Forward Search Value Iteration For POMDPs

Guy Shani and **Ronen I. Brafman** and **Solomon E. Shimony**[*]
Department of Computer Science, Ben-Gurion University, Beer-Sheva, Israel

Abstract

Recent scaling up of POMDP solvers towards realistic applications is largely due to point-based methods which quickly converge to an approximate solution for medium-sized problems. Of this family HSVI, which uses trial-based asynchronous value iteration, can handle the largest domains. In this paper we suggest a new algorithm, FSVI, that uses the underlying MDP to traverse the belief space towards rewards, finding sequences of useful backups, and show how it scales up better than HSVI on larger benchmarks.

1 Introduction

Many interesting reinforcement learning (RL) problems can be modeled as partially observable Markov decision problems (POMDPs), yet POMDPs are frequently avoided due to the difficulty of computing an optimal policy. Research has focused on approximate methods for computing a policy (see e.g. [Pineau *et al.*, 2003]). A standard way to define a policy is through a value function that assigns a value to each belief state, thereby also defining a policy over the same belief space. Smallwood and Sondik [1973] show that a value function can be represented by a set of vectors and is therefore piecewise linear and convex.

A promising approach for computing value functions is the point-based method. A value function is computed over a finite set of reachable belief points, using a *backup* operation over single belief points - *point-based* backups, generating α-vectors. The assumption is that the value function would generalize well to the entire belief space.

Two of the key issues in point-based methods are the choice of the set of belief points and the order of backups. A successful approach for addressing both of these issues uses trial-based value iteration — simulating trajectories and executing backups over the encountered belief-states in reversed order (see e.g. [Bonet and Gefner, 1998]). Such methods use some type of heuristic to find useful trajectories.

[*]Partially supported by the Lynn and William Frankel Center for Computer Sciences, and by the Paul Ivanier Center for Robotics and Production Management at BGU.

Heuristic Search Value Iteration (HSVI - [Smith and Simmons, 2005]) is currently the most successful point-based algorithm for larger domains. HSVI maintains both a lower bound ($\underline{V}$) and an upper bound ($\bar{V}$) over the optimal value function. It traverses the belief space using a heuristic based on both bounds and performs backups over the observed belief points in reversed order. Unfortunately, using $\bar{V}$ slows down the algorithm considerably as updating $\bar{V}$ and computing upper bound projections ($\bar{V}(b)$) are computationally intensive. However, this heuristic pays off, as the overall performance of HSVI is better than other methods.

In this paper we suggest a new method for belief point selection, and for ordering backups which does not use an upper bound. Rather, our algorithm uses a search heuristic based on the underlying MDP. Using MDP information is a well known technique, especially as an initial guess for the value function (starting with the Q_{MDP} method of [Littman *et al.*, 1995]), but did not lead so far to very successful algorithms. Our novel algorithm — Forward Search Value Iteration (FSVI) — traverses together both the belief space and the underlying MDP space. Actions are selected based on the optimal policy for the MDP, and are applied to the belief space as well, thus directing the simulation towards the rewards of the domain. In this sense, our approach strongly utilizes the underlying MDP policy.

We tested our algorithm on all the domains on which HSVI was tested, and it both converges faster than HSVI and scales up better, allowing us to solve certain problems for which HSVI fails to converge within reasonable time.

2 Background and Related Work

We review the POMDP model and the associated notation, and provide a short overview of relevant methods.

2.1 MDPs, POMDPs and the belief-space MDP

A Markov Decision Process (MDP) is a tuple $\langle S, A, tr, R\rangle$ where S is a set of world states, A is a set of actions, $tr(s, a, s')$ is the probability of transitioning from state s to state s' using action a, and $R(s, a)$ defines the reward for executing action a in state s.

A Partially Observable Markov Decision Process (POMDP) is a tuple $\langle S, A, tr, R, \Omega, O, b_0\rangle$ where S, A, tr, R are the same as in an MDP, Ω is a set of observations and $O(a, s, o)$ is the probability of observing o after executing a

and reaching state s. The agent maintains a *belief-state* — a vector b of probabilities such that $b(s)$ is the probability that the agent is currently at state s. b_0 defines the initial belief state — the agent belief over its initial state.

