



states. It is often denoted by  $\mathcal{B}$ . If the agent starts with  $b$ , executes  $a$  and observes  $z$ , the next belief state  $b'$  is updated as follows: for any  $s'$ ,  $b'(s') = 1/k \sum_s P(s', z|s, a)b(s)$  where  $k (= P(z|b, a))$  is a normalizer and the joint probability  $P(s', z|s, a)$  equals to  $P(s'|s, a)P(z|s', a)$ . The belief state  $b'$  is sometimes denoted by  $\tau(b, a, z)$ .

A policy  $\pi$  prescribes an action for any belief state in  $\mathcal{B}$ . Associated with a policy  $\pi$  is its value function  $V^\pi$  which is a mapping from the belief space to the real line. For any  $b$ ,  $V^\pi(b)$  is the expected total discounted reward the agent receives if it starts with belief state  $b$  and follows policy  $\pi$ . Policy  $\pi$  dominates  $\pi_2$  if  $V^{\pi_1}(b) \geq V^{\pi_2}(b)$  for any  $b$ . The optimal policy dominates any other policies. Its value function is the optimal value function. An  $\epsilon$ -optimal value function differs the optimal by at most  $\epsilon$ . An  $\epsilon$ -optimal policy is a policy whose value function is  $\epsilon$ -optimal.

Value iteration starts with an initial value function  $V_0$  and iterates using the formula:

$$V_{n+1}(b) = \max_a [r(b, a) + \lambda \sum_z P(z|b, a)V_n(\tau(b, a, z))].$$

The notation  $V_n$  is referred to as *value function at step  $n$* . The step of computing  $V_{n+1}$  from  $V_n$  is referred to as *Dynamic-Programming (DP) update*. In practice, since value functions are representable by finite sets of vectors (Sondik 1971), DP update often refers to the process of obtaining the (minimal) representing set  $\mathcal{V}_{n+1}$  of  $V_{n+1}$  from that of  $V_n$ . Value iteration terminates when the *Bellman residual*,  $\max_b |V_n(b) - V_{n-1}(b)|$ , falls below  $\epsilon(1 - \lambda)/2\lambda$ .

## Belief Subset Selection

This section defines the subset, discusses its relation to the belief space and finally shows its linear representation.

### Subset Selection

The expression  $\tau(b, a, z)$  denotes the next belief state if the agent starts with  $b$ , executes  $a$  and observes  $z$ . The set  $\{\tau(b, a, z) | b \in \mathcal{B}\}$  contains all the next belief states if the agent executes  $a$  and observes  $z$ , no matter which belief states it starts with. We denote this set by  $\tau(\mathcal{B}, a, z)$ . The union  $\cup_{a,z} \tau(\mathcal{B}, a, z)$  consists of all belief states the agent can encounter, regardless of the initial belief states, actions executed and observations collected. For simplicity, we denote it by  $\tau(\mathcal{B})$ .

The set  $\tau(\mathcal{B})$  is the subset we choose for value iteration to work with. Its definition is an application of reachability analysis (Boutilier, Brafman, & Geib 1998; Dean *et al.* 1993).

### Relation Between Belief Subset and Belief Space

Given a pair  $[a, z]$ , it turns out that the following matrix plays an important role in determining the set  $\tau(\mathcal{B}, a, z)$  where  $n$  is the number of states. We denote the matrix by  $P_{az}$ .

$$\begin{pmatrix} P(s'_1, z|s_1, a) & P(s'_2, z|s_1, a) & \cdots & P(s'_n, z|s_1, a) \\ P(s'_1, z|s_2, a) & P(s'_2, z|s_2, a) & \cdots & P(s'_n, z|s_2, a) \\ \vdots & \vdots & \ddots & \vdots \\ P(s'_1, z|s_n, a) & P(s'_2, z|s_n, a) & \cdots & P(s'_n, z|s_n, a) \end{pmatrix}$$

To see why, if a belief state is viewed as a row vector,  $\tau(b, a, z)$  can be written as  $\frac{1}{k_b} b P_{az}$  where  $k_b$  is the constant  $P(z|b, a)$ , and  $b P_{az}$  means matrix multiplication. The relationship between  $\tau(\mathcal{B}, a, z)$  and  $\mathcal{B}$  is characterized in the following lemma.

**Lemma 1** For any  $[a, z]$ , there exists a bijection between the simplex  $\tau(\mathcal{B}, a, z)$  and the space  $\mathcal{B}$  if and only if the matrix  $P_{az}$  is invertible (i.e. the determinant  $|P_{az}|$  is non-zero).

