T H E   N A T I O N A L   U N I V E R S I T Y
o f   S I N G A P O R E

NUS
National University
of Singapore

S c h o o l   o f   C o m p u t i n g
Lower Kent Ridge Road, Singapore 119260

TRA4/07

Accelerating Point-Based POMDP Algorithms through Successive Approximations of the Optimal Reachable Space

David HSU, Wee Sun, LEE and Nan RONG

April 2006

Technical Report

Foreword

Accelerating Point-Based POMDP Algorithms through Successive Approximations of the Optimal Reachable Space

David Hsu Wee Sun Lee Nan Rong

Department of Computer Science
National University of Singapore
Singapore, 117543, Singapore

Abstract

Point-based approximation algorithms have drastically improved the speed of POMDP planning. This paper presents a new point-based POMDP algorithm called SARSOP. Like earlier point-based algorithms, SARSOP performs value iteration at a set of sampled belief points; however, it focuses on sampling near the space reachable from an initial belief point under the *optimal policy*. Since neither the optimal policy nor the optimal reachable space is known in advance, SARSOP builds successive approximations to it through sampling and pruning. In our experiments, the new algorithm solved difficult POMDP problems with more than 10,000 states. Its running time is competitive with the fastest existing point-based algorithm on most problems and faster by many times on some. Our approach is complementary to existing point-based algorithms and can be integrated with them to improve their performance.

Introduction

Planning and decision-making under uncertainty are central problems in artificial intelligence and robotics. Partially observable Markov decision processes (POMDPs) provide a principled mathematical framework for solving such problems (White III 1991; Hauskrecht 2000). Unfortunately, the computational cost of solving POMDPs exactly is intractable (Papadimitriou & Tsisiklis 1987), and this has prevented their wide-scale adoption in practical applications. Recently, point-based approximation algorithms have drastically improved the speed of POMDP solution (Pineau, Gordon, & Thrun 2003; Smith & Simmons 2005; Spaan & Vlassis 2005): POMDPs with hundreds of states can now be solved in less than a minute on a desktop computer (Smith & Simmons 2005). These algorithms have the potential to make POMDP practical in many applications.

To gain computational efficiency, point-based algorithms compute an approximately optimal solution policy over a set of *points* sampled from the belief space $\mathcal{B}$ instead of an optimal policy over the entire $\mathcal{B}$. Their success hinges on two key factors. First, point-based algorithms must sample a small, representative set of points from the relevant part of $\mathcal{B}$, in order to achieve good approximation to the optimal policy. Second, under fairly general conditions, optimal policies for POMDPs can be represented as value functions that are convex and piecewise linear (Sondik 1971). Exploiting this fact, point-based algorithms represent the value function as a set of α-vectors, each corresponding to an optimal hyperplane of the value function at some sampled point. It then computes the solution by performing value iteration on the α-vectors at the sampled points.

A key question to ask is then "*Which part of $\mathcal{B}$ should be sampled and what distribution should be used for sampling?*" Some early POMDP algorithms sample the entire belief space $\mathcal{B}$, using a uniform sampling distribution, such as a grid. To improve computational efficiency, more recent point-based algorithms sample only $\mathcal{R}(b_0)$, the subset of points reachable from a given initial belief point $b_0 \in \mathcal{B}$, under arbitrary sequences of actions. To push further in this direction, we would like to sample only near $\mathcal{R}^*(b_0)$, the subset of belief points reachable from b_0 under the *optimal policy*. It is clear that $\mathcal{R}^*(b_0) \subseteq \mathcal{R}(b_0) \subseteq \mathcal{B}$, and $\mathcal{R}^*(b_0)$ is the smallest set that must be sampled and covered well. Of course, the optimal policy is unknown in advance, and we need to approximate $\mathcal{R}^*(b_0)$. In the following, to simplify the notation, we will omit the argument b_0. It is understood that $\mathcal{R}$ and $\mathcal{R}^*$ are reachable from a given initial point b_0.

