Incremental Pruning: A Simple, Fast, Exact Method for Partially Observable Markov Decision Processes

Written by Anthony Cassandra, Michael L.Littman, Nevin L.Zhang
Presented by Costas Djouvas

Since the paper assumes the knowledge of POMDPs (Partially Observable Markov Decision Processes), before presenting the paper, the speaker made a brief introduction to POMDPs. He began by explaining what an MDP is and then moved on to POMDPs:

The Markov Decision Process (MDP) is a discrete model for decision making under uncertainty. It consists of four components:

1. *States:* the world is divided into states
2. *Actions:* each state has a set of actions associated with it. These actions can be performed by an agent who is in this state
3. *Transition function:* s x $a \rightarrow s'$x r(s:state, a:action, s':next state, r:reward)
4. *Reward function :* expected reward for an agent who takes a particular action in a particular state

Policy maps a state to an action. An MDP model is used to find an optimal policy using value iteration algorithm.

1 Introduction

The difference between an MDP and a POMDP is that with POMDPs, the agent does not know its current state. It just knows the information state x (belief state), which is a probability distribution over possible states. Due to this difference, two new components need to be defined:

- Observations: set of observations
- Observation function: relationship between the state, actions and the observation Due to these additional components, the existing components(transition function, for example) will need to be modified

During explanation of POMDPs by the speaker, Dr.Parsons added that POMDPs are really MDPs, where each state is an MDP itself. At that point, one of the students asked if we can go deeper and deeper, making now the states of MDP'(state of an MDP) an MDP again. Dr.Parsons' answer was as follows: We can only go 1 level deep. At the next level, the structure collapses and we get an original MDP.

Solving a POMDP means finding a policy that maps each information (belief) state into an action so that the expected sum of discounted rewards is maximized.

Algorithms for solving POMDPs use a form of dynamic programming, called dynamic programming updates.

Some of the proposed algorithms include 'one pass', 'exhaustive', 'linear support' and 'witness'. It was shown that:

1. 'linear support' is more efficient than 'one pass'
2. 'witness' is faster than both 'exhaustive' and 'linear support'

This paper shows that although 'witness' algorithm is fast, incremental-pruning-based algorithms outperform it (time-wise) as the problem size increases.

2 DP Updates

The main idea of the dynamic programming update is to define a new value function V' in terms of a given value function V. Using value iteration, over the infinite horizon, V' represents an approximation that is very close to the optimal value function. S sets are characterized using purge, where purge takes a set of vectors and reduces the set to its unique minimum form.

3 Purging sets of vectors

In this section, the implementation of the FILTER function is provided. Given a set of vectors as an input, it returns those that have non-empty witness regions, thus pruning out the unnecessary vectors.

4 Using Purge in DP

Given the FILTER procedure, S_a^z sets can be computed from S, S' from S_a and S_a from S_a^z.

5 Complexity Analysis

The running time of the algorithms discussed in this paper is expressed in terms of the number of linear programs they solve and the size of these linear programs. This metric is used for the following reasons:

1. All the presented algorithms use linear programming as a fundamental subroutine and the solution to these linear programs is the most consuming part of the algorithm

2. The traditional "operation count" analysis are cumbersome and unenlightening

6 Incremental Pruning

This method computes S_a from S_a^z sets efficiently using purge. It is conceptually simpler than 'witness' and can be implemented to exhibit superior performance and asymptotic complexity.

There are variations of incremental pruning algorithm. The variations arise when one of the procedures that is being called from this algorithm (Filter) is implemented differently. The worst-case complexity of one of the inferior versions of the Incremental Pruning method (IP) is identical to that of a non-incremental pruning algorithm ('Witness') The best case total number of constraints for IP is asymptotically better than for 'Witness'.

7 Generalizing IP

Incremental pruning algorithm makes calls to Filter function. It was observed that the vectors passed to this function have a great deal of regularity. This observation is exploited by modifying Filter and thereby giving rise to a variety of incremental pruning algorithms.

8 Empirical Results

'IP' (incremental pruning), 'RR'(restricted region: improved version of IP), 'Exhaustive', 'Linear Support' and 'Witness' algorithms were ran on 9 different test problems. Time limit was set to 8 hours. 'Linear Support' could not solve any of the problems due to memory limitations 'Exhaustive' only solved 3/9 problems within 8 hours, but outperformed all other methods for the problems it solved. 'RR' outperformed 'Witness' in all the cases (in most cases by a factor of 5) For two problems (Network and Shuttle), both incremental pruning based algorithms ('IP' and 'RR') finished after about 1 hour whereas 'Witness' did not finish after 8 hours.

9 Discussion and Conclusions

This paper examined the incremental pruning method for performing dynamic-programming updates in partially observable Markov decision processes. The following are the advantages of the incremental pruning method over other algorithms:
- easy to implement

- asymptotic performance is as good as or better than the most efficient of the previous algorithms
- empirically the fastest algorithm of its kind for solving a variety of standard POMDP problems

10 Future works

1. Numerical precision: Develop an algorithm with a tunable precision parameter

2. Develop better best-case and worst-case analysis for RR

11 Comments

This paper appeared to be the most complex out of all presented previously. The speaker did a good job at presenting the paper as well as introducing the audience to some concepts, not covered in the paper, but necessary for understanding it.

Sumon(the discusser) noted that the authors of the paper made a good comparison between the algorithm they say is good and the other competing algorithms.

At the end, Dr.Parsons stated that the difficulty of the paper was due to his choosing it. Therefore, the lesson learned (without using any complex reinforcement learning procedure): Choose you own paper!!!