Due to this lemma, if the matrix  $P_{az}$  is degenerate (i.e.,  $|P_{az}|$  is zero),  $\tau(\mathcal{B}, a, z)$  is a proper subset of  $\mathcal{B}$ .<sup>1</sup> The subset  $\tau(\mathcal{B})$  is proper only if any  $\tau(\mathcal{B}, a, z)$  is proper. Note that these conditions can be verified *a priori*.

**Theorem 1** The set  $\tau(\mathcal{B})$  is a proper subset of belief space  $\mathcal{B}$  only if each matrix  $P_{az}$  is degenerate.

### Representation

Subset representation addresses how to represent the subsets  $\tau(\mathcal{B}, a, z)$  and  $\tau(\mathcal{B})$ . Suppose that  $b_i$  is a unit vector ( $b_i(s)$  equals 1.0 for  $s = i$  and 0 otherwise). The set  $\{b_1, b_2, \dots, b_n\}$  is a *basis* of  $\mathcal{B}$  in the sense that any  $b$  in  $\mathcal{B}$  can be represented as a linear combination of vectors in the set. For each such  $b_i$ ,  $\tau(b_i, a, z)$  belongs to  $\tau(\mathcal{B}, a, z)$  if  $P(z|b_i, a) > 0$ . It has been proven that  $\{\tau(b_i, a, z) | P(z|b_i, a) > 0\}$  is a basis of the set  $\tau(\mathcal{B}, a, z)$  (Zhang 2001). This set is denoted by  $B_{\tau(\mathcal{B}, a, z)}$ . For this reason, the set  $\tau(\mathcal{B}, a, z)$  is said to be a *belief simplex* which is specified by the *extreme belief states* in the basis  $B_{\tau(\mathcal{B}, a, z)}$ .

**Theorem 2** For any pair  $[a, z]$ , the subset  $\tau(\mathcal{B}, a, z)$  is a simplex. Therefore, the subset  $\tau(\mathcal{B})$  is a union of simplexes.

### Subset Value Iteration

This section defines an MDP whose state space is  $\tau(\mathcal{B})$ . Value iteration for the MDP works with a subset of the belief space. Two versions of value iteration are presented.

#### Subset MDP

The set  $\tau(\mathcal{B})$  is a *closed* set in the sense that if the agent starts with a belief state in the set, no action can lead it to belief states outside the set. This property of  $\tau(\mathcal{B})$  allows one to define a *subset MDP* as follows. Note that it differs from the belief space MDP transformed from the original POMDP only in the state space (e.g. (Zhang & Liu 1997)).

- The state space is  $\tau(\mathcal{B})$  and action space is  $\mathcal{A}$ .
- Transition model: given a belief  $b$  in  $\tau(\mathcal{B})$  and an  $a$ , if  $b' = \tau(b, a, z)$  for some  $z$ , then  $P(b'|b, a)$  is  $P(z|b, a)$ .
- Reward model: given a  $b$  in  $\tau(\mathcal{B})$  and  $a$ , it specifies immediate reward  $r(b, a)$  as  $r(b, a) = \sum_{s \in \mathcal{S}} b(s)r(s, a)$ .

The DP equation for the above MDP follows. In the equation below,  $V_n^{\tau(\mathcal{B})}$  denotes its value function at step  $n$ .

For any  $b$  in  $\tau(\mathcal{B})$ ,

$$V_{n+1}^{\tau(\mathcal{B})}(b) = \max_a \{r(b, a) + \lambda \sum_z P(z|b, a)V_n^{\tau(\mathcal{B})}(\tau(b, a, z))\}$$

<sup>1</sup>It is proper if there is one  $b$  in  $\mathcal{B}$  such that  $b \notin \tau(\mathcal{B}, a, z)$ .

## DP Updates

Similarly to value functions over the belief space, value functions over simplex and subset possess the same property. They can be represented by finite sets of vectors. A DP update computes the (minimal) set representing  $V_{n+1}^{\tau(\mathcal{B})}$  from that representing  $V_n^{\tau(\mathcal{B})}$ . The rest of this subsection shows how to implicitly carry out DP update. Two issues must be addressed: how to compute a set representing  $V_{n+1}^{\tau(\mathcal{B})}$ , and how to compute the minimal set w.r.t. the subset  $\tau(\mathcal{B})$ .

