Solving POMDPs: RTDP-Bel vs. Point-based Algorithms

Blai Bonet
Departamento de Computación
Universidad Simón Bolívar
Caracas, Venezuela
bonet@ldc.usb.ve

Héctor Geffner
Departamento de Tecnología
ICREA & Universitat Pompeu Fabra
08003 Barcelona, SPAIN
hector.geffner@upf.edu

Abstract

Point-based algorithms and RTDP-Bel are approximate methods for solving POMDPs that replace the full updates of parallel value iteration by faster and more effective updates at selected beliefs. An important difference between the two methods is that the former adopt Sondik's representation of the value function, while the latter uses a tabular representation and a discretization function. The algorithms, however, have not been compared up to now, because they target different POMDPs: discounted POMDPs on the one hand, and Goal POMDPs on the other. In this paper, we bridge this representational gap, showing how to transform discounted POMDPs into Goal POMDPs, and use the transformation to compare RTDP-Bel with point-based algorithms over the existing discounted benchmarks. The results appear to contradict the conventional wisdom in the area showing that RTDP-Bel is competitive, and sometimes superior to point-based algorithms in both quality and time.

1 Introduction

In recent years, a number of point-based algorithms have been proposed for solving large POMDPs [Smith and Simmons, 2005; Spaan and Vlassis, 2005; Pineau *et al.*, 2006; Shani *et al.*, 2007]. These algorithms adopt Sondik's representation of the value function but replace the complex full synchronous updates by faster and more effective updates at selected beliefs. Some of the elements of point-based algorithms are common to RTDP-Bel, an older and simpler approximate algorithm based on RTDP [Barto *et al.*, 1995], that also combines simulation and point-based updates but represents the value function in a straightforward manner by means of a table and a discretization function [Bonet and Geffner, 1998a; 1998b; 2000]. While the two type of algorithms have not been experimentally compared up to now, it is widely believed that Sondik's representation is crucial for speed and quality, and thus, that algorithms such as RTDP-Bel cannot match them. One of the reasons that this comparison has not been done is that RTDP-Bel targets a class of shortest-path POMDPs with positive action costs while point-based algorithms target *discounted POMDPs*. In this paper, we bridge the representational gap among the different types of POMDPs, showing in particular how to transform *discounted POMDPs* into equivalent *Goal POMDPs*, and use this transformation to compare point-based algorithms and RTDP-Bel over the existing *discounted benchmarks*. The results appear to contradict the conventional wisdom in the area showing that RTDP-Bel is competitive, and indeed, often superior to point-based algorithms. The transformations among POMDPs are also new, and show that Goal POMDPs are actually more expressive than discounted POMDPs, a result that generalizes one for MDPs in [Bertsekas and Tsitsiklis, 1996, pp. 39–40].

The paper is organized as follows. We consider in order Goal MDPs and RTDP (Sect. 2), Goal POMDPs and RTDP-Bel (Sect. 3), discounted POMDPs and point-based algorithms (Sect. 4), the equivalence-preserving transformations among different types of POMDPs (Sect. 5), the experimental comparison between point-based algorithms and RTDP-Bel (Sect. 6), and end with a brief discussion (Sect. 7).

2 Goal MDPs and RTDP

Shortest-path MDPs provide a generalization of the state models traditionally used in heuristic search and planning, accommodating stochastic actions and full state observability [Bertsekas, 1995]. They are given by

SP1. a non-empty state space S,
SP2. a non-empty set of target states $T \subseteq S$,
SP3. a set of actions A,[1]
SP4. probabilities $P_a(s'|s)$ for $a \in A$, $s, s' \in S$, and
SP5. costs $c(a, s)$ for $a \in A$ and $s \in S$.

The target states t are assumed to be absorbing and cost-free; meaning $P_a(t|t) = 1$ and $c(a, t) = 0$ for all $a \in A$. Goal MDPs are shortest-path MDPs with a known initial state s_0 and *positive action costs* $c(a, s)$ for all a and non-terminal states s. A Goal MDP has a solution, and hence, an optimal solution, if a target (goal) state is reachable from every state. A policy π is optimal if it minimizes the expected cost to the goal $V^\pi(s)$ for all states s. This optimal expected cost,

[1] For simplicity, and without loss of generalization, we assume that all actions in A are applicable in all states.

