Real-time Bayesian Search Control for MDPs

Scott Sanner
Statistical Machine Learning Group
NICTA, Australian National University
Canberra, Australia
ssanner@nicta.com.au

Robby Goetschalckx and **Kurt Driessens**
Department of Computer Science
Catholic University of Leuven
Heverlee, Belgium
{robby,kurtd}@cs.kuleuven.ac.be

Abstract

Many oversubscribed planning problems with multiple (potentially jointly unachievable) goals can be cast in the Markov decision process (MDP) framework when appropriate reward functions and objective criteria can be defined. Recent real-time, search-based solutions to MDPs maintain upper and lower bounds on the value function at all stages of execution prior to convergence. While many algorithms already use this information to detect convergence and to prioritize state exploration, we propose a novel Bayesian interpretation of these bounds to evaluate the myopic value of perfect information (VPI) of exactly knowing a state's value. This VPI analysis permits two general Bayesian search control modifications to existing algorithms: (1) we can use it to dynamically adapt the backup depth in real-time, search-based bounded dynamic programming approaches, and (2) we can use it for real-time policy evaluation with a bounded value function produced by an anytime algorithm. We empirically evaluate both of these modifications to existing algorithms and analyze their performance. While real-time Bayesian search control may reduce the number of backups until convergence for (1), it offers more substantive improvements for (2), making it an attractive real-time policy evaluation approach for MDPs.

Introduction

The Markov decision process (MDP) is an appropriate model for oversubscribed decision-theoretic planning problems where all goals may not be jointly achievable, but an appropriate numeric reward can be assigned to each goal and an objective criterion can be optimized that places preferences on how these rewards are obtained over time.

In order to maximize future expected reward w.r.t. some MDP objective criterion in a real-time setting, there is often a need to deliberate about the optimality of a current solution in order to focus computation where it will most greatly impact solution quality. However, assuming that time cost and reward can be expressed in commensurate units, deliberative computation should only be performed when its expected benefit outweighs its time cost. This deliberative search control problem is well-known in AI and decision-theoretic planning as we discuss further in Related Work.

In this paper, we aim to investigate the decision-theoretic tradeoff between computation time and reward in the asynchronous dynamic programming (DP) solution (Bertsekas 1982) and anytime policy evaluation of MDPs. Specifically, we focus on the class of real-time dynamic programming (RTDP) (Barto, Bradtke, & Singh 1993) algorithms that permit both asynchronous value updates (ideal for evaluating DP search control techniques) and also provide a solution estimate at all times of execution (ideal for evaluating search-based anytime policy evaluation approaches). While nothing in this paper is specific to RTDP approaches *per se*, they provide a useful family of algorithms that serve as a test-bed for evaluating our proposed modifications.

Recent advances in asynchronous DP solutions to MDPs have proposed various optimizations of the original RTDP approach (Bonet & Geffner 2003b; McMahan, Likhachev, & Gordon 2005; Smith & Simmons 2006), including maintaining upper and lower bounds and prioritizing exploration according to these bounds. In this work, we introduce a novel analysis of these bounds in a Bayesian decision-theoretic setting that allows us to trade off the potential information gain of search-based updates vs. their computational time cost in a value of information framework (Howard 1966).

One major benefit of the Bayesian decision-theoretic analysis in the derivation of our search-control approach is that its solution leads to a univariate integral that can be calculated analytically. Furthermore, the enhancement we propose is a technique that can be easily integrated into many search-based MDP solution approaches in various ways:

(1) Bayesian search control for dynamic adaptation of the backup depth in (real-time) dynamic programming.

(2) Bayesian search control for real-time policy evaluation with value bounds produced by an anytime algorithm.

Empirically, while Bayesian search control may reduce the number of backups required for (1), it offers more substantive improvements for (2), making it an attractive anytime policy evaluation approach for the MDP framework.

Preliminaries

Markov Decision Processes

We assume familiarity with the standard definition of a Markov decision process (MDP) as a tuple $\langle S, A, T, R, \gamma \rangle$ (Puterman 1994). $S = \{s_1, \ldots, s_n\}$ is a finite set of fully observable states. $A = \{a_1, \ldots, a_m\}$ is a

finite set of actions. $T : S \times A \times S \rightarrow [0, 1]$ is a known stationary, Markovian transition function. $R : S \times A \rightarrow \mathbb{R}$ is a fixed known reward function associated with every state and action. γ is a discount factor s.t. $0 \leq \gamma \leq 1$ where rewards t time steps in the future are discounted by γ^t.

If $\gamma < 1$ then we say the MDP is *discounted*. If $\gamma = 1$ then we make the following additional restrictions and we say the MDP is a *stochastic shortest path* (SSP) problem (Bertsekas & Tsitsiklis 1996): (1) we have an absorbing set of non-empty goal states $\mathcal{G}$ where $\mathcal{G} \subset S$ and $\forall a \in A, s \in \mathcal{G}\ R(s, a) = 0$, (2) negative rewards (i.e., costs) $\forall a \in A, s \notin \mathcal{G}\ R(s, a) < 0$ for all actions in non-goal states, and (3) some goal state is reachable with non-zero probability from every state. Additionally, we will assume there is a potentially restricted initial state set $\mathcal{I} \subseteq S$.

A policy $\pi : S \rightarrow A$ specifies the action $a = \pi(s)$ to take in each state s. Our goal is to find a policy that maximizes the value function, defined using the infinite horizon, expected discounted reward criterion:

$$V^{\pi}(s) = E_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t \cdot r_t \Big| s_0\right] \quad (1)$$

where r_t is the reward obtained at time t (assuming start state s_0 at time $t = 0$).

Dynamic Programming

Value iteration is a dynamic programming (DP) solution to an MDP. Starting with arbitrary $V^0(s)$, we perform the following *Bellman update* (or *backup*) that derives the optimal value function $V^t(s)$ with t-decision-stages-to-go w.r.t. the value function $V^{t-1}(s)$ with $t-1$-decision-stages-to-go:

$$V^t(s) := \max_{a \in A} \left\{ R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') \cdot V^{t-1}(s') \right\} \quad (2)$$

We repeat this process for each stage t, obtaining $V^t(s)$ from $V^{t-1}(s)$, until we have computed the intended t-stage-to-go value function. This algorithm is demonstrated graphically in Figure 1(b).