In attempt to address the above question, we have developed a new point-based POMDP algorithm called SARSOP, which stands for Successive Approximations of the Reachable Space under the Optimal Policy. The main idea of SARSOP is to compute successive approximations of the optimal value function over $\mathcal{R}^*$. In each iteration, we sample additional belief points and use the information there to update the α-vectors and improve the value function approximation. We maintain a *certificate* data structure, which helps us to efficiently prune extraneous belief points and α-vectors that lie outside $\mathcal{R}^*$ and do not contribute the final solution. The focus on sampling near $\mathcal{R}^*$, which is potentially much smaller than $\mathcal{R}$, offers two main advantages computationally. First, a smaller number of sampled points is needed to approximate $\mathcal{R}^*$. Second, by checking for dominance over a smaller space, we can prune α-vectors more aggressively.

Experiments show that the new algorithm solved difficult POMDP problems with more than 10,000 states. It was competitive with the fastest existing point-based algorithm in running times and often much faster.

The idea of approximating $\mathcal{R}^*$ through sampling and pruning is relatively straightforward, but surprisingly, to our knowledge, it has not been exploited before to improve the speed of POMDP planning. Our experiments show that the gain can be substantial.

One additional benefit of our approach is that it is complementary to existing point-based algorithms, such as point-base value iteration (PBVI) (Pineau, Gordon, & Thrun 2003) and heuristic search value iteration (HSVI2) (Smith & Simmons 2005), and can be integrated with them to improve their performance.

Preliminaries

POMDPs

A POMDP models an agent taking a sequence of actions under uncertainty to maximize its reward. Formally it is specified as a tuple $(S, A, Z, b_0, T, O, R, \gamma)$, where S is a set of discrete states, A is a set of actions, and Z is a set of observations.

At any time, the agent lies in some state $s \in S$. Commonly, we assume that the probability distribution for the agent's initial state is known and given by $b_0(s) = p(s)$. At each time step, the agent takes some action $a \in A$ and moves from a start state s to an end state s'. Due to the uncertainty in action, the end state s' is described by a conditional probability function $T(s, a, s') = p(s'|s, a)$, which gives the probability that the agent lies in s', after taking action a in state s. After taking the action, the agent makes an observation to gather information on its state. Due to the uncertainty in observation, the observation result $z \in Z$ is again described by a conditional probability function $O(s, a, z) = p(z|s, a)$ for $s \in S$ and $a \in A$.

The reward function R gives the agent a real-valued reward $R(s, a)$ if it take action a in state s, and the goal of the agent is to maximize its expected total reward by choosing a suitable sequence of actions. If the sequence of actions has infinite length, we often specify a discount factor $\gamma \in (0, 1)$ so the total reward is finite and the problem is well defined.

In a POMDP, the agent's state is only partially observable. We thus rely on the concept of a belief, which is simply a probability distribution over S. Suppose that the agent's current belief is b. It takes action a and obtains observation z. The new belief b' is given by $b'(s') = \tau(b, a, z) = \eta O(s', a, z) \sum_s T(s, a, s') b(s)$, where η is a normalizing constant.

The solution to a POMDP is a policy π that specifies the action $\pi(b)$ for any belief b encountered. We want to find the policy that maximizes the expected total reward.

Related work

POMDP is a principled approach for planning under uncertainty. Due to its computational intractability, there have been significant efforts in developing approximation algorithms. See (Hauskrecht 2000) for a recent survey. The brief review here focuses on point-based algorithms.

Early point-based algorithms sample the entire belief space $\mathcal{B}$ using fixed- or variable-resolution grids (Lovejoy 1991; Brafman 1997; Geffner & Bonet 1998). More recent algorithms sample only $\mathcal{R}$, the subset of $\mathcal{B}$ reachable from an initial belief point b_0 in order to improve efficiency (Hauskrecht 2000; Pineau, Gordon, & Thrun 2003; Roy, Gordon, & Thrun 2005; Smith & Simmons 2005; Spaan & Vlassis 2005). To our knowledge, HSVI2 (Smith & Simmons 2005) has so far demonstrated the best performance experimentally, among the point-based algorithms. HSVI2 uses heuristics to guide their sampling towards region which may potentially have higher rewards. Our algorithm is related to PBVI (Pineau, Gordon, & Thrun 2003), HSVI2, and Perseus (Spaan & Vlassis 2005), but to gain further in efficiency, it focuses on maintaining only sample points that are near $\mathcal{R}^*$, which is even smaller than $\mathcal{R}$, and builds an approximation of the optimal value function over $\mathcal{R}^*$.