1. **Start** with $s = s_0$.
2. **Evaluate** each action a in s as:
$$Q(a, s) = c(a, s) + \sum_{s' \in S} p_a(s'|s)V(s')$$
initializing $V(s')$ to $h(s')$ when s' is not in hash
3. **Select** action $\mathbf{a}$ that minimizes $Q(a, s)$.
4. **Update** $V(s)$ to $Q(\mathbf{a}, s)$
5. **Sample** next state s' with probability $P_{\mathbf{a}}(s'|s)$
6. **Finish** if s' is target, else $s := s'$ and goto 2

Figure 1: Single Trial of RTDP

denoted as $V^*(s)$, is the unique solution of the Bellman equation

$$V^*(s) = \min_{a \in A} \left\{ c(a, s) + \sum_{s' \in S} P_a(s'|s)V^*(s') \right\} \quad (1)$$

for all $s \in S \setminus T$, and $V^*(s) = 0$ for $s \in T$. The Bellman equation can be solved by the *Value Iteration* (VI) method, where a value function V, initialized arbitrarily, except $V(t) = 0$ for targets t, is updated iteratively until convergence using the right-hand side of (1) as:

$$V(s) := \min_{a \in A} \left\{ c(a, s) + \sum_{s' \in S} P_a(s'|s)V(s') \right\}. \quad (2)$$

This method for computing V^* is known as parallel value iteration. In *asynchronous value iteration*, states are updated according to (2) in any order. Asynchronous VI also delivers the optimal value function V^* provided that all the states are updated sufficiently often [Bertsekas, 1995]. With $V = V^*$, the *greedy policy* π_V is an optimal policy, where π_V is

$$\pi_V(s) = \operatorname*{argmin}_{a \in A} \left\{ c(a, s) + \sum_{s' \in S} P_a(s'|s)V(s') \right\}. \quad (3)$$

Recent AI methods for solving MDPs are aimed at making dynamic programming methods more selective by incorporating ideas developed in heuristic search. One of the first such methods is *Real-Time Dynamic Programming* (RTDP), introduced in [Barto *et al.*, 1995], which is a generalization of the LRTA* algorithm [Korf, 1990] that works for deterministic models only.

RTDP is an asynchronous value iteration algorithm of a special type: it can converge to the optimal value function and policy over the relevant states *without having to consider all the states in the problem*. For achieving this, and in contrast with standard value iteration algorithms, RTDP uses an *admissible* heuristic function or lower bound h as the initial value function. Provided with such a lower bound, RTDP selects for update the states that are reachable from the initial state s_0 through the greedy policy π_V in a way that interleaves simulation and updates. The code for a single trial of RTDP is shown in Fig. 1.

The quality of the heuristic used to initialize the value function is crucial for RTDP and LRTA*. In both cases, better heuristic values mean a more focused search, and a more focused search means more updates on the states that matter, thus allowing the value function to bootstrap more quickly.

For the implementation of RTDP, the estimates $V(s)$ are stored in a hash table that initially contains the heuristic value of the initial state only. Then, when the value of a state s that is not in the table is needed, a new entry for s with value $V(s) = h(s)$ is allocated. These entries are updated following (2) when a move from s is performed.

The simple greedy search combined with updates in both RTDP and LRTA* yields two key properties. First, a single trial of these algorithms eventually terminates in a goal state. Second, successive trials, using the same hash table, eventually deliver an optimal cost function V^*, and thus an optimal (partial) policy π_{V^*} over the relevant states.These two properties hold for Goal MDPs only, under the assumption that the goal is reachable from any state (with positive probability) and that the initial value function is a lower bound [Barto *et al.*, 1995]. They do *not* hold in general for arbitrary shortest-path MDPs nor for general *discounted cost-based (or reward-based) MDPs*. Indeed, in such cases RTDP may easily enter into a loop, as when some of the action costs are zero.

3 Goal POMDPs and RTDP-Bel

Partially Observable MDPs generalize MDPs by modeling agents that have incomplete state information [Sondik, 1978; Monahan, 1983; Kaelbling *et al.*, 1999] in the form of a prior belief b_0 that expresses a probability distribution over S, and a sensor model made up of a set of observation tokens O and probabilities $Q_a(o|s)$ of observing $o \in O$ upon entering state s after doing a. Formally, a *Goal POMDP* is a tuple given by:

PO1. a non-empty state space S,
PO2. an initial belief state b_0,
PO3. a non-empty set of target (or goal) states $T \subseteq S$,
PO4. a set of actions A,
PO5. probabilities $P_a(s'|s)$ for $a \in A$, $s, s' \in S$,
PO6. *positive* costs $c(a, s)$ for non-target states $s \in S$,
PO7. a set of observations O, and
PO8. probabilities $Q_a(o|s)$ for $a \in A, o \in O$, $s \in S$.

