*Communications and Network*, **2013, 5, 86-89**
doi:10.4236/cn.2013.51B020 Published Online February 2013 (http://www.scirp.org/journal/cn)

# Topological Order Value Iteration Algorithm for Solving Probabilistic Planning[*]

**Xiaofei Liu[1], Mingjie Li[2], Qingxin Nie[3]**
[1]Polytechnic School, SanYa University, Sanya, China
[2]School of Foundation Courses, SanYa University, Sanya, China
[3]Department of Computer Science, DunHua Vocational-technical School, Dunhua,China

Received 2012

## ABSTRACT

AI researchers typically formulated probabilistic planning under uncertainty problems using Markov Decision Processes (MDPs).Value Iteration is an inefficient algorithm for MDPs, because it puts the majority of its effort into backing up the entire state space, which turns out to be unnecessary in many cases. In order to overcome this problem, many approaches have been proposed. Among them, LAO*, LRTDP and HDP are state-of-the-art ones. All of these use reachability analysis and heuristics to avoid some unnecessary backups. However, none of these approaches fully exploit the graphical features of the MDPs or use these features to yield the best backup sequence of the state space. We introduce an improved algorithm named Topological Order Value Iteration (TOVI) that can circumvent the problem of unnecessary backups by detecting the structure of MDPs and backing up states based on topological sequences. The experimental results demonstrate the effectiveness and excellent performance of our algorithm.

**Keywords:** Probabilistic Planning; Markov Decision Processes; Dynamic Programming; Value Iteration

## 1. Introduction

In recent years, intelligent planning has developed into an important branch in artificial intelligence research, especially the uncertainty planning problem has aroused the researcher's more attention. Among a large number of research methods, probabilistic methods can be more accurate to describe the uncertainty information, so it has been widespread concerned in the research, the solving method has been gradually matured. Probabilistic planning uses probability distribution to describe the uncertainty of the initial world state and the effects of actions. In 2004, the IPC-4(2004 International Planning Competition) especially increased the competition in probabilistic planning domains; it has showed that the research of probabilistic planning is very important in the field of intelligent planning study.

Markov decision processes (MDPs) is a model for representing probabilistic planning problems. Value iteration and policy iteration are two fundamental dynamic programming algorithms for solving MDPs [1]. However, these two algorithms are sometimes inefficient. They spend too much time backing up states, often redundantly. Recently several types of algorithms have been proposed to efficiently solve MDPs. The first type uses reachability information and heuristic functions to omit some unnecessary backups, such as RTDP [2], LAO* [3], LRTDP [4] and HDP [5]. The second uses some approximation methods to simplify the problems. The third aggregates groups of states of an MDP by features, represents them as factored MDPs and solves the factored MDPs. Often the factored MDPs are exponentially simpler, but the strategies to solve them are tricky, sLAO* [6], sRTDP [7] are examples. One can use prioritization to decrease the number of inefficient backups. Faster dynamic programming [8] and ranking policies in discrete Markov Decision Processes [9] are two recent examples.

In this paper we propose an improvement of the value iteration algorithm named Topological Order Value Iteration which combines the first and last technique. It decompose a MDP into strong topological order connected components, and then using value iteration algorithm to solve the components in order, so it can prevent the calculation of a large number of useless states and make the available states arranged orderly. It does backups in the best order and only when necessary. Topological Order Value Iteration is itself not a heuristic algorithm, but it can efficiently make use of extant heuristic functions to initialize value functions.

## 2. Background

### 2.1. Markov Decision Processes

[*]Project supported by Sanya City Centre to scientific and technological cooperation Foundation (Grant No 2011YD44).

AI researchers typically use MDPs to formulate probabilistic planning problems. An MDP is defined as a four-tuple<S,A,T,C>, where S is a discrete set of states, A is a finite set of all applicable actions, T is the transition matrix describing the domain dynamics, and C denotes the cost of action transitions. The agent executes its actions in discrete time steps called stages. At each stage, the system is at one distinct states ∈S. The agent can pick any action a from a set of applicable action Ap(s) ⊆A, incurring a cost of C(s, a). The action takes the system to a new state s′ stochastically, with probability Ta (s′|s) .