The first issue can be settled by a similar procedure as in standard DP update. We enumerate all the possible vectors in the set representing  $V_{n+1}^{\tau(\mathcal{B})}$ . Each such vector can be defined by an action  $a$  and a mapping  $\delta$  from  $\mathcal{Z}$  to the set  $\mathcal{V}_n^{\tau(\mathcal{B})}$  (It is known that the pair  $[a, \delta]$  defines a policy tree in standard DP update (e.g., (Cassandra 1998)).) Given a pair  $[a, \delta]$ , the vector can be defined as follows and is denoted by  $\beta_{a,\delta}$ .

$$\beta_{a,\delta}(s) = r(s, a) + \lambda \sum_z \sum_{s'} P(s', z|s, a) \delta_z(s'). \quad (1)$$

where  $\delta_z$  is the mapped vector for observation  $z$ .

The following set is obtained by altering the actions and mappings in the above definition.

$$\{\beta_{a,\delta} | a \in \mathcal{A}, \delta : \mathcal{Z} \rightarrow \mathcal{V}_n^{\tau(\mathcal{B})} \ \& \ \forall z, \delta_z \in \mathcal{V}_n^{\tau(\mathcal{B})}\}. \quad (2)$$

The set is denoted by  $\mathcal{V}_{n+1}^{\tau(\mathcal{B})}$ . It can be proved that the set  $\mathcal{V}_{n+1}^{\tau(\mathcal{B})}$  represents value function  $V_{n+1}^{\tau(\mathcal{B})}$ . Furthermore, the set induces the same value function as  $V_{n+1}$  in  $\tau(\mathcal{B})$ .

**Theorem 3** *If  $\mathcal{V}_n^{\tau(\mathcal{B})}$  represents value function  $V_n$  in  $\tau(\mathcal{B})$ ,  $\mathcal{V}_{n+1}^{\tau(\mathcal{B})}$  represents  $V_{n+1}$  in the same subset.*

We now address the second issue: given a set of vectors, how to compute its minimal representation w.r.t.  $\tau(\mathcal{B})$ . We first explore the case for a simplex  $\tau(\mathcal{B}, a, z)$ . As in the standard *prune* operator, we need to prune useless vectors w.r.t.  $\tau(\mathcal{B})$ . Let the set be  $\mathcal{V}$  and  $\beta$  is a vector in the set. It is useful if and only if there exists a belief state  $b$  in  $\tau(\mathcal{B}, a, z)$  such that  $\beta \cdot b \geq \alpha \cdot b + x$  where  $x$  is some positive number,  $\alpha$  is a vector in the set  $\mathcal{V} - \{\beta\}$  and  $\cdot$  means inner product. On the other hand,  $b$  can be represented as  $\sum_i \lambda_i \tau(b_i, a, z)$ . If we replace  $b$  with  $\sum_i \lambda_i \tau(b_i, a, z)$ , the condition of determining  $\beta$ 's usefulness is equivalent to: whether there exists a series of nonnegative numbers  $\lambda_i$ s such that for any vector  $\alpha$ ,  $\beta \cdot \sum_i \lambda_i \tau(b_i, a, z) \geq \alpha \cdot \sum_i \lambda_i \tau(b_i, a, z) + x$ .

To determine  $\beta$ 's usefulness in the set  $\mathcal{V}$  w.r.t.  $\tau(\mathcal{B}, a, z)$ , the linear program `simplexLP` in Table 1 is used. When its optimality is reached, one checks its objective  $x$ . If it is positive, there exists a belief state in  $\tau(\mathcal{B}, a, z)$  at which  $\beta$  dominates other vectors. The belief state is represented as  $\sum_i \lambda_i \tau(b_i, a, z)$  where  $\lambda_i$ s are the solutions (values of variables). In this case,  $\beta$  is useful. Otherwise, it is useless.

To determine a vector's usefulness in a set w.r.t. the subset  $\tau(\mathcal{B})$ , one need to consider its usefulness w.r.t. each simplex. Again, let the set be  $\mathcal{V}$  and the vector be  $\beta$ . If  $\beta$  is useful

`simplexLP`( $\beta, \mathcal{V}, B_{\tau(\mathcal{B}, a, z)}$ ):

// Note:  $B_{\tau(\mathcal{B}, a, z)}$  is the basis of  $\tau(\mathcal{B}, a, z)$

