Solving Informative Partially Observable Markov Decision Processes

Weihong Zhang and **Nevin L. Zhang**

Department of Computer Science
Hong Kong University of Science & Technology
Clear Water Bay, Kowloon, Hong Kong, China

Abstract. Solving Partially Observable Markov Decision Processes (POMDPs) generally is computationally intractable. In this paper, we study a special POMDP class, namely informative POMDPs, where each observation provides good albeit incomplete information about world states. We propose two ways to accelerate value iteration algorithm for such POMDPs. First, dynamic programming (DP) updates can be carried out over a relatively small subset of belief space. Conducting DP updates over subspace leads to two advantages: representational savings in space and computational savings in time. Second, a point-based procedure is used to cut down the number of iterations for value iteration over subspace to converge. Empirical studies are presented to demonstrate various computational gains.

1 Introduction

Partially Observable Markov Decision Processes (POMDPs) provide a general framework for AI planning problems where effects of actions are nondeterministic and the state of the world is not known with certainty. Unfortunately, solving general POMDPs is computationally intractable [10]. For this reason, special classes of POMDPs incur much attention recently in the community(e.g., [8, 12]).

In this paper, we study a class of POMDPs, namely *informative POMDPs*, where any observation can restrict the world into a small set of states. Informative POMDPs come to be a median ground in terms of informative degree of observations. In one extreme case, unobservable POMDPs assume that observations do not provide any information about world states(e.g.,[9]). In other words, an observation cannot restrict the world into any range of states. In another extreme case, fully observable MDPs assume that an observation restricts the world into a unique state.

For informative POMDPs, we propose two ways to accelerate value iteration. First, for such POMDPs, we observe that dynamic programming(DP) updates can be carried out over a subset of belief space. DP updates over a subset leads to two advantages: fewer vectors are in need to represent a value function over a subset;computational savings are gained in computing sets of vectors representing value functions over the subset. Second, to further enhance our capability of solving informative POMDPs, a point-based procedure is integrated into value iteration over the subset [13]. The procedure effectively cuts down the number of iterations for value iteration to converge.

The integrated algorithm is able to solve an informative POMDP with 105 states, 35 observations and 5 actions within 430 CPU seconds.

The rest of the paper is organized as follows. In next section, we introduce background knowledge and conventional notations. In Section 3, we discuss problem characteristics of informative POMDPs and problem examples in the literature. In Section 4, we show how the problem characteristics can be exploited in value iteration. Section 5 reports experiments on comparing value iteration over belief space and over a subset of it. In Section 6, we integrate the point-based procedure to value iteration over a subset of belief space. In Section 7, we briefly discuss some related work.

2   Background

In a POMDP model, the environment is described by a set of states $\mathcal{S}$. The agent changes the states by executing one of a finite set of actions $\mathcal{A}$. At each point in time, the world is in one state s. Based on the information it has, the agent chooses and executes an action a. Consequently, it receives an *immediate reward* $r(s, a)$ and the world moves stochastically into another state s' according to a *transition probability* $P(s'|s, a)$. Thereafter, the agent receives an observation z from a finite set $\mathcal{Z}$ according to an *observation probability* $P(z|s', a)$. The process repeats itself.

Information that the agent has about the current state of the world can be summarized by a probability distribution over $\mathcal{S}$ [1]. The probability distribution is called a *belief state* and is denoted by b. The set of all possible belief states is called the *belief space* and is denoted by $\mathcal{B}$. A *belief subspace* or simply *subspace* is a subset of $\mathcal{B}$. If the agent observes z after taking action a in belief state b, its next belief state b' is updated as

$$b'(s') = kP(z|s', a) \sum_s P(s'|s, a)b(s) \tag{1}$$

where k is a re-normalization constant. We will sometimes denote this new belief state by $\tau(b, a, z)$.

A *policy* prescribes an action for each possible belief state. In other words, it is a mapping from $\mathcal{B}$ to $\mathcal{A}$. Associated with policy π is its *value function* V^π. For each belief state b, $V^\pi(b)$ is the expected total discounted reward that the agent receives by following the policy starting from b, i.e. $V^\pi(b) = E_{\pi, b}[\sum_{t=0}^\infty \lambda^t r_t]$, where r_t is the reward received at time t and λ ($0 \leq \lambda < 1$) is the *discount factor*. It is known that there exists a policy π^* such that $V^{\pi^*}(b) \geq V^\pi(b)$ for any other policy π and any belief state b. Such a policy is called an *optimal policy*. The value function of an optimal policy is called the *optimal value function*. We denote it by V^*. For any positive number ϵ, a policy π is ϵ-*optimal* if $V^\pi(b) + \epsilon \geq V^*(b)$ for any belief state b.