One way of speeding up point-based algorithms is to improve backup operations. For example, HSVI2 and Perseus perform backup only at selected belief points. Furthermore, each backup operation generates more α-vectors, which may cause some earlier α-vectors to become completely dominated and thus redundant. Many POMDP algorithms periodically prune redundant α-vectors to improve efficiency.

One crucial reason for the computational intractability of POMDPs is the high dimensionality of the belief space $\mathcal{B}$. We thus need to develop lower-dimensional representations of $\mathcal{B}$. See, *e.g.*, (Poupart & Boutilier 2003; Roy, Gordon, & Thrun 2005). These approaches are very important, but beyond the scope of this paper.

SARSOP

SARSOP is a point-based POMDP algorithm. In this section, we start with an overview of its basic structure and then give details on how it achieves strong performance by computing successive approximations of the value function over the optimal reachable space $\mathcal{R}^*$.

Basic structure of the algorithm

In many point-based POMDP algorithms, we sample a set of points from $\mathcal{B}$ and maintain a set Γ of α-vectors, which represent a piecewise linear approximation $\underline{V}$ to the optimal value function. Each α-vector in Γ represents a potentially optimal hyperplane at a sampled point. To improve the approximation $\underline{V}$, we back up the α-vectors at the sampled points. With suitable initialization (using, *e.g.*, fixed-action policies (Hauskrecht 2000)), $\underline{V}$ is always a lower bound on the optimal value function. See Algorithm 1 for a formal description of the algorithm. It iterates over three main functions, SAMPLE, BACKUP, and PRUNE, until further update produces little change in the approximation $\underline{V}$. Many point-based algorithms, including SARSOP, share this basic structure, but differ in the details of the three functions.

The sampled belief points form a tree X (Figure 1). Each node of X represents a sampled point. As there is no confusion, we use the same symbol b to denote both a sampled point and its corresponding node in X. The root of X is the initial belief point b_0. To sample a new point b', we pick a node b from X as well as an action $a \in A$ and an observation $z \in Z$ according to suitable probability distributions or heuristics. We then compute $b' = \tau(b, a, o)$ and insert b' into X as the child of b. Clearly, every point sampled this way is reachable from b_0.

Next, we perform backup at selected nodes in X. A backup operation at node b collates the information in the

Algorithm 1 The basic structure of point-based POMDP algorithms with asynchronous value backup.

```
1: Insert the initial belief point $b_0$ as the root of the tree $X$.
2: Initialize the set $\Gamma$ of $\alpha$-vectors.
3: repeat
4:     SAMPLE($X$, $\Gamma$).
5:     Choose a subset of nodes from $X$. For each chosen
       node $b$, BACKUP($X, \Gamma, b$).
6:     PRUNE($X$, $\Gamma$).
7: until termination conditions are satisfied.
8: return $\Gamma$.

SAMPLE($X$, $\Gamma$)
1: Pick $b \in X$, $a \in A$, and $z \in Z$.
2: $b' \leftarrow \tau(b, a, o)$.
3: Insert $b'$ into $X$ as a child of $b$.

BACKUP($X$, $\Gamma$, $b$)
1: For all $a \in A, z \in Z, \alpha_{a,z} \leftarrow \text{argmax}_{\alpha\in\Gamma}(\alpha \cdot \tau(b, a, z))$.
2: For all $a \in A, s \in S, \alpha_a(s) \leftarrow R(s, a) +$
       $\gamma \sum_{z,s'} T(s, a, s')O(s', a, z)\alpha_{a,z}(s')$.
3: $\alpha' \leftarrow \text{argmax}_{a\in A}(\alpha_a \cdot b)$
4: Insert $\alpha'$ into $\Gamma$.
```

children of b and propagates it back to b. In particular, we want to propagate the information from the node containing the new sample point. Experimentally, the backup operations create a bottleneck, usually taking more than 90% of the total computational time.