Rewriting Equation 2 into two steps, we define the Bellman update for state s as $V^t(s) = \textsc{Update}(V^{t-1}, s)$ where

$$Q^t(s, a) := R(s, a) + \gamma \cdot \sum_{s' \in S} T(s, a, s') \cdot V^{t-1}(s') \quad (3)$$

$$V^t(s) := \max_{a \in A} \left\{ Q^t(s, a) \right\}. \quad (4)$$

Puterman (1994) shows that for discounted MDPs, terminating once the following condition is met

$$\max_s |V^t(s) - V^{t-1}(s)| < \frac{\epsilon(1-\gamma)}{2\gamma} \quad (5)$$

guarantees ϵ-optimality w.r.t. the optimal value function $V^*(s)$, i.e., $\max_s |V^t(s) - V^*(s)| < \epsilon$.

The greedy policy $\pi(s) = \textsc{GreedyAction}(V^t, s)$ w.r.t. V^t and state s is defined as follows:

$$\pi(s) := \arg\max_{a \in A} \left\{ R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') \cdot V^t(s') \right\} \quad (6)$$

Algorithm 1: RTDP (for SSPs)

```
begin
    // Initialize V̂_h with admissible value function
    V̂_h := V_h;
    while not converged and not out of time do
        // Pick initial state
        Draw s from I at random;
        while s ∉ G do
            // Use Eqs. (3) & (4) to update V̂_h(s)
            V̂_h(s) = UPDATE(V̂_h, s);
            // Use Eq. (6) to get greedy action for s, V̂_h(s)
            a = GREEDYACTION(V̂_h, s);
            // Pick next state according to transition dist.
            s := s' ~ T(s, a, s');
    return V̂_h;
end
```

For discounted problems, the greedy policy derived from $V^t(s)$ (satisfying Equation 5) loses no more than ϵ in value over the infinite horizon in comparison to the optimal policy. While similar error bounds do not hold for ϵ-optimal value functions in SSPs, convergence to an ϵ-optimal value function is a commonly used stopping criterion for SSPs.

Asynchronous Dynamic Programming and Search

If we reexamine Equation 2, we note that we could compute this recurrence in a forward-search manner by starting at an initial state, unfolding the recurrence to horizon h, and then computing the expectation and maximization as we return to the initial state. A graphical representation of the unfolding of this computation is shown in Figure 1(a). Furthermore, *asynchronous* DP methods (Bertsekas 1982) retain convergence properties even if states are updated in an arbitrary order (e.g., by searching forward to a non-uniform depth and backing up values for all states encountered).

The Real-time dynamic programming (RTDP) (Barto, Bradtke, & Singh 1993) algorithm provided in Algorithm 1 is one popular asynchronous DP approach that updates states encountered during trial-based simulations of an MDP. This approach uses limited-depth, forward-search backups from Figure 1(a) to update the value function of the set of states visited during on-line trials, assuming that initial states were generated according to some fixed distribution. The policy used for the trials is the optimal policy for the current value function. Since updated and cached values from one step are used by other steps, this approach can be viewed as an asynchronous form of DP from Figure 1(b) (with non-uniform update depth). Convergence of RTDP requires that $V_h(s)$ be an *admissible upper bound* value function, which overestimates the true value, i.e., $\forall s \in S.\ V_h(s) \geq V^*(s)$.

One of the key advantages of the RTDP framework is that it may only need to explore a small subset of states to obtain an optimal policy π^* w.r.t. the set of initial states if the subset of states reachable from the initial states under π^* (the *rele-*

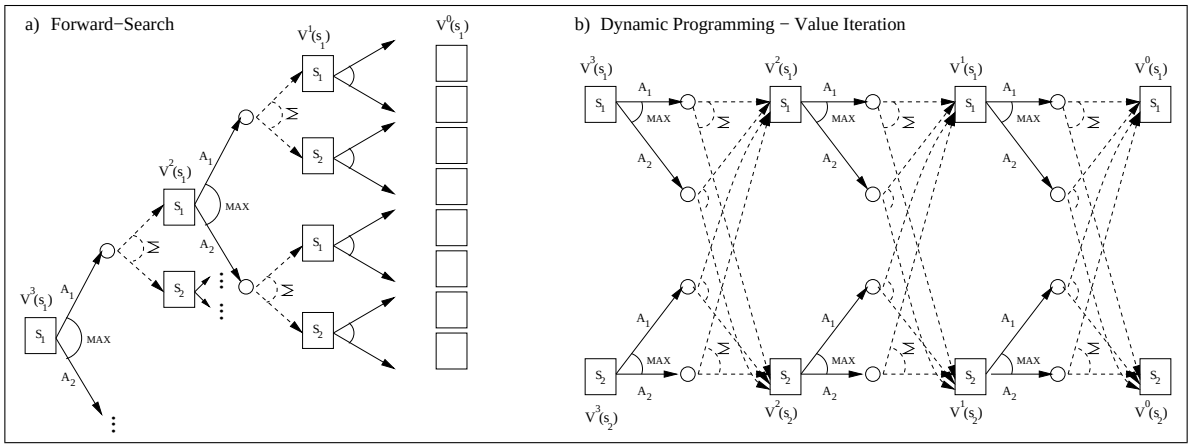


Figure 1: a) Asynchronous forward-search update for a specific state and b) synchronous value iteration update for all states.

vant states for π^*) is small. This can often be more efficient than synchronous DP approaches when the set of reachable states is small compared to the total number of states.