The *dynamic programming(DP) update operator* T maps a value function V to another value function TV that is defined as follows: for any b in $\mathcal{B}$,

$$TV(b) = \max_a [r(b, a) + \lambda \sum_z P(z|b, a)V(\tau(b, a, z))]$$

where $r(b, a) = \sum_s r(s, a)b(s)$ is the expected reward if action a is taken in b.

Value iteration is an algorithm for finding ϵ-optimal value functions. It starts with an initial value function V_0 and iterates using the formula: $V_n = TV_{n-1}$. Value iteration terminates when the *Bellman residual* $\max_b |V_n(b) - V_{n-1}(b)|$ falls below $\epsilon(1-\lambda)/2\lambda$. When it does, the value function V_n is ϵ-optimal.

Value function V_n is *piecewise linear and convex (PLC)* and can be represented by a finite set of $|\mathcal{S}|$-dimensional *vectors* [11]. It is usually denoted by $\mathcal{V}_n$. In value iteration, a DP update computes a set $\mathcal{V}_{n+1}$ representing V_{n+1} from $\mathcal{V}_n$ representing V_n.

3 Problem Characteristics

In general, a POMDP agent perceives the world by receiving observations. Starting from any state, if the agent executes an action a and receives an observation z, world states can be categorized into two classes by the observation model: states the agent can reach and states it cannot. Formally, the set of reachable states is $\{s | s \in \mathcal{S} \text{ and } P(z|s,a) > 0\}$. We denote it by $\mathcal{S}_{az}$.

An $[a, z]$ pair is said to be *informative* if the size $|\mathcal{S}_{az}|$ is much smaller than $|\mathcal{S}|$. Intuitively, if the pair $[a, z]$ is informative, after executing a and receiving z, the agent knows that the true world states are restricted into a small set. An observation z is said to be *informative* if $[a, z]$ is informative for every action a giving rise to z. Intuitively, an observation is informative if it always gives the agent an good idea about world states regardless of the action executed at previous time point. A POMDP is said to be *informative* if all observations are informative. In other words, any observation the agent receives always provides it a good idea about world states. Since one observation is received at each time point, a POMDP agent always has a good albeit imperfect idea about the world.

Informative POMDPs are especially suitable and appropriate for modeling a class of problems. In this class, a problem state is described by a number of variables(fluents). Some variables are observable while others are not. The possible assignments to observable variables form the observation space. A specific assignment to observable variables restricts the world states into a small range of them. A *slotted Aloha* protocol problem belongs to this class [2, 4]. Similar problem characteristics also exist in a non-stationary environment model proposed for reinforcement learning [7].

4 Exploiting Problem Characteristics

In this section, we show how informativeness can be exploited in value iteration. We start from belief subspace representation.

4.1 Belief subspace

We are interested in particular subspace type: *belief simplex*. It is specified by a list of *extreme belief states*. The simplex with extreme belief states $b_1, b_2, ...,b_k$ consists of all belief states of the form $\sum_{i=1}^{k} \lambda_i b_i$ where $\lambda_i \geq 0$ and $\sum_{i=1}^{k} \lambda_i = 1$.

Suppose the current belief state is b. If the agent executes an action a and receives an observation z, its next belief state is $\tau(b, a, z)$. If we vary the belief state in the belief

space $\mathcal{B}$, we obtain a set $\{\tau(b,a,z)|b \in \mathcal{B}\}$. Abusing notation, we denote this set by $\tau(\mathcal{B},a,z)$. In words, no matter which belief state the agent starts from, if it receives z after performing a, its next belief state must be in $\tau(\mathcal{B},a,z)$. Obviously, $\tau(\mathcal{B},a,z) \subseteq \mathcal{B}$.

Belief states in the set $\tau(\mathcal{B},a,z)$ have nice property which can be explored in context of informative POMDPs. By belief state update equation, if state s' is not in the set $\mathcal{S}_{az}$, the belief $b'(s')$ equals 0. The nonzero beliefs must distribute over states in $\mathcal{S}_{az}$. To reveal the relation between belief states and the set $\mathcal{S}_{az}$, we define a subset of $\mathcal{B}$:

$$\phi(\mathcal{B},a,z) = \{b| \sum_{s \in \mathcal{S}_{az}} b(s) = 1.0, \ \forall s \in \mathcal{S}_{az}, b(s) \geq 0\}.$$

It can be proven that for any belief state b, $\tau(b,a,z)$ must be in the above set. Therefore, $\tau(\mathcal{B},a,z)$ is a subset of $\phi(\mathcal{B},a,z)$ for a pair $[a,z]$. It is easy to see that $\phi(\mathcal{B},a,z)$ is a simplex in which each extreme point has probability mass on one state.

We consider the union of subspaces $\cup_{a,z}\phi(\mathcal{B},a,z)$ for all possible combinations of actions and observations. It consists of all the belief states the agent can encounter. In other words, the agent can never get out of this set. To ease presentation, we denote this set by $\phi(\mathcal{B},\mathcal{A},\mathcal{Z})$. Since each simplex in it is a subset of $\mathcal{B}$, so is $\phi(\mathcal{B},\mathcal{A},\mathcal{Z})$.

