Bayesian Real-time Dynamic Programming

Scott Sanner
SML Group
National ICT Australia
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

Guy Shani
MLAS Group
Microsoft Research
Redmond, WA, USA
guyshani@microsoft.com

Abstract

Real-time dynamic programming (RTDP) solves Markov decision processes (MDPs) when the initial state is restricted, by focusing dynamic programming on the envelope of states reachable from an initial state set. RTDP often provides performance guarantees without visiting the entire state space. Building on RTDP, recent work has sought to improve its efficiency through various optimizations, including maintaining upper and lower bounds to both govern trial termination and prioritize state exploration. In this work, we take a Bayesian perspective on these upper and lower bounds and use a value of perfect information (VPI) analysis to govern trial termination and exploration in a novel algorithm we call VPI-RTDP. VPI-RTDP leads to an improvement over state-of-the-art RTDP methods, empirically yielding up to a three-fold reduction in the amount of time and number of visited states required to achieve comparable policy performance.

1 Introduction

Markov Decision Processes (MDPs) [Puterman, 1994] provide a convenient framework for modeling fully-observable stochastic planning problems. In an MDP, the agent computes a policy — a mapping from states to actions — in order to maximize a stream of rewards. A popular approach to policy computation is through a value function — a function that assigns a value to each world state. The computation of the value function can be either synchronous, where all states are updated during each iteration, or asynchronous, where the agent updates some states more than others.

Recent years have seen a resurgence of interest in asynchronous dynamic programming solutions to MDPs [Bertsekas, 1982]. Of particular interest has been the trial-based real-time dynamic programming (RTDP) approach [Barto *et al.*, 1993] as evidenced by a variety of recent work [Bonet and Geffner, 2003a; 2003b; McMahan *et al.*, 2005; Smith and Simmons, 2006]. RTDP algorithms have a number of distinct advantages for practical MDP solutions, specifically:

(1) *Anytime performance:* RTDP algorithms can be interrupted at any time, generally yielding a better solution the longer they are allowed to run.

(2) *Optimality without exhaustive exploration:* By focusing trial-based search on states reachable from the set of initial states, RTDP algorithms may obtain an optimal policy while visiting only a fraction of the state space.

Recent state-of-the-art advances in RTDP algorithms such as bounded RTDP (BRTDP) [McMahan *et al.*, 2005] and focused RTDP (FRTDP) [Smith and Simmons, 2006] propose (a) maintaining upper and lower bounds on the value function; (b) using the policy derived from the lower bound to provide guarantees on policy performance; and (c) directing exploration (and termination) by the uncertainty of a state's value, as measured by the gap between its upper and lower value bounds. As BRTDP and FRTDP both prioritize search according to value uncertainty, they may execute needless updates in areas of the state space where the policy has converged, but the values have not.

To address this deficiency, we take a Bayesian perspective on the upper and lower bounds in order to express a belief distribution over value functions. We then use this distribution in a myopic value of perfect information (VPI) [Howard, 1966] framework to approximate the expected improvement in decision quality resulting from the update of a state's value. This leads us to the development of a novel algorithm called VPI-RTDP that directs exploration (and termination) according to this VPI analysis. Empirically, VPI-RTDP results in an improvement over state-of-the-art RTDP methods, yielding up to a three-fold reduction in the amount of time and unique states visited to achieve comparable policy performance.

2 Background

2.1 Markov Decision Processes

A Markov decision process (MDP) is 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. $0 \le \gamma \le 1$ where rewards k time steps in the future are discounted by γ^k. There is a set of initial states $\mathcal{I} \subseteq S$, and a possibly empty set of absorbing goal states $\mathcal{G} \subset S$ where all actions lead to a zero-reward self-transition with probability 1.

