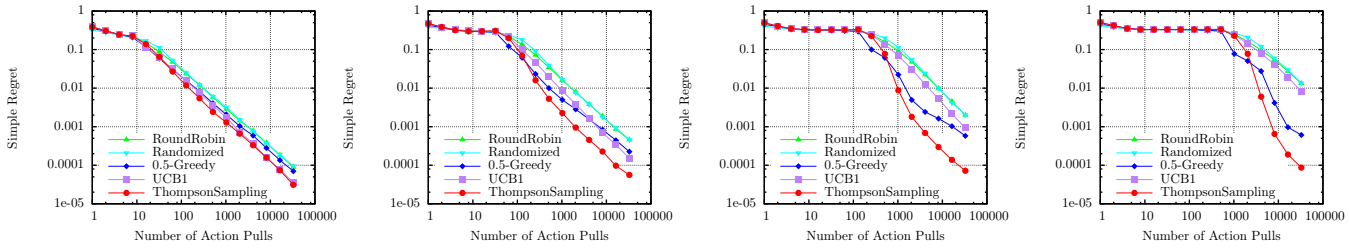
Convergence

For Thompson sampling in stationary MABs, Agrawal and Goyal (2013) have shown that: 1) the probability of selecting any suboptimal action a at the current step is bounded by a linear function of the probability of selecting the optimal action; 2) the coefficient in this linear function decreases exponentially fast with the increase in the number of selections of optimal action. Thus, the probability of selecting the optimal action in an MAB is guaranteed to converge to 1 in the limit using Thompson sampling.

In our settings, the distribution of $X_{b,\pi}$ is determined by both the underlying transition and observation functions and the Q values given the policy π. When the Q values converge, the distribution of $X_{b,\pi}$ becomes stationary with the optimal policy. For the leaf nodes (level H) of the search tree, Thompson sampling will converge to the optimal actions with probability 1 in the limit since the MABs of leaf nodes are stationary with respect to the underlying stationary rollout policy. When all the leaf nodes converge, the distributions of the return values from them will remain unchanged. So the MABs of the nodes in level $H-1$ become stationary as well. Thus, Thompson sampling will also converge to the optimal actions for nodes in level $H-1$. Recursively, this holds for all the upper-level nodes. Therefore, we conclude that D^{2}NG-POMCP can find the optimal policy for the root node by given sufficient computational resources.

Experiments

To empirically illustrate our motivation, we simply compared Thompson sampling in terms of simple regret with other common algorithms in MABs, including RoundRobin, Randomized, 0.5-Greedy and UCB1. The RoundRobin algorithm selects an arm in a round-robin fashion among all arms (Feldman and Domshlak 2012); the Randomized algorithm uniformly selects a random arm; the 0.5-Greedy algorithm selects the best seen arm with probability 0.5, and a random arm otherwise (Tolpin and Shimony 2012); the UCB1 algorithm selects the arm that maximizes the UCB1 heuristic. In our experiments, each arm returns a random reward sampled from a Bernoulli distribution; UCB1 is implemented with exploration constant $\sqrt{2}$; Thompson sampling chooses Beta distributions as the conjugate priors, initialized as $(\alpha = 1, \beta = 1)$. We ran each of the algorithms over 10,000 experiments of randomly generated instances for different numbers of arms, and reported the average simple regret as shown in Figure 2. It can be seen from the results that Thompson sampling yields lower simple regret, and per-



(a) 8 arms (b) 32 arms (c) 128 arms (d) 512 arms

Figure 2: Performance of Thompson sampling in terms of simple regret in MABs.

forms much better if the action space is large. While Thompson sampling theoretically achieves logarithmic optimal and empirically performs very well in terms of cumulative regret in MABs, to the best of our knowledge, it is observed for the first time in the literature that Thompson sampling empirically outperforms other common algorithms in terms of simple regret also, providing a potential of success of applying Thompson sampling to Monte-Carlo planing domains.