It is assumed that target states t are cost-free, absorbing, and fully observable; i.e., $c(a, t) = 0$, $P_a(t|t) = 1$, and $t \in O$, so that $Q_a(t|s)$ is 1 if $s = t$ and 0 otherwise. The *target beliefs* or goals are the beliefs b such that $b(s) = 0$ for $s \in S \setminus T$.

The most common way to solve POMDPs is by formulating them as completely observable MDPs over the *belief states* of the agent [Astrom, 1965; Sondik, 1978]. Indeed, while the effects of actions on states cannot be predicted, the effects of actions on *belief states* can. More precisely, the belief b_a that results from doing action a in the belief b, and the belief b_a^o that results from observing o after doing a in b, are:

$$b_a(s) = \sum_{s' \in S} P_a(s|s')b(s')\,, \quad (4)$$

$$b_a(o) = \sum_{s \in S} Q_a(o|s)b_a(s)\,, \quad (5)$$

$$b_a^o(s) = Q_a(o|s)b_a(s)/b_a(o) \quad \text{if } b_a(o) \neq 0\,. \quad (6)$$

As a result, the *partially observable* problem of going from an initial state to a goal state is transformed into the *completely observable* problem of going from one *initial belief state* into

a *target belief state*. The Bellman equation for the resulting *belief* MDP is

$$V^*(b) = \min_{a \in A} \left\{ c(a,b) + \sum_{o \in O} b_a(o) V^*(b_a^o) \right\} \qquad (7)$$

for non-target beliefs b and $V^*(b_t) = 0$ otherwise, where $c(a,b)$ is the expected cost $\sum_{s \in S} c(a,s)b(s)$.

The RTDP-Bel algorithm, depicted in Fig. 2 is a straightforward adaptation of RTDP to Goal POMDPs where states are replaced by belief states updated according to (7). There is just one difference between RTDP-Bel and RTDP: in order to bound the size of the hash table and make the updates more effective, each time that the hash table is accessed for reading or writing the value $V(b)$, the belief b is *discretized*. The discretization function d is extremely simple and maps each entry $b(s)$ into the entry:

$$d(b(s)) = \text{ceil}(D * b(s)) \qquad (8)$$

where D is a positive integer (the discretization parameter), and ceil(x) is the least integer $\geq x$. For example, if $D = 10$ and b is the vector $(0.22, 0.44, 0.34)$ over the states $s \in S$, $d(b)$ is the vector $(3, 5, 4)$.

Notice that the discretized belief $d(b)$ is not a belief and does not have to represent one. It just represents the unique cell in the hash table that stores the value function for b and of all other beliefs b' such that $d(b') = d(b)$. The discretization is used only in the operations for accessing the hash table and it does not affect the beliefs that are generated during a trial. Using a terminology that is common in Reinforcement Learning, the discretization is a simple *function approximation* device [Bertsekas and Tsitsiklis, 1996; Sutton and Barto, 1998], where a single parameter, the value stored at cell $d(b)$ in the hash table is used to represent the value of all beliefs b' that discretize into $d(b') = d(b)$. This approximation relies on the assumption that the value of beliefs that are close, should be close as well. Moreover, the discretization preserves supports (the states s with $b(s) > 0$) and thus never collapses the value of two beliefs if there is a state that is excluded by one but not by the other.[2]

The theoretical consequences of the discretization are several. First, convergence is not guaranteed and actually the value in a cell may oscillate. Second, the value function approximated in this way does not remain necessarily a lower bound.[3] These problems are common to most of the (non-linear) function approximations schemes used in practice, and the simple discretization scheme used in RTDP-Bel is no exception. The question, however, is how well this function approximation scheme works in practice.

[2]This property becomes more important in problems involving action preconditions [Bonet, 2002].

[3]Some of the theoretical shortcomings of RTDP-Bel can be addressed by storing in the hash table the values of a set of selected belief points, and then using suitable interpolation methods for approximating the value of beliefs points that are not in the table. This is what grid-based methods do [Hauskrecht, 2000]. These interpolations, however, involve a large computational overhead, and do not appear to be cost-effective.