One example on belief space and subspaces is shown in Figure 1. A POMDP has four states and four observations. Its belief region is the tetrahedron ABCD where A, B, C and D are extreme belief states. For simplicity, we also use these letters to refer to the states. Suppose that $\mathcal{S}_{az}$ sets are independent of the actions. More specifically, for any action a, $\mathcal{S}_{az_0} = \{A,B,C\}$, $\mathcal{S}_{az_1} = \{A,B,D\}$, $\mathcal{S}_{az_2} = \{A,C,D\}$, and $\mathcal{S}_{az_3} = \{B,C,D\}$. In this POMDP, belief simplexes are four facets ABC, ABD, ACD and BCD and belief subspace $\phi(\mathcal{B},\mathcal{A},\mathcal{Z})$ is the surface of the tetrahedron. We also note that the subspace $\phi(\mathcal{B},\mathcal{A},\mathcal{Z})$ is much smaller than $\mathcal{B}$ in size.
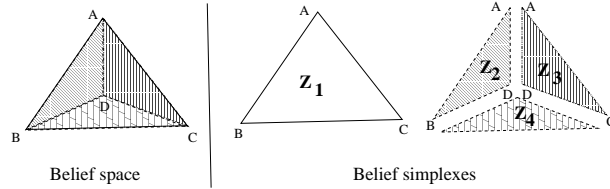

Fig. 1. Belief space, belief simplexes and belief subspace

4.2 Value functions over subspaces

A value function V_n over belief space $\mathcal{B}$ is a mapping from the belief space $\mathcal{B}$ to real line. Conceptually, for any b in $\mathcal{B}$, $V_n(b)$ is the maximum rewards the agent can receive in n steps if it starts from b. *Value function over subspace* is defined similarly. A n-step *value function over simplex* $\phi(\mathcal{B},a,z)$ is a mapping from the simplex. We denote it by $V_n^{\phi(\mathcal{B},a,z)}$. Conceptually, for a belief state b in subspace $\phi(\mathcal{B},a,z)$, $V_n^{\phi(\mathcal{B},a,z)}(b)$ is the

maximum rewards the agent can receive if it starts from b. An n-step value function $V_n^{\phi(\mathcal{B},\mathcal{A},\mathcal{Z})}(b)$ can be defined similarly and its domain is restricted to $\phi(\mathcal{B},\mathcal{A},\mathcal{Z})$.

Value function V_n can be represented by a set $\mathcal{V}_n$ of $|\mathcal{S}|$-dimensional vectors. If V_n is restricted to a simplex $\phi(\mathcal{B},a,z)$, it is a value function $V_n^{\phi(\mathcal{B},a,z)}$ over the simplex. It preserves the PLC property and can be represented by a set of vectors. For informative POMDPs, the restriction will result in a representational advantage. Specifically, for a pair $[a,z]$, since the beliefs over states outside $\mathcal{S}_{az}$ are zero, we need to allocate only $|\mathcal{S}_{az}|$ components for a vector. Typically, a value function is represented by a great number of vectors. If one represents the same value function over a simplex, it would lead to tremendous savings because the vectors are of smaller dimensions.

Given a collection $\{\mathcal{V}_n^{\phi(\mathcal{B},a,z)}\}$ in which each set $\mathcal{V}_n^{\phi(\mathcal{B},a,z)}$ is associated with an underlying set $\mathcal{S}_{az}$, defining a value function $\mathcal{V}_n^{\phi(\mathcal{B},\mathcal{A},\mathcal{Z})}$ over subspace $\phi(\mathcal{B},\mathcal{A},\mathcal{Z})$ exhibits a little bit difficulty. This is because the underlying set $\mathcal{S}_{az}$ contains different states for different $[a,z]$ pairs. It makes no sense if one defines the value function by computing the inner product of a vector and a belief state because possibly the dimension of the vector differs from that of the belief state. We define $\mathcal{V}_n^{\phi(\mathcal{B},\mathcal{A},\mathcal{Z})}$ this way: for any b in $\phi(\mathcal{B},\mathcal{A},\mathcal{Z})$,

$$\mathcal{V}_n^{\phi(\mathcal{B},\mathcal{A},\mathcal{Z})}(b) = \mathcal{V}_n^{\phi(\mathcal{B},a,z)}(b) \tag{2}$$

where $[a,z]$ is a pair such that $b \in \phi(\mathcal{B},a,z)$. The set $\mathcal{V}_n^{\phi(\mathcal{B},\mathcal{A},\mathcal{Z})}$ can be regarded as a two-dimensional array of sets over simplexes. When it needs to determine a value for a belief state, one (1) identifies a simplex containing it and (2) computes the value using the corresponding set of vectors. Obviously, the set $\mathcal{V}_n^{\phi(\mathcal{B},\mathcal{A},\mathcal{Z})}$ represents value function $V_n^{\phi(\mathcal{B},\mathcal{A},\mathcal{Z})}$.