We also tested D^{2}NG-POMCP and compared the results with POMCP in *RockSample* and *PocMan* domains. We implemented our codes and conducted the experiments based on `POMCP` — an open source software implementing the POMCP algorithm and some benchmark problems.[1]

For each problem instance, we 1) ran the algorithms for a number of iterations from the current history, 2) applied the best action based on the resulting action-values, 3) repeated the loop until terminating conditions (e.g., a goal state is satisfied or the maximal number of running steps is reached), and 4) reported the total discounted reward and average computation time per action selected. The performance of algorithms is evaluated by the average total discounted reward over 1,000 independent runs. In all experiments, we initialized $(\mu_{s,h,0}, \lambda_{s,h}, \alpha_{s,h}, \beta_{s,h})$ to $(0, 0.01, 1, 100)$, $\psi_{h,a,r}$ to 0.01, and $\rho_{h,a,o}$ to 0.01 for all $s \in S$, $a \in A$, $r \in \mathcal{I}$, $o \in O$ and encountered history h in the search tree. When testing D^{2}NG-POMCP and POMCP, we used the same preferred actions based rollout policy as described and implemented in (Silver and Veness 2010) and `POMCP` respectively. For fair comparison, we also applied the same settings as in `POMCP`: for each decision node, 1) only applicable actions are selected, and 2) applicable actions are forced to be selected once before any of them is selected twice. All experiments were ran on Linux 3.8.0 computers of 2.80 GHz quad-core CPUs and 8 GB RAM.

The RockSample$[n, k]$ problem simulates a robot exploring in a $n \times n$ grid map containing k rocks. The goal is to determine which rocks are valuable, collect valuable ones as much as possible and leave the map finally, by taking moving, checking and sampling actions. Sampling a valuable rock yields a reward of $+10$; sampling an invaluable rock yields a reward of -10; moving into the exit area yields a reward of $+10$. All other actions have no cost or reward.

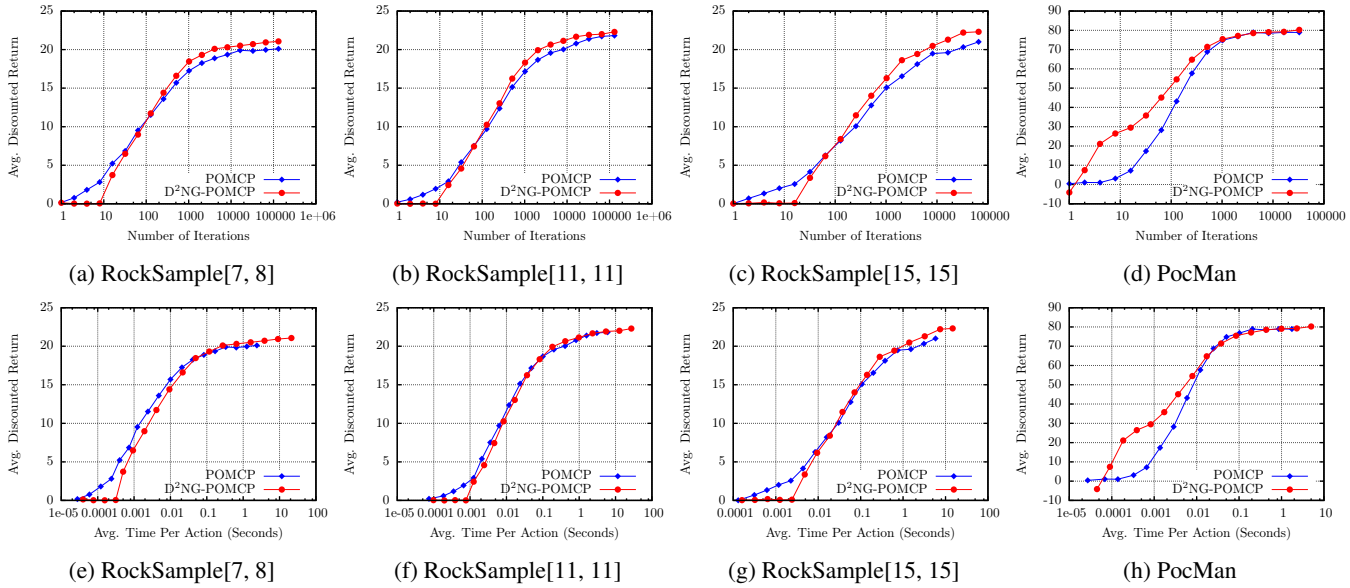
[1] `POMCP` can be downloaded from http://www0.cs.ucl.ac.uk/staff/D.Silver/web/Applications.html

RockSample	[7, 8]	[11,11]	[15,15]
States $\|S\|$	12,544	247,808	7,372,800
AEMS2	21.37 ± 0.22	N/A	N/A
HSVI-BFS	21.46 ± 0.22	N/A	N/A
SARSOP	21.39 ± 0.01	21.56 ± 0.11	N/A
POMCP	20.71 ± 0.21	20.01 ± 0.23	15.32 ± 0.28
D^{2}NG-POMCP	20.87 ± 0.20	21.44 ± 0.21	20.20 ± 0.24

Table 1: Comparison of D^{2}NG-POMCP with existing approaches in RockSample for discounted cumulative reward.