The horizon of an MDP is the number of stages for which costs are accumulated. There are a set of sink goal states G⊆S, reaching which terminates the execution. To solve the MDP we need to find an optimal policy (S→A), a probabilistic execution plan that reaches a goal state with the minimum expected cost. Any optimal policy must satisfy the following system of Bellman equations, the value function of a policy π is defined as:

$$V^{\pi}(s) = C(s,\pi(s)) + \gamma \sum_{s'\in S} T(s' \mid s) V^{\pi}(s'), \gamma \in [0,1] \quad (1)$$

and the optimal value function is defined as:

$$V^{*}(s) = \min_{a\in A(s)} [C(s,a) + \gamma \sum_{s'\in S} T(s' \mid s) V^{*}(s')], \gamma \in [0,1] \quad (2)$$

## 2.2. Dynamic Programming

Most optimal MDP algorithms are based on dynamic programming. Its usefulness was first proved by a simple yet powerful algorithm named value iteration [10].Value iteration first initializes the value function arbitrarily. Its basic idea is to iteratively update the value functions of every state until they converge. And in each iterm, the value function is updated according to Equation 2. We call one such update a Bellman backup. The Bellman residual of a state s is defined to be the difference between the value functions of s in two consecutive iterations. The Bellman error is defined to be the maximum Bellman residual of the state space. When this Bellman error is less than some threshold value, we conclude that the value functions have converged sufficiently.

The main drawback of the value functions algorithm is that, and in each iterm, the value functions of every state are updated, which is highly unnecessary. Firstly, some states are backed up before their successor states, and often this type of backup is fruitless. Secondly, different states converge with different rates. When only a few states are not converged, we may only need to back up a subset of the state space in the next iteration.

## 3. Topological Order Value Iteration

We have studied the sequence of state backups according to an MDP's graphical structure, which is the intrinsic property of an MDP and potentially decides the complexity of solving it [11]. Our first observation is that states and their value functions are causally related. If in an MDP M, one state s′ is a successor state of s after applying action a, then V (s) is dependent on V (s′). For this reason, we want to back up s′ ahead of s. The causal relation is transitive.

Topological Order Value Iteration solves an MDP problem by using the problem's graphical structure wisely. Given an MDP, TOVI first builds a directed reachability graph Gsr, where G has one vertex per state s ∈ S. A directed edge from vertex s1 to s2 exists if there is an action such that Ta (s2|s1) > 0. TOVI then finds all the strongly connected components of Gsr, and the topological order of the components. Then, it solves every connected component individually, by value iteration, according to their topological order. **Figure 1** shows the graphical representation of one simple MDP that has 7 states and 12 actions. In the figure, successors of probabilistic actions are connected by an arc. For simplicity reason, transition probabilities Ta and costs C(s, a) are omitted. Using TOVI, we can divide the MDP into two connected components C1 and C2. Based on the remaining actions, C1 and C2 can be subdivided into three and two smaller components respectively. By decomposing an MDP into smaller components, TOVI's convergence can be much faster than VI.

We use Kosaraju's algorithm of detecting the topological order of strongly connected components in a directed graph [12]. Note that Bonet and Geffner used Tarjan's algorithm in detection of strongly connected components in a directed graph in their solver [5], but they do not use the topological order of these components to systematically back up each component of an MDP. Kosaraju's algorithm is simple to implement and its time complexity is only linear in the number of states, so when the state space is large, the overhead in ordering the state backup sequence is acceptable. Our experimental results also demonstrate that the overhead is well compensated by the computational gain.

