# Accelerating Point-Based POMDP Algorithms via Greedy Strategies*

Zongzhang Zhang and Xiaoping Chen

University of Science and Technology of China, Hefei, China
zzz@mail.ustc.edu.cn, xpchen@ustc.edu.cn

**Abstract.** Many planning tasks of autonomous robots can be modeled as partially observable Markov decision process (POMDP) problems. Point-based algorithms are well-known algorithms for solving large-scale POMDP problems. Several leading point-based algorithms eschew some flawed but very useful heuristics to find an $\epsilon$-optimal policy. This paper aims at exploiting these avoided heuristics by a simple framework. The main idea of this framework is to construct a greedy strategy and combine it with the leading algorithms. We present an implementation to verify the framework's validity. The greedy strategy in this implementation stems from some common ignored heuristics in three leading algorithms, and therefore can be well combined with them. Experimental results show that the combined algorithms are more efficient than the original algorithms. On some benchmark problems, the combined algorithms have achieved about an order of magnitude improvement in runtime. These results provide an empirical evidence for our proposed framework's efficiency.

**Keywords:** planning of autonomous robots, POMDP, point-based algorithms, framework, greedy strategy, reinforcement learning.

## 1 Introduction

Partially observable Markov decision process (POMDP) has been widely recognized as a powerful probabilistic model for planning of autonomous robots in uncertain environments. It is often challenging to exactly solve POMDP problems due to their computational complexity, however. In recent years, various efficient approximate algorithms have been proposed and have been successfully applied to various robotic tasks [1,2]. Among them, point-based algorithms are a family of particularly efficient algorithms [3,4]. They focus on sampling a set of representative beliefs that are reachable from the initial belief, and are adept at tackling large POMDP problems [3]. All point-based algorithms repeatedly take two steps. The first step is to sample a set of beliefs according to some rules.

* This work is supported in part by the Natural Science Foundations of China under Grant No. 60745002, and the National Hi-Tech Project of China under Grant No. 2008AA01Z150.

N. Ando et al. (Eds.): SIMPAR 2010, LNAI 6472, pp. 545–556, 2010.

The second step is to perform an $\alpha$-vector backup operation for each sampled belief in some order.

Several leading point-based algorithms have a wonderful property that an $\epsilon$-optimal policy can be found theoretically for a POMDP problem given enough running time. To satisfy this property, these algorithms ignore some very useful heuristics. The main reason to do this is that these heuristics have some shortcomings, such as, sometimes easily trap algorithms into local optima. However, by avoiding these heuristics, algorithms lose a precious opportunity to further improve their performance.

This paper aims at taking advantage of these underused heuristics to accelerate the leading point-based algorithms. To address the local-optima issue, we propose a generic framework. The main idea of this framework is to construct a greedy strategy using these ignored heuristics and insert it as a component into the leading point-based algorithms (see Fig. 1(a)). The combined algorithms use an $\epsilon$-optimal strategy to guide search toward an optimal policy, and meanwhile, use a greedy strategy to explore some shortcuts via some extra heuristics (see Fig. 1(b)). The use of a systematic search for finding a better policy with greedy action choices to exploit the best known decisions is reminiscent of exploration-exploitation tradeoff mechanisms in reinforcement learning [5]. However, to our knowledge, this idea hasn't been well applied into solving POMDP problems.

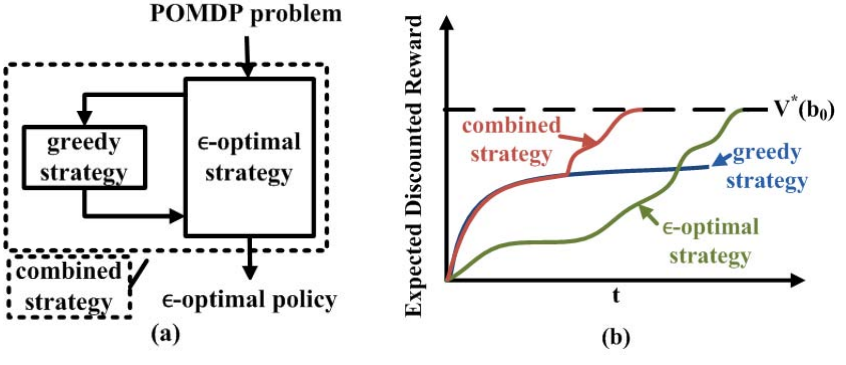


**Fig. 1.** (a) A generic framework, in which a combined strategy is a combination of an $\epsilon$-optimal strategy and a greedy strategy. (b) The lines represent the process of finding a better policy (corresponding to a higher expected discounted reward) over time for different strategies. A combined strategy takes advantage of a greedy strategy to explore possible shortcuts, and hence has extra opportunities to speed up the convergence rate. $V^*$ represents the optimal value function.