4.3 DP update over subspace

In this subsection, we show how to conduct implicit DP update over belief subspace. The problem is cast as: given an array $\mathcal{V}_n^{\phi(\mathcal{B},\mathcal{A},\mathcal{Z})}$ representing $V_n^{\phi(\mathcal{B},\mathcal{A},\mathcal{Z})}$ over subspace $\phi(\mathcal{B},\mathcal{A},\mathcal{Z})$, how to compute an array $\mathcal{V}_{n+1}^{\phi(\mathcal{B},\mathcal{A},\mathcal{Z})}$?

To compute the set $\mathcal{V}_{n+1}^{\phi(\mathcal{B},\mathcal{A},\mathcal{Z})}$, we construct one set $\mathcal{V}_{n+1}^{\phi(\mathcal{B},a',z')}$ for any possible pair $[a',z']$. Before doing so, we recall how DP update over belief space constructs a vector in set $T\mathcal{V}_n$.

DP update $T\mathcal{V}_n$ computes a set $\mathcal{V}_{n+1}$ from a current set $\mathcal{V}_n$. It is known that each vector in $\mathcal{V}_{n+1}$ can be defined by a pair of action and a mapping from the set of observations to the set $\mathcal{V}_n$. Let us denote the action by a and the mapping by δ. For an observation z, we use δ_z to denote the mapped vector in $\mathcal{V}_n$. Given an action a and a mapping δ, the vector, denoted by $\beta_{a,\delta}$, is defined as follows: for each s in $\mathcal{S}$,

$$\beta_{a,\delta}(s) = r(s,a) + \lambda \sum_z \sum_{s'} P(s'|s,a)P(z|s',a)\delta_z(s').$$

By enumerating all possible combinations of actions and mappings, one can define different vectors. All these vectors form a set $\mathcal{V}_{n+1}$, i.e., $\{\beta_{a,\delta} | a \in \mathcal{A}, \delta : \mathcal{Z} \to \mathcal{V}_n\}$. It turns out that this set represents value function V_{n+1}.

We move forward to define a vector in $\mathcal{V}_{n+1}^{\phi(\mathcal{B},a',z')}$ given an array $\mathcal{V}_n^{\phi(\mathcal{B},\mathcal{A},\mathcal{Z})}$. Similar to the case in DP update $T\mathcal{V}_n$, a vector in set $\mathcal{V}_{n+1}^{\phi(\mathcal{B},a',z')}$ can be defined by a pair of action a and a mapping δ but with two important modifications. First, the mapping δ is from set of observations to the array $\mathcal{V}_n^{\phi(\mathcal{B},\mathcal{A},\mathcal{Z})}$. Moreover, for an observation z, δ_z is a vector in $\mathcal{V}_n^{\phi(\mathcal{B},a,z)}$. Second, the vector only need to be defined over the set $\mathcal{S}_{a'z'}$. To be precise, given a pair $[a',z']$, an action a and a mapping δ, a vector, denoted by $\beta_{a,\delta}$, can be defined as follows:

for each s in $\mathcal{S}_{a'z'}$,

$$\beta_{a,\delta}(s) = r(s,a) + \lambda \sum_z \sum_{s' \in \mathcal{S}_{az}} P(s'|s,a)P(z|s',a)\delta_z(s').$$

A couple of remarks are in order for the above definition. First, $\beta_{a,\delta}$ has only $|\mathcal{S}_{a'z'}|$ components. For states outside $\mathcal{S}_{a'z'}$, it is unnecessary to allocate space for them. Second, given the action a and observation z, when we define the component $\beta_{a,\delta}(s)$, we only need to account for next states in $\mathcal{S}_{az}$. This is true because for other states the probabilities of observing z are zero. It is important to note that an $|\mathcal{S}_{a'z'}|$-dimensional vector $\beta_{a,\delta}$ is constructed by making use of $|\mathcal{Z}|$ vectors: these vectors are of different dimensions because they come from different representing sets over simplexes.

If we enumerate all possible combinations of actions and mappings above, we can define various vectors. These vectors form a set

$$\{\beta_{a,\delta} | a \in \mathcal{A}, \delta : \mathcal{Z} \to \mathcal{V}_n^{\phi(\mathcal{B},\mathcal{A},\mathcal{Z})} \ \& \ \forall z, \delta_z \in \mathcal{V}_n^{\phi(\mathcal{B},a,z)}\}.$$

The set is denoted by $\mathcal{V}_{n+1}^{\phi(\mathcal{B},a',z')}$. The following lemma reveals the relation between the set and value function $V_{n+1}^{\phi(\mathcal{B},a',z')}$.