The transition from belief state b to belief state b' using action a is deterministic given an observation o and defines the τ transition function. We denote $b' = \tau(b, a, o)$ where:

$$b'(s') = \frac{O(a, s', o) \sum_s b(s) tr(s, a, s')}{pr(o|b, a)} \quad (1)$$

$$pr(o|b, a) = \sum_s b(s) \sum_{s'} tr(s, a, s') O(a, s', o) \quad (2)$$

While the agent is unable to observe the true state of the world, the world itself still behaves as an MDP, which we call the underlying MDP.

In many MDP and POMDP examples the agent should either reach some state (called the goal state) where it receives a reward, or collect rewards that can be found in a very small subset of the state space. Other problems provide varying rewards in each state.

2.2 Value Functions for POMDPs

It is well known that the value function V for the belief-space MDP can be represented as a finite collection of $|S|$-dimensional vectors known as α vectors. Thus, V is both piecewise linear and convex [Smallwood and Sondik, 1973]. A policy over the belief space is defined by associating an action a to each vector α, so that $\alpha \cdot b = \sum_s \alpha(s) b(s)$ represents the value of taking a in belief state b and following the policy afterwards. It is therefore standard practice to compute a value function — a set V of α vectors. The policy π_V is immediately derivable using: $\pi_V(b) = \text{argmax}_{a:\alpha_a \in V} \alpha_a \cdot b$.

V can be iteratively computed using point-based *backup* steps, efficiently implemented [Pineau *et al.*, 2003] as:

$$backup(b) = \text{argmax}_{g_a^b : a \in A} b \cdot g_a^b \quad (3)$$

$$g_a^b = r_a + \gamma \sum_o \text{argmax}_{g_{a,o}^\alpha : \alpha \in V} b \cdot g_{a,o}^\alpha \quad (4)$$

$$g_{a,o}^\alpha(s) = \sum_{s'} O(a, s', o) tr(s, a, s') \alpha^i(s') \quad (5)$$

The vector representation is suitable only for lower bounds. An upper bound can be represented as a direct mapping between belief states and values. The H operator, known as the Bellman update, updates such a value function:

$$Q_V(b, a) = b \cdot r_a + \gamma \textstyle\sum_o pr(o|a, b) V_n(\tau(b, a, o)) \quad (6)$$

$$HV(b) = \max_a Q_V(b, a) \quad (7)$$

2.3 Trial-Based Value Iteration

Synchronous value iteration assumes that the entire state space is updated over each iteration. Asynchronous value iteration allows some states to be updated more than others, based on the assumption that value function accuracy is more crucial in these states. A well known form of asynchronous value iteration is trial-based updates, where simulation trials are executed, creating trajectories of states (for MDPs) or belief states (for POMDPs). Only the states in the trajectory are updated and then a new trial is executed. RTDP [Barto *et al.*, 1995] is a trial-based algorithm for MDPs (Algorithm 1).

Algorithm 1 RTDP

```
1: Initialize $Q(s, a) \leftarrow 0$
2: **while** true **do**
3:     $s \leftarrow s_0$
4:     **while** $s$ is not a goal state **do**
5:         **for each** $a \in A$ **do**
6:             $Q(s, a) \leftarrow R(s, a) + \sum_{s'} tr(s, a', s')V(s')$
7:         $a \leftarrow \text{argmax}_{a'} Q(s, a')$
8:         $s \leftarrow$ sample from $tr(s, a, *)$
```

RTDP-BEL — an adaptation of RTDP to POMDPs was suggested by Bonet and Geffner [Bonet and Gefner, 1998]. RTDP-BEL discretizes belief states and maintains Q-values for the discretized belief states only. Trials are executed over the POMDP mapping the real belief state into the closest discretized belief state which is then updated.