Algorithm 1: RTDP

begin
  // Initialize $\hat{V}_h$ with admissible value function
  $\hat{V}_h := V_h$
  while convergence not detected and not out of time **do**
    $depth := 0$
    $visited.$CLEAR() // Clear visited states stack
    Draw s from $\mathcal{I}$ at random // Pick initial state
    while $(s \notin \mathcal{G}) \wedge (s \neq null) \wedge (depth < \text{max-depth})$
    do
      $depth := depth + 1$
      $visited.$PUSH(s)
      $\hat{V}_h(s) :=$ UPDATE($\hat{V}_h, s$) // See (2) & (3)
      $a :=$ GREEDYACTION($\hat{V}_h, s$) // See (4)
      $s :=$ CHOOSENEXTSTATE(s, a) // See (5)

    // The following end-of-trial update is an optimization
    // not appearing in the original RTDP
    while ¬$visited.$EMPTY() **do**
      $s := visited.$POP()
      $\hat{V}_h(s) :=$ UPDATE($\hat{V}_h, s$)

  return $\hat{V}_h$
end

Algorithm 2: CHOOSENEXTSTATE-BRTDP(s, a)

begin
  // Compute bound gap of reachable states s', use to select
  $\forall s', \ b(s') := T(s, a, s')(\hat{V}_h(s') - \hat{V}_l(s'))$
  $B := \sum_{s'} b(s')$
  if $B < \frac{\hat{V}_h(s) - \hat{V}_l(s)}{\tau}$ **then**
    return $null$
  return $s' \sim \frac{b(\cdot)}{B}$
end

A policy $\pi : S \to A$ specifies the action $a = \pi(s)$ to take in state s. Our goal is to find a policy that maximizes the value function, defined as the sum of expected discounted rewards

$$V_\pi(s) = E_\pi \left[\sum_{k=0}^{\infty} \gamma^k \cdot r_k \middle| s_0 = s \right] \quad (1)$$

where r_k is the reward obtained at time step k.

2.2 Synchronous Dynamic Programming (DP)

Value iteration (VI) is a *synchronous* dynamic programming (DP) solution to an MDP. Starting with an arbitrary $V^0(s)$, VI performs value updates for *all* states s, computing the next value function $V^k(s) :=$ UPDATE(V^{k-1}, s):

$$Q^k(s, a) := R(s, a) + \gamma \cdot \sum_{s' \in S} T(s, a, s') \cdot V^{k-1}(s') \quad (2)$$

$$V^k(s) := \max_{a \in A} \left\{ Q^k(s, a) \right\}. \quad (3)$$

This update is known as a *Bellman update*.

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

$$\pi(s) := \arg\max_{a \in A} \left\{ R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') \cdot V^k(s') \right\} \quad (4)$$

After some finite number of iterations k of VI, the greedy policy with respect to V^k is provably optimal [Puterman, 1994].

2.3 Asynchronous DP and Real-time DP

Asynchronous DP methods [Bertsekas, 1982] are a variant of dynamic programming that apply the Bellman update to states in an arbitrary order while still retaining convergence properties under certain conditions. The real-time dynamic programming (RTDP) [Barto *et al.*, 1993] algorithm (Algorithm 1) is an asynchronous DP approach that updates states encountered during trial-based MDP simulations. RTDP explores the state space in depth-limited trials and performs Bellman updates at each visited state. RTDP visits states s' *sampled* from the transition distribution ($s' \sim T(s, a, \cdot)$) for the current greedy action a and current state s, i.e.,

$$\text{CHOOSENEXTSTATE}(s, a) := s' \sim T(s, a, \cdot). \quad (5)$$

We say that V_h is an *admissible* upper bound over the optimal value function V^* if $V_h(s) \geq V^*(s)$ for every state s. Similarly, V_l is an admissible lower bound if $V_h(s) \leq V^*(s)$. Given an admissible upper bound, RTDP converges to the optimal value function in the limit of trials [Barto *et al.*, 1993].

One key advantage of RTDP is that it may only need to explore a small subset of states to obtain an optimal policy π^* w.r.t. $\mathcal{I}$ if the subset of states reachable from $\mathcal{I}$ under π^* (the *relevant states* for π^*) is small.

2.4 RTDP Extensions

One drawback of RTDP is that its random exploration of states in (5) may focus search in areas of the state space that already have converged values. Since this wastes computation, one improvement is to focus exploration on states with high value uncertainty and terminate when there is low uncertainty. This is the motivation behind the bounded RTDP (BRTDP) [McMahan *et al.*, 2005] and focused RTDP (FRTDP) [Smith and Simmons, 2006] extensions to RTDP.

