Analysis of Methods for solving MDPs

(Extended Abstract)

Marek Grześ and Jesse Hoey
David R. Cheriton School of Computer Science, University of Waterloo, Canada
{mgrzes, jhoey}@cs.uwaterloo.ca

ABSTRACT

New proofs for two extensions to value iteration are derived when the type of initialisation of the value function is considered. Theoretical requirements that guarantee the convergence of backward value iteration and weaker requirements for the convergence of backups based on best actions only are identified. Experimental results show that standard value iteration performs significantly faster with simple extensions that are investigated in this work.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search

General Terms

Algorithms, Experimentation, Theory

Keywords

Policy Iteration, Markov Decision Process, Value Iteration

1 INTRODUCTION

We consider the problem of finding an optimal policy in discrete time, finite state and action, discounted (by factor $\gamma < 1$) as well as undiscounted ($\gamma = 1$) Markov Decision Processes (MDPs) [6]. A standard MDP notation is used from [3]. The following definitions are considered:

DEFINITION 1. *Q is pessimistic if $Q(x,a) \leq Q^*(x,a)$ and optimistic if $Q(x,a) \geq Q^*(x,a)$.*

DEFINITION 2. *Q is monotone pessimistic if $Q(x,a) \leq R_x(a) + \gamma \sum_{x'} T_{x,a}(x')V(x')$ and is monotone optimistic if $Q(x,a) \geq R_x(a) + \gamma \sum_{x'} T_{x,a}(x')V(x')$ for all x and a, where $V(x) = \max_a Q(x,a)$.*

2 ANALYSIS

In our recent work [3], a new backup of the value function was proposed that exploits the idea of updating best actions only (BAO). The approach was shown to be very successful in PAC-MDP reinforcement learning that requires frequent replanning of a changing MDP. The current work investigates how this idea can help in general MDP planning where every MDP is solved once. We also show a new theorem which allows applying the BAO operator in a more general scenario:

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.
Copyright © 2012, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

THEOREM 1. *Planning based on backups that, in every state, keep updating all best actions until the Bellman error of best actions is smaller than ϵ (BAO) converges to the optimal value function when the initial value function is optimistic.*

Our recent work [3] has identified specific problems with the convergence of backward value iteration (BVI) [2]. Here, we show new, formal theoretical requirements that guarantee that backward value iteration will converge.

THEOREM 2. *In the backward value iteration algorithm specified in [2], the policy induced by the current value function is proper (i.e., every state reaches the goal state with probability 1 [1]) after every iteration when:*

1. *the initial value function is monotone pessimistic, i.e., the conditions of Definition 2 are satisfied*
2. *the initial policy is proper, i.e., at least one goal state is in the policy graph of each state*

When the policy induced by the current value function of the BVI algorithm is proper after every iteration, the algorithm will update all states in every iteration and upon termination the Bellman error satisfies the termination condition on all states.

3 RESULTS

In order to test BAO in general MDPs, the following algorithms are evaluated: (1) VI: standard Gauss-Seidel value iteration [1], (2) MPI(k): modified policy iteration [7] where k is the constant number of iterations in policy evaluation, (3) PI: policy iteration [4], and (4) PS: prioritised sweeping with priority based on the Bellman error [5]. If BAO is applicable, it is used as one of the options and added to the name of the algorithm in the results. Also, a simplified version of BAO is used, named BAOnce, that updates best actions only once during every visit to the state. $V(i)$, V_{max}, V_{min}, V^+, and V^- mean that the value function of a particular algorithm was initialised with i, $R_{max}/(1-\gamma)$, $R_{min}/(1-\gamma)$, and upper and lower bounds on V^* correspondingly. Every domain was evaluated 10 times, for every randomly generated domain 10 instances were generated, the precision ϵ was 10^{-5}, and the standard error of the mean is shown in the results which display the planing time and the number of performed backups (the best results are in boldface).

VI, by default, cannot beat PI/MPI on domains with a high number of actions. For this reason, the first set of domains is generated according to [6] and has a high number of actions: the number of states and actions in every state is 100, and an action can lead to three randomly selected states with a probability sampled from a truncated Gaussian distribution with mean 20 and standard deviation 5 or from a uniform distribution on [1-100].