To verify this framework's utility, we designed a greedy algorithm using some common avoided or unnoticed heuristics among three leading point-based algorithms: heuristic search value iteration 2 (HSVI2) [6], focused real-time dynamic programming (FRTDP) [3,7] and successive approximations of the reachable space under optimal policies (SARSOP) [4]. We called this greedy algorithm as the "second best policy-guided" search algorithm (SBPG). The three leading

algorithms are trial-based algorithms. They sample a set of beliefs by exploring forward in each trial. During forward exploration, they decide the next sampled belief via both action-selection strategy and observation-selection strategy. In general, the two strategies are, in fact, the functions of both the lower and upper bound over the optimal value function. To guarantee cconvergence, these algorithms adopt the same action-selection strategy, which only depends on the upper bound. As a result, the lower bound is ignored in the design of the action-selection strategy. However, the lower bound is very useful in finding a better policy. A direct insight is that the current best policy of these algorithms is obtained according to the lower bound, but not the upper bound.

SBPG originates from this insight. It has two novel features. First, it has a heuristic search strategy based on classification and error-minimization search techniques. Second, it has a greedy, often more efficient action-selection strategy, which mainly depends on the lower bound. Using these techniques, SBPG can greedily focus on sampling a small belief space, which is "promising" in terms of the significance and easiness of finding a better policy. In addition, SBPG can easily be combined with the three leading algorithms. Our experimental results indicate the combined algorithms SBPG+HSVI2, SBPG+FRTDP and SBPG+SARSOP, for most of the studied instances, significantly outperform the original algorithms HSVI2, FRTDP and SARSOP, respectively. These results provide a powerful empirical evidence for our proposed framework's utility.

Our first contribution is a novel framework for speeding up point-based algorithms via greedy strategies. The second contribution is an implementation of this framework that includes a methodology for constructing a greedy strategy and combining it with the three leading algorithms, and a validation of the framework via empirical results.

## 2 Background and Related Work

In this section, we go over POMDP model and some point-based algorithms for POMDPs, and briefly describe HSVI2, one of the fastest point-based algorithms.

### 2.1 POMDP Model

POMDP provides a natural and expressive mathematical model for planning of autonomous robots with hidden state and uncertainty in action effects. A POMDP model can be officially defined by a tuple $< S, A, Z, Tr, \Omega, R, \gamma, b_0 >$. In the tuple, $S$ is a set of all possible states, $A$ is a set of all possible actions, $Z$ is a set of observations available to the agent, $Tr(s, a, s') = Pr(s'|s, a) : S \times A \times S \rightarrow [0, 1]$ is a state transition function, $\Omega(a, s', z) = Pr(z|a, s') : A \times S \times Z \rightarrow [0, 1]$ is an observation function, $R(s, a)$ is a reward function, $\gamma$ $(0 \leq \gamma < 1)$ is a discount factor, and $b_0$ represents the agent's *initial belief*. A *belief* $b$ is a probability distribution over the set of states $S$, where $b(s)$ gives the probability that the agent's state is $s$. When the agent takes action $a$ at belief $b$ and receives

observation $z$, it will arrive at a new belief $\tau(b, a, z)$. $b_a^z = \tau(b, a, z)$ is defined using Equation 1:

$$b_a^z(s') = \eta\Omega(a, s', z) \sum_{s \in S} Tr(s, a, s')b(s). \quad (1)$$

Here, $\eta$ denotes a normalizing constraint. Every POMDP problem has at least an *optimal policy* $\pi^*$ that maximizes the *expected discounted reward* $V^\pi(b) = E[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)|b, \pi]$, where $\pi$ denotes a *policy*, and $s_t$ and $a_t$ are the agent's state and action at time t, respectively. We denote $\Pi$ as the set of all possible policies. The function $V^* = V^{\pi^*}$ is called the *optimal value function*, which is usually represented as a set of vectors. When $V^*$ is available, $\pi^*$ can be achieved using Equation 2:

$$\pi^*(b) = \arg\max_{a \in A} Q^*(b, a) = \arg\max_{a \in A} \Big\{ R(b, a) + \gamma \sum_{z \in Z} Pr(z|b, a)V^*(b_a^z) \Big\}. \quad (2)$$

Here, $R(b, a) = \sum_{s \in S} R(s, a)b(s)$, and $Pr(z|b, a)$ means the probability of receiving observation $z$ after taking action $a$ at $b$. We represent the lower and upper bounds of $V^*(b)$ as $\underline{V}(b)$ and $\bar{V}(b)$, and the lower and upper bounds of $Q^*(b, a)$ as $\underline{Q}(b, a)$ and $\bar{Q}(b, a)$, respectively. Generally speaking, it is intractable to get $\pi^*$ for large POMDP problems given limited time. Hence, approximate algorithms concentrate on finding an approximate optimal policy $\pi$ with high expected discounted reward. A policy $\pi$ is called an *$\epsilon$-optimal policy* when it satisfies $V^*(b_0) - V^\pi(b_0) \leq \epsilon$. An algorithm is called an *$\epsilon$-optimal algorithm* if it always returns an $\epsilon$-optimal policy, otherwise it is called a *greedy algorithm*. The strategy used in an $\epsilon$-optimal algorithm and a greedy algorithm are called an *$\epsilon$-optimal strategy* and a *greedy strategy*, respectively.