BRTDP and FRTDP both maintain upper and lower bounds on the value function. Assuming that the upper and lower bounds are admissible, subsequent DP updates preserve admissibility [McMahan *et al.*, 2005].

In BRTDP and FRTDP, we update both bounds using $\hat{V}_h(s) :=$ UPDATE($\hat{V}_h, s$) and $\hat{V}_l(s) :=$ UPDATE($\hat{V}_l, s$). In BRTDP, the CHOOSENEXTSTATE routine that directs exploration is modified (Algorithm 2), prioritizing states by the gap between their upper and lower value bounds. Trials terminate when the sum of these gaps is below a threshold. FRTDP provides a similarly motivated uncertainty-based state selection and trial termination criteria.

3 Bayesian Real-time Dynamic Programming

While BRTDP and FRTDP often converge faster than RTDP, they can still execute redundant updates in areas of the state space where the policy has already converged, but the values

have not. Thus, in place of focusing exploration and updates on states with the highest value uncertainty, what we would really like is to *prioritize updates on those states whose value update may lead to the greatest improvement in policy value*.

Performing this value of information analysis would be prohibitively expensive if done exactly and non-myopically so we use a myopic Bayesian value of information framework [Howard, 1966] to approximate the expected improvement in decision quality resulting from a state's value update.

We begin by rewriting (2) and (3) using a vector notation where $\vec{\Gamma}_{a,s} = \gamma T(s, a, \cdot)$:

$$Q_{a,s}^t := R(s, a) + \vec{\Gamma}_{a,s} \cdot \vec{V}^{t-1} \qquad (6)$$

$$V^t(s) := \max_{a \in A} Q_{a,s}^t \qquad (7)$$

3.1 Bounds and Belief Distributions

In order to take an expectation in a Bayesian framework, we need to assign probabilities to our current value beliefs.

Let $\vec{V}_l$ and $\vec{V}_h$ represent vectors of lower and upper bounds respectively, and $\vec{\theta} = \langle \vec{V}_l, \vec{V}_h \rangle$. 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. Determining these correlations is tantamount to performing DP backups, which is precisely the operation that we are trying to optimize.

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. This assumption is not simply for 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. 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.

We use $\vec{\theta}$ to parameterize a multivariate uniform distribution $P(\vec{v}|\vec{\theta})$ for $\vec{v} \in \mathbb{R}^{|S|}$ that is consistent with the upper and lower bounds $\vec{\theta}$. $P(\vec{v}|\vec{\theta})$ can be conveniently factorized as

$$P(\vec{v}|\vec{\theta}) = \prod_{s \in S} P(v_s|\vec{\theta})$$

$$P(v_s|\vec{\theta}) = \begin{cases} V_h(s) > V_l(s) : & Uniform(v_s; V_l(s), V_h(s)) \\ V_h(s) = V_l(s) : & \delta_{V_l(s)}(v_s) \end{cases}$$

where $\delta_{V_l(s)}(v_s)$ is a Dirac delta function.

3.2 The Myopic Value of Exploration

With a value belief distribution that is the uniform hyper-rectangle between upper and lower value bounds, we can now write out the integral for the *expected* value of $Q(a, s)$ under the current beliefs and evaluate it in closed-form:

$$E[Q_{a,s}|\vec{\theta}] = R(s, a) + \int_{\vec{v}} \prod_{s'} P(v_{s'}|\vec{\theta}) \left[\vec{\Gamma}_{a,s} \cdot \vec{v}\right] d\vec{v}$$

$$= R(s, a) + \vec{\Gamma}_{a,s} \cdot \frac{\vec{V}_h + \vec{V}_l}{2} \qquad (8)$$

We now use this to determine the states $t \in S$ for which the expected information gain of updating value $V(t)$ is greatest. Let us assume that we are only interested in the impact of the value v_t on the current policy value. Given that we do not know the true value v_t^*, we can use a value of perfect information (VPI) analysis [Howard, 1966] where we assume that a clairvoyant source informs of the true value $v_t^* = V^*(t)$.