Nr	Time [ms]		Backups		Algorithm
1	3869.5	± 159.0	7970000.0	± 332699	VI-V(0)
2	3780.1	± 172.2	7662000.0	± 367979	VI-Vmax
3	2546.5	± 127.2	5158000.0	± 251183	VI-V+
4	840.4	± 61.2	641943.6	± 41327	VI-Vmax-BAO
5	**104.1**	**± 3.9**	114576.1	± 4805	**VI-V+-BAO**
6	**91.3**	**± 2.8**	73694.2	± 2044	**VI-V+-BAOnce**
7	5569.2	± 143.0	6421040.0	± 177804	PS-V+
8	1907.7	± 78.3	94820.0	± 3445	MPI(2)-V(0)
9	441.5	± 20.6	99680.0	± 4283	MPI(10)-V(0)
10	238.9	± 10.8	97060.0	± 4028	MPI(20)-V(0)
11	**122.9**	**± 4.3**	255330.0	± 10614	**MPI(500)-V(0)**
12	136.5	± 5.4	309910.0	± 14962	PI-V(0)
13	1079.2	± 58.3	57700.0	± 2579	MPI(2)-V+
14	133.6	± 6.6	303910.0	± 16916	PI-V+

Table 1: Results on non-terminating MDPs, Gaussian rewards and $\gamma = 0.99$

Nr	Time [ms]		Backups		Algorithm
1	3545.9	± 147.0	7526000.0	± 310506	VI-V(0)
2	3024.4	± 127.4	6305000.0	± 255679	VI-Vmax
3	**170.9**	**± 4.6**	172349.5	± 5251	**VI-Vmax-BAO**
4	169.3	± 3.0	127090.0	± 2314	VI-Vmax-BAOnce
5	6958.2	± 142.7	7819750.0	± 155515	PS-Vmax
6	1963.9	± 72.2	96840.0	± 3460	MPI(2)-V(0)
7	431.8	± 14.2	98630.0	± 3279	MPI(10)-V(0)
8	250.6	± 6.8	102980.0	± 2862	MPI(20)-V(0)
9	**101.1**	**± 4.8**	209310.0	± 10885	**MPI(500)-V(0)**
10	**111.4**	**± 5.4**	251550.0	± 12444	**PI-V(0)**

Table 2: Results on non-terminating MDPs, uniformly distributed rewards and $\gamma = 0.99$

The first experiment evaluates domains with Gaussian reward (see Table 1). MPI improves its performance and gets closer to the performance of PI when k grows. All rewards are positive (and similar due to Gaussian distribution) here, and evaluation of every policy makes progress towards an optimal solution, and for that reason it makes sense to advance evaluation of every policy (high k) and do fewer policy updates - the situation where VI is poor. BAO with V_{max} is better than standard VI, but loses against MPI. Only a more informative initialisation, V^+, allowed BAO to be both faster and to reduce the number of backups beyond what was achieved by the best MPI settings. Certainly, one could argue that V^+ is usually not known exactly in the real situation, however sometimes (see the car replacement example below) a bound, far better than V_{max}, can be determined and the discussed experiment shows that such a bound would be very convenient for the BAO update.

Since BAO continuously adapts its evaluated policy, our guess was that it may waste time on evaluating all actions which are similar due to a low variance in the Gaussian rewards. Therefore, the same set of domains was generated with a uniform reward distribution. Results in Table 2 show the evidence that higher variance in values of rewards made BAO perform better even with uninformative V_{max} initialisation. Here, there are actions which are proved to be non-optimal initially and BAO can help.