Variables:  $x, \lambda_i$  for each  $i$

Maximize:  $x$

Constraints:

$$\begin{aligned} \beta \cdot \sum_i \lambda_i \tau(b_i, a, z) &\geq \alpha \cdot \sum_i \lambda_i \tau(b_i, a, z) + x \\ \text{for } \alpha \in \mathcal{V} \text{ and for each } i, \tau(b_i, a, z) &\in B_{\tau(\mathcal{B}, a, z)} \\ \sum_i \lambda_i &= 1, \lambda_i \geq 0 \text{ for } i. \end{aligned}$$

Table 1: determining a vector's usefulness w.r.t. a simplex

w.r.t. a simplex, it must be useful w.r.t. the subset. However, if it is useless w.r.t. a simplex, it may be useful w.r.t. another simplex. Hence, for a simplex, if  $\beta$  has been identified as useful, there is no need to check it again for subsequent simplexes. After all the simplexes have been examined, if  $\beta$  is useless w.r.t. all simplexes, it is useless w.r.t. the subset.

## Stopping Criterion

By MDP theory, as value iteration continues, the Bellman residual,  $\max_{b \in \tau(\mathcal{B})} |\mathcal{V}_n^{\tau(\mathcal{B})} - \mathcal{V}_{n-1}^{\tau(\mathcal{B})}|$ , becomes smaller. If it is smaller than  $\epsilon(1 - \lambda)/2\lambda$ , the algorithm terminates. The output set  $\mathcal{V}_{n-1}^{\tau(\mathcal{B})}$  is  $\epsilon$ -optimal for the subset MDP.

## Complexity Analysis

The performance of subset value iteration heavily depends on the "largeness" of the subset  $\tau(\mathcal{B})$ . If it is a proper subset of  $\mathcal{B}$ , this leads to computational benefits. First, the algorithm is expected to be more efficient than the standard one. This is because each DP update accounts for a smaller space than  $\mathcal{B}$ . Second, the complexity of value functions is reduced when the domain is restricted to the subset. This means that subset value iteration generates fewer vectors.

## Simplex-By-Simplex Value Iteration

The above value iteration works in a collective fashion in the sense that it directly computes value functions over  $\tau(\mathcal{B})$ . This subsection proposes value iteration in a simplex-by-simplex fashion. At each iteration, it computes value functions for individual simplexes. The rationale is, by letting value iteration work with the finer-grained belief subsets, it could be more efficient than its collective version.

The DP update is formulated as follows: given a collection  $\{\mathcal{V}_n^{\tau(\mathcal{B}, a, z)} | a \in \mathcal{A}, z \in \mathcal{Z}\}$  where each  $\mathcal{V}_n^{\tau(\mathcal{B}, a, z)}$  represents  $V_n^{\tau(\mathcal{B})}$  only in the simplex  $\tau(\mathcal{B}, a, z)$ , how to construct a set  $\mathcal{V}_{n+1}^{\tau(\mathcal{B}, a, z)}$  for each simplex  $\tau(\mathcal{B}, a, z)$ .

Likewise, a vector  $\beta_{a,\delta}$  in  $\mathcal{V}_{n+1}^{\tau(\mathcal{B}, a, z)}$  can be defined by an action  $a$  and a mapping  $\delta$ . The fact that  $\tau(b, a, z)$  must be in  $\tau(\mathcal{B}, a, z)$  for any  $b$  implies that for any  $z$ ,  $\delta_z$  can be restricted to a vector in the set  $\mathcal{V}_n^{\tau(\mathcal{B}, a, z)}$ . By altering the actions and mappings, one obtains the following set:

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

It differs from (2) in the mapping  $\delta$ . The above set is denoted by  $\mathcal{V}_{n+1}^{\tau(\mathcal{B}, a, z)}$ . The value function it induces possesses

the ideal property as expected. To obtain the minimal representation, one prunes the set w.r.t.  $\tau(\mathcal{B}, a, z)$ .

**Theorem 4** For any  $a$  and  $z$ , the set  $\mathcal{V}_{n+1}^{\tau(\mathcal{B}, a, z)}$  represents value function  $V_{n+1}^{\tau(\mathcal{B})}$  and  $V_{n+1}$  over  $\tau(\mathcal{B}, a, z)$  if each  $\mathcal{V}_n^{\tau(\mathcal{B}, a, z)}$  represents  $V_n^{\tau(\mathcal{B})}$  in the individual simplex.

Though subset value iteration can be conducted in either collective or individual fashion, they are essentially the same in terms of value functions induced. This is stated in the following theorem.