Theorem 1 (Barto, Bradtke, & Singh, 1993) *If $\hat{V}_h$ is initialized with an admissible value function then it eventually converges to V^* on the set of relevant states of an optimal policy under repeated RTDP trials for (a) any discounted MDP and (b) any SSP with restricted initial state where at least one policy reaches the goal with probability one.*

As noted in the original description of the RTDP algorithm (Barto, Bradtke, & Singh 1993), the *depth of* UPDATE*($\hat{V}_h, s$) can be optimized*. However, this is not generally done in the current literature, with the exception of a pass to backup all states in a trial in reverse order, e.g., as done in Labeled RTDP (LRTDP) (Bonet & Geffner 2003b).

Policy Evaluation and Search

We note that most dynamic programming approaches to solving MDPs are *anytime* algorithms, i.e., they can be stopped at any point during execution before convergence and provide a solution estimate $\hat{V}$. It is typically assumed (and often easily proved) that the more time an anytime DP algorithm is given, the closer $\hat{V}$ is to convergence.

When an anytime DP algorithm returns a value function estimate $\hat{V}$ that has not converged, one important question is how to evaluate a policy w.r.t. $\hat{V}$? We could evaluate the greedy policy for $\hat{V}$ as given in Equation 6, but we note that this may be suboptimal since $\hat{V}$ itself may not be optimal. Noting that policy evaluation in Equation 6 is the same as an $\arg\max_a$ applied to the Bellman update of Equations 3 and 4, we could *optimize the depth of a forward-search update* as previously discussed in order to obtain a potentially better value estimate and thus a potentially better policy.

But then the same question arises here as did for the case of the RTDP update discussed previously. How do we optimize the update search depth in both of these cases to ensure that this search computation (and thus time) is not wasted? We answer this in a Bayesian search control framework next.

Bayesian Search Control

The current UPDATE($\hat{V}_h, s$) procedure used in most algorithms is the uniform depth-one update of Equations 3 and 4. However, as motivated previously, the question we now want to ask is how we can dynamically adapt this update depth? That is, given the task of updating $V(s)$ for some $s \in S$ w.r.t. the value $V(t)$ of potential successor states $t \in S$ of s, how do we determine which $V(t)$ to recursively update, and which $V(t)$ to use as currently estimated? More precisely, we ask whether the information gain of knowing $V(t)$ more precisely justifies the time for the additional search required?

We answer this question through a Bayesian value of information framework (Howard 1966) where we balance the potential gain of exploration with its additional cost based on our current beliefs. To obtain a Bayesian framework, we need some manner of assigning probabilities to our current value beliefs, which we provide shortly.

First, however, we define a simple vectorial notation that will simplify the notation throughout our analysis. We often switch between function notation $V(s)$ for our value function and a vector $\vec{V}$ of length $|S|$. Noting that the $Q^t(s,a)$ computation in Equation 3 is a linear function of $\vec{V}^{t-1}$, we extract the coefficients of $\vec{V}^{t-1}$ from Equation 3 and store them in a vector $\vec{\Gamma}_{a,s}$ of length $|S|$. Then we can rewrite Equations 3 and 4 in the following form:

$$Q_{a,s} := \vec{\Gamma}_{a,s} \cdot \vec{V}^{t-1} \quad (7)$$

$$V^t(s) := \max_{a \in A} Q_{a,s} \quad (8)$$

Bounds and Belief Distributions

Let $\vec{V}_l$ and $\vec{V}_h$ respectively represent an entire vector of lower and upper bounds for all states and let specific elements of these vectors be denoted by $V_l(s)$ and $V_h(s)$. Let $\vec{\theta} = \langle \vec{V}_l, \vec{V}_h \rangle$. While the bounds $V_h(s)$ and $V_l(s)$ for state s may be correlated with the bounds $V_h(s')$ and $V_l(s')$ for any state s' reachable from s under some policy π on account of the Bellman equations, determining these correlations is

tantamount to performing DP backups and thus they cannot be determined without solving the underlying MDP.

Given that we have no additional immediate knowledge about possible belief values $v_s \in [V_h(s), V_l(s)]$ for state s, we can only reasonably assume that v_s is uniformly distributed between these lower and upper bounds.[1] Overloading notation, we let $\vec{V} = \mathbb{R}^{|S|}$ represent a random vector of value beliefs. Then $\vec{\theta}$ can be used to parameterize a multivariate uniform distribution $P(\vec{v}|\vec{\theta})$ for $\vec{v} \in \vec{V}$ that is consistent with the upper and lower bounds $\vec{\theta}$. We can explicitly represent $P(\vec{v}|\vec{\theta}) = \prod_s P(v_s|\vec{\theta})$ where $P(v_s|\vec{\theta})$ is a univariate uniform density over the values $v_s \in [V_l(s), V_h(s)]$ for state s (and a Dirac delta function $\delta_{V_l(s)}(v_s)$ in the special case that $[V_l(s) = V_h(s)]$).

Before we defined admissibility for the initial upper bound $V_h(s)$, now we must also define initial *admissible lower bounds* $V_l(s)$: $\forall s \in S.\ V_l(s) \leq V^*(s)$. We assume that both upper and lower bounds are respectively admissible and note that subsequent DP updates preserve this property (McMahan, Likhachev, & Gordon 2005).

Optimizing the Update Depth

With a uniform parameterization over our value beliefs, we can write out the integral for our *expected* value of $Q(a, s)$ under the current beliefs and evaluate it in closed form:

$$E[Q_{a,s}|\vec{\theta}] = \int_{\vec{v}} \prod_{s'} P(v_{s'}|\vec{\theta}) \left[\vec{\Gamma}_{a,s} \cdot \vec{v}\right] d\vec{v}$$
$$= \vec{\Gamma}_{a,s} \cdot \frac{\vec{V}_h + \vec{V}_l}{2} \quad (9)$$

Now we return to the original question of how to determine the states $t \in S$ for which the potential information gain of updating value $V(t)$ exceeds the time cost required for this additional update. Given that we do not know this gain exactly, we use a value of perfect information framework (Howard 1966) where we assume that a clairvoyant source informs us of the true value $v_t^* = V^*(t)$ for state t.