1. **Start** with $b = b_0$.
2. **Sample** state s with probability $b(s)$.
3. **Evaluate** each action a in b as:
$$Q(a,b) = c(a,b) + \sum_{o \in O} b_a(o) V(b_a^o)$$
initializing $V(b_a^o)$ to $h(b_a^o)$ if b_a^o is not in the hash.
4. **Select** action $\mathbf{a}$ that minimizes $Q(a,b)$.
5. **Update** $V(b)$ to $Q(\mathbf{a},b)$.
6. **Sample** next state s' with probability $P_{\mathbf{a}}(s'|s)$.
7. **Sample** observation o with probability $Q_{\mathbf{a}}(o|s')$.
8. **Compute** b_a^o using (6).
9. **Finish** if b_a^o is target belief, else $b := b_a^o$ and $s := s'$ and goto 3.

Figure 2: RTDP-Bel is RTDP over the belief MDP with an additional provision: for reading or writing the value $V(b)$ in the hash table, b is replaced by $d(b)$ where d is the discretization function.

4 Discount and Point-based Algorithms

Discounted *cost-based* POMDPs differ from Goal POMDPs in two ways: target states are not required and a discount factor $\gamma \in (0,1)$ is used instead. The Bellman equation for discounted models becomes:

$$V^*(b) = \min_{a \in A} \left\{ c(a,b) + \gamma \sum_{o \in O} b_a(o) V^*(b_a^o) \right\}. \qquad (9)$$

The expected cost $V^\pi(b)$ for policy π is always well-defined and bounded by $C/(1-\gamma)$ where C is the largest cost $c(a,s)$ in the problem.

Discounted *reward-based* POMDPs are similar except that rewards $r(a,s)$ are used instead of costs $c(a,s)$, and a max operator is used in the Bellman equation instead of min. A discounted reward-based POMDP can be easily translated into an equivalent discounted cost-based POMDP by simply multiplying all rewards by -1.

Point-based algorithms target discounted POMDPs. They use a finite representation of the (exact) value function for POMDPs due to Sondik [1971] that enables the computation of exact solutions. In Sondik's representation, a value function V is represented as a finite set Γ of $|S|$-dimensional real vectors such that

$$V(b) = \max_{\alpha \in \Gamma} \sum_{s \in S} \alpha(s) b(s) = \max_{\alpha \in \Gamma} \alpha \cdot b\,. \qquad (10)$$

Point-based algorithms, however, replace the updates of full synchronous value iteration with faster and more effective updates at selected beliefs. The update at a single belief b of a function V, represented by a set Γ of vectors α, results in the addition of a single vector to Γ, called backup(b), whose computation in vector notation is [Pineau *et al.*, 2006;

Shani *et al.*, 2007]:

$$backup(b) = \operatorname*{argmax}_{g_a^b, a \in A} g_a^b \cdot b\,, \tag{11}$$

$$g_a^b = r(a,\cdot) + \gamma \sum_{o \in O} \operatorname*{argmax}_{g_{a,o}^\alpha, \alpha \in \Gamma} \left\{ g_{a,o}^\alpha \cdot b \right\}, \tag{12}$$

$$g_{a,o}^\alpha(s) = \sum_{s' \in S} \alpha(s') Q_a(o|s') P_a(s'|s)\,. \tag{13}$$

This computation can be executed efficiently in time $O(|A||O||S||\Gamma|)$ and forms the basis of all point-based POMDP algorithms that differ mainly in the selection of the belief points to update and the order of the updates.

5 Transformation of POMDPs

For transforming discounted POMDPs into equivalent Goal POMDPs, we need to make precise the notion of equivalence. Let us say that the *sign* of a POMDP is *positive* if it is a cost-based POMDP, and *negative* if it is a reward-based POMDP. Also, let $V_M^\pi(b)$ denote the expected cost (resp. reward) from b in a positive (resp. negative) POMDP M. Then, if the target states are absorbing, cost-free, and observable, an equivalence relation among two POMDPs of any types can be expressed as follows:

Definition 1. *Two POMDPs R and M are equivalent if they have the same set of non-target states, and there are two constants α and β such that for every policy π and non-target belief b, $V_R^\pi(b) = \alpha V_M^\pi(b) + \beta$ with $\alpha > 0$ if R and M are POMDPs of the same sign, and $\alpha < 0$ otherwise.*

That is, two POMDPs R and M are equivalent if their value functions are related by a simple linear transformation over the same set of non-target beliefs. It follows that if R and M are equivalent, they have the same optimal policies and the same 'preference order' among policies. Thus, if R is a discounted reward-based POMDP and M is an equivalent Goal POMDP, we can obtain optimal and/or suboptimal policies for R by running an optimal and/or approximate algorithm over M.