Using this assumption of perfect external knowledge about v_t^* to refine our beliefs, we replace the previous upper and lower bounds for state t in $P(v|\vec{\theta})$ with v_t^*:

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

Integrating the above, rearranging terms, and substituting (8) yields a simplified form of this expectation:

$$E[Q_{a,s}|\vec{\theta}, v_t^*] = \qquad (9)$$
$$\underbrace{E[Q_{a,s}|\vec{\theta}] - T(s, a, t) \left(\frac{V_h(t) + V_l(t)}{2}\right)}_{c_{(a,s,t,\vec{\theta})}} + \underbrace{T(s, a, t)}_{d_{(a,s,t)}} v_t^*$$

Now, whereas (8) evaluated to a constant since all values in $\vec{V}_h$ and $\vec{V}_l$ are known, (9) evaluates to the function $c_{(a,s,t,\vec{\theta})} + d_{(a,s,t)} v_t^*$ linear in v_t^* since all values other than v_t^* are constants.

To evaluate the gain of knowing v_t^*, we take the analytical framework of [Dearden *et al.*, 1998] used for analyzing the value of *action* selection in the *model-free* MDP setting and adapt it for analyzing the value of *state* exploration in the *model-based* framework. Let $a^* = \text{GREEDYACTION}(\vec{V}_h, s)$ (since convergence of RTDP requires exploring states reachable from the best upper bound action). We could evaluate the gain in Q-value for state s by using a rather than a^* if we knew the exact value of successor state t is v_t^*:

$$Gain_{s,t,a,a*}(v_t^*) = \qquad (10)$$
$$\max \left(0, E[Q_{a,s}|\vec{\theta}, v_t^*] - E[Q_{a^*,s}|\vec{\theta}, v_t^*]\right)$$

$Gain_{s,t,a,a^*}(v_t^*)$ is only non-zero when knowledge of v_t^* indicates that a^* is better than a in s and the gain is then the difference in utility. We note that $Gain_{s,t,a,a^*}(v_t^*)$ must always be non-negative because more information can never reduce policy quality.

In reality, we do not know v_t^*. However, we can still write out the *expected* myopic VPI of being in state s with current best policy a^* and knowing the value of t is v_t^* by integrating over it w.r.t. our beliefs $P(v_t^*|\vec{\theta})$:

$$VPI_{s,a^*}(t) = \max_{a \neq a^*} \int_{v_t^* = -\infty}^{\infty} P(v_t^*|\vec{\theta}) Gain_{s,t,a,a^*}(v_t^*) dv_t^*$$

$$= \frac{1}{V_h(t) - V_l(t)} \max_{a \neq a^*} \int_{v_t^* = V_l(t)}^{V_h(t)} Gain_{s,t,a,a^*}(v_t^*) dv_t^* \qquad (11)$$

Since we are looking at the gain over multiple actions and only one of them can be optimal, we take the maximum expected gain possible. $VPI_{s,a^*}(t)$ provides us with an approximate estimate of the myopic VPI of the impact that an update of state t's value will have on the policy quality at s.
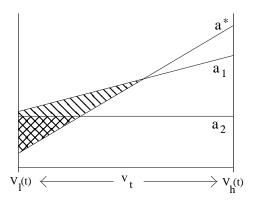
Figure 1: A graphical representation of the $VPI_s(t)$ calculation. Note that a^* is the current greedy optimal action. $Gain_{s,t,\cdot,a^*}(v_t^*)$ is non-zero for a_1 in the union of the hatched and crosshatched areas and non-zero for a_2 in the crosshatched area. Thus, the action yielding maximal gain is a_1 and the $VPI_{s,a^*}(t)$ is then the combined hatched and crosshatched area multiplied by $1/(V_h(t) - V_l(t))$.

$VPI_{s,a^*}(t)$ might seem difficult to evaluate. But in Figure 1 we show that the calculation intuitively only requires a maximization over the expected gains of taking each $a \neq a^*$ instead of a^*. This can be computed efficiently with the same $O(|S| \cdot |A|)$ computational complexity as the Bellman backup at a single state — for every next state t, $VPI_{s,a^*}(t)$ is the maximizing $Gain_{s,t,a,a^*}(v_t^*)$ over all actions $a \neq a^*$, which is a constant time calculation consisting of (a) computing the line equations (9) for a and a^* used by $Gain_{s,t,a,a^*}(v_t^*)$, (b) determining the intersection point of these two lines and (c) calculating the triangular area between the lines and value bounds where a dominates. As the Bellman backup must already be computed four times at each state visited during a trial, this does not change the complexity of the algorithm.