Lemma 1. *For any pair $[a,z]$, the set $\mathcal{V}_{n+1}^{\phi(\mathcal{B},a,z)}$ represents value function $V_{n+1}^{\phi(\mathcal{B},a,z)}$ over simplex $\phi(\mathcal{B},a,z)$.* □

For now, we are able to construct a set $\mathcal{V}_{n+1}^{\phi(\mathcal{B},a,z)}$ for a pair $[a,z]$. A complete DP update over $\phi(\mathcal{B},\mathcal{A},\mathcal{Z})$ needs to construct such sets for all possible pairs of actions and observations. After these sets are constructed, they are pooled together to form an array $\mathcal{V}_{n+1}^{\phi(\mathcal{B},\mathcal{A},\mathcal{Z})}$. It induces a value function by (2). It can be proved that the array $\mathcal{V}_{n+1}^{\phi(\mathcal{B},\mathcal{A},\mathcal{Z})}$ represents value function $V_{n+1}^{\phi(\mathcal{B},\mathcal{A},\mathcal{Z})}$ over the set $\phi(\mathcal{B},\mathcal{A},\mathcal{Z})$. The following theorem means that $V_{n+1}^{\phi(\mathcal{B},\mathcal{A},\mathcal{Z})}$ defines the same value function as V_{n+1} over the set $\phi(\mathcal{B},\mathcal{A},\mathcal{Z})$.

Theorem 1. *For any b in $\phi(\mathcal{B},\mathcal{A},\mathcal{Z})$, $V_{n+1}^{\phi(\mathcal{B},\mathcal{A},\mathcal{Z})}(b) = V_{n+1}(b)$.* □

As a corollary of the above theorem, we remark that, if b is a belief state in the intersection of two simplexes $\phi(\mathcal{B},a_1,z_1)$ and $\phi(\mathcal{B},a_2,z_2)$ for two pairs $[a_1,z_1]$ and $[a_2,z_2]$, $V_{n+1}^{\phi(\mathcal{B},a_1,z_1)}(b) = V_{n+1}^{\phi(\mathcal{B},a_2,z_2)}(b)$.

4.4 Complexity analysis

DP update $T\mathcal{V}_n$ improves values for belief space $\mathcal{B}$, while DP update of computing $\mathcal{V}_{n+1}^{\phi(\mathcal{B},\mathcal{A},\mathcal{Z})}$ from $\mathcal{V}_n^{\phi(\mathcal{B},\mathcal{A},\mathcal{Z})}$ improves values for subspace $\phi(\mathcal{B},\mathcal{A},\mathcal{Z})$. Since the subspace is much smaller than $\mathcal{B}$ in an informative POMDP, one expects:(1)fewer vectors are in need to represent a value function over a subspace;(2)since keeping useful vectors needs solve linear programs, this would lead to computational gains in time cost. Our empirical studies confirmed these two expectations.

4.5 Value iteration over subspace

Value iteration over subspace starts with a value function $\mathcal{V}_0^{\phi(\mathcal{B},\mathcal{A},\mathcal{Z})}$. Each set in it is initialized to contain a zero-vector of $|\mathcal{S}_{az}|$-dimension.

As value iteration continues, the Bellman Residual becomes smaller between two consecutive value functions over $\phi(\mathcal{B},\mathcal{A},\mathcal{Z})$. When the residual over $\phi(\mathcal{B},\mathcal{A},\mathcal{Z})$ falls below a predetermined threshold, it is also the case for the residual over any simplex. This suggest that the stopping criterion depend on residuals over simplexes. When the quantity $\max_{a,z} \max_{b \in \phi(\mathcal{B},a,z)} |\mathcal{V}_{n+1}^{\phi(\mathcal{B},a,z)}(b) - \mathcal{V}_n^{\phi(\mathcal{B},a,z)}(b)|$, the maximal difference between two consecutive value functions over all simplexes, falls below a threshold η, value iteration should terminate.

When value iteration terminates, it outputs the array $\mathcal{V}_n^{\phi(\mathcal{B},\mathcal{A},\mathcal{Z})}$. A value function V over the entire belief space can be defined by one step lookahead operator as follows:

$$V(b) = \max_a \{r(b,a) + \sum_z P(z|b,a)\mathcal{V}_n^{\phi(\mathcal{B},a,z)}(\tau(b,a,z))\} \ \forall \ b \in \mathcal{B}. \tag{3}$$

The value function V defined is said to be $\mathcal{V}_n^{\phi(\mathcal{B},\mathcal{A},\mathcal{Z})}$-greedy.

The hope is that if $\mathcal{V}_n^{\phi(\mathcal{B},\mathcal{A},\mathcal{Z})}$ is a good value function over $\phi(\mathcal{B},\mathcal{A},\mathcal{Z})$, so is $\mathcal{V}_n^{\phi(\mathcal{B},\mathcal{A},\mathcal{Z})}$-greedy value function. The following theorem shows how the threshold η impacts the quality of value function $\mathcal{V}_n^{\phi(\mathcal{B},\mathcal{A},\mathcal{Z})}$ and $\mathcal{V}_n^{\phi(\mathcal{B},\mathcal{A},\mathcal{Z})}$-greedy value function.

