Academia Mexicana de la Ciencia de Sistemas

Universidad de Guanajuato

Memoria Técnica

V Congreso Internacional de la Ciencia de Sistemas

"Conocimiento Sistemico para el desarrollo"

Diseño Gráfico: *V. H. Jiménez Arredondo.*

Editores: A. Lara-López

M. A. Ibarra-Manzano

O. G. Ibarra-Manzano

ISBN: 978-607-441-263-5

Yuriria, Gto., México

3 y 4 de octubre de 2013

Contenido

Título:

Solving Discounted Markov Decision Processes by Using Prioritized Approach

Autores:

1) Ma. de Guadalupe García-Hernández
2) José Ruiz-Pinales
3) J. G. Aviña-Cervantes
4) Andrea Huerta-Fuerte
5) Horacio Rostro-González
6) Sergio Ledesma-Orozco
7) Rafael Guzmán-Cabrera

Afiliación:

Universidad de Guanajuato

División de Ingenierías Campus Irapuato-Salamanca

Dirección:

Carretera Salamanca-Valle de Santiago Km 3.5+1.8 Km

Comunidad de Palo Blanco, C.P. 36885 Salamanca, Gto.

Email:

1) garciag@ugto.mx

2) ruizpinales@hotmail.com

3) avina@ugto.mx

Tel y Fax:

Tel. +52 (464) 64 7 99 40, FAX 2311

Solving Discounted Markov Decision Processes by Using Prioritized Approach

Solución de Procesos de Decisión de Markov Descontados Usando Enfoque Priorizado

Abstract. The problem of solving large Markov decision processes accurately and quickly is a challenging one. For the special case of additive Markov decision processes fast solution algorithms exist. For instance, improved prioritized value iteration and improved prioritized sweeping are two fast algorithms for the solution of additive Markov decision processes. Here, the possibility of transforming discounted Markov decision processes into additive Markov decision processes for solving them by means of fast algorithms such as improved prioritized value iteration and improved prioritized sweeping is explored.

Resumen. El problema de la solución de procesos de decisión de Markov de gran talla de forma exacta y rápida es un gran reto. Para el caso especial de procesos de decisión de Markov aditivos ya existen algoritmos de solución rápidos. Por ejemplo, iteración de valor priorizado mejorado y barrido priorizado mejorado son dos algoritmos rápidos para resolver procesos de decisión de Markov aditivos. Aquí, se explora la posibilidad de transformar cualquier proceso de decisión de Markov descontado en procesos de decisión de Markov aditivos para que puedan resolverse por medio de algoritmos rápidos tales que iteración de valor priorizado mejorado y barrido priorizado mejorado.

Keywords. Discounted Markov Decision Processes, Additive Markov Decision Processes, Prioritization, transformation of Markov Decision Processes.

1. Introduction

In planning under uncertainty, the planner's objective is to find a policy that optimizes some expected utility. Most approaches for finding such policies are based on decision-theoretic planning (Boutilier, 1999) (Bellman, 1954) (Puterman, 1994). Among these, Markov decision processes (MDP) constitute a mathematical framework for modeling and deriving optimal policies. Value iteration is a dynamic programming algorithm (Bellman, 1957) for solving MDP, but it is usually not considered because of its slow convergence (Littman, 1995). This is because its speed of convergence depends strongly on the order of the computations (or backups).