**Theorem 5** Let  $\mathcal{U} = \cup_{a,z} \mathcal{V}_{n+1}^{\tau(\mathcal{B}, a, z)}$ . For any  $b \in \tau(\mathcal{B})$ ,  $\mathcal{U}(b) = \mathcal{V}_{n+1}^{\tau(\mathcal{B})}(b)$ .

It is worthwhile of noting that for two action/observation pairs, the simplexes  $\tau(\mathcal{B}, a_1, z_1)$  and  $\tau(\mathcal{B}, a_2, z_2)$  might not be disjoint. Couple of remarks are in order for this case. First, the representing sets  $\mathcal{V}_{n+1}^{\tau(\mathcal{B}, a_1, z_1)}$  and  $\mathcal{V}_{n+1}^{\tau(\mathcal{B}, a_2, z_2)}$  might contain duplicates. Therefore the size  $\sum_{a,z} |\mathcal{V}_{n+1}^{\tau(\mathcal{B}, a, z)}|$  (the total number of vectors generated) could be greater than  $|\mathcal{V}_{n+1}^{\tau(\mathcal{B})}|$ . Second, by Theorem 4, for any  $b$  in the intersection of the simplexes,  $\mathcal{V}_{n+1}^{\tau(\mathcal{B}, a_1, z_1)}(b) = \mathcal{V}_{n+1}^{\tau(\mathcal{B}, a_2, z_2)}(b)$ . This is true because both sets of vectors represent  $V_{n+1}$  in  $\tau(\mathcal{B})$ .

## Subset, Decision-Making and Value Iteration

This section shows that value functions generated by subset value iteration can be used for decision-making in the entire belief space. It also studies an interesting relationship between subset selection and value iteration.

### Decision-making for the Entire Belief Space

When the algorithm terminates, the output set  $\mathcal{V}_{n-1}^{\tau(\mathcal{B})}$  is  $\epsilon$ -optimal w.r.t. the subset MDP. The agent can choose the  $\epsilon$ -optimal action for belief states in  $\tau(\mathcal{B})$ . However, for the original POMDP, the agent can start from any belief state in  $\mathcal{B}$ . What action should it choose for its initial belief state if it is not in  $\tau(\mathcal{B})$ ? It turns out that with a slightly stricter stopping criterion, the output value function generated by subset value iteration can be used for any belief states.

First, a value function over  $\tau(\mathcal{B})$  can be used to define a value function over  $\mathcal{B}$ . In fact, given a  $V^{\tau(\mathcal{B})}$ , a value function  $V$  over  $\mathcal{B}$  can be defined as follows: for any  $b$  in  $\mathcal{B}$ ,

$$V(b) = \max_a \{r(b, a) + \lambda \sum_z P(z|b, a) V^{\tau(\mathcal{B})}(\tau(b, a, z))\}.$$

The function  $V$  is said to be  $V^{\tau(\mathcal{B})}$ -greedy.

Second, if  $\mathcal{V}_n^{\tau(\mathcal{B})}$  represents the same value function as  $V_n$  in the subset  $\tau(\mathcal{B})$ , the  $\mathcal{V}_n^{\tau(\mathcal{B})}$ -greedy value function is actually  $V_{n+1}$ . In this sense, the set  $\tau(\mathcal{B})$  is said to be *sufficient* in terms of value function representation. If subset value iteration starts with the same value function as standard value iteration, inductively, they generate the same series of value functions in terms of the subset  $\tau(\mathcal{B})$ . Moreover, the step value functions in standard value iteration can always be derived from those in subset value iteration.

Third, if one slightly changes the stopping criterion in subset value iteration as in Theorem 6, its output value function can directly be used for  $\epsilon$ -optimal decision-making over the belief space (Zhang 2001). Note that to achieve the  $\epsilon$ -optimality, subset value iteration uses a stricter criterion and therefore takes more iterations than the standard algorithm. Meanwhile, stricter criterion means that value function returned by subset value iteration is closer to the optimality.

**Theorem 6** If subset value iteration terminates when  $\max_{b \in \tau(\mathcal{B})} |\mathcal{V}_n^{\tau(\mathcal{B})}(b) - \mathcal{V}_{n-1}^{\tau(\mathcal{B})}(b)| \leq \epsilon(1 - \lambda)/(2\lambda^2|\mathcal{Z}|)$  and it outputs  $\mathcal{V}_{n-1}^{\tau(\mathcal{B})}$ , then  $V_{n-1}^{\tau(\mathcal{B})}$ -greedy value function is  $\epsilon$ -optimal over the belief space.