3.3 VPI Exploration Heuristic

We use the $VPI_{s,a^*}(t)$ calculation to prioritize state exploration and trial termination in Algorithm 3. Using dual bound updates and Algorithm 3 in place of CHOOSENEXTSTATE(s,a) in RTDP (Algorithm 3) yields the *VPI-RTDP* algorithm. This approach has roughly the same structure as BRTDP and FRTDP except that $VPI_{s,a^*}(t)$ is used in conjunction with bound gap $V_h(s) - V_l(s)$ heuristics.

The $VPI_{s,a^*}(t)$ calculation is uninformative when bounds are close to their maximal uncertainty. We thus revert to the BRTDP heuristic in this case; in practice we use a threshold β that is 95% of the maximum possible bound. With probability α we avoid termination when $VPI_{s,a^*}(t)$ is zero for all states t since this is a local termination heuristic that ignores the need of predecessor states to reduce uncertainty.

VPI-RTDP has the same theoretical guarantees as BRTDP [McMahan *et al.*, 2005] when $\alpha > 0$ quite trivially since all states with non-zero probability of an update by BRTDP must then also have a non-zero probability of update under VPI-RTDP. However, this is mainly of theoretical concern since we have found it advantageous to use very small α in practice, e.g., $\alpha = 0.001$ as used in our experiments.

Algorithm 3: CHOOSENEXTSTATE-VPI(s, a)

begin
  // *Check for large bound gap*
  $\forall t, \ b(t) := T(s, a, t)(\hat{V}_h(t) - \hat{V}_l(t))$
  $B := \sum_t b(t)$
  if $\max_t b(t) > \beta$ **then**
    | **return** $t \sim \frac{b(\cdot)}{B}$
  // *If VPI non-zero, focus on value of information*
  $\forall t, \ v(t) := VPI_{s,a}(t)$
  $V := \sum_t v(t)$
  if $V > 0$ **then**
    | **return** $t \sim \frac{v(\cdot)}{V}$
  // *VPI is zero, continue with probability α*
  $r \sim Uniform(0,1)$
  if $r < \alpha \wedge B \neq 0$ **then**
    | **return** $t \sim \frac{b(\cdot)}{B}$
  return *null*
end

---

4 Empirical Results

We evaluated RTDP, BRTDP ($\tau = 10$), FRTDP ($\epsilon = .001$) and VPI-RTDP on the racetrack benchmark domain [Barto *et al.*, 1993]. Example racetrack topologies that we use are provided in Figure 2. We borrow some topologies from [Smith and Simmons, 2006] as well as variations of their slippage and wind enhancements to the original racetrack problem.

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 $\langle x', y' \rangle = \langle 0, 0 \rangle$, receives -1 for every action taken, except for 0 in the absorbing goal states. 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 $\langle 0, 0 \rangle$. Nominally, the car accelerates according to the intended action. However, a car may skid with probability $.1$, thus resulting in 0 acceleration in each direction. Or with probability $.8$ the wind may perturb the commanded acceleration by a uniform choice of $\{\langle -1, 0 \rangle, \langle 0, -1 \rangle, \langle 1, 0 \rangle, \langle 0, 1 \rangle\}$. We use discount $\gamma = 1$ so this is a stochastic shortest paths (SSP) MDP. We set *max-depth* $= 200$ for these problems and thus use $-max\text{-}depth$ to initialize the lower bounds. For informed upper bound initialization we used the negative of the Manhattan distance to the closest goal divided by twice the maximum velocity (clearly an optimistic estimate).