The slow convergence of value iteration for solving large MDP is usually tackled up by using one of two approaches (Dai, 2007a): heuristic search (Hansen, 2001) (Bhuma, 2003) (Bonet, 2003a,b, 2006), or prioritization (Moore, 1993) (Ferguson, 2004) (Dai, 2007b) (Wingate, 2005). In the first case, heuristic search combined with dynamic programming are used to reduce the number of relevant states as well as the number of expansions of the search. Hansen *et al*. (Hansen, 2001) considered only part of the state space by constructing a partial solution graph, searching implicitly from the initial state towards the goal state, and expanding the most promising branch of an MDP according to a heuristic function. Bhuma *et al*. (Bhuma, 2003) extended this approach by using a bidirectional heuristic search algorithm. Bonet *et al.* (Bonet, 2003a,b) proposed two other heuristic algorithms that use a clever labeling technique to mark irrelevant states. Later on, they explored depth-first search for the solution of MDP (Bonet, 2006). In the second case, prioritization methods are based on the observation that, in each iteration, the value function usually changes only for a reduced set of states. So, they prioritize each backup in order to reduce the number of evaluations (Moore, 1993) (Dai, 2007b). Ferguson *et al.* (Ferguson, 2004) proposed another prioritization method called focused dynamic programming, where priorities are calculated in a different way than in prioritized sweeping. Dai *et al.* (Dai, 2007a) extended Bhuma *et al.*'s idea (Bhuma, 2003) by using concurrently different starting points. In addition, they also proposed (Dai, 2007b) a topological value iteration algorithm, which groups states that are mutually and causally related together in a meta-state for the case of strongly connected states (or MDP with cyclic graphs). Likewise, other approaches such as topological sorting (Wingate, 2005) and shortest path methods (McMahan, 2005a,b) have been proposed. On the first hand, topological sorting algorithms can be used to find good backup orderings but their computational cost is usually high (Wingate, 2005). On the other hand, shortest path methods have been applied to the solution of MDP with some success (McMahan, 2005a,b).

In this paper, the problem of finding an optimal policy of discounted MDPs accurately and quickly is considered. It has been found that any MDP can be transformed into a goal-based MDP and viceversa (Barry, 2009). This opens up the possibility of solving discounted MDPs by using the fastest existing algorithms for solving additive MDPs. Thus, different equivalences of MDPs in terms of additive MDPs that can be solved by using fast algorithms based on prioritization are considered.

This paper is organized as follows: first a brief introduction to MDPs as well as solution methods is presented, then, different equivalences of MDPs are studied and finally experimental results and conclusions are presented.

I.1 Markov Decision Processes

Markov decision processes (MDP) provide a mathematical framework for modeling sequential decision problems in uncertain dynamic environments (Bellman, 1957) (Puterman, 2005).

Formally, an MDP is a four-tuple (S, A, P, R), where S is a finite set of states, A is a finite set of actions, $P: S, A, S \mapsto \mathbb{R}$ is the transition probability function, and $R: S, A \mapsto \mathbb{R}$ is the reward function. A policy (or strategy) $\pi(s)$ is a rule that specifies which action should be taken in each state. The core problem of MDP is to find the optimal policy that maximizes the expected total (or average) reward (Puterman, 2005). The value function is the expected reward, starting at state s and following policy π, that is given by:

$$V(s) = E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t))\right]$$

where $\gamma \in [0,1]$ is the discount factor, which may be used for decreasing exponentially future rewards implying that future rewards have less value than current rewards (Russell, 2004). For the case of additive MDP ($\gamma = 1$) the expected total reward may be infinite.

Thus, absorbing terminal states are necessary in additive MDP to guarantee that the expected total reward is finite.

Let $V^*(s)$ be the optimal value function of any state given by:

$$V^*(s) = \max_{\pi} V^{\pi}(s)$$

The optimal value function satisfies the Bellman equation (Bellman, 1954) (Puterman, 2005) that is given by:

$$V^*(s) = \max_{a} \left[R(s,a) + \gamma \sum_{s' \in S} P(s,a,s') V^*(s') \right]$$

Value iteration, policy iteration and linear programming are three of the most well-known techniques for finding the optimal value function $V^*(s)$ and the optimal policy $\pi^*(s)$ for infinite horizon problems (Chang, 2007). However, policy iteration and linear programming are computationally expensive techniques when dealing with problems with large state spaces because they both require solving a linear system (of equations) of the same size as the state space. In contrast, value iteration avoids this problem by using a recursive approach that it is typically used in dynamic programming (Chang, 2007).