For every POMDP problem, beliefs that are reachable from the initial belief $b_0$ can be represented as an *and/or tree* [4]. We denote it as $\triangle_{\text{POMDP}}$. Each node in the tree represents a belief. The root node in $\triangle_{\text{POMDP}}$ is $b_0$. Beliefs that are reachable from $b_0$ by following some policy $\pi$ make up an and/or subtree in $\triangle_{\text{POMDP}}$ [3]. We define the subtree corresponding to $\pi$ as $\triangle$. Without the risk of confusion, we also use $\triangle$ to represent the set of all nodes in the subtree, since a subtree consists of nodes. Note that the notation of $\triangle$ is important and will be mentioned several times in this paper. $D_\triangle$ is the depth of $\triangle$, and $\triangle(i)$ is the set of nodes at depth $i$, where $0 \leq i \leq D_\triangle$.

### 2.2 Point-Based Algorithms

We can sort point-based algorithms into three camps. The first camp focuses on how to sample a set of beliefs $B$. HSVI2, FRTDP and SARSOP are three of the most successful algorithms in this camp. In addition, other algorithms, such as point-based value iteration (PBVI) [8], Perseus [9] and Forward search value iteration (FSVI) [10] are also well known and widely accepted. The second camp concentrates on finding an optimal order of performing backup operations. Except for the above point-based algorithms, some approximate algorithms which

require low computational cost are used to initialize $\underline{V}$ and $\bar{V}$, respectively. Among them, the blind policy [11] and the fast informed bound (FIB) [11] have been widely accepted. HSVI2, FRTDP and SARSOP use the blind policy to initialize $\underline{V}$ and use FIB to initialize $\bar{V}$. The initial bounds $\underline{V}_0$ and $\bar{V}_0$ obtained by these two methods are *uniformly improvable* [12], since they satisfy $\max_{a\in A}\{R(b,a)+\gamma\sum_{z\in Z} Pr(z|b,a)\underline{V}_0(b_a^z)\} \geq \underline{V}_0(b)$ and $\max_{a\in A}\{R(b,a)+\gamma\sum_{z\in Z} Pr(z|b,a)\bar{V}_0(b_a^z)\} \leq \bar{V}_0(b)$ for any $b$, respectively. A more detailed survey of point-based algorithms can be find in [3,11].

### 2.3 HSVI2

Because of HSVI2's simple structure, we describe HSVI2 briefly (see Algorithm 1). Later, we discuss how to combine SBPG with HSVI2. In Algorithm 1, $d_b$ represents $b$'s depth in $\triangle_{\text{POMDP}}$, excess($b$,$d_b$) is specified as $\bar{V}(b)-\underline{V}(b)-\epsilon\gamma^{-d_b}$. The procedure update($b$) updates both $\underline{V}(b)$ and $\bar{V}(b)$ by performing an $\alpha$-vector backup operation and a Bellman update at $b$, respectively. HSVI2 adopts the action-selection strategy that only depends on the upper bound (see line 7).

**Algorithm 1.** HSVI2

| | |
|---|---|
| **Function** $\pi$ = HSVI2($\epsilon$) | **Function** trialRecurse($b, \epsilon, d_b$) |
| 1: Initialize the bounds $\underline{V}$ and $\bar{V}$; | 5: **if** $\bar{V}(b)-\underline{V}(b) \leq \epsilon\gamma^{-d_b}$ **then** |
| 2: **while** $\bar{V}(b_0)-\underline{V}(b_0) > \epsilon$ | 6: return ; |
| 3: trialRecurse($b_0, \epsilon, 0$); | 7: $a^* = \arg\max_{a\in A} \bar{Q}(b,a)$; |
| 4: **return** $\pi$ corresponding to $\underline{V}(b_0)$; | 8: $z^* = \arg\max_{z\in Z}[Pr(z\|b,a^*)\cdot$ $excess(\tau(b,a^*,z),d_b+1)]$; |
| | 9: trialRecurse($\tau(b,a^*,z^*),\epsilon,d_b+1$); |
| | 10: update($b$); |