Since we can assume external knowledge of v_t^*, we can use this to refine our beliefs where v_t^* replaces the previous upper and lower bounds for state t in $P(v|\vec{\theta})$:

$$E[Q_{a,s}|\vec{\theta}, v_t^*] = \int_{\vec{v}} \delta_{v_t^*}(v_t) \prod_{s' \neq t} P(v_{s'}|\vec{\theta}) \left[\vec{\Gamma}_{a,s} \cdot \vec{v}\right] d\vec{v} \quad (10)$$

Now, whereas Equation 9 evaluated to a constant since all values in $\vec{V}_h$ and $\vec{V}_l$ are known, this equation evaluates to a linear function of v_t^* since all values except for v_t^* are constants. This linear function may simply be expressed as

$$E[Q_{a,s}|\vec{\theta}, v_t^*] = c_{(a,s,t,\vec{\theta})} + d_{(a,s,t,\vec{\theta})} v_t^* \quad (11)$$

where $c_{(a,s,t,\vec{\theta})}$ and $d_{(a,s,t,\vec{\theta})}$ are constants that can be easily determined from Equation 10 given their subscripts.

To evaluate the gain of knowing v_t^*, we use the analytical framework of Dearden *et al.* (Dearden, Friedman, & Russell 1998) adapted to our setting. Let $a^* = \arg\max_a E[Q_{a,s}|\vec{\theta}]$ (i.e., the current best action in expectation for state s). We can evaluate the gain of knowing v_t^* by determining how much it will increase future reward:

$$Gain_t(v_t^*) = \quad (12)$$
$$\max\left(0, \max_{a \in A}\{E[Q_{a,s}|\vec{\theta}, v_t^*] - E[Q_{a^*,s}|\vec{\theta}, v_t^*]\}\right)$$

In short, $Gain_t(v_t^*)$ is only non-zero when knowledge of v_t^* indicates that some other action a is more optimal than the current best choice of a^* and then the gain is the difference in utility. Since we are looking at the gain over multiple actions, we must take the maximum positive gain possible.

In reality, we do not know v_t^*, but we do have current beliefs over its value. Thus, we can write out the expected value of perfect information about v_t^*:

$$VPI(t) = \int_{v_t^*=-\infty}^{\infty} Gain_t(v_t^*) P(v_t^*|\vec{\theta}) dv_t^*$$
$$= \frac{1}{V_h(t) - V_l(t)} \int_{v_t^*=V_l(t)}^{V_h(t)} Gain_t(v_t^*) dv_t^* \quad (13)$$

$VPI(t)$ now provides us with an optimistic upper bound estimate on the myopic value of perfect information for updating the value of state t. If we consider the cost for this node expansion to be some constant η measured in units commensurable with reward[2], then our analysis shows that we should recursively update the value $V(t)$ if $VPI(t) > \eta$.

Implementation

At first, $VPI(t)$ might seem difficult to evaluate due to the integral and implicit maximization in the $Gain_t(v_t^*)$ equation. But in Figure 2 we show that the calculation only requires a univariate integral over a piecewise linear function and it also admits a simple and efficient approximation.

Algorithmically, *Bayesian search control* reduces to performing recursive applications of UPDATE(s) to all reachable states t that have $VPI(t) > \eta$ (with an additional modification to avoid infinite loops). This provides us with the new UPDATE($\vec{V}_h, \vec{V}_l, s$) in Algorithm 2 that together with maintaining admissibly initialized upper and lower bounds is the only modification required to convert RTDP to *Bayesian RTDP*[3] (or its variant using the approximation to VPI from Figure 2). We can also define *Bayesian policy evaluation* (and its approximate variant) simply by calling UPDATE($\vec{V}_h, \vec{V}_l, s$) to refine the value of s's successor states before evaluating the action a to take in state s.

[1] We note that this assumption is not simply one of convenience. Without knowing how values were updated or being able to determine correlations between them, we have no more reason to believe that the true value v_s^* is at the mean $[V_h(s) + V_l(s)]/2$ rather than at one of the boundaries $V_h(s)$ or $V_l(s)$, or anywhere in between. It is important to note that for model-based DP, the value updates are not sampled in a statistical sense and thus the central limit theorem and associated normality assumptions do not apply.

[2] Or it could scale with the number of actions and/or next states, which might be pre-cached for some problems.

[3] We trivially note that Theorem 1 still holds for Bayesian RTDP since additional DP updates can never hurt the convergence of asynchronous value iteration methods such as RTDP.

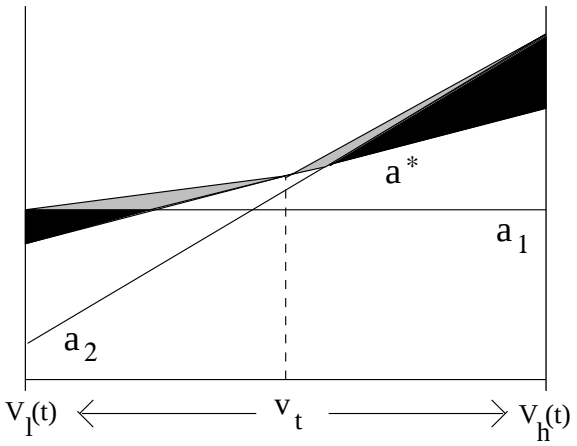


Figure 2: A graphical representation of the $VPI(t)$ calculation. Note that a^* is the current optimal action (for the current midpoint belief estimates) and the only values for which $Gain_t(v_t^*)$ is non-zero are indicated by the black filled area. Thus the $VPI(t)$ calculation in Equation 13 simply corresponds to the black area multiplied by $1/(V_h(t) - V_l(t))$. The *approximation* of this is the union of the black and gray area; it bounds the maximum $VPI(t)$ by evaluating only the endpoints and the midpoint.