Starting from an initial value function, value iteration applies successive updates to the value function for each $s \in S$ by using:

$$V^t(s) = \max_{a} \left[R(s,a) + \gamma \sum_{s' \in S} P(s,a,s') V^{t-1}(s') \right]$$

Let $\{V_n | n = 0,1,\ldots\}$ be the sequence of value functions obtained by value iteration. Then, it can be shown that every value function obtained by value iteration satisfies $|V_n - V^*| \leq \gamma^n |V_0 - V^*|$. Thus, from the Banach fixed-point theorem, it can be inferred that value iteration converges to the optimal value function $V^*(s)$. One advantage of value iteration comes from the fact that the value functions obtained can be used as bounds for the optimal value function (Tijms, 2003).

The convergence of value iteration may be quite slow for γ close to one. For this reason, several improvements to value iteration have been proposed (Puterman, 2005). For instance, common techniques may improve convergence rate, reduce the time taken per iteration and/or use better stopping criteria.

One of the easiest ways to improve convergence rate is to update the value functions as soon as they become available (also known as asynchronous updates). For instance, Gauss-Seidel value iteration uses the following update equation (Puterman, 2005):

$$V^t(s) = \max_a \left[R(s,a) + \gamma \sum_{s'<s} P(s,a,s')V^t(s') + \gamma \sum_{s' \geq s} P(s,a,s')V^{t-1}(s') \right]$$

It is well known that policy iteration converges faster than value iteration does, but it is more expensive per. A combined approach (modified policy iteration) can exploit the advantages of both. Thus, modified policy iteration uses a partial policy evaluation step based on value iteration (Puterman, 2005).

Another way of improving the convergence rate as well as the iteration time is using prioritization and partitioning (Wingate, 2005). Generally, prioritization methods are based on the observation that, at each iteration, the value function usually changes only for a reduced set of states. Thus, by restricting the computation to only those states, a reduction of the iteration time is expected. It has been outlined that for acyclic problems the ordering of the states, where the transition matrix becomes triangular may result in a significant reduction in time (Wingate, 2005).

II. Materials and Methods

II.1 Prioritized Sweeping

Although value iteration is a powerful algorithm for solving MDP, it has some potential problems. First, some backups are useless because not all states change in a given iteration (Dai, 2007b). Second, backups are not performed in an optimal order. Priority-based methods such as prioritized sweeping (PS) (Moore, 1993) avoid these problems by ordering and performing backups so as to perform the least number of backups (Dai, 2007b). To be more precise, PS maintains a priority queue for ordering backups intelligently. This priority queue is updated as the algorithm sweeps through the state space. PS can begin by inserting the goal state in the priority queue when it is used in an offline dynamic programming algorithm, such as value iteration. At each step, PS pops a state s from the queue with the highest priority and performs a Bellman backup of that state. If the Bellman residual of state s is greater than some threshold value ε or if s the goal state, then PS is inserts its predecessors into the queue according to their priority (Dai, 2007b). Unfortunately, the use of a priority queue for all the states of the model may result in an excessive overhead for real-world problems (Wingate, 2005), especially for cyclic MDP.

Focused dynamic programming (Ferguson, 2004) is another variant of prioritized sweeping that exploits the knowledge of the start state to focus its computation on states that are reachable from that state. To do this, focused dynamic programming uses a priority metric that it is defined using two heuristic functions: an admissible estimate of the expected cost for reaching the current state from the start state and an estimate of the expected cost for reaching the goal state from the current state. In contrast to other forms of prioritized sweeping, this approach removes the state with the lowest priority value from the priority queue, instead of removing the state with the highest priority value, since it is interested in states through which the shortest path passes.

Dibangoye *et al.* (Dibangoye, 2008) proposed an improved topological value iteration algorithm (iTVI) which uses a static backup order. Instead of minimizing the number of backups per iteration or eliminating useless updates, this algorithm attempts to minimize the number of iterations by using a good backup order. First, depth-first-search is used to collect all reachable states from the start state. Next, breadth-first-search is used to build a metric $d(s)$, which is defined as the distance from the start state to state s. A static backup order is built from the resulting metric in such a way that states that are closer to the start state be updated first. The algorithm is guaranteed to converge to the optimal value function because it updates all states recursively in the same way as value iteration does.