### Belief Subset and Value Iteration

Since subset value iteration retains the quality of value functions, it can be regarded as an *exact* algorithm. One interesting problem is, if value iteration intends to retain quality, can it work with a proper subset of  $\tau(\mathcal{B})$ ?

In general, the answer is no. The reason follows. To compute  $V_{n+1}$ , one needs to keep values  $\mathcal{V}_n^{\tau(\mathcal{B})}$  for belief states in  $\tau(\mathcal{B})$ . Otherwise, if one accounts for a smaller set  $\mathcal{B}'$ , it can be proved that there exists a belief state  $b$  in  $\mathcal{B}$ , an action  $a$  and an observation  $z$  such that  $\tau(b, a, z)$  does not belong to  $\mathcal{B}'$ . It's known that the value update of  $\mathcal{V}_{n+1}(b)$  depends on the values for all possible next belief states. Due to the unavailability of  $V_n^{\tau(\mathcal{B})}(\tau(b, a, z))$ , the value  $V_{n+1}(b)$  can not be calculated exactly. Consequently, if subset value iteration works with a subset of  $\tau(\mathcal{B})$ , it can not be exact. In other words, it should be an *approximate* algorithm. To make it be exact, value iteration needs consider at least  $\tau(\mathcal{B})$ . In this sense, the subset is said to be a *minimal* sufficient set.

## Empirical Studies

We implemented subset value iteration in the simplex-by-simplex version using incremental pruning (Zhang & Liu 1997; Cassandra 1998). For convenience, it is denoted by *ssVI* and standard algorithm by *VI*. Here we focus on a simple maze problem because firstly we would compare the performance for both algorithms to run to completion, and secondly it eases the analysis of performance differences. We compare *VI* and *ssVI* along two dimensions: the sizes of representing sets and time costs of DP update. For *ssVI* and *VI*, the sizes are  $\sum_{a,z} |\mathcal{V}_n^{\tau(\mathcal{B}, a, z)}|$  and  $|\mathcal{V}_n|$  respectively.

Figure 1 presents the layout of the maze problem. There are 10 locations (states) and the goal is location 9. The agent can execute one of five actions: four “move” actions along four directions and a declare-goal action. The “move” actions can achieve intended effect with probability 0.8. The declare-goal action does not change the agent’s position. In the figure thick lines stand for walls and thin lines for nothing(open). At each time point, the robot reads four sensors which inform it of whether there is a wall or nothing along a direction. So an observation is a string of four letters. For example, at location 2, the observation is *owow* where *o* means nothing and *w* means wall. In the following, we present two versions of the maze problem: in one version, *ssVI* is superior; in the other version, *ssVI* is inferior.

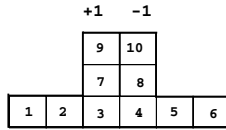


Figure 1: A maze problem

### Case I: $\tau(\mathcal{B}) \subset \mathcal{B}$

In this version, the observations(strings of letters) are collected deterministically. The reward model is defined as follows: when the agent declares goal at location 9, it receives a reward of 1 unit; if it does so at location 10, it receives a penalty of -1.

The statistics are presented in Figure 2. The first chart depicts the time cost of each DP update in log-scale for VI and  $\text{ssVI}$  with the strict stopping criterion. To compute a 0.01-optimal value function, VI terminates within 20,000 seconds after 162 iterations while  $\text{ssVI}$  terminates within 1,000 seconds after 197 iterations. We note that  $\text{ssVI}$  needs more iterations but it still takes much less time. The performance difference is big. Moreover, more iterations means that the value function generated by  $\text{ssVI}$  is closer to the optimality. This is not a surprising result if we take a look at

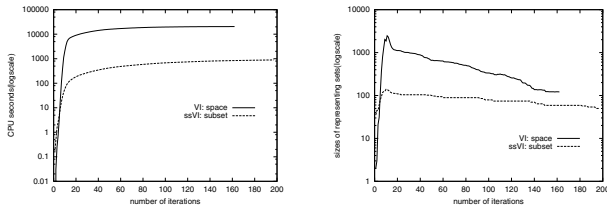


Figure 2: Performances for VI and  $\text{ssVI}$  on maze problem. Note that  $y$ -axis is in log-scale.

the matrix  $P_{az}$  for an action  $a$  and observation  $z$ . Let us assume that the observation is `owow` and hence the possible locations might be 2 or 5. Regardless of actions executed, only entries in column 2 and 5 in  $P_{az}$  can be non-zero. Therefore the matrix is highly sparse and degenerate and the simplex  $\tau(\mathcal{B}, a, z)$  is much smaller than  $\mathcal{B}$ . This analysis holds similarly for other combinations of actions and observations. This means  $\text{ssVI}$  accounts for only a small portion of the belief space and thus explains why  $\text{ssVI}$  is more efficient. In addition, we expect that the size of the sets representing value functions over subset is much smaller.