An important aspect for the convergence speed of trial-based algorithms is a good heuristic that leads towards important states and updates. Unfocused heuristics may cause the algorithm to spend much time updating useless states.

2.4 Heuristic Search Value Iteration (HSVI)

Computing an optimal value function over the entire belief space does not seem to be a feasible approach. A possible approximation is to compute an optimal value function over a subset of the belief space. An optimal value function for a subset of the belief space is no more than an approximation of a full solution. We hope, however, that the computed value function will generalize well for unobserved belief states.

Point-based algorithms [Pineau *et al.*, 2003; Spaan and Vlassis, 2005; Smith and Simmons, 2005] compute a value function only over a reachable subset of the belief space using point-based backups (Equations 3- 5).

Of this family of algorithms, HSVI (Algorithm 2) has shown the best performance over large scale problems. HSVI [Smith and Simmons, 2005] maintains both an upper bound ($\bar{V}$) and lower bound ($\underline{V}$) over the value function. It traverses the belief space following the upper bound heuristic, greedily selecting successor belief points where $\bar{V}(b) - \underline{V}(b)$ is maximal, until some stopping criteria have been reached. It then executes backup and H operator updates over the observed belief points on the explored path in reversed order.

HSVI can be viewed as a trial-based value iteration for POMDPs, even though the next belief state is not sampled but selected using the upper and lower bounds.

Even though HSVI is able to handle large problems, the H operator update and the interpolations used to find the value $\bar{V}(b)$ are computationally intensive, becoming more time consuming with each explored belief state. As such, HSVI spends much time maintaining the upper bound that is used only as a heuristic directing the belief space traversal.

We note, however, that Smith and Simmons also present theoretical results for the convergence of HSVI, relying on the existence of the upper bound. As we suggest to remove the upper bound, these theoretical results no longer apply.

Algorithm 2 HSVI

```
Function HSVI
1: Initialize $\underline{V}$ and $\bar{V}$
2: while $\bar{V}(b_0) - \underline{V}(b_0) > \epsilon$ do
3:     Explore($b_0, \underline{V}, \bar{V}$)

Function Explore($b, \underline{V}, \bar{V}$)
1: if Stopping criteria have not been reached then
2:     $a^* \leftarrow \text{argmax}_a Q_{\bar{V}}(b, a')$ (see Equation 6)
3:     $o^* \leftarrow \text{argmax}_o(\bar{V}(\tau(b, a, o)) - \underline{V}(\tau(b, a, o))$
4:     Explore($\tau(b, a^*, o^*), \underline{V}, \bar{V}$)
5:     add($\underline{V}, backup(b, \underline{V})$)
6:     $\bar{V} \leftarrow HV(b)$
```

Algorithm 3 FSVI

```
Function FSVI
1: Initialize $V$
2: while $V$ has not converged do
3:     Sample $s_0$ from the $b_0$ distribution
4:     MDPExplore($b_0, s_0$)

Function MDPExplore($b, s$)
1: if $s$ is not a goal state then
2:     $a^* \leftarrow \text{argmax}_a Q(s, a)$
3:     Sample $s'$ from $tr(s, a^*, *)$
4:     Sample $o$ from $O(a^*, s', *)$
5:     MDPExplore($\tau(b, a^*, o), s'$)
6: add($V, backup(b, V)$)
```

2.5 Using the underlying MDP

Using the underlying MDP optimal policy for the POMDP is a very well known idea and has been explored from many aspects in the past. Littman et al. [Littman *et al.*, 1995] suggest to use the optimal Q-values of the underlying MDP to create the Q_{MDP} value function for a POMDP:

$$Q_{MDP}(b) = \max_a Q(s, a)b(s) \quad (8)$$

Many grid-based techniques (e.g. [Zhou and Hansen, 2001]) initialize the upper bound over the value function using the underlying MDP. RTDP-BEL initializes a Q function for the POMDP using the optimal Q function for the MDP.