Meuleau *et al*. (Meuleau, 2006) solved stochastic over-subscription planning problems (SOSP) by means of a two-level hierarchical model. They exploit this hierarchy by solving a number of smaller factored MDP. Shani *et al*. (Shani, 2008) extended the use prioritization to partially observable MDP. In this case, backups are prioritized by using the Bellman error as a priority metric and no priority queue is used.

In contrast with the above methods, it is worth to mention a prioritization method that does not require a priority queue (Dai, 2007c), instead, it uses a FIFO (first input, first output) queue if the backwards traversal of the policy graph is breadth-first (forwards value iteration), or a LIFO (last input, first output) queue if the backwards traversal is depth-first (backwards value iteration). In both cases, unnecessary backups can be avoided by using a labeling technique (Bonet, 2003a,b) and the decomposition of the state space into a number of strongly connected components. Unfortunately, it has been shown that the backup order induced by these algorithms is not optimal (Dai, 2007c).

Since the performance of PS depends on the priority metric that it is used to order states in the priority queue, several researchers have investigated alternative priority metrics. For instance, IPS (McMahan, 2005a,b) uses a combination of a value change metric, and an upper bound metric. In fact, it has been shown that IPS may outperform other prioritized sweeping algorithms (Dai, 2007a,b).

Dijkstra's algorithm is an efficient greedy algorithm for solving the single-source shortest path problem in a weighted acyclic graph. This algorithm is a special case of the A* algorithm but unlike the A* algorithm, Dijkstra's algorithm is not goal oriented. This is because Dijkstra's algorithm computes all shortest paths from a single source node to all nodes and thus solves the one-to-all shortest path problem. In fact it has been shown that Dijkstra's algorithm is a successive approximation method to solve the dynamic programming equation for the shortest path problem (Sniedovich, 2006, 2010) and therefore it is based on the Bellman's optimality principle. One interesting feature of Dijkstra's algorithm is that it processes states according to a greedy best first rule.

One of the advantages of value iteration and its variants is that their convergence to the optimal value function is guaranteed for the case of discounted MDP and for the case of additive MDP with absorbing states (Bertsekas, 1995) (Li, 2009). This is because successive application of the Bellman equation has guaranteed convergence to the optimal value function.

II.2 Proposed Method

As we have already mentioned, any discounted MDP can be transformed into an additive MDP (Barry, 2009). As a matter of fact, elimination of the discount factor γ from the Bellman equation is an equivalence preserving transformation.

Given an MDP (M) with discount factor γ and no absorbing states, it can be transformed into an additive MDP (M') by using the following procedure:

- Add to the set of states S_M of M a single absorbing state s_g with reward 0. Let $S_{M'}$ be the set of states of the transformed MDP.
- For each $s, s' \in S_M$, set $P_{M'}(s,a,s') = \gamma P_M(s,a,s')$. Set $P_{M'}(s,a,s') = 1-\gamma$.

Other kinds of equivalence preserving transformations are possible. For instance, an MDP can be transformed by adding a constant to the reward function. In this way, for instance, it is possible to transform any positive MDP into a negative MDP and *vice versa*.

Given a discounted MDP D with reward function $R(s,a)$, and $D+C$ be the MDP resulting from adding the constant C to the reward function. Then

$$V_D^\pi(s) = R(s,\pi(s)) + \gamma \sum_{s'\in S} P(s,\pi(s),s') V_D^\pi(s')$$

$$V_D^\pi(s) + \frac{C}{1-\gamma} = R(s,\pi(s)) + \frac{C}{1-\gamma} + \gamma \sum_{s'\in S} P(s,\pi(s),s') V_D^\pi(s')$$

$$V_D^\pi(s) + \frac{C}{1-\gamma} = R(s,\pi(s)) + C + \gamma \sum_{s'\in S} P(s,\pi(s),s') \left(V_D^\pi(s') + \frac{C}{1-\gamma}\right)$$