Invocation of SAMPLE and BACKUP generates new sampled points and α-vectors. However, not all of them are useful for constructing the optimal policy and are pruned to improve computational efficiency.

Reachable space under the optimal policy

The efficiency of point-based POMDP algorithms depends on two key factors. First, how are the sampled points distributed over $\mathcal{B}$? Since our goal is to compute the optimal policy for a given initial belief point b_0, we should avoid sampling far away form $\mathcal{R}^*$ to save computational costs. Second, the cost of a single backup operation is directly proportional to the number of α-vectors in Γ. One way of speeding up backup, which dominates the total running time, is then to keep the set Γ small. Existing point-based algorithms usually prune an α-vector if it is dominated by others over the entire $\mathcal{B}$. We would like to prune more aggressively: an α-vector is pruned if it is dominated by other α-vectors over $\mathcal{R}^*$. This results in potentially much smaller Γ.

Unfortunately we do not know the optimal policy or the optimal reachable space $\mathcal{R}^*$ in advance. Even if we do, the cost of checking α-vector dominance over $\mathcal{R}^*$ exactly may be prohibitive. Instead, we use the set of sampled points in X to approximate $\mathcal{R}^*$ and iteratively improve the approximation as the policy improves. By checking for dominance only at the sampled points, the cost of pruning is greatly reduced. The details are described in next two subsections.

Approximating $\mathcal{R}^*$ through belief point pruning

To use the sampled points in X as an approximation of $\mathcal{R}^*$, we must prune those points far away from $\mathcal{R}^*$. For this, we

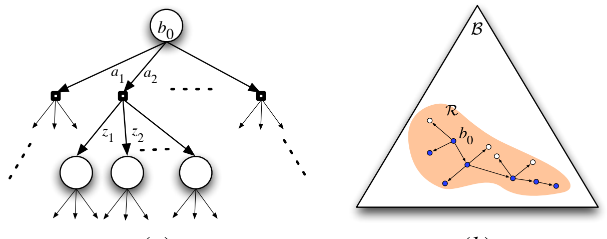

Figure 1. (a) The POMDP tree with the root b_0. (b) The corresponding reachable belief space $\mathcal{R}$. The shaded nodes indicate points in $\mathcal{R}^*$.

maintain not only a lower bound $\underline{V}$, but also an upper bound $\overline{V}$ on the optimal value function. Various methods for maintaining the upper bound can be used. Currently SARSOP uses the same method as HSVI2. The upper bound computation is very fast, but the bounds are somewhat weak. See (Smith & Simmons 2005) for details.

By maintaining both upper and lower bounds, we can improve our approximation of $\mathcal{R}^*$ as these bounds improve. Consider a node b in X. Let

$$\underline{Q}(b, a) = \sum_s R(s, a)b(s) + \gamma \sum_z p(z|b, a)\underline{V}(\tau(b, a, z))$$

be the lower bound on the value of taking action a at b. The upper bound $\overline{Q}$ is defined similarly, using $\overline{V}$. If $\overline{Q}(b, a) < \underline{Q}(b, a')$ for some actions a and a', the optimal policy will never take the action a at node b and traverse the subtree underneath. We thus prune the subtree along with all the associated sampled points. Some pruned points may possibly lie in $\mathcal{R}^*$, as there are other paths in X to reach them under the optimal policy. However, the benefit of reducing the number of sampled points usually outweighs the loss in value function approximation due to over-pruning. These points can also be recovered from the other paths in X eventually.

Belief point pruning improves computational efficiency in two ways. As the suboptimal branches of X are pruned, SARSOP avoids sampling the part of $\mathcal{B}$ unreachable under the optimal policy (see SAMPLE in Algorithm 1), and thus the sample distribution is automatically adapted to bias towards $\mathcal{R}^*$. Furthermore, reducing the number of sampled points helps in α-vector pruning. We explain why in the next subsection.