An important drawback of using the underlying MDP is the inability of the MDP to assess actions that gather information. Agents in a partially observable environment occasionally need to execute actions that do not move them towards a reward, but only improve their state of information about the current world state, such as activating a sensor. In an MDP the world state is always known and therefore information gathering actions produce no additional value. Agents relying on the Q_{MDP} heuristic, for example, will therefore never execute any such actions.

3 Forward Search Value Iteration

We propose a new algorithm, Forward Search Value Iteration (FSVI), using trial-based updates over the belief space of the POMDP. FSVI maintains a value function using α-vectors and updates it using point-based backups.

The heuristic FSVI uses to traverse the belief space is based on the optimal solution to the underlying MDP. We assume that such a solution is given as input to FSVI in the form of a Q-function over MDP states. This assumption is reasonable, as a solution to the underlying MDP is always simpler to compute than a solution to the POMDP.

FSVI (Algorithm 3) simulates an interaction of the agent with the environment, maintaining both the POMDP belief state b and the underlying MDP state s — the true state of the environment the agent is at within the simulation. While at policy execution time the agent is unaware of s, in simulation we may use s to guide exploration through the environment.

FSVI uses the MDP state to decide which action to apply next based on the optimal value function for the underlying MDP, thus providing a path in belief space from the initial belief state b_0 to the goal (or towards rewards). As we assume that the value function of the underlying MDP is optimal, this heuristic will lead the agent towards states where rewards can be obtained.

The trial is ended when the state of the underlying MDP is a goal state. When the MDP does not have a goal state we can use other criteria such as reaching a predefined sum of rewards or number of actions. If the goal is unreachable from some states we may also add a maximal number of steps after which the trial ends.

FSVI (Algorithm 3) is apparently very simple. Its simplicity translates into increased speed and efficiency in generating belief points and updating the values associated with belief points. FSVI's method for selecting the next action is very fast, requiring to check only $O(|A|)$ values (MDP Q-values) as opposed to any action selection method based on the current belief state. For example, RTDP-BEL takes $O(|S|)$ operations for discretizing the belief state before it can select the next action and HSVI needs $O(|A||O||S||\bar{V}|)$ operations, where $|\bar{V}|$ is the number of points in the upper bound, to compute the values of all the successors of the current belief state. As FSVI generates trajectories using forward projection, it is easy to determine good sequences of backups, simply going in reversed order. This ability is shared by HSVI, but not by other point-based algorithms such as Perseus [Spaan and Vlassis, 2005] and PBVI [Pineau *et al.*, 2003].

Other algorithms, such as HSVI and RTDP-BEL, use a heuristic that is initialized based on the MDP Q-function and use some form of interpolation over these Q-values. These algorithms also improve the heuristic by updating the Q-values to fit the POMDP values. Such algorithms therefore work initially much like the Q_{MDP} heuristic which is known to preform badly for many POMDP problems and in many cases gets stuck in local optima. These algorithms can potentially need many updates to improve the heuristic to be able to reach rewards or goal states.

As noted earlier, a major drawback of MDP based approaches is their inability to perform in information gathering tasks. FSVI is slightly different in that respect. FSVI uses point-based backups in which information gathering actions are also evaluated and considered. It is therefore able to perform single step information gathering actions such as the activation of a sensor. For example, in the RockSample domain [Smith and Simmons, 2005], the robot should activate a sensor to discover whether a rock is worthwhile, and indeed FSVI performs very well in the RockSample domain, executing such actions. However, when the information gathering requires a lengthy sequence of operations, such as in the heaven-hell problem [Bonet and Gefner, 1998] where the agent is required to pass a corridor to read a map and then return to take a reward, FSVI's heuristic will fail to lead it through the corridor, and therefore it cannot learn of the existence of the map. FSVI can learn to read the map (using an information gathering action) if it is on the path from an initial state to the goal.

This limitation can be removed by adding an exploration factor causing FSVI to occasionally choose a non-optimal action. In practice, however, it is unlikely that random exploration will rapidly lead towards meaningful trajectories.