Preliminary Empirical Results

We evaluated Bayesian search control for the DP solution and real-time policy evaluation of racetrack benchmark problems (Barto, Bradtke, & Singh 1993; Smith & Simmons 2006) shown in Figure 4. The state in this problem is a combination of a car's coordinate position $\langle x, y \rangle$ and velocity $\langle x', y' \rangle$ in each coordinate direction. A car begins at one of the initial start states (chosen uniformly randomly) with velocity $\langle x', y' \rangle = \langle 0, 0 \rangle$. Actions $\langle x'', y'' \rangle$ available to the car are integer accelerations $\{-1, 0, 1\}$ in each coordinate direction. If the car hits a wall, then its velocity $\langle x', y' \rangle$ is reset to $\langle 0, 0 \rangle$. A car may skid with probability .1, thus resulting in an action outcome equivalent to 0 acceleration in each direction. With probability .1 the wind may perturb the commanded acceleration by a uniform choice of $\{\langle -1, -1 \rangle, \langle -1, 0 \rangle, \langle -1, 1 \rangle, \langle 0, -1 \rangle, \langle 0, 1 \rangle, \langle 1, -1 \rangle, \langle -1, 0 \rangle, \langle 1, 1 \rangle\}$. Finally, if the car passes over a grayed "rough" patch in the diagram, the immediate reward is -10, for all other "clear" patches, the immediate reward is -1. We use discount $\gamma = 1$.

While these problems represent a small *subset* of possible oversubscribed MDPs (i.e., only one goal state of many can be achieved, each with a different cumulative cost), these preliminary results are indicative of the ability of Bayesian search control methods to reason about the trade-offs of achieving each goal in oversubscribed planning.

Dynamic Programming and Search Results

We have already defined the *Bayesian RTDP* algorithm for dynamically adapting the backup depth along with its approximate variant that we evaluate. We also evaluate other RTDP algorithms and their (approximate) Bayesian variants: Labeled RTDP (LRTDP) (Bonet & Geffner 2003b) and

Algorithm 2: UPDATE($\hat{V}_h, \hat{V}_l, s$) for Bayesian Search

begin
  // Mark state as visited
  $s.visited :=$ *true*;
  foreach $t \in$ {*states reachable from* s} **do**
    // Update t *if its VPI exceeds the computation cost*
    if $(\neg t.visited \ \wedge \ VPI(t) > \eta)$ **then**
      UPDATE($\hat{V}_h, \hat{V}_l, t$);
  // Unmark state as visited
  $s.visited :=$ *false*;
  // Update upper and lower bounds (after recursion)
  $\hat{V}_l(s) := \max_{a \in A} R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') \cdot \hat{V}_l(s')$;
  $\hat{V}_h(s) := \max_{a \in A} R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') \cdot \hat{V}_h(s')$;
end

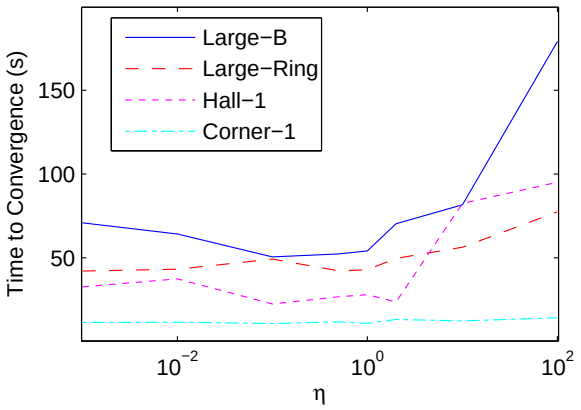
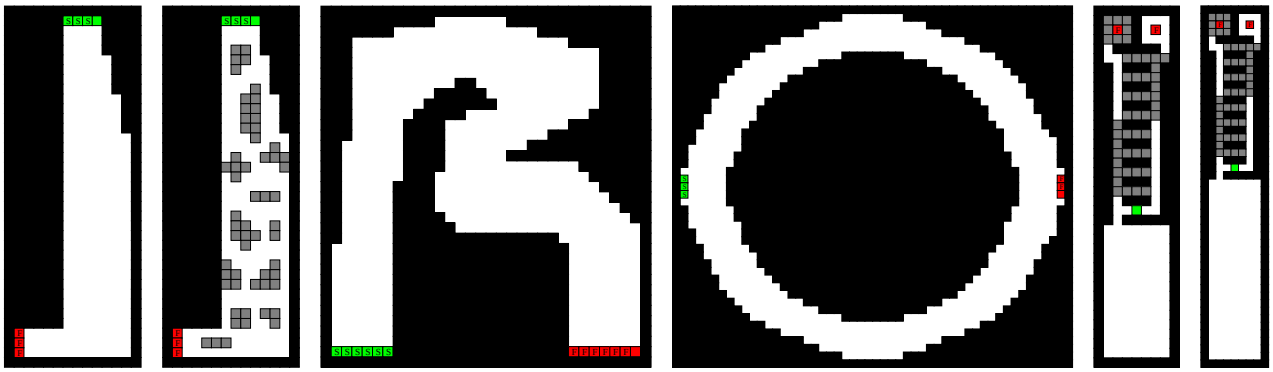


Figure 3: Time to convergence for different values of η for Bayesian RTDP on various problems.

Focused RTDP (FRTDP) (Smith & Simmons 2006). We discuss these variants in the Related Work, it suffices to state that the Bayesian variants of these algorithms simply use the same dynamic backup depth modification as Bayesian RTDP. As one final algorithm variant to evaluate, we replace the $VPI(t)$ calculation in UPDATE($\vec{V}_h, \vec{V}_l, s$) with the *bound gap* corresponding to $\vec{V}_h(t) - \vec{V}_l(t)$ that represents the true value uncertainty for state t. The bound gap is a heuristic used to focus trial trajectories in Bounded RTDP (McMahan, Likhachev, & Gordon 2005) and FRTDP.