## 3 Framework Definition

This section provides a definition of the framework used in this paper (see Framework 1). Trading off exploration and exploitation is one of central issues of reinforcement learning. In the past decades, some simple, ad hoc exploration strategies have been developed in the reinforcement-learning field. Among them, $\varepsilon$-greedy exploration, Boltzmann exploration, interval estimation and Rmax have been popular in practice [5]. These exploration methods are viewed as reasonable and computationally tractable approaches. However, greedy exploration methods have been only used to initialize $\underline{V}$ and $\bar{V}$ in the POMDP field. The main reason seems to be that previous POMDP researchers always pursue general useful heuristics to find an $\epsilon$-optimal policy for all common POMDP problems. However, the demand of these general heuristics is harsh and unfair for many very useful heuristics with some shortcomings, such as, local optima. The framework is proposed to address this issue via the combination of an $\epsilon$-optimal strategy and a greedy strategy. Note that the concrete representations of exploration and exploitation in reinforcement learning and our framework are different. However,

they have the common feature in exploring a better action strategy via heuristics which are eschewed in exploitation. In Framework 1, $T_O$ and $T_G$ represent the $\epsilon$-optimal algorithm's running time and the greedy algorithm's running time, respectively. The $condition(T_O, T_G)$ can be defined in different forms. Equation 3 is its simplest form, which is similar to the $\varepsilon$-greedy strategy.

$$condition(T_O, T_G) = \frac{T_G}{T_O + T_G} < \varepsilon. \tag{3}$$

**Framework 1.** Framework for Accelerating Point-Based POMDP Algorithms

1: Choose an original $\epsilon$-optimal algorithm $O$;
2: Construct a greedy algorithm $G$;
3: **while** $\bar{V}(b_0) - \underline{V}(b_0) > \epsilon$
4: call a main step of the original $\epsilon$-optimal algorithm $O$;
5: **if** $condition(T_G, T_O) ==$ true **then**
6: call the greedy algorithm $G$;
7: **return** $\pi$ corresponding to $\underline{V}(b_0)$;

As shown in Framework 1, the invocation of a greedy strategy is not limited to the combined algorithm's initialization. Therefore, the combined algorithm can use the greedy strategy to explore possible shortcuts over its "whole" running time. The advantage of such a framework is that the combined algorithm has more extra opportunities to accelerate the original $\epsilon$-optimal algorithm if the greedy strategy has some wonderful property, such as, very low computational cost, or very useful heuristics.

## 4 Mathematical Foundation for a Greedy Strategy

We now describe an important theorem for constructing SBPG (see Sect. 5). First, we define a policy as the *current best policy* $\pi_{best}$ and the *current second best policy* $\pi_{second}$ if they, respectively, satisfy Equation 4 and Equation 5:

$$\underline{V}^{\pi_{best}}(b_0) \geq \underline{V}^{\pi}(b_0), \pi \in \Pi, \tag{4}$$

$$\underline{V}^{\pi_{second}}(b_0) \geq \underline{V}^{\pi}(b_0), \pi \in \Pi - \{\pi_{best}\}. \tag{5}$$

Following $\pi_{best}$'s definition, we know $\pi_{best}(b) = \arg\max_{a \in A} \underline{Q}(b, a)$. In fact, it is possible that many policies $\pi_{best}$ and $\pi_{second}$ exist. In these situations, we can select a unique $\pi_{best}$ according to an appropriate (lexicographic) tie-breaking rule. In what follows, we assume $\pi_{best}$ is unique to simplify the discussion and denote a set of all policies $\pi_{second}$ that satisfies Equation 5 as $\Pi_{second}$. We represent the subtree that corresponds to $\pi_*$ by $\triangle_*$, where $*$ can be "best", "second", "candidate" or "promising". Theorem 1 shows the relationship between $\pi_{best}$ and $\pi_{second}$. We don't include theorem 1's proof due to space limitations.

**Theorem 1.** *Given* $\underline{V}$ *is uniformly improvable, there exist at least one* $\pi_{second}$ *in* $\Pi_{second}$ *and only one node* $b^*$ *in both* $\triangle_{best}$ *and* $\triangle_{second}$ *such that* $\pi_{second}(b^*)$ *is in* $\arg\max_{a \in A - \{\pi_{best}(b^*)\}} \underline{Q}(b^*, a)$ *and* $\pi_{second}(b') = \pi_{best}(b')$ *for any* $b'$ *in* $\triangle_{second} - \{b^*\}$.