This is confirmed in the second chart. We see that at the same iteration VI always generates much more vectors than  $\text{ssVI}$ . The sizes at both curves increase sharply at first iterations and then stabilize. The size for VI reaches its peak of 2466 at iteration 11 and the maximum size for  $\text{ssVI}$  is 139 at iteration 10. This size in VI is about 20 times many as that in  $\text{ssVI}$ . This is a magnitude consistent with the performance difference. After the sizes stabilize, they are 130 in VI and around 50 in  $\text{ssVI}$ .

### Case II: $\tau(\mathcal{B}) = \mathcal{B}$

In this version, the actions set is enlarged to include a new one `stay`. If it is executed, it receives either a null observation with a probability 0.9 or a string with 0.1. The revised problem has more complications on the observation model. At most locations the agent receives a string as before. But due to hardware limitations, with a probability of 0.1, it wrongly reports the string `owow` as `owww` and `woww` as `wowo`. The reward model is changed to reflect new design considerations: the agent needs to pay for its information about states. For this purpose, `stay` yields no cost. In contrast, the “move” actions always cause a cost of 2.

The results are collected and presented in Figure 3. First we note both VI and  $\text{ssVI}$  are able to run only 11 iterations within a reasonable time limit (8 hours). The first chart presents the time costs along iterations. To run 11 iterations,  $\text{ssVI}$  takes 53,000 seconds while VI takes around 30,900 seconds. Therefore  $\text{ssVI}$  is slower than VI for this problem. However, the magnitude of performance difference is not big. To explain this, let us consider the matrix  $P_{az}$  de-

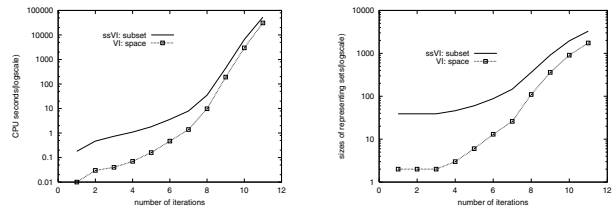


Figure 3: Performances for VI and  $\text{ssVI}$  on noisy maze.

termined by the action `stay` and observation null. The transition matrix is an identity and each state can lead to the null observation with probability of 0.9 if `stay` is executed. Therefore, the matrix  $P_{az}$  is invertible and the simplex  $\tau(\mathcal{B}, a, z)$  is actually  $\mathcal{B}$ . Because  $\text{ssVI}$  needs to account for additional simplexes for other combinations of actions and observations,  $\text{ssVI}$  must be less efficient than VI. This explains the performance difference in time between  $\text{ssVI}$  and VI. For the same reason,  $\text{ssVI}$  should produce more vectors than VI due to the intersection of simplexes. This is verified and demonstrated in the second chart of Figure 3. The curve for  $\text{ssVI}$  is always on the upper side of that for VI. For the 11th iteration,  $\text{ssVI}$  generates 3,300 vectors and VI generates around 1,700 vectors.

### Related Work

Two basic ideas behind subset value iteration in this paper are (1) reducing the complexity of DP updates and (2) reducing the complexity of value functions.

In a broad sense, most value iteration algorithms and their variations are common in that the efforts are devoted to reducing DP complexity. However, different algorithms and approaches take different forms for the similar purpose. These algorithms include grid-based algorithms where only belief states in the grid are considered at each iteration (Lovejoy 1991), state-space decomposition algorithms where the solution to the original MDP is

constructed from the solutions of a collection of smaller state-space MDPs (e.g. (Dean & Lin 1995)), model minimization technique where states are grouped together if they lead to similar behaviors via stochastic bisimulation equivalence (Dean & Givan 1997), algorithms using reachability analysis where new states are added to the subset being considered so far (Boutilier, Brafman, & Geib 1998; Dean *et al.* 1993), and real time dynamic programming where only belief states which have been explored are added into the set for value updates (Geffner & Bonet 1998).