We examined the performance of the various RTDP algorithms initialized with uninformed admissible upper and lower bounds set to respective maximum and minimum values for a max trial length of 100. We found $\eta = 1$ to generally yield good results for both Bayesian and bound gap RTDP. In Figure 3 we analyze the performance of Bayesian RTDP for various settings of the computation-information tradeoff parameter η. If η is too small, it encourages useless exploration that costs time, while if η is too large then it does not take advantage of value of information except in extreme cases, thus effectively wasting the time required to compute it. We note that empirically, the best setting for η across these four problems seems to be between 0.1 and 1.



(a) Corner-1 (b) Corner-2 (c) Large-B (d) Large-Ring (e) Hall-1 (f) Hall-2

Figure 4: Various racetrack domains evaluated in this paper. Initial states are green and labelled 'S', terminal states are red and labelled 'F'. Black squares delineate walls, gray squares incur cost -10 for passing over, and white squares incur cost -1 for passing over.

	Corner-1	Corner-2	Large-B	Large-Ring	Hall-1	Hall-2
RTDP	**7.8** (1.7e+04)	**42.0** (8.8e+05)	85.6 (6.4e+07)	38.8 (1.3e+07)	23.2 (1.9e+07)	426.7 (2.9e+08)
Bound Gap RTDP	10.4 (2.9e+04)	50.1 (3.5e+05)	**28.5** (5.4e+06)	**28.6** (**3.1e+06**)	**10.1** (2.4e+06)	**63.6** (**1.1e+07**)
Approx. Bay. RTDP	9.8 (3e+04)	**42.3** (7.9e+05)	153.3 (7.9e+07)	40.0 (8.6e+06)	12.8 (**1.9e+06**)	775.2 (1.4e+08)
Bayesian RTDP	8.3 (**9e+03**)	52.5 (**2.7e+05**)	92.0 (**3.6e+06**)	43.1 (9.8e+06)	17.2 (5.7e+06)	557.5 (8.7e+07)
LRTDP	7.4 (1.9e+05)	**26.2** (9.5e+04)	**28.7** (1.9e+06)	34.3 (1.0e+06)	9.2 (1.5e+05)	**54.9** (7.6e+05)
Bound Gap LRTDP	**7.1** (1.8e+05)	29.3 (**1.1e+05**)	29.1 (2.4e+06)	**32.5** (9.4e+05)	**8.7** (1.5e+05)	**54.8** (7.4e+05)
Approx. Bay. LRTDP	7.4 (1.8e+05)	29.3 (1.3e+05)	29.5 (2.2e+06)	32.9 (9.8e+05)	**8.9** (**1.3e+05**)	**54.7** (**7e+05**)
Bayesian LRTDP	7.6 (**1.7e+05**)	29.2 (**1.1e+05**)	29.6 (**2.1e+06**)	**32.7** (**9.3e+05**)	**9.1** (**1.3e+05**)	55.1 (7.5e+05)
FRTDP	7.2 (**1.6e+05**)	**29.0** (1.3e+05)	22.1 (5.6e+05)	28.7 (4.3e+05)	9.0 (1.3e+05)	**49.0** (3e+05)
Bound Gap FRTDP	**7.1** (1.7e+05)	29.7 (1.4e+05)	**21.3** (5.8e+05)	**27.1** (4.4e+05)	**8.7** (1.4e+05)	52.0 (3e+05)
Approx. Bay. FRTDP	7.6 (**1.6e+05**)	29.9 (**1.2e+05**)	22.1 (**5e+05**)	28.5 (**3.8e+05**)	8.9 (**1.1e+05**)	52.9 (**2.9e+05**)
Bayesian FRTDP	7.7 (1.7e+05)	30.1 (1.3e+05)	22.1 (**5e+05**)	28.8 (**3.8e+05**)	9.0 (**1.1e+05**)	53.0 (**2.9e+05**)

Table 1: Time (s) and (# of backups) to convergence ($\epsilon = 10^{-3}$) for RTDP variants on each problem (statistically indistinguishable best values within 95% confidence bounds for a problem are bolded). Overall, Bayesian approaches often require the least number of backups.

Time and number of backups to convergence on the racetrack benchmarks are recorded in Table 1. Values were averaged over 30 runs. One feature of the top four rows is that Bayesian RTDP (and its approximate variant) often converged with an order of magnitude fewer backups than plain RTDP. However, the computational overhead of (approximate) Bayesian RTDP can be substantial and thus this backup savings did not always translate to time savings. The bound gap approach to dynamically adapting the backup depth also performed well for reasons we discuss later.

The bottom eight rows of Table 1 show the performance of other RTDP variants on these same problems demonstrating the same general trends as before, although with lower variance among search approaches for any given algorithm and problem. It appears that unlike RTDP, the heuristics and other provable mechanisms for restricting search in LRTDP and FRTDP perform well at focusing the search so that additional dynamic backup depth modifications do not contribute substantial additional savings.

To further investigate the performance of Bayesian RTDP vs. RTDP, in Figure 5 we provide a plot of the area between the lower and upper bounds for both algorithms on Corner-2. As a function of the number of backups it is immediately clear that Bayesian RTDP manages to quickly reduce its error bounds demonstrating that the expected value of information analysis in the backups has a substantial payoff in terms of reducing error early on.

So, if Bayesian variants of RTDP reduce error more quickly and perform fewer backups, why is convergence time not substantially improved? One obvious issue is the computational overhead of the $VPI(t)$ calculation. Another important issue *seemingly unacknowledged in the Bayesian search-control literature* (see Related Work) is that $VPI(t)$ is not always an informative heuristic for obtaining dynamic programming convergence.[4] An example of this is shown in Figure 6 where we see that a^* completely dominates the other actions, resulting in $VPI(t) = 0$. However, the value of Q_{s,a^*} is still highly uncertain as v^t varies; thus further refining the value of Q_{s,a^*} could be useful for the convergence of other states that *depend* on the accuracy of Q_{s,a^*}. While $VPI(t)$ may not yield ideal search control for the convergence of dynamic programming, it is sensitive to how value changes may affect the optimal policy, indicating that it may be better applied to policy search as we evaluate next.