Car replacement from [4] was evaluated as a realistic domain with many actions: there are 41 states and 41 actions. Results are in Table 3. $\gamma = 0.97$ since in [4], it is justified as having a real meaning of around 12% annual interest rate. Rewards have high variance, but this time there is another property that strongly influences the performance of evaluated algorithms. Specifically, a short horizon policy is very sub-optimal when compared with a long horizon policy. Actions that yield high instantaneous reward are sub-optimal in the long term (selling a good car now and buying a cheap one may result in getting money now but incurs losses in the long term). Hence, BAO first learns actions which seem promising in short term and then unlearns them. The same applies to MPI. Small k makes MPI slower. With sufficiently large k, policies are evaluated 'almost exactly', and this helps avoiding short horizon policies. This also explains why MPI with lowest k is even slower than BAO because MPI applies full backups during policy improvement. $V(0)$ could be used to initialise the value function in BAO because in this domain there is never a positive long term reward (the possession of a car always incurs costs). With this knowledge, BAO can be competitive even on this challenging domain. If the bound can be improved, BAO gains further speed-up. Thus, $V(0)$, V_{max}, and V^+ yields optimistic initialisation required by BAO, and V_{min} and V^- pessimistic which was originally required by the theory of MPI [7], however the recent literature shows that this requirement can be avoided [1].

Nr	Time [ms]		Backups		Algorithm
1	206.5	± 8.0	591880.1	± 24543	VI-Vmax
2	144.6	± 6.6	429999.8	± 21736	VI-V+
3	169.6	± 5.3	494214.0	± 15791	VI-V(0)
4	123.5	± 8.0	378729.3	± 21790	VI-V-
5	160.2	± 5.6	498248.4	± 18250	VI-Vmin
6	126.8	± 0.8	176371.1	± 592	VI-Vmax-BAO
7	**30.7**	**± 2.0**	46615.6	± 882	**VI-V+-BAO**
8	**55.9**	**± 1.1**	81765.3	± 1350	**VI-V(0)-BAO**
9	124.5	± 3.0	159412.1	± 494	VI-Vmax-BAOnce
10	**25.8**	**± 0.4**	36149.7	± 498	**VI-V+-BAOnce**
11	**48.8**	**± 0.8**	64849.7	± 281	**VI-V(0)-BAOnce**
12	397.1	± 3.1	734117.3	± 3216	PS-Vmax
13	279.6	± 1.1	540306.2	± 2237	PS-V+
14	314.4	± 1.4	596263.0	± 3923	PS-V(0)
15	226.8	± 1.8	447260.8	± 2255	PS-V-
16	277.7	± 1.6	537243.5	± 1959	PS-Vmin
17	**16.1**	**± 0.6**	18158.9	± 487	**MPI(20)-Vmax**
18	**13**	**± 0.2**	14559.1	± 292	**MPI(20)-V+**
19	**13.1**	**± 0.2**	14883	± 251	**MPI(20)-V(0)**
20	**13.7**	**± 0.4**	15293	± 360	**MPI(20)-V-**
21	**15.8**	**± 0.5**	17535.7	± 562	**MPI(20)-Vmin**
22	26.3	± 0.7	80097.6	± 2332	PI-Vmax
23	25.2	± 0.5	76264.1	± 1598	PI-V+
24	26.6	± 0.8	81413.7	± 2403	PI-V(0)
25	25.8	± 0.9	77067.7	± 3203	PI-V-
26	27.8	± 0.4	84660.9	± 1870	PI-Vmin

Table 3: Results on car replacement

4 CONCLUSION

Our experiments have shown that, thanks to BAO updates, the gap between MPI and VI is significantly reduced on challenging domains with many actions. Unpublished comparisons with BVI on stochastic shortest path problems showed that standard VI can also outperform prioritised approaches when BAO is used.

Acknowledgement This research was sponsored by American Alzheimer's Association through the Everyday Technologies for Alzheimer's Care (ETAC) program. The first author was supported by a fellowship from the Ontario Ministry of Research and Innovation.

5 REFERENCES

[1] D. P. Bertsekas. *Dynamic Programming and Optimal Control (2 Vol Set)*. Athena Scientific, 3rd ed., 2007.

[2] P. Dai and E. A. Hansen. Prioritizing Bellman backups without a priority queue. In *Proc. of ICAPS*, 2007.

[3] M. Grześ and J. Hoey. Efficient planning in R-max. In *Proc. of AAMAS*, 2011.

[4] R. A. Howard. *Dynamic Programming and Markov Processes*. MIT Press, Cambridge, 1960.

[5] A. W. Moore and C. G. Atkenson. Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning*, 13:103–130, 1993.

[6] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994.

[7] M. L. Puterman and M. C. Shin. Modified policy iteration algorithms for discounted Markov decision problems. *Management Science*, 24:1127–1137, 1978.