4 Empirical Evaluations

4.1 Evaluation Metrics

We evaluate the following criteria:

Value function evaluation — Average discounted reward (ADR): $\frac{\sum_{i=0}^{\#trials}\sum_{j=0}^{\#steps}\gamma^j r_j}{\#trials}$. ADR is filtered using a first order filter to reduce the noise in ADR.

Execution time — we report the CPU time but as this is an implementation specific measurement, we also report the amount of basic operations such as backup, τ function, dot product ($\alpha \cdot b$) and $g_{a,o}^{\alpha}$ (Equation 5) computations .

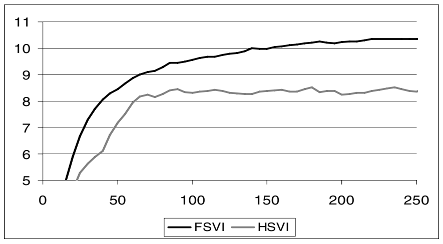
Memory — we show the size of the computed value function and the number of maintained belief points.

4.2 Experimental Setup

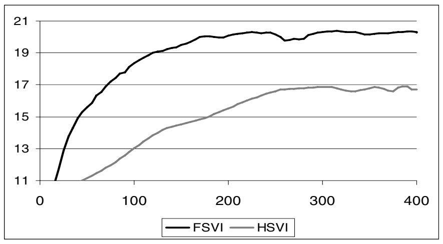
As HSVI is known to perform better than other point-based algorithms such as Perseus [Spaan and Vlassis, 2005] and PBVI [Pineau *et al.*, 2003], we compare FSVI only to HSVI. Comparison is done on a number of benchmarks from the point-based literature: Hallway, Hallway2 [Littman *et al.*, 1995], TagAvoid [Pineau *et al.*, 2003], RockSample [Smith and Simmons, 2005] and Network Ring [Poupart and Boutilier, 2003]. These include all the scalability domains on which HSVI was tested in [Smith and Simmons, 2005], except for Rock Sample 10,10 which could not be loaded on our system due to insufficient memory. Table 1 contains the problem measurements for the benchmarks including the size of the state space, action space and observation space, the number of trials used to evaluate the solutions, and the error in measuring the ADR for each problem.

The Rock Sample domain provides an opportunity for testing the scalability of different algorithms. However, these problems are somewhat limited: they assume deterministic state transitions as well as full observability for the robot location, making the problems easier to solve. To overcome these limitations we added a new domain — Noisy Rock Sample, in which agent movements are stochastic and it receives noisy observations as to its current location.

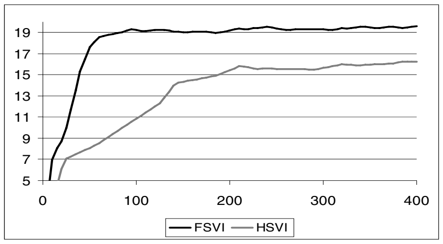
We implemented in Java a standard framework that incorporated all the basic operators used by all algorithms such as vector dot products, backup operations, τ function and so



(a)



(b)



(c)

Figure 1: Convergence on the Noisy Rock Sample 5,5 problem (a), the Rock Sample 7,8 problem (b) and the Rock Sample 8,8 problem (c). The X axis shows CPU time and the Y axis shows ADR.

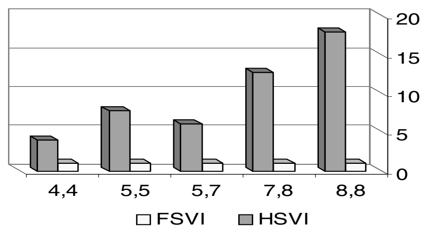


Figure 2: Normalized comparison of CPU time for the Rock Sample problems.

forth. All experiments were executed on identical machines: x86 64-bit machines, dual-proc, 2.6Ghz CPU, 4Gb memory, 2Mb cache, running linux and JRE 1.5.

We focus our attention on convergence speed of the value function to previously reported ADR. We executed HSVI and FSVI, interrupting them from time to time to compute the efficiency of the current value function using ADR. Once the filtered ADR has reached the same level as reported in past publications, execution was stopped. The reported ADR was then measured over additional trials (number of trials and error in measurement is reported in Table 1).

