REAL-TIME VS. CLASSIC DYNAMIC PROGRAMMING: AN EMPIRICAL EVALUATION*

Mohammad Pezeshki
Department of Computer Science and Operations Research
Université de Montréal
mohammad.pezeshki@umontreal.ca

ABSTRACT

This technical report outlines common methods of Dynamic Programming in the context of Reinforcement Learning and Planning from an empirical, computer science perspective. In common control and prediction algorithms such as value- and policy- iteration, Dynamic Programming is used extensively in each iteration. Consequently, it is beneficial to take advantage of specific task properties in order to solve the task in a more efficient way. In this report, we specifically compare classic Dynamic Programming against Real-Time Dynamic Programming (RTDP) for the task of value iteration of Bellman Equations.

1 INTRODUCTION

In the context of Planning, we always assume that the environment is known meaning that a Markov Decision Process (MDP) with known rewards and states. Given an MDP, Prediction amounts to evaluation of a given policy. A well-known method of Prediction is *Iterative Policy Evaluation*. That is while Control tries to improve upon policies to find the optimal policy, which *Policy Iteration* and *Value Iteration* are two methods of that. Bellman equations are an essential part of any of these algorithms as summarized in the following table:

Table 1: Planning algorithms and their characteristics. Equations column indicates which Bellman equations are used.

Algorithm	Category	Equations
Iterative Policy Evaluation	Prediction	Bellman Expectation Equation
Policy Iteration	Control	Bellman Expectation Equation + Greedy Policy Improvement
Value Iteration	Control	Bellman Optimality Equation

All of these algorithms take advantage of Dynamic Programming to do the computations in an efficient way. However, depending on the task at hand, there are variants on Dynamic Programming which could be even more efficient. In this report we empirically compare RTDP (Barto et al., 1995) and DP for an artificial task of path finding. In the following sections, we first introduce Synchronous DP and later bridge to RTDP by introducing Gauss-Seidel DP and Asynchronous DP. The experimental results are also presented in Section 3.

2 VARIANTS OF DYNAMIC PROGRAMMING

To introduce different versions of Dynamic Programming, we begin with the algorithm of value iteration. Note that the same procedure applies to other algorithms which have Bellman Expectation Equation. Consider an MDP with a set of n states $S = \{s_1, ..., s_n\}$, a set of m possible actions

*This technical report is presented at Prof. Doina Precup's Reinforcement Learning course at McGill University, Winter 2017.

$A = \{a_1, ..., a_m\}$ and a discount factor of γ. The immediate reward at state s_i with action a is denoted by $R^a_{s_i}$. The transition probability from state s_i to state s_j given action a is also denoted by $P_{s_i s_j}$. The following sections define a step of updating of the Bellman Optimality Equation.

2.1 SYNCHRONOUS DP

In synchronous DP (classical DP), among total K iterations of Value Iteration algorithm, $(k+1)^{th}$ update for $(i)^{th}$ state is defined as follows:

$$v_{k+1}(s_i) = \max_{a \in A} \big(R^a_{s_i} + \gamma \sum_{s_j \in S} P^a_{s_i s_j} v_k(s_j)\big). \tag{1}$$

This update is synchronous as no two states are updated independent of each other. In other words, $v_{k+1}(.)$ is always computed based on $v_k(.)$. Note that in this variant of DP, the final result is independent of ordering of visiting states. In terms of time complexity, given m possible action from each state and n total states, each iteration requires $O(mn^2)$ operations. Theoretically, synchronous DP converges to the optimal value function which corresponds to the optimal policy.

2.2 GAUSS-SEIDEL DP

In Gauss-Seidel DP updates of the value function are done based on the most recent values of other states. In other words, Gauss-Seidel DP "sweeps" between all states and update the values right away. As a result, the model no longer needs to store new values before actually updating states. Thus the update equation is as follows, for all state $s_i \in S$:

$$v_{k+1}(s_i) = \max_{a \in A} \big(R^a_{s_i} + \gamma \sum_{s_j \in S} P^a_{s_i s_j} v(s_j)\big), \tag{2}$$

where,

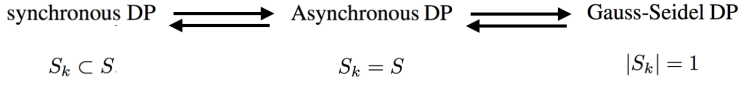
$$v(s_j) = \begin{cases} v_{k+1}(s_j) & \text{if } j < i, \\ v_k(s_j) & \text{else.} \end{cases} \tag{3}$$