In Figure 3, we show the average policy reward for the lower bound greedy policy vs. the execution time (top) and the number of *unique* states visited (bottom) for each algorithm on four Racetrack problems. 95% confidence intervals are shown on all average reward estimates. VPI-RTDP outperforms all competing algorithms on each problem and reaches optimality visiting a smaller fraction of the state space (i.e., number of unique states) than the competing algorithms. On the largest problem (Block-80), VPI-RTDP shows roughly a three-fold reduction in the amount of time and number of unique states visited required to achieve performance comparable to the best competitor (BRTDP). For the *Block* problems especially, VPI-RTDP managed to avoid

(a) Large-B   (b) Large-Ring   (c) Block-10 (d) Block-20 (e) Block-(30–60) (f) Block-70 (g) Block-80
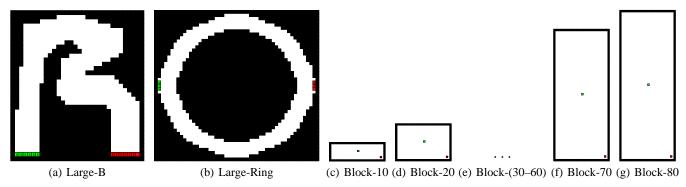
Figure 2: Various racetrack domains evaluated in this paper. Initial states are labeled 'S', terminal states are labeled 'F'. Black squares delineate walls and whitespace indicate legal car coordinates. There are 8 *Block* domains ranging in length from 10 to 80 with a fixed width of 30. For our evaluation, the *Block* domains have the important property that a large fraction of the states are irrelevant to the optimal policy.
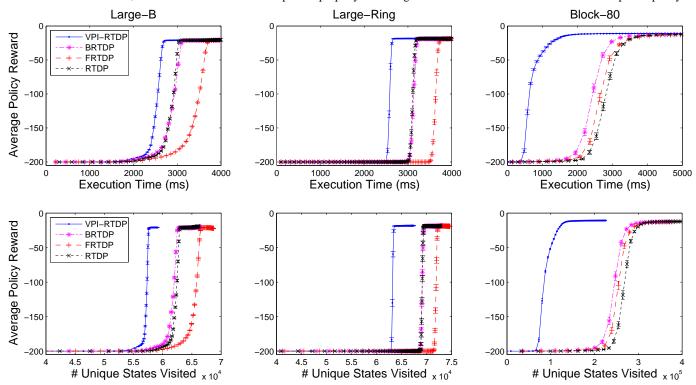

Figure 3: Average reward for lower bound policy vs. the execution time and # of unique states visited by each algorithm on three Racetrack problems. Most importantly, we note that (a) the upper-leftmost line is always VPI-RTDP (representing the best performance vs. time/space tradeoff) and (b) VPI-RTDP asymptotes at the optimal policy return visiting only a fraction of the unique states visited by the other algorithms.

visiting the many irrelevant states leading away from the initial state and goal — the VPI of these states was low (so VPI avoided these states) even while their bound gap was still large (BRTDP and FRTDP could not avoid these states).

In Figure 4, we analyze the scaling behavior of each algorithm as a function of problem size. Shown are the time and number of unique states visited required to achieve average policy return of at least -100 vs. the length parameter in the Block problem (results averaged over 120 runs of each algorithm). The number of states in the Block problem grows slightly superlinearly with length as higher velocities can be achieved on longer tracks. We chose -100 as a policy performance comparison point since it is roughly the point of inflection in performance for all algorithms shown in Figure 3

— after -100 is reached, the quality of the policy rapidly improves with future trials. The figure shows that as problem size increases, the time and space performance gap between VPI-RTDP and competing algorithms significantly widens.

5 Related Work

Other extensions of RTDP and other efficient algorithms that focus dynamic programming on reachable states were suggested in the past. Labeled RTDP (LRTDP) [Bonet and Geffner, 2003b] improves on basic RTDP by labeling solved states when their values (and the values of their successors) have converged, thus not requiring future updates. Heuristic Dynamic Programming (HDP) [Bonet and Geffner, 2003a]
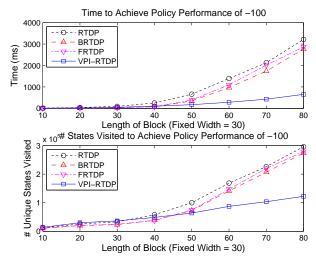
Figure 4: The amount of time and number of unique states visited required to achieve an average policy return of at least -100 as the length parameter of the *Block* problem increases from 10 to 80.

combines an even stronger version of this labeling approach with a heuristic search algorithm. LAO* and Improved LAO* [Hansen and Zilberstein, 2001] are policy iteration approaches to solving MDPs while limiting themselves to relevant states for the current greedy policy. As BRTDP and FRTDP were shown to outperform LRTDP and HDP [McMahan *et al.*, 2005; Smith and Simmons, 2006] and LRTDP was shown to outperform (Improved) LAO* [Bonet and Geffner, 2003b], we focused our experimental comparison on BRTDP and FRTDP as the current state-of-the-art RTDP algorithms. That BRTDP outperforms LRTDP is not surprising — both avoid visiting converged states (LRTDP labels them), but BRTDP *also* prioritizes states by their degree of convergence.