Next, by defining $W^\pi(s) = V_D^\pi(s) + \frac{C}{1-\gamma}$, the Bellman equation of the resulting MDP is:

$$W^{\pi}(s) = R(s,\pi(s)) + C + \gamma \sum_{s' \in S} P(s,\pi(s),s')W^{\pi}(s')$$

Thus, the value function of the MDP resulting from adding a constant to the reward function is:

$$V^{\pi}_{D+C}(s) = V^{\pi}_{D}(s) + \frac{C}{1-\gamma}.$$

In the previous approach an MDP is transformed into an additive MDP by adding a single goal state connected to all states. The advantage is that the resulting MDP can be found by using methods such as IPS or IPVI.

The previous approach can be improved by adding more than one goal state. In this case, it is proposed that only certain states must be connected to an absorbing state and that there must be several goal states. The reason is that, for the goal-based MDP obtained by adding a single goal state, most prioritized algorithms such as IPS must update at the beginning all predecessors of the goal state. That is, IPS and IPVI must update at the beginning all the whole set of states S, which incurs in a high computational overhead mainly because of the large number queue operations (insertions and priority changes).

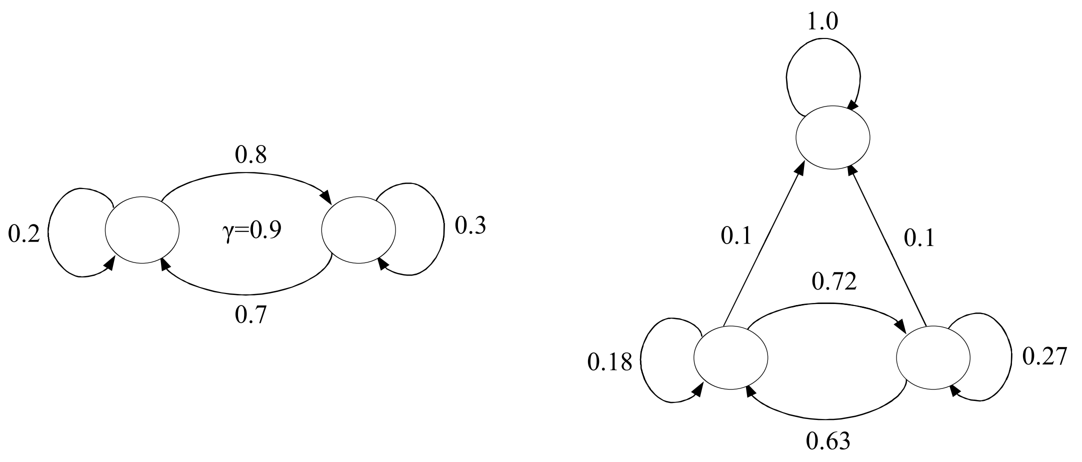

Figure 1. Transformation of a discounted MDP into an additive MDP. Left is original MDP, right is transformed MDP. A new action with transition probability $1-\gamma$ is added to each state and all other transition probabilities are multiplied by the discount factor.

II.3 Experiment Description

For the validation of the proposed algorithm, discounted MDPs with different numbers of states, actions, and reachable states were generated**.** Next**,** each generated MDP was transformed into an additive MDP by adding an absorbing state and connecting it to all states.

All the experiments were performed on workstation with 4GB RAM running Windows 7. All the tested algorithms were implemented using the Java language. The initial and maximum size of the stack of the Java virtual machine was set to 1024 MB and 1536 MB, respectively. For all the experiments, we set $\varepsilon = 10^{-7}$ and $\gamma = 0.9$. Obviously, the transformed MDPs had no discount factor.

III. Results

We tested our approach by solving the additive MDPs resulting from different transformations: transformation by adding a single goal state. In addition, we compared that approach with two methods: asynchronous value iteration with sparse coding and improved prioritized value iteration.