α-vector pruning

Let P denote the set of sampled belief points in X. SARSOP prunes away an α-vector if it is dominated by others over $\mathcal{R}^*$, which is approximated by the current P. A simple criterion for dominance is to say that for two α-vectors α_1 and α_2, α_1 dominates α_2 at a belief point b if $\alpha_1 \cdot b \geq \alpha_2 \cdot b$. However, this is not robust. The set P is a finitely sampled approximation of $\mathcal{R}^*$. Since SARSOP computes an approximately optimal policy over P only, the computed policy may choose an action that causes it to slightly veer off $\mathcal{R}^*$ and get into a region in which the value function approximation is poor. To address this issue, we impose the more stringent requirement of dominance over a δ-neighborhood: α_1 dominates α_2 at a belief point b if $\alpha_1 \cdot b' \geq \alpha_2 \cdot b'$ at every point

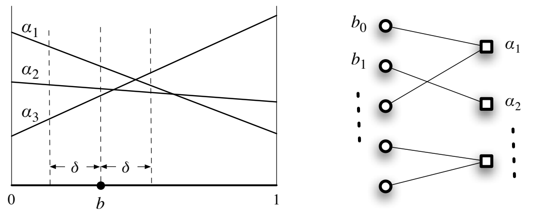

Figure 2. (*a*) δ-dominance. α_1 dominates α_2, but not α_3 in the δ-neighborhood of b. (*b*) The certificate structure.

b' whose distance to b is less than δ, for some fixed constant δ. We call this δ-*dominance*. We can check δ-dominance very quickly by computing the distance d from b to the intersection of the hyperplanes represented by α_1 and α_2 and making sure that $d \geq \delta$. See Figure 2*b* for an example.

To prune α-vectors efficiently, SARSOP maintains a *certificate* structure, using δ-dominance. The certificate structure is a bipartite graph consisting of two sets of nodes, P and Γ. There is an edge between two nodes $b \in P$ and $\alpha \in \Gamma$, if α is not dominated by another α-vector over the δ-neighborhood of b. Thus, every edge (b, α) represents a certificate that demonstrates the usefulness of α. A sampled point b issues a certificate to an α-vector, when it is created through a backup operation at b. The certificates from each sampled point b are checked periodically and revoked if the corresponding α-vectors are δ-dominated at b. Following the checks, any α-vector holding no certificates can be immediately pruned.

By maintaining the certificate structure and checking δ-dominance, SARSOP prunes away the useless α-vectors, but retain all the optimal α-vectors over the space within a distance δ of $\mathcal{R}^*$.

The pruning method above works well if P is a representative sample of $\mathcal{R}^*$. At the beginning, when there are few sampled points in P, pruning the α-vectors too aggressively may result in poor performance. To mitigate this effect, we incorporate all the corner points of the belief simplex $\mathcal{B}$ into the certificate structure. These additional points are not used for backup operations, but only for checking δ-dominance. They introduce little overhead, as the sparsity of these points (with only one nonzero coefficient per point) allows us to compute the certificates involving them very quickly.

We expect each belief point to be involved in only a few certificates. Thus, as a number of belief points decreases, the number of certified α-vector decreases as well. This is why belief point pruning helps in α-vector pruning.

Relationship with existing point-based algorithms

As mentioned earlier, the main idea of SARSOP is to build successive approximations of the value function over $\mathcal{R}^*$ through belief point and α-vector pruning. It should be clear by now that the idea is general and independent of how a point-based algorithm samples $\mathcal{B}$ and performs backup (see Algorithm 1). Therefore, belief point and α-vector pruning can be integrated with existing point-based algorithms, in particular, PBVI, HSVI2, and Perseus, and improve their performance.

In fact, we implemented SARSOP on top of HSVI2 by incorporating our belief point and α-vector pruning methods. SARSOP uses the same sampling strategy introduced in HSVI2, because of its strong performance experimentally. To sample a new belief point, it traverses a single path down the tree X by choosing the action with the highest upper bound and the observation that maximally reduces the gap between the lower and the upper bounds at the root of X, until it reaches a node b at a desired level. It then samples a new point b' and insert b' into X as a child of b (see SAMPLE in Algorithm 1). After creating the new node, it performs backup operations at all nodes along the path leading to b'.

Convergence

SARSOP prunes only belief points and α-vectors that are provably suboptimal. The pruning does not affect the value function approximation over the unpruned belief points in X. Therefore, SARSOP converges, if its sampling strategy converges. Since SARSOP uses the same sampling strategy as HSVI2, to compute a policy with a given reward quality at the root, it is sufficient to build X to a fixed depth. The convergence result of HSVI2 holds for SARSOP as well (Smith & Simmons 2004).