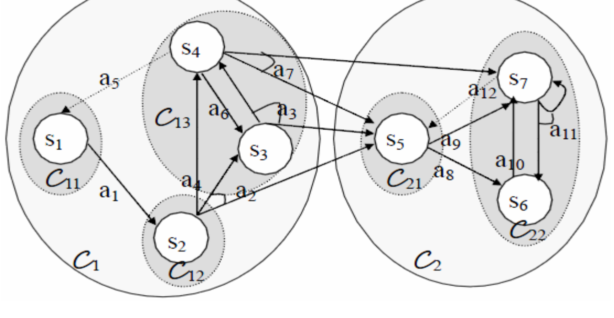

**Figure 1. A simplified MDP and its set of strongly connected components.**

The pseudo code of TOVI is shown in Algorithm 1. We first use Kosaraju's algorithm to find the set of strongly connected components C in graph Gsr, and their sequential order. Note that each c ∈ C maps to a set of states in M. We then use value iteration to solve each c. Since there are no cycles in those components, we only need to solve them once.

```
Algorithm 1 Topological Order Value Iteration

1: Input:an MDP <S,A,T,C>
   VI(S: a set of states, δ)
2: while (true)
3: for each state s ∈ S
4: $V(s)=\min_{a\in Ap(s)}[C(s,a)+\gamma\sum_{s'\in S}T(s'|s)V(s')]$
5: if (Bellman error is less than δ)
6: return
   Scc(MDP M)(Kosaraju's algorithm)
7: build the graph Gsr
8: compute the strongly connected components of
   Gsr,order them by topological order C1,...,Ck
9: for c←1 to k do
10: solve component Cc by value iteration
Search(s)
11: if $s \notin G$ then mark s as visited
12: $a \leftarrow \arg\min_a Q(s,a)$
13: for every unvisited successor s′of action a do
14: Search(s′)
15: Back-up(s)
16: for each action a do
17: $Q(s,a)\leftarrow C(a,s)+\gamma\sum_{s'\in S}T(s'|s)V_l(s')$
18: if Q(s,a)>Vu(s) then eliminate a from Ap(s)
19: $V_l(s)\leftarrow \min_{a\in Ap(s)} Q(s,a)$
20: $V_u(s)\leftarrow\min_{a\in Ap(s)}[C(s,a)+\sum_{s'\in S}T_{a'}(s'|s)V_u(s')]$
```

## 4. Experiment

We tested the Topological Order Value Iteration and compared its running time against value iteration (VI), LAO*, and LRTDP. All the algorithms are coded in C and properly optimized, and run on the same Intel Core2 Duo CPU E7400 2.80GHz processor with 4G main memory. The operating system is Linux version 2.6.15 and the compiler is gcc version 3.3.4.

We use seven MDP test domains for our experiments. They are Mountain Car, Single Arm Pendulum, Wetfloor2, and three domains from International Planning Competition 2006–Drive, Elevators and TireWorld. The Performance of the different algorithms in various test domains is listed in **Table 1**. All running times are in seconds, fastest times are bolded. BC size means the size of the biggest connected component. "-" means that the algorithm failed to solve the problem within 5 minutes.

The experimental results has showed TOVI algorithm is better than the other three algorithms on most of domains, it can fast convergence due to only update the appropriate path to calculate the sequence to avoid a large number of useless state calculations. However, in Single Arm Pedulum and TireWorld test domains, TOVI algorithm grouped the state diagram and ordered each connected component has speeded more time in the overall running time, so its performance less than the LAO * algorithm and LRTDP.

**Table 1. Running time of different algorithms in various domains.**

| Test Domain | VI Time | LAO* Time | LRTDP Time | TOVI | |
|---|---|---|---|---|---|
| | | | | BC size | Time |
| MCar100 | 1.91 | 1.23 | 2.81 | 7,799 | **0.68** |
| MCar300 | 229.70 | 93.91 | 117.23 | 71,751 | **23.22** |
| MCar700 | - | 216.01 | - | 390,191 | **233.98** |
| SAP100 | 9.39 | **1.81** | 2.58 | 9,999 | 2.37 |
| SAP300 | - | **32.4** | - | 89,999 | 44.2 |
| SAP500 | - | **130.7** | - | - | - |
| WF200 | - | 11.22 | 22.08 | 39,999 | **10.58** |
| WF400 | - | 98.97 | 97.73 | 159,999 | **90.78** |
| DAP10 | 51.45 | 3.04 | 1.02 | 9,454 | **0.75** |
| DAP20 | - | 144.12 | 32.68 | 150,489 | **21.95** |
| Drive | - | - | - | 75,840 | **74.70** |
| Ele(p13) | - | 227.53 | - | 1053 | **58.46** |
| Ele(p15) | - | 27.53 | - | 1053 | **14.59** |
| *TireW(5)* | 0.14 | **0.01** | 1.26 | 23 | 2.26 |
| *TireW(6)* | 0.16 | **0.01** | 1.44 | 618,448 | 48.81 |