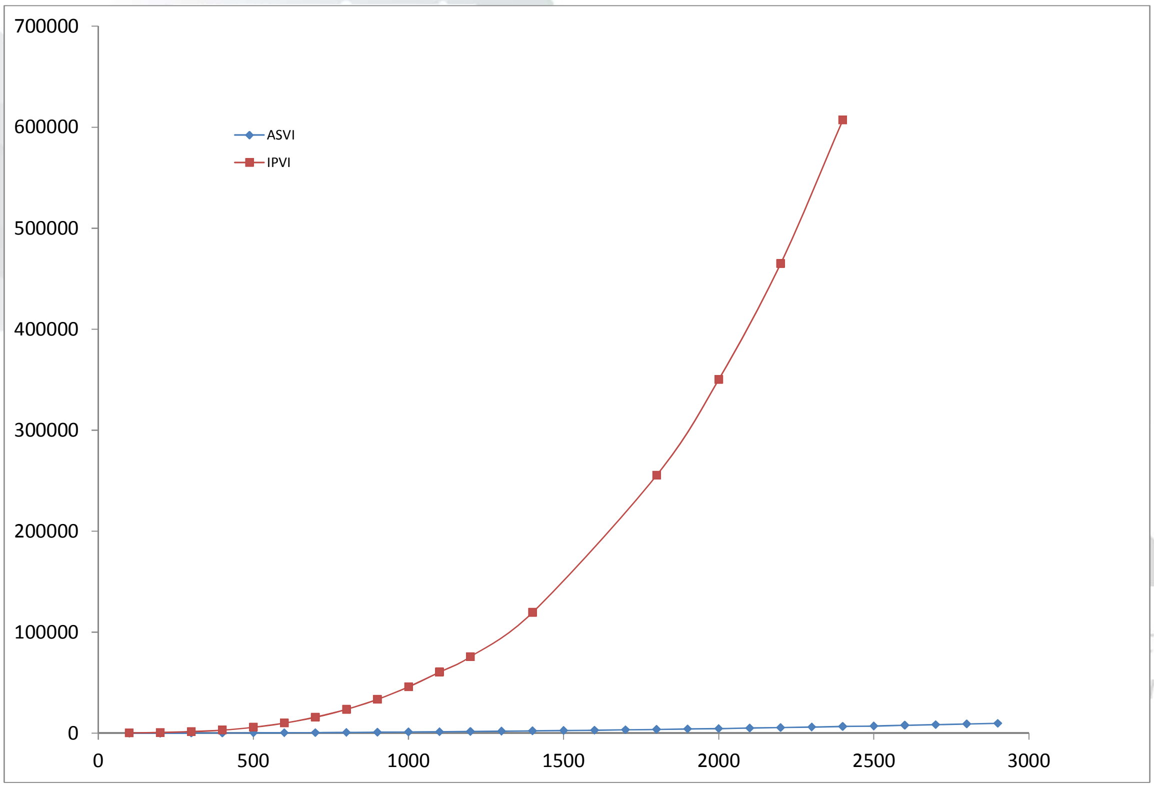

Figure 2. Solution time as a function of the number of states for dense MDPs.

Figure 2 shows the solution time for the case of difficult dense MDPs with 4 actions and different numbers of states. As we had expected, asynchronous value iteration with sparse coding was by far the fastest algorithm.

IV. Discussion

In this paper a new approach for solving discounted MDPs by transforming them to additive MDPs and using fast algorithms such as improved prioritized sweeping and improved prioritized value iteration has been studied and tested.

We compared the performance of the method using asynchronous value iteration with sparse coding and improved prioritized value iteration. For the case of dense MDPs, the performance of asynchronous value iteration with sparse coding with sparse coding was the fastest one. As we had expected, improved prioritized value iteration incurred in a large overhead caused by the large number of queue operations. Further work will focus on the evaluation of the second method proposed which transforms discounted MDPs by using more than one absorbing state.

V. References

Agrawal, S. and Roth, D. (2002). Learning a Sparse Representation for Object Detection. 7th European Conference on Computer Vision, 1-15, Copenhagen, Denmark.