We say that a transformation that maps a POMDP M into M' is *equivalence-preserving* if M and M' are equivalent. In this work, we consider three transformations, all of which leave certain parameters of the POMDP unaffected. In the following transformations, we only mention the parameters that change:

- T1. $R \mapsto R + C$ is about the addition of a constant C (positive or negative) to all rewards/costs. It maps a discounted POMDP R into the discounted POMDP $R + C$ by replacing the rewards (resp. costs) $r(a,s)$ (resp. $c(a,s)$) by $r(a,s) + C$ (resp. $c(a,s) + C$).
- T2. $R \mapsto kR$ is about the multiplication by a constant $k \neq 0$ (positive or negative) of all rewards/costs. It maps a discounted POMDP R into the discounted POMDP kR by replacing the rewards (resp. costs) $r(a,s)$ (resp. $c(a,s)$) by $kr(a,s)$ (resp. $kc(a,s)$). If k is negative, R and kR have opposite signs.
- T3. $R \mapsto \overline{R}$ eliminates the discount factor. It maps a discounted cost-based POMDP R into a shortest-path POMDP $\overline{R}$ by adding a fully observable target state t so that $P_a(t|s) = 1 - \gamma$, $P_a(s'|s) = \gamma P_a^R(s'|s)$, $O_a(t|t) = 1$ and $O_a(s|t) = 0$ where γ is the discount factor in R and $P_a^R(s'|s)$ refers to the transition probabilities in model R.

Theorem 2. *The transformations $R \mapsto R + C$, $R \mapsto kR$ and $R \mapsto \overline{R}$ are all equivalence-preserving.*

Proof (sketch): One shows that if $V_R^\pi(b)$ is the value function for the policy π in R over the non-target beliefs b, and hence the (unique) solution of the Bellman equation for π in R:

$$V_R^\pi(b) = r(b,a) + \gamma \sum_{o \in O} b_a(o) V_R^\pi(b_a^o)\,, \tag{14}$$

then $V_R^\pi(b) + C/(1-\gamma)$ is the solution of the Bellman equation for the same policy π in the model $R + C$, $kV_R^\pi(b)$ is the solution of the Bellman equation for π in kR, and $V_R^\pi(b)$ itself is the solution of Bellman equation for π in $\overline{R}$. This all involves the manipulation of the Bellman equation for π in the various models. In particular, for proving $V_{R+C}^\pi(b) = V_R^\pi(b) + C/(1-\gamma)$, for example, we add the expression $C' = C/(1-\gamma)$ to both sides of (14), to obtain:

$$V_R^\pi(b) + C' = \big[C + r(b,a)\big] + \gamma \sum_{o \in O} b_a(o) \big[V_R^\pi(b_a^o) + C'\big]$$

as $C' = C + \gamma C'$, and thus $V_R^\pi(b) + C'$ is the solution of the Bellman equation for π in $R + C$. □

Theorem 3. *Let R be a discounted reward-based POMDP, and C a constant that bounds all rewards in R from above; i.e. $C > \max_{a,s} r(a,s)$. Then, $M = \overline{-R + C}$ is a Goal POMDP equivalent to R. Indeed,*

$$V_R^\pi(b) = -V_M^\pi(b) + C/(1-\gamma) \tag{15}$$

for all non-target beliefs b and policies π.

Proof (sketch): For such C, $-R$ is a discounted cost-based POMDP, $-R + C$ is a discounted cost-based POMDP with positive costs, and $M = \overline{-R + C}$ is a Goal POMDP. A straightforward calculation shows that $V_R^\pi(b) = -V_M^\pi(b) + C/(1-\gamma)$ for all non-target beliefs b, and hence $V_R^\pi(b) = \alpha V_M^\pi(b) + \beta$ with a negative $\alpha = -1$, in agreement with the signs of R and M. □

As an illustration, the transformation of the discounted reward-based POMDP Tiger with 2 states into a Goal POMDP with 3 states is shown in Fig. 3, where the third state is a target (absorbing, costs-free, and observable). Although not shown, the rewards are transformed into positive costs by multiplying them by -1 and adding them the constant 11, as the maximal reward for the original problem is 10.