Experimental results

We now compare the performance of SARSOP with that of HSVI2, which, to our knowledge, has so far got the best experimental performance among the point-based POMDP algorithms.

Implementation details

HSVI2 uses a heuristic called α-*vector masking*. During a backup operation at a point b, it updates only the entries of an α-vector corresponding to the non-zero entries of b, rather than all the entries. We found that the heuristic sometimes improves the performance. Other times, it slows down the performance substantially (see Table 1). Therefore, we implemented SARSOP on top of a modified version of HSVI2 without this heuristic.

We made several small improvements on the sparse matrix operations. These improvements are applied to the implementations of all the algorithms (SARSOP, HSVI2, and HSVI2 without masking) tested in our experiments.

Experimental setup

Our test set consists of six problems (see Table 1). The first four are standard ones for testing point-based algorithms. The fifth, Rock Sample, was introduced in the work on HSVI2. It has a large state space of more than 10,000 states. We added a new problem, Bridge, which has a moderately-sized state space of 2,653 states, but is difficult because it easily misleads heuristics based on full observability. The discount factor γ was set to 0.95 for all the problems.

In the Bridge problem, a robot needs to get across a river with many bridges on a foggy day. It has a map of the environment, represented on a 51×52 grid (Figure 3). The robot is uncertain of its initial position, which is uniformly distributed in a region along left border of the map. It is unable to sense its location in the fog, but can localize itself

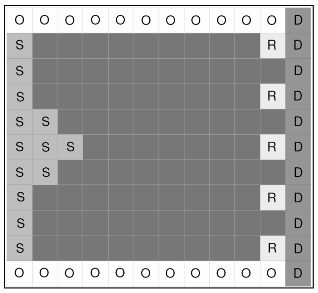

Figure 3. A scaled down version of the Bridges problem on a 11 × 12 grid. “S” marks the possible initial positions of The robot. The robot is equally likely to start in any of these squares. “D” marks the destinations. “R” marks the river. “O” marks places that the robot can fully localize itself.

exactly anywhere along the top or bottom borders. With appropriate control, the robot can either stay in the same place or move deterministically to any of the five squares adjacent to its current position: directly above, below, or to the right. The cost of movement is 1 in the horizontal and vertical directions and $\sqrt{2}$ in the diagonal direction. The robot gets a reward of 10,000 if it reaches the right border of the map and a penalty of 2,000 if it falls into the river. Roughly, the optimal policy for the robot is to move diagonally until it reaches the top or bottom border to localize itself. It can then safely cross the river and get to the destinations on the right border. A feature of this problem is that heuristics based on full observability favor the shorter horizontal movements rather than the diagonal movements and thus often choose the wrong action.

In the experiments, we performed 100 simulation runs for each computed policy to estimate its total reward. We ran all the algorithms on each problem multiple times (at least 5) until the variances of the reward estimates are small enough. The experiments were conducted on a PC with a 2.4GHz Intel Xeon processor and 2GB of memory.

Results

Table 1 shows the performance statistics of SARSOP and HSVI2, averaged over the multiple runs for each problem. The statistics for HSVI2 without masking are also shown for reference. Columns 2–3 of the table give the running times and the estimated rewards of the computed policies. Columns 4–6 give the number of α-vectors ($|\Gamma|$), the number of sampled belief points ($|X|$), and the number of backup operations (N_{B}).

The results show that SARSOP is much faster than HSVI2 on all problems, except for Rock Sample. On Tag, Tiger Grid, and Bridge, SARSOP is faster by 5 to 10 times.

The results also show that reducing the number of belief points and α-vectors significantly improves computational efficiency. This is especially clear if we compare the results of SARSOP and HSVI2 without masking so that the effect of α-vector masking does not interfere. Note that α-vector pruning is highly effective in all the test problems, often reducing the number of α-vectors by several times. In comparison, belief pruning is less effective. The reason, we believe, is that the current upper bound method used in SARSOP is too weak. It would be interesting to try better methods to compute tighter upper bounds.