Theorem 2. *If $\eta \leq \epsilon(1-\lambda)/(2\lambda|\mathcal{Z}|)$ and value iteration over $\phi(\mathcal{B},\mathcal{A},\mathcal{Z})$ outputs $\mathcal{V}_n^{\phi(\mathcal{B},\mathcal{A},\mathcal{Z})}$, then $\mathcal{V}_n^{\phi(\mathcal{B},\mathcal{A},\mathcal{Z})}$-greedy value function is ϵ-optimal over the entire belief space.* $\square$

This theorem is important for two reasons. First, although value iteration over subspace computes a value function over a subset of belief space, ϵ-optimal value function over the entire belief space can be obtained by one step lookahead operator. Second, due to the availability of ϵ-optimal value function over $\mathcal{B}$, the agent can use it to select action for any belief state in $\mathcal{B}$. This is true for any initial belief states. Although $\phi(\mathcal{B},\mathcal{A},\mathcal{Z})$ consists of all belief states the agent can encounter after receiving any observation, the initial belief state does not necessarily belong to this set. The theorem implies that the $\mathcal{V}_n^{\phi(\mathcal{B},\mathcal{A},\mathcal{Z})}$-greedy value function can be used to guide the agent to select near optimal action for any initial belief state.

Finally, we note that to guarantee the ϵ-optimality, the threshold η (set to $\epsilon(1-\lambda)/(2\lambda|\mathcal{Z}|)$) in value iteration over subspace is smaller than that over belief space.

This stopping criterion is said to be *strict* one. If η is set to be $\epsilon(1 - \lambda)/(2\lambda)$ for value iteration over subspace, the condition is the *loose stopping criterion*. In our experiments, we use the loose stopping criterion.

5 Experiments

Experiments have been designed to test the performances of value iteration algorithms with and without exploiting the informative characteristics. Here we report results on a 3x3 grid world problem in Figure 2. It has nine states and the location 8(marked by $*$) is the goal state. The grid is divided by three rows and three columns. There are three locations along any row or column. The agent can perform one of four nominal-direction moving actions or a declaring success action. After performing a moving action, the agent reaches a neighboring location with probability 0.80 and stays at the same location with probability 0.20. Reasonable constraints are imposed to moving actions. For instance, if the agent is in location 0 and moves north, it stays at the same location. A declare-success action does not change the agent's location. After performing any action, the agent is informed of the column number with certainty. As such, the problem has three observations: col-0, col-1 and col-2. A move action incurs a cost of -1. If the agent declares success in location 8, it receives a reward of 10;If it does so in other locations, it receives a cost of -2.
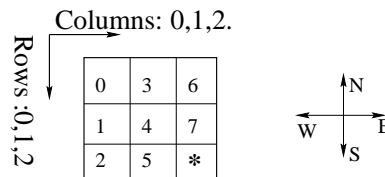

Fig. 2. A 3x3 grid world

This POMDP is informative. If value iteration is conducted without exploiting informativeness, one need to improve values over space $\mathcal{B}(= \{b| \sum_{i=0}^{8} b(s_i) = 1.0\}$. Since the observations are column numbers and independent of actions, DP update over subspace need to account for three simplexes: $\mathcal{B}_j = \{b| \sum_{3j,3j+1,3j+2} b(s_j) = 1.0\}$ for j=0,1,2 where j is the column number.

Our experiments are conducted on a SUN SPARC workstation. The discount factor is set to 0.95. The precision parameter is set to 0.000001. The quality requirement ϵ is set to 0.01. We use the loose stopping criterion. In our experiments, incremental pruning [12, 5] is used to compute sets of vectors representing value functions over belief space or subspace. For convenience, we use VI1 and VI to refer to the value iteration algorithms with and without exploiting regularities respectively. We compare VI and VI1 at each iteration along two dimensions: the size of set representing value function and time cost to conduct a DP update. The results are presented in Figure 3.
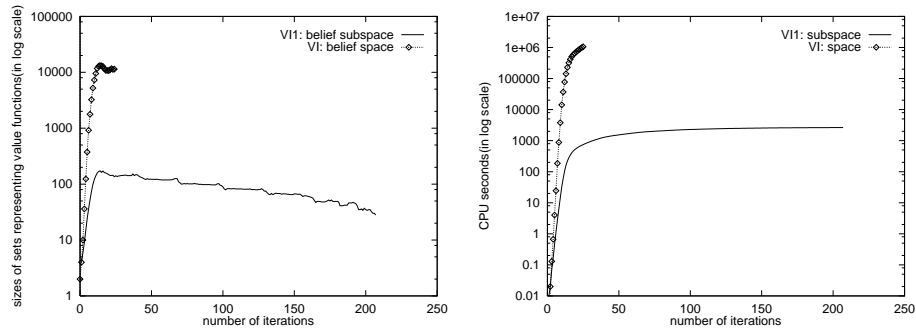
Fig. 3. Comparative study on value iterations over belief space and belief subspace