6 RTDP-Bel vs. Point-based Algorithms

For comparing RTDP-Bel with point-based algorithms over discounted benchmarks R, we run RTDP-Bel over the equivalent Goal POMDPs M and evaluate the resulting policies back in the original model R in the standard way. In the experiments, we use the discretization parameter $D = 15$. We

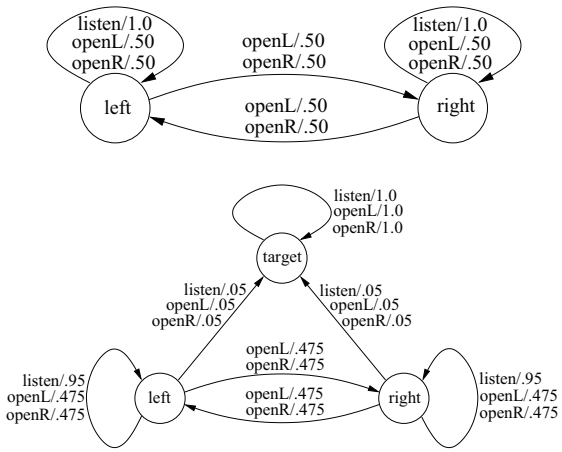

Figure 3: Transformation of the discounted reward-based POMDP problem Tiger with two states (top) and $\gamma = 0.95$ into a Goal POMDP problem (bottom) with an extra target state.

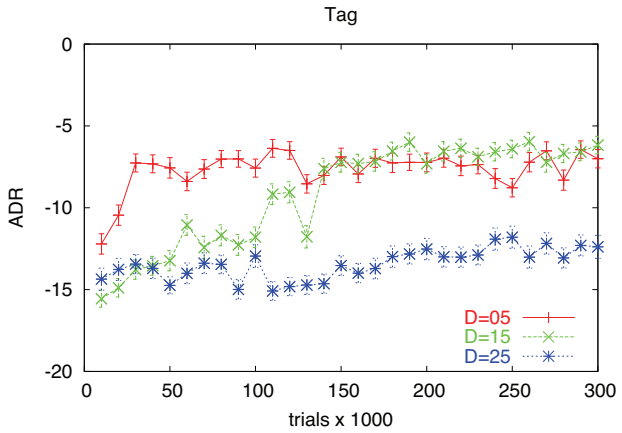

Figure 4: Performance of RTDP-Bel over Tag for three discretization parameters: $D = 5$, $D = 15$ and $D = 25$.

have also found it useful to avoid transitions in the simulation to the dummy target state t created in the transformation which makes long trials unlikely. Notice that target beliefs b_t do not need to be explored or visited as $V^*(b_t) = h(b_t) = 0$.

The point-based algorithms considered are HSVI2 [Smith and Simmons, 2005], PBVI [Pineau *et al.*, 2006], Perseus [Spaan and Vlassis, 2005] and FSVI [Shani *et al.*, 2007]. As usual, we also include the QMDP policy.

Table 1 displays the average discounted reward (ADR) of the policies obtained by each of the algorithms, the time in seconds for obtaining such policies, and some algorithm-specific information: for HSVI2 the number of planes in the final policy, for Perseus and PBVI+GER the number of beliefs points in the belief set, for FSVI the size of the final value function, and for RTDP-Bel the final number of entries in the hash table and the number of trials.

Except when indicated, the data has been collected by running the algorithms on our machine (a Xeon at 2.0GHz), from code made available by the authors. For FSVI, the experiments on Tag and LifeSurvey1 did not finish, with the algorithm apparently cycling with a poor policy.

The ADR is measured over 1,000 simulation trials of 250 steps each, except in three problems: Hallway, Hallway2, and Tag. These problems include designated 'goal states' that are not absorbing. In these cases, simulation trials are terminated when such states are reached.

As it can be seen from the table, RTDP-Bel does particularly well in the 6 largest problems (Tag, RockSample, and LifeSurvey1): in 4 of these problems RTDP-Bel returns the best policies (highest ADR) and in the other two, it returns policies that are within the confidence interval of the best algorithm. In addition, in 4 of these problems (the largest 4), RTDP-Bel is also the fastest. On the other hand, RTDP-Bel does not perform as well in the 3 smallest problems, and in particular in the two Hallway problems where it produces weaker policies in more time than all the other algorithms.