Table 1. Performance comparison.

	Time (s)	Reward	$\|\Gamma\|$	$\|X\|$	N_{B}
Hallway $\|S\|=61, \|A\|=5, \|Z\|=21$					
HSVI2	2.6	1.00	73	78	218
HSVI2 w/o mask	2.9	1.00	73	78	228
SARSOP	1.7	1.04	42	59	170
Hallway2 $\|S\|=93, \|A\|=5, \|Z\|=17$					
HSVI2	17.2	0.57	222	104	284
HSVI2 w/o mask	10.8	0.57	219	108	296
SARSOP	10.1	0.57	87	119	326
Tiger Grid $\|S\|=36, \|A\|=5, \|Z\|=17$					
HSVI2	161.6	2.37	1716	595	1711
HSVI2 w/o mask	139.6	2.41	1441	534	1436
SARSOP	15.1	2.40	148	351	949
Tag $\|S\|=870, \|A\|=5, \|Z\|=30$					
HSVI2	26.8	-5.51	699	357	964
HSVI2 w/o mask	12.2	-5.67	366	135	361
SARSOP	5.4	-5.71	123	86	237
Rock Sample [7,8] $\|S\|=12{,}545, \|A\|=13, \|Z\|=2$					
HSVI2	1007	21.80	2169	1932	5646
HSVI2 w/o mask	1792	22.26	1269	1162	3289
SARSOP	1879	21.74	922	1029	2896
Bridge $\|S\|=2653, \|A\|=6, \|Z\|=103$					
HSVI2*	>40000				
HSVI2 w.o mask	17987	745.63	1401	1425	21186
SARSOP	4016	742.08	348	813	11880

*In Bridge, HSVI2 did not converge after running for more than 11 hours and was terminated.

The α-vector masking technique used by HSVI2 works well for some problems, but degrades the performance substantially on others. In Rock Sample, the robot always knows its own position perfectly. This greatly reduces uncertainty and enables masking to work extremely well, and HSVI2 is much faster than the other two algorithms.

The ideas of α-vector pruning and masking are in principle complementary. It should be possible to make masking more rigorous and remove its negative effect. We can then combine pruning and masking, but we have not explored this possibility yet.

The effect of δ

Recall that to prune α-vectors, we need to check for δ-dominance, where δ is a parameter. We examined the effect of δ on performance through experiments. The general trend is that smaller δ values allow more aggressive α-vector pruning and lead to better performance (see Table 2). For some problems, if δ is too small, it may slow down the performance due to over-pruning. However, this issues does not always occur, and $\delta = 0$ sometimes gives the best result. See, *e.g.*, the Tiger Grid problem in Table 2. Of course, some problems are more sensitive to δ than others. In particular, we found that Rock Sample is quite sensitive to δ. In the neighborhoods of some belief points, the optimal policy performs delicate switching of actions. Setting the δ value too small easily causes over-pruning.

Table 2. The performance of SARSOP for various δ values.

δ	Time (s)	Reward	$\|\Gamma\|$	$\|X\|$	N_{B}
Tiger Grid					
1×10^{-2}	41.9	2.39	223	518	1419
1×10^{-3}	26.9	2.40	163	442	1166
1×10^{-4}	15.1	2.40	148	351	949
0	12.5	2.39	168	329	876
Tag					
1×10^{-2}	22.1	-5.66	208	165	432
1×10^{-3}	20.3	-5.57	167	270	704
1×10^{-4}	5.4	-5.71	123	86	237
0	21.2	-5.66	164	301	785

In our current implementation of SARSOP, δ is set manually. For our experiments, δ was set to 1×10^{-4} for all the problems, except for Rock Sample, which had $\delta = 1 \times 10^{-3}$. It is possible to set δ automatically. We can keep a small subset of dominated α-vector unpruned and observe their effect on the value function approximation during the backup operations. We then use this information to update δ so that it is as small as possible, but still avoids over-pruning. We will soon implement and experiment with this technique.