Russell and Wefald [1991] proposed the idea of using myopic value of information heuristics for search node expansion. This idea was later applied to MDPs in the form of Bayesian Q-learning [Dearden *et al.*, 1998], which presents a technique to balance exploration and exploitation in a *model-free* Q-learning approach. While Bayesian Q-learning inspired this work, there are subtle, but important differences. Bayesian Q-learning is concerned with the expected information gain of *action* selection and *not* with the information gain of exploring individual *states* as required here by the RTDP framework. Furthermore, Bayesian Q-learning models its beliefs using a normal-gamma distribution leading to a VPI integral that *cannot* be computed in closed-form thus requiring sampling methods for evaluation. Our *model-based* framework and uniform hyper-rectangle Bayesian belief distribution allows us to derive a *closed-form* computation for the VPI of the same complexity as the Bellman backup.

The sensitivity analysis used in the hierarchical planner DRIPS [Haddawy *et al.*, 1995] is the closest approximation we have found in the literature of the model-based VPI analysis we derived here. However, in our experiments adapting DRIPS to the RTDP setting (unreported here due to space limitations), VPI outperformed DRIPS.

6 Concluding Remarks

We contributed a novel, efficiently computable, and closed-form Bayesian value of information analysis that can be used as a state exploration and trial termination heuristic in a new Bayesian RTDP algorithm: VPI-RTDP. VPI-RTDP attempts to avoid the pitfalls of BRTDP and FRTDP by focusing search on those states with the greatest potential impact on policy quality, *not* just value uncertainty as done previously. Empirically, VPI-RTDP leads to an improvement over state-of-the-art RTDP methods, yielding up to a three-fold reduction in the amount of time and fraction of state space visited to achieve comparable policy performance. Theoretically, VPI-RTDP has the same optimal convergence guarantees as BRTDP.

One potential extension of this work would be to continuous state or action MDPs. Certain special cases such as MDPs with linear Q-functions over the state or action space might admit computationally efficient, closed-form derivation of VPI heuristics.

Acknowledgements

References

[Barto *et al.*, 1993] A. G. Barto, S. J. Bradtke, and S. P. Singh. Learning to act using real-time dynamic programming. Tech. Report UM-CS-1993-002, U. Mass. Amherst, 1993.

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

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

[Bonet and Geffner, 2003b] B. Bonet and H. Geffner. Labeled RTDP: Improving the convergence of real-time dynamic programming. In *ICAPS*, 12–21, Trento, Italy, 2003.

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

[Haddawy *et al.*, 1995] P. Haddawy, A. Doan, and R. Goodwin. Efficient decision-theoretic planning techniques. In *UAI*, 229–236, Montreal Canada, August 1995.

[Hansen and Zilberstein, 2001] E A. Hansen and S. Zilberstein. LAO * : A heuristic search algorithm that finds solutions with loops. *Artif. Intell.*, 129(1-2):35–62, 2001.

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

[McMahan *et al.*, 2005] H. Brendan McMahan, M. Likhachev, and G. J. Gordon. Bounded real-time dynamic programming: RTDP with monotone upper bounds and performance guarantees. In *ICML*, pages 569–576, Bonn, Germany, 2005.

[Puterman, 1994] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, NY, 1994.

[Russell and Wefald, 1991] Stuart Russell and Eric Wefald. Principles of metareasoning. *Artif. Intell.*, 49(1-3):361–395, 1991.

[Smith and Simmons, 2006] T. Smith and R. G. Simmons. Focused real-time dynamic programming for MDPs: Squeezing more out of a heuristic. In *AAAI*, 2006.