2.3 ASYNCHRONOUS DP

Asynchronous DP is similar to Gauss-Seidel DP but the sweeps are not in order. In a multi-processor hardware, asynchronous DP assigns different sets of states to each processor. For example, one processor works on states s_1 to $s_{t'}$ and another processor works on states $s_{t'}$ to $s_{t"}$. For consistency with the previous definitions, we refer to each processor as an iteration processing $S_k \subset S$.

$$v_{k+1}(s_i) = \begin{cases} \max_{a \in A} \big(R^a_{s_i} + \gamma \sum_{s_j \in S} P^a_{s_i s_j} v_k(s_j)\big), & \text{if } s_i \in S_k, \\ v_k(s_i) & \text{else.} \end{cases} \tag{4}$$

Both synchronous DP and Gauss-Seidel DP are special cases of asynchronous DP. If in each iteration k the whole S is processed i.e. $S_k = S$, the algorithm is synchronous DP. If $|S_k| = 1$ and successive states are processed at each iteration, then the algorithm is Gauss-Seidel DP.



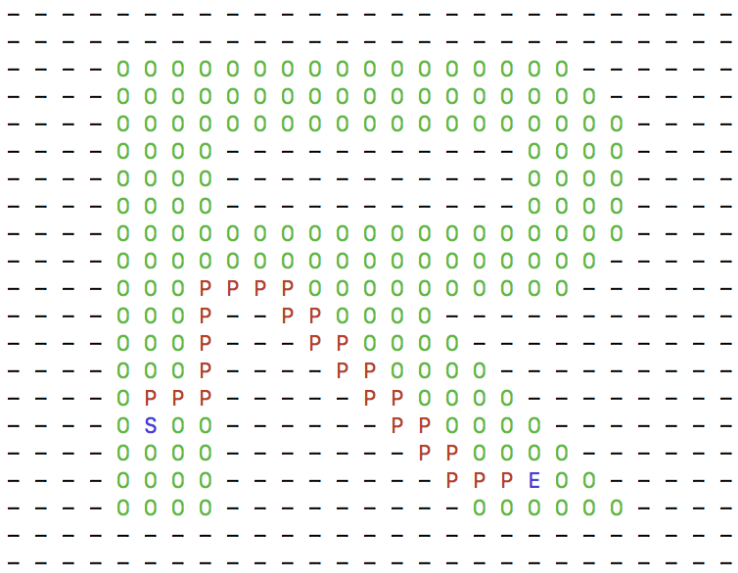
2.4 REAL-TIME DP

All the previous variants of DP we mentioned update the policy after a complete evaluation of the value function. In RTDP, each step of DP affects the policy. In other words, updating the value function and the policy are done concurrently. The procedure is that policy decisions are based on the most recent value function. In the same time, sequences of states, generated by the policy, affect the ordering of DP computations. The RTDP tries to concentrate computations on states which are more probable to happen (to be reached from current state).

In problems in which some state have a reward of zero, RTDP enables the algorithms to find a shortest path from a given state s_i to the goal state without necessarily visiting all the states. Thus the Equation (1) remains the same but only on states in all paths from the starting state.

3 Experimental Results

We define a task of grid-world. The objective is to reach the "E" state starting from the "S" state. Cells with "-" have an immediate reward of -100. The end state has an immediate reward of 0.0. The immediate reward for other states are set to be -1.0. The RTDP algorithm concentrates the computations only on the states which are more probable to be reached from the starting state. The final path found by the algorithm is shown in the following figure, denoted by P.



The code is available online on author's Github.

4 Conclusion

In this report, we briefly surveyed different variants of Dynamic Programming in the context of Planning. In experimental results, we empirically compared classic DP with RTDP on an artificial task. As shown in the previous section, RTDP allocates more of the computations on the states that are more likely to happen. Although both classic DP and RTDP converge asymptotically, in practice, RTDP achieves the optimal policy faster significantly.

References

Barto, Andrew G, Bradtke, Steven J, and Singh, Satinder P. Learning to act using real-time dynamic programming. *Artificial intelligence*, 72(1-2):81–138, 1995.