Now, we define a set of policies $\Pi_{candidate}$, which is a useful tool for constructing SBPG. A policy $\pi_{candidate}$ is in the set $\Pi_{candidate}$ if and only if it satisfies that there exists a node $b^*$ in both $\triangle_{best}$ and $\triangle_{candidate}$ such that $\pi_{candidate}(b^*) \neq \pi_{best}(b^*)$ and $\pi_{candidate}(b') = \pi_{best}(b')$ for any $b'$ in $\triangle_{candidate} - \{b^*\}$. Let's look at this definition, we can easily conclude that there exists at least one $\pi_{second}$ in $\Pi_{candidate}$, and the definition of $\pi_{candidate}(b^*)$ seems to be confusing. Here, we first ignore this confusion since the intent becomes clear in Sect. 5.2. All policies $\pi_{candidate}$ in $\Pi_{candidate}$ are different in their definition of what $b^*$ is and what $\pi_{candidate}(b^*)$ is. Since there are $|\triangle_{best}|$ possible $b^*$ and $|A| - 1$ possible actions $\pi_{candidate}(b^*)$ at every $b^*$, $\triangle_{candidate}$ has $|\triangle_{best}|(|A| - 1)$ policies. All policies $\pi_{candidate}$ in $\Pi_{candidate}$ have a remarkable property: if $\underline{Q}(b^*, \pi_{candidate}(b^*))$ is greater than $\underline{Q}(b^*, \pi_{best}(b^*))$, then a better policy will be found. Note that SBPG's objective is to speed up the process of finding a better policy, it hence focuses on re-evaluating a set of "promising" policies' reward in $\Pi_{candidate}$.

As we see, $\triangle_{best}$ is a tree with infinite depth, and therefore, $\triangle_{best}$ may consist of an infinite number of nodes. To focus on the key part for $V^{\pi_{best}}(b_0)$, we prune the vast majority of nodes in the tree by adding a constraint condition: excess($b$,$d_b$) $> 0$ for all $b$ in $\triangle_{best}$. Here, the definition of excess($b$,$d_b$) is different, which depends on the algorithm that SBPG is combined with. When SBPG is combined with HSVI2 or SARSOP, it is defined as $(\bar{V}(b) - \underline{V}(b)) - \epsilon\gamma^{-d_b}$, whereas it is $(\bar{V}(b) - \underline{V}(b)) - \epsilon/2$ when SBPG is combined with FRTDP. In the next section, we use this definition as $\triangle_{best}$, but not the prior one.

## 5 Framework Implementation via a Greedy Strategy

This section provides an algorithmic implementation of Framework 1. Sect. 5.1-5.3 describe how to construct a greedy algorithm SBPG using some common ignored heuristics in HSVI2, FRTDP and SARSOP. Sect. 5.4 describes how to combine it with the three leading algorithms. Here, we first briefly describe the exploration mechanism in SBPG.

In a running race, the most "promising" player for surpassing the currently No.1 ranked player is the current No.2 player. SBPG's exploration mechanism stems from this principle. In HSVI2, FRTDP and SARSOP, policy's reward evaluation ($\underline{V}^{\pi}(b_0)$) by the lower bound is not exact, and sometimes very rough. Therefore, $\pi_{second}$'s reward re-evaluation is a good way of finding a better policy. The main difficulty is to locate $b^*$'s position in $\triangle_{second}$ (see Theorem 1). Fortunately, it is not a big obstacle since $\pi_{second}$'s property shown in Theorem 1 offers a guidance for selecting some "promising" policies from $\Pi_{candidate}$. Following the definition of $\Pi_{candidate}$, whether a policy in $\Pi_{candidate}$ is "promising" depends on its $b^*$ and the definition of $\pi(b^*)$. Sect. 5.1 provides a heuristic method via classification and error-minimization search to find a set of "promising" nodes

$b^*$. In Sect. 5.2, we use the second best action selection in defining $\pi(b^*)$. Sect. 5.3 introduces a sampling strategy that re-evaluates "promising" policies' reward.

### 5.1 Heuristic Search via Classification and Error-Minimization

A "promising" node plays a significant and easy role in finding a better policy. SBPG selects the "promising" nodes from $\triangle_{best}$ via the following strategy.

**Classification.** Each set $\triangle_{best}(i)$ has a weight to measure its importance in finding a better policy. For a node $\tau(b, a, z)$ at the $i^{th}$ depth of $\triangle_{best}$, let its lower bound improve $\delta$, then the backup operation at its parent node $b$ will bring an improvement of at least $\gamma\delta Pr(z|b, a)$ at $b$, further, the backup operations along the path from its parent node to the root node $b_0$ will bring $\underline{V}(b_0)$'s improvement of at least $\gamma^i\delta Pr(b|b_0, \pi_{best})$, where $Pr(b|b_0, \pi_{best})$ represents the probability of arriving at $b$ from $b_0$ by following the policy $\pi_{best}$. Let the lower bound of all nodes in $\triangle_{best}(i)$ improve $\delta$, then the backup operations from every node's parent node to $b_0$ will improve the lower bound at least $\gamma^i\delta \sum_{b\in\triangle_{best}(i)} Pr(b|b_0, \pi_{best})$ at $b_0$. Therefore, SBPG uses $\gamma^i \sum_{b\in\triangle_{best}(i)} Pr(b|b_0, \pi_{best})$ as $weight(\triangle_{best}(i))$.