The second idea behind subset value iteration is concerned with the representational complexity of value functions. Intuitively, the representing set of a value function over a belief subset contains fewer vectors than that of the same function over a belief set. This fact has been observed in (Hauskrecht & Fraser 1998). In their medical treatment example, a problem state is specified by several variables. They noted that the inconsistency between the assignments of different variables can be exploited to reduce the complexity of value functions.

## Conclusions and Extensions

In this paper, we study value iterations working with belief subset. We use reachability analysis to select a particular subset. The subset is (1) *closed* in that no actions can lead the agent to belief states outside it; (2) *sufficient* in that value function defined over it can be extended into the belief space; and (3) *minimal* in that value iteration needs to consider at least the subset if it intends to achieve the quality of value functions. The closedness enables one to formulate a subset MDP. We address the issues of representing the subset and pruning a set of vectors w.r.t. the subset. We then describe the subset value iteration algorithm. For a given POMDP, whether the subset is proper can be determined *a priori*. If this is the case, subset value iteration carries the advantages of representation in space and efficiency in time.

As for future directions, it would be interesting to ask the question, which POMDP classes in practice have the property whose belief subset is smaller than or the same as the belief space? (Zhang 2001) presents two POMDP classes where in one class the subset is proper while in the other class the subset is the same as the belief space. Different algorithms exploiting the subset structures have been proposed to solve different POMDP classes.

Another interesting direction is to exploit some asynchronous scheme for DP updates. At each iteration, one can conduct DP update over only a few simplexes other than their union. The residuals of consecutive value functions over individual simplexes can be used to select which simplexes to work with. One popular choice is to select those regions whose value functions have larger residuals.

Finally, the subset value iteration algorithm implemented in this paper works with POMDPs represented flatly. To achieve economy of representation from the model side (e.g. (Hansen & Feng 2000)), one promising direction is to implement a “structured” version of the algorithm for structured POMDPs. If this is feasible, one expects that the algorithm would be able to solve larger POMDPs.

**Acknowledgment** This work was supported in part by Hong Kong Research Grants Council under grant HKUST 6088 /01. The first author would like to thank Eric Hansen for insightful discussions, and Judy Goldsmith for valuable comments on an earlier writeup of the ideas in this paper.

## References

- Aström, K. J. 1965. Optimal control of Markov decision processes with incomplete state estimation. *Journal of Mathematical Analysis and Applications* 10:403–406.
- Boutilier, C.; Brafman, R. I.; and Geib, C. 1998. Structured reachability analysis for Markov decision processes. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI)*.
- Cassandra, A. R. 1998. *Exact and approximate algorithms for partially observable Markov decision processes*. Ph.D. Dissertation, Department of Computer science, Brown university.
- Dean, T., and Givan, R. 1997. Model minimization in Markov decision processes. In *Proceedings of National Conference on Artificial Intelligence (AAAI)*, 106–111.
- Dean, T. L., and Lin, S. H. 1995. Decomposition techniques for planning in stochastic domains. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI)*, 1121–1127.
- Dean, T. L.; Kaelbling, L. P.; Kirman, J.; and Nicholson, A. 1993. Planning with deadlines in stochastic domains. In *Proceedings of the 9th National Conference on Artificial Intelligence (AAAI)*, 574–579.
- Geffner, H., and Bonet, B. 1998. Solving large POMDPs using real time dynamic programming. In *Working Notes Fall AAAI Symposium on POMDPs*, 61–68.
- Hansen, E., and Feng, Z. 2000. Dynamic programming for POMDPs using a factored state representation. In *Proceedings of the 5th International Conference on Artificial Intelligence Planning and Scheduling (AIPS)*. Breckenridge, Colorado.
- Hauskrecht, M., and Fraser, H. 1998. Modeling treatment of ischemic heart disease with partially observable Markov decision processes. In *American Medical Informatics Association annual symposium on Computer Applications in Health Care*, 538–542. Orlando, Florida.
- Lovejoy, W. S. 1991. Computationally feasible bounds for partially observed Markov decision processes. *Operations Research* 39(1):162–175.
- Sondik, E. J. 1971. *The optimal control of partially observable decision processes*. Ph.D. Dissertation, Stanford University, Stanford, California, USA.
- Zhang, N. L., and Liu, W. 1997. A model approximation scheme for planning in partially observable stochastic domains. *Journal of Artificial Intelligence Research* 7:199–230.
- Zhang, W. 2001. *Algorithms for partially observable Markov decision processes*. Ph.D. Dissertation, Department of Computer science, the Hong Kong University of Science and Technology.