We also studied the effect of using different discretizations. Fig. 4 shows the quality of the policies resulting from up to 300,000 trials, for the Tag problem using $D = 5, 15, 25$. The cases for $D = 5$ and $D = 15$ are similar in this problem except that in the first case RTDP-Bel takes 290 seconds and uses 250k hash entries, and in the second, 493 seconds and 2.5m hash entries. The case of $D = 25$ is different as RTDP-Bel needs then more than 300,000 trials to reach the target ADR. In this case, RTDP-Bel takes 935 seconds and uses 7.7m hash entries. In this problem $D = 5$ works best, but in other problems finer discretizations are needed. All the results in Table 1 are with $D = 15$.

7 Discussion

RTDP-Bel is a simple adaptation of the RTDP algorithm to goal POMDPs that precedes the recent generation of point-based algorithms and some of its ideas. The experiments show, however, that RTDP-Bel can be competitive, and even superior to point-based algorithms in both time and quality. This raises two questions. First, why RTDP-Bel has been neglected up to now, and second, why RTDP-Bel does as well? We do not know the answer to the first question but suspect that the reason is that RTDP-Bel has been used over discounted benchmarks where it doesn't work. RTDP-Bel, is for Goal POMDPs, in the same way that RTDP is for Goal MDPs. In this work, we have shown how discounted POMDPs can be translated into equivalent Goal POMDPs for RTDP-Bel to be used. About the second question, the reasons that RTDP-Bel does well on goal POMDPs appear to be similar to the reasons that RTDP does well on goal MDPs: restricting the updates to the states generated by the stochastic simulation of the greedy policy appears to be quite effective, ensuring convergence without getting trapped into loops. The discretization in POMDPs, if suitable, doesn't appear to hurt this. On the other hand, this property does not result from value functions represented in Sondik's form that encode upper bounds rather than lower bounds (in the cost setting). The two type of methods, however, are not incompatible and it may be possible to combine the strengths of both.

Acknowledgments

We thank J. Pineau and S. Thrun, T. Smith and G. Shani for helping us with PBVI, HSVI2 and FSVI. The work of H. Geffner is partially supported by grant TIN2006-15387-C03-03 from MEC/Spain.

References

[Astrom, 1965] K. Astrom. Optimal control of Markov Decision Processes with incomplete state estimation. *J. Math. Anal. Appl.*, 10:174–205, 1965.

[Barto *et al.*, 1995] A. Barto, S. Bradtke, and S. Singh. Learning to act using real-time dynamic programming. *Art. Int.*, 72:81–138, 1995.

[Bertsekas and Tsitsiklis, 1996] D. Bertsekas and J. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[Bertsekas, 1995] D. Bertsekas. *Dynamic Programming and Optimal Control, (2 Vols)*. Athena Scientific, 1995.

[Bonet and Geffner, 1998a] B. Bonet and H. Geffner. Learning sorting and decision trees with POMDPs. In *Proc. ICML*, pages 73–81, 1998.

[Bonet and Geffner, 1998b] B. Bonet and H. Geffner. Solving large POMDPs using real time dynamic programming. In *Proc. AAAI Fall Symp. on POMDPs*. AAAI Press, 1998.

[Bonet and Geffner, 2000] B. Bonet and H. Geffner. Planning with incomplete information as heuristic search in belief space. In *Proc. ICAPS*, pages 52–61, 2000.

[Bonet, 2002] B. Bonet. An ϵ-optimal grid-based algorithm for Partially Observable Markov Decision Processes. In *Proc. ICML*, pages 51–58, 2002.

[Hauskrecht, 2000] M. Hauskrecht. Value-function approximations for partially observable Markov decision processes. *JAIR*, 13:33–94, 2000.

[Kaelbling *et al.*, 1999] L. P. Kaelbling, M. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Art. Int.*, 101:99–134, 1999.

[Korf, 1990] R. Korf. Real-time heuristic search. *Art. Int.*, 42(2–3):189–211, 1990.

[Monahan, 1983] G. Monahan. A survey of partially observable Markov decision processes: Theory, models and algorithms. *Management Science*, 28(1):1–16, 1983.

[Pineau *et al.*, 2006] J. Pineau, G. J. Gordon, and S. Thrun. Anytime point-based approximations for large POMDPs. *JAIR*, 27:335–380, 2006.

[Shani *et al.*, 2007] G. Shani, R. I. Brafman, and S. E. Shimony. Forward search value iteration for POMDPs. In *Proc. IJCAI*, pages 2619–2624, 2007.

