On the Convergence of Techniques that Improve Value Iteration

Marek Grześ and Jesse Hoey
David Cheriton School of Computer Science, University of Waterloo, Canada

WATERLOO CHERITON SCHOOL OF COMPUTER SCIENCE

Goals

- (1) Establish requirements that guarantee convergence of backward value iteration
- (2) Improve performance of value iteration on domains with many actions where policy iteration has advantages
- (3) Show simple ways of improving standard value iteration

Significance

- Backward value iteration needs correct initialisation that we identified
- In contrast to policy iteration, value iteration is easy to parallelise and we show how to make value iteration work better in those cases where policy iteration is traditionally faster
- Basic value iteration can be improved without any sophisticated methods and we show empirical evidence that such simple improvements could be considered in the field

Introduction

- The problem of planning in Markov Decision Processes is considered
- Gauss-Seidel Value Iteration is investigated, i.e. the value of the updated state is available immediately
- Standard Bellman backup
  $\mathbf{V^{i+1}(s) = max_a \left\{ Q^{i+1}(s,a) = R(s,a) + \gamma \sum_{s'} P(s,a,s') V^{i}(s') \right\}}$
- *Definition 1:* $\mathbf{Q}$ *is pessimistic if* $\mathbf{Q(x,a) \leq Q^*(x,a)}$ *and optimistic if* $\mathbf{Q(x,a) \geq Q^*(x,a)}$.
- *Definition 2:* $\mathbf{Q}$ *is monotone pessimistic if* $\mathbf{Q(x,a) \leq R_x(a) + \gamma \sum_{x'} T_{x,a}(x') V(x')}$ *and is monotone optimistic if* $\mathbf{Q(x,a) \geq R_x(a) + \gamma \sum_{x'} T_{x,a}(x') V(x')}$ *for all* $\mathbf{x}$ *and* $\mathbf{a}$*, where* $\mathbf{V(x) = max_a Q(x,a)}$.

Backward Value Iteration: the Algorithm

Repeat:

- Start backward breath-first search from the goal state
- Visit states once in every iteration
- When a new state is visited, backup occurs, and only its policy predecessors are added to the open list

Huge savings because the order is good to propagate values from the goal state, and traversal in reduced to policy predecessors only. Introduced in: *P. Dai and E. A. Hansen. Prioritizing Bellman backups without a priority queue. In Proc. of ICAPS, 2007.*

Backward Value Iteration: Policy Loops

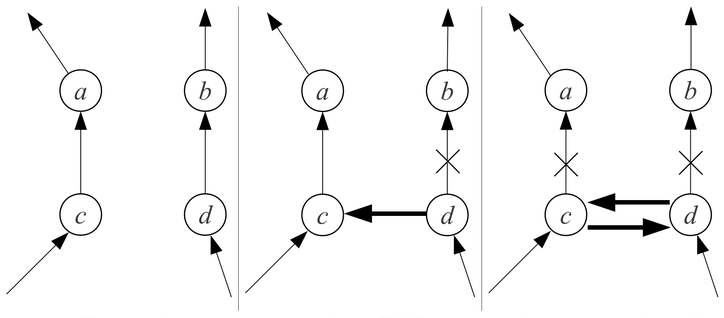

Figure: Example situation when BVI is caught in a policy loop

(1) Backward Value Iteration: Required Initialisation

Theorem: In the backward value iteration algorithm, the policy induced by the current value function is proper (i.e., every state reaches the goal state with probability 1) after every iteration when:

- *the initial value function is monotone pessimistic, i.e., the conditions of Definition 2 are satisfied*
- *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.

Best Actions Only (BAO) Backup

old_val $\leftarrow$ **V(s)**
repeat
 best_actions = all **a** in **A(s)** st. $|\mathbf{Q(s,a) - max_i\, Q(s,i)}| < \epsilon$
 $\delta = \mathbf{0}$
 for each **a** in **best_actions do**
  old_q $= \mathbf{Q(s,a)}$
  $\mathbf{Q(s,a) = R(s,a) + \gamma \sum_{s'} T(s,a,s') max_{a'} Q(s',a')}$
  if $|\mathbf{old\_q - Q(s,a)}| > \delta$ **then**
   $\delta = |\mathbf{old\_q - Q(s,a)}|$
  end if
 end for
until $\delta < \epsilon$
return $|\mathbf{old\_val - V(s)}|$

Introduced in: M. Grześ and J. Hoey. Efficient planning in R-MAX. In Proc. of AAMAS, 2011.

(2) Best Actions Only (BAO) Backup: Requirements

Theorem 2: 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.

Results

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: Results on non-terminating MDPs, uniformly distributed rewards and $\gamma = \mathbf{0.99}$

- BAO yields considerable improvements

(3) Improved Default Order

Nr	Time [ms]	Backups	Algorithm
1	889.8 ± 2.1	1186660.8 ± 2254	VI-V(0)-random
2	862.6 ± 1.8	1183280.0 ± 0	VI-V(0)-BFS
3	648.9 ± 3.9	867175.2 ± 3608	VI-V(Eucl)-random
4	323.8 ± 3.8	422600.0 ± 0	VI-V(Eucl)-BFS
5	**163.1 ± 1.2**	202274.0 ± 0	**VI-V(Eucl)-BAO-BFS**
6	295.1 ± 0.5	345545.0 ± 0	VI-V(Eucl)-BAOnce
7	1493.6 ± 4.0	1529072.0 ± 0	PS-V(-100)
8	211.6 ± 0.5	69729.0 ± 0	MPI(2)-V(Eucl)-BFS
9	200.5 ± 0.5	107763.0 ± 0	MPI(5)-V(Eucl)-BFS
10	228.5 ± 0.3	175379.0 ± 0	MPI(10)-V(Eucl)-BFS
11	405.9 ± 0.8	464437.4 ± 282	MPI(100)-V(Eucl)-BFS
12	869.2 ± 1.4	1289352.6 ± 1035	MPI(500)-V(Eucl)-BFS
13	565.0 ± 1.1	456200.0 ± 0	BVI-V(-100)-SS
14	611.5 ± 0.9	544074.0 ± 0	BVI-V(DS-MPI)
15	819.4 ± 1.4	680856.0 ± 0	BVIPC-V(-100)
16	528.3 ± 2.1	422408.0 ± 0	LBVI-V(Eucl)
17	366.8 ± 2.0	202122.0 ± 0	LBVI-V(Eucl)-BAO

Table: Results on the navigation maze

- Default breath-first ordering outperforms more sophisticated dynamic ordering

Conclusion

- (1) We identified the loop invariant of the main loop of the BVI algorithm and derived the proof and the initial conditions which guarantee that the BVI algorithm will converge
- (2) We proved that updates of only best actions can be applied when initialisation is optimistic
- (3) The default order of states is important and it can be easily improved

Acknowledgements

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.

School of Computer Science - University of Waterloo - Canada e-mail: {mgrzes|jhoey}@cs.uwaterloo.ca www: http://www.cs.uwaterloo.ca/~mgrzes