The discount factor γ is 0.95. Empirical results are depicted in Figure 3. Each data point shows the average result over 1,000 runs or at most 12 hours of total computation. The top four figures depict the performance with respect to number of iterations; the bottom four figures depict the performance with respect to average time per action. Number of iterations is the total number of calls to **`D2NG-POMCP`** allowed for each decision of action in the **`OnlinePlanning`** function; average time per action is the average computation time for each action selected in the **`OnlinePlanning`** function. We can see from the figures that D^{2}NG-POMCP converges faster than POMCP in terms of number of iterations, and appears to be competitive with POMCP in terms of average time per action. Table 1 presents the comparison of D^{2}NG-POMCP with prior work, including AEMS2, HSVI-BFS, SARSOP and POMCP. AEMS2 and HSVI-BFS are online algorithms; SARSOP is an offline algorithm. They are all provided with full factored representations of the underlying problems. AEMS2 and HSVI-BFS used knowledge computed offline by PBVI; results are taken from (Ross et al. 2008). SARSOP was given approximately 1,000 seconds of offline computation; results are taken from (Kurniawati, Hsu, and Lee 2008). POMCP and D^{2}NG-POMCP used the same informed rollout policy. The results of POMCP are taken from (Silver and Veness 2010). Each online algorithm was given exactly 1 second per action. Performance is evaluated by average discounted return over 1,000 runs or at most 12 hours of total computation. The results indicate that D^{2}NG-POMCP is able to provide competitive results on RockSample$[7, 8]$ and RockSample$[11, 11]$, and advanced POMCP with much better return on RockSample$[15, 15]$.

The PocMan problem is firstly introduced in (Silver and



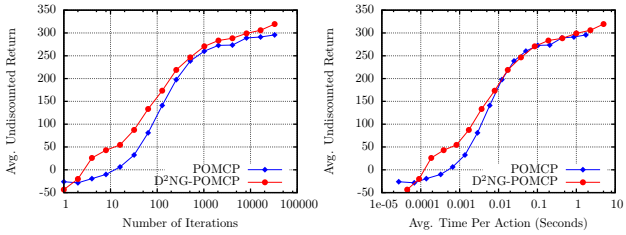
(a) RockSample[7, 8] (b) RockSample[11, 11] (c) RockSample[15, 15] (d) PocMan

(e) RockSample[7, 8] (f) RockSample[11, 11] (g) RockSample[15, 15] (h) PocMan

Figure 3: Performance of D^{2}NG-POMCP in RockSample and PocMan evaluated by average discounted return.

Veness 2010). The PocMan agent navigates in a 17×19 maze, while trying to eat some randomly distributed food pellets and power pills. Four ghost agents roam the maze, according to a given stochastic strategy. The PocMan agent dies if it touches any ghost, unless it has eaten any power pills within the last 15 time steps. It receives a reward of -1 at each step, $+10$ for each food pellet, $+25$ for eating a ghost and -100 for dying. A 10-bit observation is observed at every time step, corresponding to the PocMan agent's senses of sight, hearing, touch and smell. The PocMan problem has approximately 1,056 states, 4 actions, and 1,024 observations. The discount factor γ is 0.95. The performance of D^{2}NG-POMCP in PocMan evaluated by average discounted return is shown in Figure 3 (d and h). Each data point shows the average result over 1,000 runs or at most 12 hours of total computation. It is worth noting that the algorithm's performance in PocMan experiment is evaluated by average discounted return, instead of average undiscounted return as in the original paper (Silver and Veness 2010). We believe that using average discounted return to show the performance is more reasonable, since average discounted return is what the algorithms really intent to optimize. To provide a more comprehensive view, we also included the respective results with regard to average undiscounted return in addition as shown in Figure 4. In this domain, D^{2}NG-POMCP performs much better than POMCP with regard to both number of iterations and average time per action.

Regarding to sample and computational complexities, although the total computation time of D^{2}NG-POMCP is linear with the total number of simulations, which is at most $width \times depth$ (where $width$ is the number of iterations and $depth$ is the maximal planning horizon), our approach does require more computation than POMCP, due to the time-consuming operations of samplings from various distributions. However, if the simulations are expensive (e.g., computational physics in 3D environment or stochastic environment with multiple agents where the cost of executing the simulation steps greatly exceeds the time needed by action-selection steps in MCTS), D^{2}NG-POMCP can obtain better performance in terms of computational complexity because D^{2}NG-POMCP is expected to have lower sample complexity (as confirmed by the resutls in PocMan).