Discussion

Our experimental results show that the idea of approximating the optimal value function over $\mathcal{R}^*$ through belief point and α-vector pruning brings significant computational advantages. Currently, SARSOP improves the approximation to $\mathcal{R}^*$ by pruning branches of the tree X known to be suboptimal. The pruning removes sampled points lying far away from $\mathcal{R}^*$. This can be considered as rejection sampling. Ideally, we want to improve the SAMPLE function in Algorithm 1 so that it does not generate these irrelevant points at all. The pruning partially achieves this, as it effectively adapts the sampling distribution so that SAMPLE no longer explores the pruned branches. Much more can be done. Indeed, in robot motion planning, another PSPACE-hard problem that shares many difficulties with POMDPs, probabilistic roadmap (PRM) algorithms have made tremendous progress in the last decade by using specialized sampling strategies that exploit the geometric structure of the space (Choset *et al.* 2005; Hsu, Latombe, & Kurniawati 2006). Nowadays, we can solve many difficult motion planning problems for robots with many degrees of freedom in complex geometric environments within minutes. We believe that the development of good sampling strategies that exploit the geometric structure of the belief space will bring similar benefits to POMDP solution and is a promising direction for future research.

Conclusion

Point-based algorithms have greatly improved the speed of POMDP solution, using sampled approximations of the space reachable from an initial belief point. We propose a new point-based algorithm, SARSOP, which pushes this idea further by focusing on the space reachable under the optimal policy. SARSOP builds successive approximations of this optimal reachable space through sampling and pruning. Experiments shows that it outperformed the fastest existing point-based algorithm by many times on some problems, while remaining competitive on others. Our approach is complementary to those used in existing point-based algorithms and can be combined with them to improve their performance.

References

Brafman, R. 1997. A heuristic variable grid solution method for POMDPs. In *Proc. Nat. Conf. on Artificial Intelligence*, 727–733.

Choset, H.; Lynch, K.; Hutchinson, S.; Kantor, G.; Burgard, W.; Kavraki, L.; and Thrun, S. 2005. *Principles of Robot Motion : Theory, Algorithms, and Implementations*. The MIT Press. chapter 7.

Geffner, H., and Bonet, B. 1998. Solving large POMDPs using real time dynamic programming. In *Working Notes of AAAI 1998 Fall Symposium on Planning with Partially Observable Markov Decision Processes*, 61–68.

Hauskrecht, M. 2000. Value-function approximations for partially observable Markov decision processes. *J. Artificial Intelligence Research* 13:33–94.

Hsu, D.; Latombe, J.; and Kurniawati, H. 2006. On the probabilistic foundations of probabilistic roadmap planning. *Int. J. Robotics Research* 25(7):627–643.

Lovejoy, W. 1991. Computationally feasible bounds for partially observed Markov decision processes. *Operations Research* 39(1):162–175.

Papadimitriou, C., and Tsisiklis, J. 1987. The complexity of Markov decision processes. *Mathematics of Operations Research* 12(3):441–450.

Pineau, J.; Gordon, G.; and Thrun, S. 2003. Point-based value iteration: An anytime algorithm for POMDPs. In *Proc. Int. Jnt. Conf. on Artificial Intelligence*, 477–484.

Poupart, P., and Boutilier, C. 2003. Value-directed compression of POMDPs. In *Advances in Neural Information Processing Systems*, volume 15. The MIT Press. 1547–1554.

Roy, N.; Gordon, G.; and Thrun, S. 2005. Finding aproximate POMDP solutions through belief compression. *J. Artificial Intelligence Research* 23:1–40.

Smith, T., and Simmons, R. 2004. Heuristic search value iteration for POMDPs. In *Proc. Uncertainty in Artificial Intelligence*, 520–527.

Smith, T., and Simmons, R. 2005. Point-based POMDP algorithms: Improved analysis and implementation. In *Proc. Uncertainty in Artificial Intelligence*.

Sondik, E. 1971. *The optimal control of partially observable Markov processes*. Ph.D. Dissertation, Stanford University, Stanford, California, USA.

Spaan, M., and Vlassis, N. 2005. Perseus: Randomized point-based value iteration for POMDPs. *J. Artificial Intelligence Research* 24:195–220.

White III, C. 1991. Partially observed Markov decision processes: A survey. *Annals of Operations Research* 32:215–230.