[4]The bound gap serves as a surrogate for such a heuristic although it may not always focus computation where uncertainty can be reduced. E.g., bound gap needlessly explores the "room" at the bottom of Hall-1 where it may be difficult to reduce uncertainty.

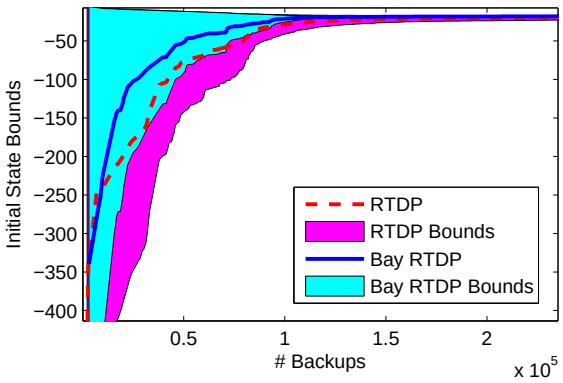


Figure 5: Plot of area between lower and upper bounds on the initial state value for RTDP and Bayesian RTDP vs. the number backups made for an uninformed solution to Corner-2. Note that the Bayesian RTDP bounds overlay on the RTDP bounds. A line following the midpoint of the bounds is shown for each algorithm.

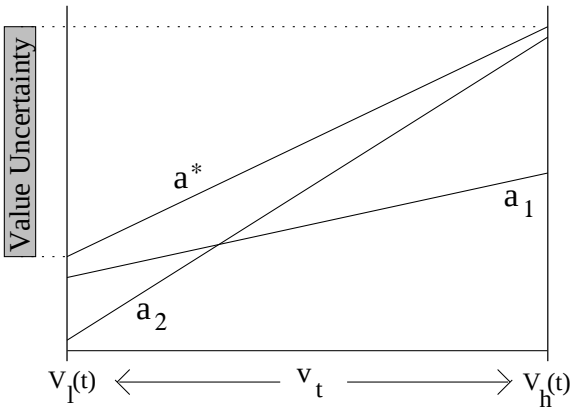


Figure 6: A hypothetical case where $VPI(t) = 0$, yet further refining the value of Q_{s,a^*} could be useful for other states that depend on this value.

Policy Evaluation and Search Results

We experimented with various policy evaluation approaches using the upper and lower bound value function estimates produced by an anytime version of RTDP stopped before convergence. The four approaches that we evaluate are *no search* (just direct policy evaluation from Equation 6), *Bayesian* and its approximate variant as previously discussed, and *bound gap* that uses the same approach as the *Bayesian* policy evaluation methods except that $VPI(t)$ is replaced with the bound gap corresponding to $\vec{V}_h(t) - \vec{V}_l(t)$.

Results of these policy evaluation approaches for non-converged value functions on various problems are provided in Table 2 and at various stages of convergence for Large-Ring in Figure 7. The overall result is that given a fixed time budget, Bayesian search control methods always perform as well as other methods and most often outperform the other methods by a large margin (e.g., Large-B, Hall-1, and Hall-2 in Table 2 and at most time points in Figure 7). These results are consistent with our previous analysis — Bayesian VPI analysis will avoid updating uncertain states if it knows a better value estimate is unlikely to change the optimal policy, thereby leading to effective search-control for policy evaluation. On the other hand, policy search prioritized by bound gap shows major deficiencies w.r.t. Bayesian approaches indicating that the uncertainty bound gap may not be the best search heuristic for policy evaluation.

Related Work

Trading off computation time and reward in a fully decision-theoretic framework where deliberative actions are afforded the same status as actions that directly interact with the environment is a well-known problem in AI and decision-theoretic planning. We refer the reader to Russell and Wefald (Russell & Wefald 1991) for an excellent historical review of such ideas as well as a meta-reasoning approach that can be seen as a high-level motivating thread of the research presented here. Search control itself is major focus of the planning community and thus we focus our related work discussion on search control methods used for RTDP as well as other Bayesian search control approaches.

Various extensions of RTDP provide different search control strategies. Labeled RTDP (LRTDP) (Bonet & Geffner 2003b) improves on basic RTDP by labeling states as solved when their values (and the values of states reachable from them) have converged, thus not needing to consider them for updates later on. Heuristic Dynamic Programming (HDP) (Bonet & Geffner 2003a) combines an even stronger version of this labeling approach with a heuristic algorithm that searches for states with inconsistent values to update. Bounded RTDP (BRTDP) (McMahan, Likhachev, & Gordon 2005) uses the bound gap[5] to focus trial trajectories and thus updates, on the states with most uncertainty. Focused RTDP (FRTDP) (Smith & Simmons 2006) improves on this by taking into account the state occupancy probability under the current policy. However, *none* of these RTDP extensions optimize the dynamic programming backup depth using Bayesian VPI search methods nor do they propose using Bayesian search methods for the anytime policy evaluation of their unconverged solutions. As such, the Bayesian search control approaches introduced in this paper can be easily integrated into the above RTDP variants as we partially demonstrated in the experimental results.

There are also a variety of search control strategies that attempt to exploit value of information in some manner. Bayesian Q-learning (Dearden, Friedman, & Russell 1998) presents a technique to balance exploration and exploitation in a *model-free* Q-learning approach. To estimate the expected gains on future decisions, a normal-gamma distribution is assumed for the utility values of *sampled* state-action pairs, which allows the computation of an information value for actions (not states as done here). Work on control strategies (Tash & Russell 1994) also enables a planning agent to choose where to focus its computational effort, but requires externally provided variance estimates of state-values. The DRIPS planner (Haddawy, Doan, & Goodwin 1995) searches for plans using a hierarchical approach, start-

[5] See the experimental results for a detailed discussion of how the bound gap used by BRTDP and FRTDP differ from the Bayesian VPI approach in this paper.