The first chart in the figure depicts the number of vectors in log-scale generated at each iteration for `VI` and `VI1`. In `VI`, at each iteration, we collects the sizes of sets representing value functions. In `VI1`, we compute three sets representing value functions over three simplexes and report the sum of the sizes of these three sets. For this problem, except the first iterations, `VI` generates significantly more vectors than `VI1`. In `VI`, after a severe growth, the number of vectors tends to be stable. In this case, value functions over belief space are represented by over 10,000 vectors. In contrary, the number of vectors generated by `VI1` is much smaller. Our experiments show that the maximum number is below 150. After `VI1` terminates, the value function is represented by only 28 vectors.

Due to the big difference between numbers of vectors generated by `VI1` and `VI`, `VI1` is significantly efficient than `VI`. This is demonstrated in the second chart in Figure 3. Note that CPU times in the figure are drawn in log-scale. When `VI1` terminates after 207 iterations, it takes around 2,700 seconds. On average, one DP update takes less than 13 seconds. For `VI`, it never terminates within reasonable time limit. By our data, it takes 1,052,590 seconds for first 25 iterations. On average, each iteration takes around 42,000 seconds. Comparing with `VI1`, we see that `VI1` is drastically efficient.

6 Integrating Point-based Improvement

In this section, we integrate a point-based improving procedure into value iteration over subspace and report our experiments on a larger POMDP problem.

6.1 Point-based improvement

The standard DP update $T\mathcal{V}$ is difficult because it has to account for infinite number of belief states. However, given a set $\mathcal{V}$ and a belief state b, computing the vector in the set $T\mathcal{V}$ at b is much easier. This can be accomplished by using a so-called *backup operator*.

Given a set $\mathcal{V}$ of vectors, a point-based procedure heuristically generates a finite set of belief points and backs up on the set to obtain a set of vectors. It is designed to have

this property: the value function represented by the set of backup vectors is better than the input set $\mathcal{V}$. Because the set of belief states are generated heuristically, point-based improvements is much cheaper than DP improvements.

A point-based value iteration algorithm interleaves standard DP update with multiple steps of point-based improvements. The standard DP update ensures that the output value function is ϵ-optimal when value iteration terminates.

6.2 Backup operator

In value iteration over subspace, DP update computes $\mathcal{V}_{n+1}^{\phi(\mathcal{B},\mathcal{A},\mathcal{Z})}$ from $\mathcal{V}_n^{\phi(\mathcal{B},\mathcal{A},\mathcal{Z})}$. To do so, it computes the set $\mathcal{V}_{n+1}^{\phi(\mathcal{B},a',z')}$ for each $[a', z']$ pair. It is conceivable that this is still not so easy because $\phi(\mathcal{B}, a', z')$ usually consists of infinite number of belief states.

Consequently, it is necessary to design a point based procedure to improve $\mathcal{V}_n^{\phi(\mathcal{B},a',z')}$ before it is fed to DP update over subspace $\phi(\mathcal{B}, a', z')$. We can generate heuristically a finite set of belief states in the simplex and back up on this set to obtain a set of vectors. The key problem is, given a set $\mathcal{V}_n^{\phi(\mathcal{B},\mathcal{A},\mathcal{Z})}$ and a belief state b in $\phi(\mathcal{B}, a', z')$, how to compute a vector in the set $\mathcal{V}_{n+1}^{\phi(\mathcal{B},a',z')}$? We can define a backup operator in this context. The backup vector can be built by three steps as follows.

1. For each action a and each observation z, find the vector in $\mathcal{V}_n^{\phi(\mathcal{B},a,z)}$ that has maximum inner product with $\tau(b, a, z)$. Denote it by $\beta_{a,z}$.
2. For each action a, construct a vector β_a by: for each s in the set $\mathcal{S}_{a'z'}$,

$$\beta_a(s) = r(s,a) + \gamma \sum_{z \in \mathcal{Z}} \sum_{s' \in \mathcal{S}_{az}} P(s', z|s, a)\beta_{a,z}(s')$$

   where $P(s', z|s, a)$ equals to $P(s'|s, a)P(z|s, a)$.
3. Find the vector, among the β_a's, that has maximum inner product with b. Denote it by β.

It can be proven that β is a vector in $\mathcal{V}_{n+1}^{\phi(\mathcal{B},a',z')}$. With the backup operator, before the set $\mathcal{V}_n^{\phi(\mathcal{B},a,z)}$ is fed to DP update over subspace, it is improved by multiple steps of point-based procedure. After these preliminary steps, the improved sets are fed to DP update over subspace. As such, we expect that the number of iterations can be reduced as value iteration converges.