Method	**ADR**	$\|V\|$	**Time** (secs)	**# Backups**	**#**$g^{\alpha}_{a,o}$ x 10^6	$\|B\|$ x 10^4	**#**τ x 10^3	**#** $\alpha \cdot b$ x 10^6	$\|\bar{V}\|$ 10^3	$\bar{V}(b)$ 10^3	$\#H\bar{V}$
Hallway				187ms	0.13μs		0.37μs	33ns		0.5μs	65ms
HSVI	0.516	182	314	634	5.85	3.4	34.52	6.67	0.42	106.1	1268
FSVI	0.517	233	50	655	7.71	0.05	0.51	7.78			
	± 0.0024	± 71	± 15	± 100	± 2.46	± 0.01	± 0.08	± 2.49			
Hallway2				222ms	0.17μs		1μs	36ns		1μs	126ms
HSVI	0.341	172	99	217	1.56	2.11	11.07	1.81	0.23	37.2	434
FSVI	0.345	296	49	355	4.28	0.03	0.3	4.33			
	± 0.0027	± 22	± 8	± 33	± 0.73	± 0	± 0.03	± 0.74			
Tag Avoid				311ms	0.92μs		0.56μs	8.3ns		0.6μs	24.1ms
HSVI	-6.418	100	52	304	0.5	0.29	1.74	0.53	1.1	29.3	1635
FSVI	-6.612	174	45	182	0.37	0.01	0.14	0.39			
	± 0.15	± 25	± 8	± 27	± 0.1	± 0	± 0.02	± 0.11			
Rock Sample 4,4				92ms	0.2μs		0.46μs	0.031μs		0.2μs	4.67ms
HSVI	18.036	123	4	207	1.08	0.1	1.17	1.09	0.34	6.06	414
FSVI	18.029	84	1	204	0.24	0	0.04	0.25			
	± 0.024	± 76	± 1	± 122	± 0.41	± 0	± 0.01	± 0.42			
Rock Sample 5,5				114ms	1.1μs		2.5μs	0.033μs		0.64μs	24.1ms
HSVI	18.74	348	85	2309	10.39	0.26	2.34	10.5	0.8	101.1	1883
FSVI	19.206	272.5	11.1	626.2	1.47	0.02	0.1	1.49			
	± 0.063	± 75	± 5	± 247	± 0.88	± 0	± 0.02	± 0.89			
Noisy Rock Sample 5,5				314ms	0.86μs		1.9μs	0.035μs		0.69μs	44.5ms
HSVI	10.318	2639	1586	3528	129.11	2.01	23.05	132.4	0.88	264.9	9294
FSVI	10.382	924	210	1153	11.93	0.48	0.53	12.09			
	± 0.069	± 170	± 52	± 224	± 4.79	± 0.12	± 0.14	± 4.8			
Rock Sample 5,7				143ms	3.7μs		26.6μs	0.11μs		3.0μs	256.6ms
HSVI	22.67	274	205	350	0.65	1.0	4.3	0.99	3.44	14.09	702
FSVI	22.82	306.9	34.3	500	39684	3722	0.4	2.1			
	± 0.63	± 91.5	± 13.6	± 400.1	± 0.02	± 0.014	± 0.001	± 2.9			
Rock Sample 7,8				567ms	20μs		0.25ms	1.1μs		15μs	3.4sec
HSVI	20.029	533	3045	2089	14.51	1.5	4.1	14.66	3.42	9.53	628
FSVI	20.369	343.1	239	512.1	2.389	0.049	0.024	2.457			
	± 0.265	± 146.6	± 78.7	± 284.8	± 2.32	± 0.059	± 0.013	± 2.357			
Rock Sample 8,8				570ms	25μs		0.35ms	2.6μs		18μs	2.7sec
HSVI	19.474	762	13917	1317	10.83	4.01	58.09	11.43	17.3	58.38	2638
FSVI	19.646	261.4	783	367.8	0.042	0.816	0.176	1.183			
	± 0.337	± 76.9	± 295	± 125.1	± 0.02	± 0.013	± 0.06	± 0.661			
Network Ring 8				164ms	1μs		2.6ms	4.5ns		11μs	31.8ms
HSVI	42.27	80	19	153	0.004	0.25	8.44	0.31	0.39	8.46	307
FSVI	42.39	40.5	6.75	252	0.004	0.022	0.24	0.19			
	± 0.18	± 16	± 6.1	± 146	± 0.002	± 0.012	± 0.14	± 0.19			
Network Ring 10				553ms	10.6μs		13.3ms	23.3ns		99.4μs	369ms
HSVI	51.43	69	141	103	0.0036	0.29	6.9	0.144	1.11	6.9	206
FSVI	51.44	33.25	47	267.7	0.002	0.025	0.255	0.19			
	± 0.03	± 6.14	± 14.3	± 85.78	± 0.0004	± 0.008	± 0.081	± 0.086			