According to the above definition of $weight$, we can conclude that $weight(\triangle_{best}(0)) = 1$, and $0 < weight(\triangle_{best}(i)) \leq 1$, for $1 \leq i \leq D_{\triangle_{best}}$. When SBPG is called, it decides whether to select a "promising" node from each $\triangle_{best}(i)$ using $weight(\triangle_{best}(i))$ as its selection probability. To avoid randomness, SBPG uses an equivalent deterministic algorithm to make decision.

**Error-Minimization Search.** When SBPG has decided to choose a node from $\triangle_{best}(i)$, the error-minimization search strategy will be useful to select a "promising" node from $\triangle_{best}(i)$. For each node in $\triangle_{best}(i)$, SBPG computes its heuristic value $H$ using Equation 6:

$$H(b) = \begin{cases} Pr(b|b_0, \pi_{best})(\bar{V}(b) - \underline{V}(b)) & \text{if } a_s^b \text{ exists} \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

Here, $Pr(b|b_0, \pi_{best})$ represents the significance that $\underline{V}(b)$'s improvement will bring into $\underline{V}(b_0)$'s, and $\bar{V}(b) - \underline{V}(b)$, to some extent, represents the easiness of improving $\underline{V}(b)$. Therefore, $H(b)$ is a trade-off between the significance and easiness. By the error-minimization search strategy, SBPG chooses the node with the highest non-zero heuristic value from $\triangle_{best}(i)$. The strategy is inspired by a similar search strategy in an efficient online POMDP algorithm [13].

### 5.2 Second Best Action Selection

According to Theorem 1, we should define $\pi(b^*) = \arg\max_{a\in A-\{\pi_{best}(b^*)\}} \underline{Q}(b^*, a)$. However, this action selection sometimes guides search towards a suboptimal action branch. We address it by a refined action selection $a_s^b$ in Equation 7:

$$a_s^b = \arg\max_{a\in A_s^b-\{\pi_{best}(b)\}} \underline{Q}(b, a). \tag{7}$$

Here, $A_s^b$ represents $\{a \in A | \bar{Q}(b,a) > \underline{Q}(b, \pi_{best}(b)\}$. We call $a_s^b$ as "$b$'s second best action". The second best action selection uses the upper bound $\bar{V}$ to clear away suboptimal action branches, just like the $\alpha$-$\beta$ pruning technique in game theory. Therefore, it is a good way of leveraging off the lower and upper bounds.

### 5.3 Second Best Policy-Guided Sampling and Updates

According to Sect. 5.1 and Sect. 5.2, we can obtain a set of "promising" policies $\Pi_{promising}$ from $\Pi_{candidate}$, whose elements are different in $b^*$ and $\pi(b^*)$. Now, we need to re-evaluate each $\pi_{promising}$ in $\Pi_{promising}$. Algorithm 2 first re-evaluates the subtree rooted at $Q^{\pi_{promising}}(b^*, a_s^{b^*})$ (see line 1-2 in Algorithm 2), in the hope that $Q^{\pi_{promising}}(b^*, a_s^{b^*})$ is greater than $Q^{\pi_{best}}(b^*, \pi_{best}(b^*))$. If so, it updates each node along the path from $b_0$ to $b^*$ in reversed order. Thus a better policy will be found. Here, $D_{max}$ represents the maximal sampling depth from $b^*$. We increase $D_{max}$ if SBPG hasn't found a better policy during a trial (see line 3-4 in Algorithm 3). Algorithm 3 shows a sketch of SBPG. Note that SBPG defines its action selection mainly according to the lower bound (see line 1-2 in Algorithm 2 and the definition of $\pi_{candidate}$ in Sect. 4), whereas HSVI2, FRTDP and SARSOP's action selection only depends on the upper bound. This is the deep reason that SBPG can be well combined with the three leading algorithms.

**Algorithm 2.** samplingAndUpdates

**Procedure** samplingAndUpdates($b^*$,$a_s^{b^*}$)
1: **for** i← 0 : $D_{max}$ **do**
2: update($b$), for all $b$ in $\{b \in \triangle_{promising}(d_b + D_{max} - i) | excess(b, d_b) > 0\}$;
3: **if** $\underline{V}(b^*)$ has been improved **then**
4: update each node along the path from $b_0$ to $b^*$ in reversed order;

**Algorithm 3.** SBPG

**Procedure** SBPG($\epsilon$)
1: select a set of "promising" policies $\Pi_{promising}$ from $\triangle_{best}$;
2: samplingAndUpdates($b^*$,$a_s^{b^*}$), for all $\pi_{promising}$ in $\Pi_{promising}$;
//Each $\pi_{promising}$ has unique $b^*$ and $a_s^{b^*}$.
3: **if** a better policy hasn't been found **then**
4: $D_{max}$++;