## 5. Conclusions

We have introduced and analyzed a probabilistic planning MDP solver, Topological Order Value Iteration that studies the dependence relation of the value functions of the state space and use the dependence relation to decide the sequence to back up states. The algorithm is based on the idea that different MDPs have different graphical structures, and the graphical structure of an MDP intrinsically determines the complexity of solving that MDP. We notice that no current solvers detect this information and use it to guide state backups. Thus, they solve MDPs of the same problem sizes but with different graphical structure with almost the same strategies. In this sense, they are not "intelligent". Topological Order Value Iteration is proposed to solve this problem. It is guaranteed to find the optimal solution of a Markov decision process sequentially.

Topological Order Value Iteration also is a flexible algorithm, which can use the initial state information and apply reachability analysis. Our results have shown that TOVI is extremely useful in MDPs with many connected components. The complexity increase of TOVI is not as great as other algorithms as the number of layers increase, which shows that TOVI is very suitable for solving MDPs with layered structures.

## REFERENCES

[1] S. Y. Yan, M. H. Yin, W. X. Gu and X. F. Liu, “Research and Advances on Probabilistic Planning,” *Caai Transactions on Intelligent Systems,* Vol. 1, 2008, pp. 9-22.

[2] A. Barto, S. Bradke and S. Singh, “Learning to Act using Real-time Dynamic Programming,” *Artificial Intelligence*, Vol. 72, 1995, pp. 81-138. doi:10.1016/0004-3702(94)00011-O

[3] E. Hansen and S. Zilberstein, “LAO*: A Heuristic Search Algorithm that Finds Solutions Withloops,” *Artificial Intelligence*, Vol. 129, 2001, pp. 35-62. doi:10.1016/S0004-3702(01)00106-0

[4] B. Bonet and H. Geffner, “Labeled RTDP: Improving the Convergence of Real-time Dynamic Programming,” *Proceedings of 13th ICAPS,* 2003, pp. 12-21.

[5] B. Bonet and H. Geffner, “Faster Heuristic Search Algorithms for Planning with Uncertainty and Full Feedback,” *Proceedings of IJCAI-03,* 2003, pp. 1233-1238.

[6] C. Guestrin, D. Koller, R. Parr and S. Venkataraman, “Efficient Solution Algorithms for Factored MDPs,” *Journal of Artificial Intelligence Research,* Vol. 19, 2003, pp. 399-468.

[7] Z. Feng and E. Hansen, “Symbolic Heuristic Search for Factored Markov Decision Processes,” *In Proceedings of AAAI-05*, 2002, pp. 44-50.

[8] P. Dai, Mausam and S. Daniel, “Focused Value Iteration,” *The Nineteenth International Conference on Automated Planning and Scheduling (ICAPS-09)*, 2009, pp. 82-89.

[9] P. Dai and J. Goldsmith, “Ranking Policies in Discrete Markov Decision Processes,” *Annals of Mathematics and Artificial Intelligence,* Vol. 59, 2010, pp. 107-123. doi:10.1007/s10472-010-9216-8

[10] M. Pterman and Markov, “Decision Processes: Discrete Stochastic Dynamic Programming,” Wiley-Interscience, 2005.

[11] M. Littman, T. Dean and P. Kaelbling, “On the Complexity of Solving Markov Decision Problems,” *In Proceedings of UAI-95*, 1995, pp. 394-402.

[12] H. Cormen, C. Leiserson and R. Rivest, “Introduction to Algorithms,” Second Edition, The MIT Press, 2001.