Table 2: Performance measurements. Model rows show rough estimates of basic operations execution time.

Problem	$\|S\|$	$\|A\|$	$\|O\|$	#trials	ADR Error
Hallway	61	5	21	10,000	± 0.0015
Hallway2	93	5	17	10,000	± 0.004
Tag Avoid	870	5	30	10,000	± 0.045
Rock Sample 4,4	257	9	2	10,000	± 0.075
Rock Sample 5,5	801	10	10	10,000	± 0.3
Noisy RS 5,5	801	10	27	10,000	± 0.3
Rock Sample 5,7	3,201	12	2	10,000	± 0.25
Rock Sample 7,8	12,545	13	2	1,000	± 0.25
Rock Sample 8,8	16,385	13	2	1,000	± 0.25
Network Ring 8	256	17	2	2,000	± 1.1
Network Ring 10	1024	21	2	2,000	± 0.98

Table 1: Benchmark problem parameters

4.3 Results

Table 2 presents our experimental results. For each problem and method we report:

1. Resulting ADR.
2. Size of the final value function ($|V|$).
3. CPU time until convergence.
4. Backups count.
5. $g^{\alpha}_{a,o}$ operations count.
6. Number of computed belief states.
7. τ function computations count.
8. Dot product operations count.
9. Number of points in the upper bound ($|\bar{V}|$).
10. Upper bound projection computations count ($\bar{V}(b)$).
11. Upper bound value updates count ($H\bar{V}(b)$).

The last 3 items refer only to HSVI as FSVI does not maintain an upper bound. The reported numbers do not include the repeated expensive computation of the ADR, or the initialization time (identical for both algorithms).

To illustrate the convergence rate of each algorithm, we also plotted the ADR as a function of CPU time in Figure 1. These graphs contain data collected over separate executions with fewer trials, so Table 2 is more accurate.

4.4 Discussion

Our results clearly indicate that FSVI converges faster than HSVI and scales up better. The convergence rate shown in Figure 1 is always faster, the time required to reach optimal ADR is also always faster, and this difference become more pronounced as problem size increases. Figure 2 presents a more focused comparison of the relative ability of the two solvers to scale up. Overall, it appears that HSVI is slowed down considerably by its upper bound computation and its updates. While it is possible that HSVI's heuristic leads to a more informed selection of belief points, FSVI is able to handle more belief points, and faster. In fact, in the Rock Sample domains, we can see that FSVI is able to converge with fewer belief points and even slightly improved ADR. Thus, on these domains, the heuristic guidance in point selection offered by the MDP policy is superior to the bounds-based choice of HSVI. Indeed, we can also see that the number of backups FSVI executes is in most cases less than HSVI, hinting that the chosen trajectories (and hence backups) are also better than those selected by HSVI.