### 5.4 Combination of SBPG and Leading Algorithms

Algorithm 4 presents the SBPG+HSVI2 algorithm. In a similar way, the combined algorithms SBPG+FRTDP and SBPG+SARSOP can also be obtained. The three combined algorithms can easily be modified into anytime algorithms. In this paper, we define simply $condition(T_{SBPG}, T_*) = \frac{T_{SBPG}}{T_{SBPG}+T_*} \leq \frac{1}{2}$, where * represents one of HSVI2, FRTDP and SARSOP.

**Algorithm 4.** SBPG+HSVI2

**Function** $\pi$ = SBPG+HSVI2($\epsilon$)
1: Initialize the bounds $\underline{V}$ and $\bar{V}$;
2: $D_{max} = 1$;
3: **while** $\bar{V}(b_0) - \underline{V}(b_0) > \epsilon$
4: trialRecurse($b_0, \epsilon, 0$);
5: **if** condition($T_{SBPG}$, $T_{HSVI2}$) == true **then**
6: SBPG($\epsilon$);
7: **return** $\pi$ corresponding to $\underline{V}(b_0)$;

## 6 Empirical Evaluations

In this section, we present empirical results of our combined algorithms and compare them with HSVI2, FRTDP and SARSOP, respectively.

### 6.1 Experimental Setup

We implemented both SBPG+HSVI2 and SBPG+FRTDP based on zmdp-v1.1.7, in which HSVI2 and FRTDP have been included. For SBPG+SARSOP and SARSOP, we use appl-v0.3. The experimental platform is AMD Athlon(tm) 64 X2 Dual Core Processor 3600+ 2.00GHz, 2GB memory. The operation system is Ubuntu Linux 9.04. Table 1 lists all benchmark problems we have tested.

**Table 1.** Benchmark problems

| Problem | $\|S\|$ | $\|A\|$ | $\|Z\|$ | Problem | $\|S\|$ | $\|A\|$ | $\|Z\|$ |
|---|---|---|---|---|---|---|---|
| TagAvoid | 870 | 5 | 30 | LifeSurvey1 | 7,001 | 7 | 28 |
| RockSample_5_5 | 801 | 10 | 2 | RockSample_5_7 | 3,201 | 12 | 2 |
| RockSample_7_8 | 12,545 | 13 | 2 | RockSample_10_10 | 102,401 | 15 | 2 |

### 6.2 Results

We compare the performance of the combined algorithms with the original algorithms in terms of the reward and the time using via Table 2. Now, we describe how an algorithm's reward and time are obtained on a benchmark problem. Firstly, we ran every algorithm for a maximum of half an hour. Every algorithm returned a policy. We evaluated the expected discounted rewards of these policies using sufficiently large number (more than 10,000) of simulation runs, and then selected the minimal and maximal reward among them. We empirically set a threshold as (minimal reward + maximal reward) / 2 for the benchmark problem. Secondly, we ran each algorithm again and evaluated its policy for every "evaluated time interval". The evaluated time interval is dependent on the POMDP problem's size. The term of "time" records the first time that an

algorithm obtains a policy whose reward is greater than the threshold. And the term of "reward" is the corresponding policy's reward and the reward's 95% confidence interval.