(a) PocMan (b) PocMan

Figure 4: Performance of D^{2}NG-POMCP in PocMan evaluated by average undiscounted return.

Conclusion

We proposed the D^{2}NG-POMCP algorithm — a novel Bayesian modeling and inference based Thompson sampling approach to MCTS that balances between cumulative and simple regrets for online planning in POMDPs — and presented the overall Bayesian framework for representing, updating, decision-making and propagating of probability distributions over rewards in the partially observable Monte-Carlo search tree. Experimental results showed that D^{2}NG-POMCP outperformed the state-of-the-art algorithms (i.e.,

MEMS2, HSVI-BFS, SARSOP, and POMCP) in both RockSample and PocMan domains. One of the basic assumptions of D^{2}NG-POMCP is to model the uncertainty of the accumulated reward of following policy π from belief b as a mixture of Normal distributions, which in principle only holds in the limit. In future work, we plan to extend this assumption to more general distributions and test our algorithm on real-world applications.

Acknowledgements

This work is supported in part by the National Hi-Tech Project of China under grant 2008AA01Z150 and the Natural Science Foundation of China under grant 60745002 and 61175057. Feng Wu is supported in part by the ORCHID project (`http://www.orchid.ac.uk`). Zongzhang Zhang is supported in part by MoE ARF grant MOE2010-T2-2-071. We are grateful to the anonymous reviewers for their constructive comments and suggestions.

References

Agrawal, S., and Goyal, N. 2013. Further optimal regret bounds for Thompson sampling. In *Artificial Intelligence and Statistics*, 99–107.

Asmuth, J., and Littman, M. L. 2011. Learning is planning: near Bayes-optimal reinforcement learning via Monte-Carlo tree search. In *Uncertainty in Artificial Intelligence*, 19–26.

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

Bai, A.; Wu, F.; and Chen, X. 2013. Bayesian mixture modelling and inference based Thompson sampling in Monte-Carlo tree search. In *Advances in Neural Information Processing Systems 26*. 1646–1654.

Bellman, R. 1957. *Dynamic Programming*. Princeton, NJ, USA: Princeton University Press, 1 edition.

Browne, C.; Powley, E. J.; Whitehouse, D.; Lucas, S. M.; Cowling, P. I.; Rohlfshagen, P.; Tavener, S.; Perez, D.; Samothrakis, S.; and Colton, S. 2012. A survey of Monte Carlo tree search methods. *IEEE Trans. Comput. Intellig. and AI in Games* 4(1):1–43.

Bubeck, S.; Munos, R.; and Stoltz, G. 2011. Pure exploration in finitely-armed and continuous-armed bandits. *Theor. Comput. Sci* 412(19):1832–1852.

Chapelle, O., and Li, L. 2011. An empirical evaluation of Thompson sampling. In *Advances Neural Information Processing Systems*, 2249–2257.

DasGupta, A. 2008. *Asymptotic theory of statistics and probability*. Springer.

Feldman, Z., and Domshlak, C. 2012. Simple regret optimization in online planning for Markov decision processes. *arXiv preprint arXiv:1206.3382*.

Gelly, S., and Silver, D. 2011. Monte-Carlo tree search and rapid action value estimation in computer go. *Artificial Intelligence* 175(11):1856–1875.

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

Jones, G. L. 2004. On the Markov chain central limit theorem. *Probability surveys* 1:299–320.

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

Kaufmann, E.; Korda, N.; and Munos, R. 2012. Thompson sampling: An optimal finite time analysis. In *Algorithmic Learning Theory*, 199–213.

Kocsis, L., and Szepesvári, C. 2006. Bandit based Monte-Carlo planning. In *European Conference on Machine Learning*, 282–293.

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*, 65–72.

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.

Silver, D., and Veness, J. 2010. Monte-Carlo planning in large POMDPs. In *Advances in Neural Information Processing Systems*, 2164–2172.

Thompson, W. R. 1933. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika* 25:285–294.

Tolpin, D., and Shimony, S. E. 2012. MCTS based on simple regret. In *AAAI Conference on Artificial Intelligence*.

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*, 956–963. ACM.

Winands, M. H.; Bjornsson, Y.; and Saito, J. 2010. Monte Carlo tree search in lines of action. *IEEE Transactions on Computational Intelligence and AI in Games* 2(4):239–250.

Wu, F.; Zilberstein, S.; and Chen, X. 2011. Online planning for ad hoc autonomous agent teams. In *International Joint Conference on Artificial Intelligence*, 439–445.