Observe that the upper bound of HSVI allows the definition of additional properties such as a stopping criterion based on the gap between bounds over the initial belief point b_0, and an upper bound over the number of trials required to close that gap. In practice, however, HSVI never manages to reduce this gap considerably (as also noted in [Smith and Simmons, 2005]) and reaches maximal performance when the gap is still very large. This is to be expected, as the the convergence guarantees for HSVI relies on the eventual exploration of the entire And-Or search graph for the domain until an appropriate depth that grows as γ approaches 1.

As noted above, FSVI is not able to deal adequately with longer information-gathering sequences, such as those required in the heaven-hell domain. HSVI performs better than FSVI on these domains, although it, too, does not do too well. We tested both algorithms on a deterministic version of heaven-hell. HSVI took 52ms to converge and required 71 backups, while FSVI needed 137ms and 937 backups using an exploration factor of 0.5. When only the activation of single step information gathering actions is needed, such as all the Rock Sample and Network domains, FSVI performs well.

Given that different approximation algorithms for POMDPs are likely to perform well on different classes of domains, it is desirable to be able to determine prior to execution of an algorithm whether it will work well on a given problem. FSVI appears to be the best current candidate for domains in which long information seeking plans are not required. Thus, it is interesting to ask whether such domains are easy to recognize. We believe that the following initial schema we are currently exploring might work well. Roughly, we start by defining informative states as states where observations are available that result in radically reduced belief state entropy. If all such states are visited with reasonably high probability by following the underlying MDP optimal policy, then FSVI should perform well. In comparison to the cost of solving the POMDP, this test is relatively cheap to perform.

5 Conclusions

This paper presents a new point-based algorithm — Forward Search Value Iteration (FSVI) — that executes asynchronous value iteration. FSVI simulates trajectories through the belief space choosing actions based on the underlying MDP optimal action. Once the trajectory has reached a goal state, FSVI performs backups on the belief states in reversed order.

FSVI is a simple algorithm that is easy to implement and leads to good quality solutions rapidly. This was demonstrated by comparing FSVI to a state-of-the-art algorithm, HSVI, a closely related point-based algorithm that uses a different heuristic for belief space traversals. Our experiments show that FSVI is much faster and scales up better on almost all benchmark domains from the literature.

The underlying MDP provides a heuristic that is both fast to compute and use and also always directs the agent towards rewards. This heuristic, however, limits the ability of the agent to recognize traversals that will improve its state information. Namely, information gathering tasks that require lengthy traversals to gather information cannot be solved by FSVI. This deficiency was demonstrated on the heaven-hell problem which requires a detour in order to collect important information. Thus, it is apparent that future work should consider how to revise FSVI so that its generated trajectories will visit such states. This can perhaps be done by executing simulations that try to reduce the entropy of the current belief space, using information visible to the underlying MDP, or by examining some augmented state space. Given FSVI's performance and simplicity, we believe that such useful modifications are possible.

References

[Barto *et al.*, 1995] A. G. Barto, S. J. Bradtke, and S. P. Singh. Learning to act using real-time dynamic programming. *Journal of AI*, 72(1):81–138, 1995.

[Bonet and Gefner, 1998] B. Bonet and H. Gefner. Solving large POMDPs using real time dynamic programming. In *AAAI Fall Symposium on POMDPs*, 1998.

[Littman *et al.*, 1995] M. L. Littman, A. R. Cassandra, and L. P. Kaelbling. Learning policies for partially observable environments: Scaling up. In *ICML'95*, 1995.

[Pineau *et al.*, 2003] J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In *IJCAI*, August 2003.

[Poupart and Boutilier, 2003] P. Poupart and C. Boutilier. Bounded finite state controllers. In *NIPS 16*, 2003.

[Smallwood and Sondik, 1973] R. Smallwood and E. Sondik. The optimal control of partially observable processes over a finite horizon. *OR*, 21, 1973.

[Smith and Simmons, 2005] T. Smith and R. Simmons. Point-based pomdp algorithms: Improved analysis and implementation. In *UAI 2005*, 2005.

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

[Zhou and Hansen, 2001] R. Zhou and E. A. Hansen. An improved grid-based approximation algorithm for POMDPs. In *IJCAI*, pages 707–716, 2001.