[Smith and Simmons, 2005] T. Smith and R. Simmons. Point-based POMDP algorithms: Improved analysis and implementation. In *Proc. UAI*, pages 542–547, 2005.

[Sondik, 1971] E. Sondik. *The Optimal Control of Partially Observable Markov Decision Processes*. PhD thesis, Stanford University, 1971.

[Sondik, 1978] E. Sondik. The optimal control of partially observable Markov decision processes over the infinite horizon: discounted costs. *Oper. Res.*, 26(2), 1978.

[Spaan and Vlassis, 2005] M. T. J. Spaan and N. A. Vlassis. Perseus: Randomized point-based value iteration for POMDPs. *JAIR*, 24:195–220, 2005.

[Sutton and Barto, 1998] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

algorithm	reward	time	data / trials
Tiger-Grid (36s, 5a, 17o)			
Perseus*	2.34	104	10k
HSVI2	2.31 ± 0.35	22	735
RTDP-Bel	2.31 ± 0.10	379	409k / 12k
FSVI	2.30 ± 0.13	872	2,654
PBVI+GER	2.28 ± 0.06	1,770	1,024
QMDP	0.24 ± 0.06	n/a	n/a
Hallway (60s, 5a, 21o)			
HSVI2†	0.52	2.4	147
FSVI	0.52 ± 0.02	57	341
PBVI+GER	0.51 ± 0.02	9	64
Perseus*	0.51	35	10k
RTDP-Bel	0.49 ± 0.02	50	57k / 12k
QMDP	0.23 ± 0.02	n/a	n/a
Hallway2 (92s, 5a, 17o)			
HSVI2†	0.35	1.5	114
Perseus*	0.35	10	10k
FSVI	0.35 ± 0.03	60	458
PBVI+GER	0.33 ± 0.02	8	32
RTDP-Bel	0.25 ± 0.02	627	711k / 28k
QMDP	0.10 ± 0.01	n/a	n/a
Tag (870s, 5a, 30o)			
RTDP-Bel	-6.16 ± 0.53	493	2.5m / 300k
Perseus*	-6.17	1,670	10k
HSVI2†	-6.36	24	415
PBVI+GER	-6.75 ± 0.50	9,659	512
QMDP	-16.57 ± 0.65	n/a	n/a
FSVI	—	—	—
RockSample[4,4] (257s, 9a, 2o)			
PBVI+GER	18.34 ± 0.49	17	256
FSVI	18.21 ± 0.82	104	353
RTDP-Bel	18.12 ± 0.52	16	26k / 20k
HSVI2	18.00 ± 0.14	0.7	268
QMDP	3.97 ± 0.35	n/a	n/a
RockSample[5,5] (801s, 10a, 2o)			
RTDP-Bel	19.59 ± 0.55	21	28k / 20k
PBVI+GER	19.56 ± 0.55	1,495	256
HSVI2	19.21 ± 0.25	41	2,417
FSVI	19.07 ± 1.01	31	413
QMDP	4.16 ± 0.49	n/a	n/a
RockSample[5,7] (3,201s, 12a, 2o)			
RTDP-Bel	23.75 ± 0.60	35	8,232 / 20k
FSVI	23.67 ± 0.90	297	1,049
HSVI2	23.64 ± 0.32	126	2,431
PBVI+GER	14.59 ± 0.79	4,265	512
QMDP	6.35 ± 0.53	n/a	n/a
RockSample[7,8] (12,545s, 13a, 2o)			
FSVI	21.03 ± 0.84	916	1,578
RTDP-Bel	20.70 ± 0.57	62	16k / 16k
HSVI2	20.66 ± 0.39	224	1,748
PBVI+GER	14.70 ± 0.54	7,940	256
QMDP	7.50 ± 0.46	n/a	n/a
LifeSurvey1 (7,001s, 7a, 28o)			
RTDP-Bel	95.94 ± 3.13	170	82k / 58k
PBVI+GER	92.63 ± 2.75	11,859	256
HSVI2	92.29 ± 0.65	731	971
QMDP	90.25 ± 3.48	n/a	n/a
FSVI	—	—	—

*From [Spaan and Vlassis, 2005]: Pentium IV 2.4GHz.
†From [Smith and Simmons, 2005]: Pentium IV 3.4GHz.

Table 1: RTDP-Bel vs. Point-based algorithms. Figures show ADR, time in seconds, and data for each algorithm. Algorithms ordered by ADR and experiments conducted in a Xeon 2.0GHz except where indicated.