6.3 Experiments

The problem is an extended version of the 3x3 grid world. It is illustrated in Figure 4. It has 35 columns. The goal location is 104, marked by $*$ in the figure. The agent is informed of its column number. So the problem has 105 states, 35 observations and 5 actions. The transition and observation models are similar to those in 3x3 grid world.

For simplicity, we use `PB-VI1` and `VI1` to refer to the algorithms conducting DP over subspace with and without integration of point-based procedure. Due to space

0	3	6	9																													102
1																																103
2																																*

Columns: 0...34

Fig. 4. A 3x35 grid world

limit, we report only on the time cost of DP update in `VI1` and `PB-VI1`. For convenience, the time reported for a DP update of `PB-VI1` consists of two portions: the time for multiple point-based improvements and the time for a DP update over subspace. The results are collected in Figure 5. Note that the time axis is drawn in log-scale.

Fig. 5. Comparative study on value iteration over subspace with/o point-based improving

We see that `VI1` stand-alone is still insufficient to solve the problem. It takes little time in first 15 iterations but the DP time grows later on. For instance, for the 20th iteration, it takes 2,500 CPU seconds and for 27th iteration, it takes 9,000 seconds. It is believed that `VI` can by no means solve this problem.

The situation changes when the point-based procedure is integrated. `PB-VI1` converges in 425 seconds after 36 DP updates (plus point-based improvements) over belief subspace. On average, each iteration takes less than 15 seconds. It is much faster than `VI1`. In addition, `VI1` needs to run 207 iterations to converge for 3x3 grid world (mentioned in previous section). Taking this as a reference, we also note that the point-based procedure is still efficient in cutting down the number of iterations for `VI1` to converge.

7   Related Work and Future Directions

Our concept of informative POMDPs is very similar to that of regional observable POMDPs in [12]. Both of them assume that any observation restricts the world into a

small set of states. In [12], a regional observable POMDP is proposed to approximate an original POMDP and value iterations for regional observable POMDPs are conducted over the entire belief space. Our work focuses on accelerating value iterations for such POMDP class by restricting them over a subset of belief space.

The approach we use to exclude belief states from being considered works much like that in reachability analysis (e.g., see [6, 3]). In fully observable MDP, this technique is used to restrict value iteration over a small subset of state space. Even although value iteration is restricted into a subspace for informative POMDPs, we show that value function of good quality over entire belief space can be obtained from value functions over its subset. In addition, as mentioned in Subsection 4.5, the value function generated by value iteration over subspace is guaranteed to be ϵ-optimality without much effort.

Acknowledgments

References

1. Astrom, K. J.(1965). Optimal control of Markov decision processes with incomplete state estimation. *Journal of Mathematical Analysis and Applications*, 10, 403-406.
2. Bertsekas, D. P. and Gallagher, R. G.(1995). *Data Networks*. Prentice Hall., Englewood Cliffs, N. J..
3. Boutilier, C., Brafman, R. I. and Geib, C. (1998). Structured reachability analysis for Markov decision processes. In *Proceedings of UAI-98*.
4. Cassandra, A. R.(1998). *Exact and approximate algorithms for partially observable Markov decision processes*. PhD Thesis, Department of Computer Science, Brown University.
5. Cassandra, A. R., Littman, M. L. and Zhang, N. L.(1997). Incremental pruning: A simple, fast, exact method for partially observable Markov decision processes. *Proceedings of Thirteenth Conference on Uncertainty in Artificial Intelligence*, 54-61.
6. Dean, T., Kaelbling, L. P., Kirman, J. and Nicholson, A. (1995). Planning under time constraints in stochastic domain. *Artificial Intelligence*, volume 76, number 1-2, Pages 35-74.
7. Choi, S.P.M., Yeung, D. Y. and Zhang, N.L. *An environment model for non-stationary reinforcement learning*. Advances in Neural Information Processing Systems 12(NIPS-99), 987-993.
8. Hansen, E. A. (1998). *Finite-memory controls of partially observable systems*. PhD thesis, Depart of Computer Science, University of Massachusetts at Amherst.
9. Hauskrecht, M.(2000). Value-function approximations for partially observable Markov decision processes. *Journal of Artificial Intelligence Research*, 13, 33-94.
10. Papadimitriou, C. H. and Tsitsiklis, J. N.(1987). The complexity of Markov decision processes. *Mathematics of Operations Research*, Vol. 12, No. 3, 441-450.
11. Sondik, E. J. (1971). The optimal control of partially observable decision processes. Ph D thesis, Stanford University, Stanford, California, USA.
12. Zhang, N. L. and Liu, W. (1997). A model approximation scheme for planning in stochastic domains. *Journal of Artificial Intelligence Research*, 7, 199-230.
13. Zhang, N. L. and Zhang, W. (2001). Speeding up the convergence of value iteration in partially observable Markov decision processes. *Journal of Artificial Intelligence Research*, Vol. 14, 29-51.