	Corner-1	Corner-2	Large-B	Large-Ring	Hall-1	Hall-2
No Search	-44.6 ± 3.6	-92.9 ± 2.7	-54.6 ± 1.5	-47.5 ± 1.5	-86.1 ± 1.7	-191.1 ± 5.7
Bound Gap	**-30.6 ± 1.6**	-74.9 ± 1.7	-53.1 ± 1.3	**-40.1 ± 1.1**	-89.1 ± 1.6	-149.4 ± 3.7
Approx. Bayesian	**-33.4 ± 2.0**	-57.9 ± 1.1	-54.8 ± 1.5	**-38.2 ± 0.9**	-65.5 ± 1.5	-141.8 ± 3.5
Bayesian	**-33.0 ± 2.4**	**-53.5 ± 1.2**	**-44.1 ± 0.8**	**-40.6 ± 1.0**	**-57.0 ± 1.3**	**-103.2 ± 2.8**

Table 2: Average cumulative reward per trial (with 95% confidence intervals and best values bolded) using various (search-based) policy evaluation approaches on the non-converged value function produced by stopping RTDP (with upper and lower bounds) after running approximately 1/4 of the time it would need to converge. Search-based methods used $\eta = 1$ and were given a maximum of 100ms.

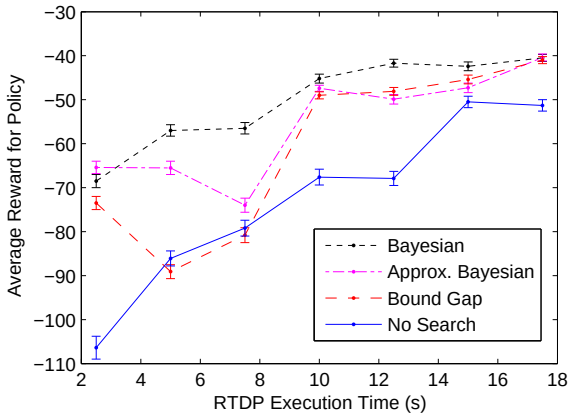


Figure 7: Average cumulative reward per trial (with 95% confidence intervals) using various (search-based) policy evaluation approaches on the non-converged value function produced by stopping upper- and lower-bounded RTDP on Large-Ring at various times (on the x-axis). The search-based methods used $\eta = 1$ and were given a maximum of 100ms.

ing from an abstract, high-level plan and iteratively specializing it until a ground plan is computed. Estimates of plan performances are used both to eliminate suboptimal plans and a sensitivity analysis of these performances is used to select which abstract actions to specify first. The sensitivity analysis used in DRIPS resembles a crude approximation of the value of perfect information we derived here that only evaluates alternate actions at the upper and lower bounds, but we note that even better approximations provided here did not perform as well on policy evaluation as the exact Bayesian VPI calculation that we derived. The work of Dearden, et al. (2003) uses plan value estimates in planning problems with temporally extended actions and continuous resources. Employing a seed-plan, their approach tries to find beneficial contingent branches through a value of information analysis similar to what we employ here, however, they express beliefs over the resources, but not over the value function itself as we do here. Finally, Goodwin (1996) provides a general reference list of other meta-level search control heuristics although none of these approaches adopt the Bayesian value framework and value of information analysis proposed here.

Concluding Remarks

We presented a novel Bayesian value of information analysis that can be used for various search control tasks in the real-time solution of MDPs (and thus many oversubscribed planning problems as explained in the Introduction). We then applied these Bayesian search control ideas to dynamically adjust the backup depth in RTDP algorithms and to perform search control for real-time policy evaluation. While Bayesian search control for RTDP often reduces the number of backups required to converge, this did not always translate to a clear time savings for various reasons that we analyzed. A key observation, however, was that Bayesian search control proved an excellent tool for avoiding state updates that do not change the optimal policy, thus leading to excellent empirical performance on real-time policy evaluation experiments. As such, Bayesian search control poses an attractive real-time policy evaluation approach for MDPs; certainly, the encouraging results here warrant further exploration of such ideas.

References

Barto, A. G.; Bradtke, S. J.; and Singh, S. P. 1993. Learning to act using real-time dynamic programming. Technical Report UM-CS-1993-002, U. Mass. Amherst.

Bertsekas, D. P., and Tsitsiklis, J. N. 1996. *Neuro-Dynamic Programming*. Belmont, MA: Athena Scientific.

Bertsekas, D. P. 1982. Distributed dynamic programming. *IEEE Transactions on Automatic Control* 27:610–617.

Bonet, B., and Geffner, H. 2003a. Faster heuristic search algorithms for planning with uncertainty and full feedback. In *Proc. IJCAI-2005*, 1233–1238. Acapulco, Mexico.

Bonet, B., and Geffner, H. 2003b. Labeled RTDP: Improving the convergence of real-time dynamic programming. In *ICAPS-03*.

Dearden, R.; Meuleau, N.; Ramakrishnan, S.; Smith, D.; and Washington, R. 2003. Incremental contingency planning. In *ICAPS Workshop on Planning under Uncertainty*.

Dearden, R.; Friedman, N.; and Russell, S. J. 1998. Bayesian Q-learning. In *AAAI/IAAI-1998*, 761–768.

Goodwin, R. 1996. *Meta-Level Control for Decision-Theoretic Planners*. Ph.D. Dissertation, Carnegie Mellon University, PA.

Haddawy, P.; Doan, A.; and Goodwin, R. 1995. Efficient decision-theoretic planning techniques. In *AAAI-1995*, 229–236.

Howard, R. A. 1966. Information value theory. *IEEE Transactions on Systems Science and Cybernetics* SSC-2(1):22–26.

McMahan, H. B.; Likhachev, M.; and Gordon, G. J. 2005. Bounded real-time dynamic programming In *ICML-05*, 569–576.

Puterman, M. L. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York: Wiley.

Russell, S., and Wefald, E. 1991. Principles of metareasoning. *Artificial Intelligence* 49(1-3):361–395.

Smith, T., and Simmons, R. G. 2006. Focused real-time dynamic programming for MDPs In *AAAI-06*.

Tash, J., and Russell, S. 1994. Control strategies for a stochastic planner. In *(AAAI-94)*, vol. 2, 1079–1085.