**Table 2.** Performance comparison on benchmarks (e.t.i. = evaluated time interval)

| Method | Reward | Time (s) | Method | Reward | Time (s) |
|---|---|---|---|---|---|
| **TagAvoid** | e.t.i. = 1s | | **LifeSurvey1** | e.t.i. = 200s | |
| SBPG+HSVI2 | -6.09±0.09 | 13 | SBPG+HSVI2 | 93.68±0.14 | 1200 |
| HSVI2 | -6.05±0.09 | 591 | HSVI2 | 92.22±0.14 | 1800 |
| SBPG+FRTDP | -6.04±0.10 | 6 | SBPG+FRTDP | 94.39±0.15 | 200 |
| FRTDP | -6.09±0.12 | 46 | FRTDP | 94.30±0.15 | 600 |
| SBPG+SARSOP | -6.02±0.10 | 1.5 | SBPG+SARSOP | 94.42±0.15 | 200 |
| SARSOP | -6.06±0.12 | 6 | SARSOP | 94.40±0.15 | 800 |
| **RockSample_5_5** | e.t.i. = 0.5s | | **RockSample_5_7** | e.t.i. = 10s | |
| SBPG+HSVI2 | 19.23±0.07 | 1.0 | SBPG+HSVI2 | 24.62±0.06 | 60 |
| HSVI2 | 19.23±0.07 | 19.5 | HSVI2 | 24.61±0.07 | 460 |
| SBPG+FRTDP | 19.23±0.06 | 1.0 | SBPG+FRTDP | 24.60±0.05 | 90 |
| FRTDP | 19.23±0.06 | 3.5 | FRTDP | 24.62±0.06 | 480 |
| SBPG+SARSOP | 19.24±0.10 | 1.5 | SBPG+SARSOP | 24.61±0.10 | 40 |
| SARSOP | 19.23±0.10 | 4.0 | SARSOP | 24.61±0.12 | 360 |
| **RockSample_7_8** | e.t.i. = 200s | | **RockSample_10_10** | e.t.i. = 200s | |
| SBPG+HSVI2 | 21.51±0.13 | 600 | SBPG+HSVI2 | 20.39±0.10 | 1800 |
| HSVI2 | 21.37±0.12 | 1800 | HSVI2 | 19.47±0.09 | 1800 |
| SBPG+FRTDP | 21.51±0.11 | 800 | SBPG+FRTDP | 20.86±0.09 | 1200 |
| FRTDP | 21.49±0.10 | 800 | FRTDP | 20.11±0.12 | 1800 |
| SBPG+SARSOP | 21.49±0.11 | 800 | SBPG+SARSOP | 20.49±0.11 | 200 |
| SARSOP | 21.28±0.11 | 1800 | SARSOP | 20.49±0.11 | 600 |

## 7 Conclusions and Future Work

This paper presents a novel framework and an implementation of this framework for accelerating point-based POMDP algorithms. This implementation mainly constructs a greedy strategy SBPG via some common ignored heuristics in three leading point-based POMDP algorithms. It is important to note that the heuristics used in constructing a greedy strategy must be very useful, and the method of using them is key to our framework's performance. In our implementation, the lower bound over the optimal value function not used in the three leading algorithms' action-selection is very useful. The second best policy guided exploration mechanism in SBPG provides a promising way to accelerate these algorithms. As a future work, we will examine constructing more greedy strategies to speed up current point-based POMDP algorithms, and provide further empirical evidences for our proposed framework's efficiency.

**Acknowledgments.** Thanks to Feng Wu, Michael L. Littman, Trey Smith, Stéphane Ross, and David Hsu for ideas and discussions.

## References

1. Hoey, J., von Bertoldi, A., Poupart, P., Mihailidis, A.: Assisting Persons with Dementia during Handwashing Using a Partially Observable Markov Decision Process. In: Proceedings of International Conference on Vision Systems (2007)
2. Hsu, D., Lee, W.S., Rong, N.: A Point-Based POMDP Planner for Target Tracking. In: Proceedings of IEEE Conference on Robotics and Automation, pp. 2644–2650 (2008)
3. Smith, T.: Probabilistic Planning for Robotic Exploration. Ph.D. Dissertation, The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA (2007)
4. Kurniawati, H., Hsu, D., Lee, W.S.: SARSOP: Efficient Point-Based POMDP Planning by Approximating Optimally Reachable Belief Spaces. In: Proceedings of Robotics: Science and Systems (2008)
5. Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement Learning: A Survey. Journal of Artificial Intelligence Research 4, 237–285 (1996)
6. Smith, T., Simmons, R.: Point-Based POMDP Algorithms: Improved Analysis and Implementation. In: Proceedings of Uncertainty in Artificial Intelligence (2005)
7. Smith, T., Simmons, R.: Focused Real-Time Dynamic Programming for MDPs: Squeezing More Out of A Heuristic. In: Proceedings of the Twenty-First Conference on Artificial Intelligence, pp. 1227–1232 (2006)
8. Pineau, J., Gordon, G., Thrun, S.: Point-Based Value Iteration: An Anytime Algorithm for POMDPs. In: Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, pp. 1025–1032 (2003)
9. Spaan, M.T.J., Vlassis, N.: Perseus: Randomized Point-Based Value Iteration for POMDPs. Journal of Artificial Intelligence Research 24, 195–220 (2005)
10. Shani, G., Brafman, R.I., Shimony, S.E.: Forward Search Value Iteration for POMDPs. In: Proceedings of Twentieth International Joint Conference on Artificial Intelligence, pp. 2619–2624 (2007)
11. Hauskrecht, M.: Value-Function Approximations for Partially Observable Markov Decision Processes. Journal of Artificial Intelligence Research 13, 33–94 (2000)
12. Zhang, N.L., Zhang, W.: Speeding Up the Convergence of Value Iteration in Partially Observable Markov Decision Processes. Journal of Artificial Intelligence Research 14, 29–51 (2001)
13. Ross, S., Pineau, J., Paquet, S., Chaib-Draa, B.: Online Planning Algorithms for POMDPs. Journal of Artificial Intelligence Research 32(1), 663–704 (2008)