Determining efficient methods for solving Markov Decision Processes

O. Forghieri[1], E. Hyon[2,3]

[1] École Polytechnique, Palaiseau, France
{orso.forghieri}@polytechnique.edu
[2] LIP6, Université Paris Sorbonne, CNRS, Paris, France
[3] Université de Paris Nanterre, Nanterre, France
{emmanuel.hyon}@parisnanterre.fr

Mots-clés : *Markov Decision Process, Linear Programming, Dynamic Programming, Numerical Solvers*

1 Introduction

We consider here Markovian Decision Processes (MDPs) with a total knowledge of the model i.e. known transitions and rewards functions. There are primarily two effective methods for exactly solving a MDP using model-based approaches : dynamic programming or linear programming, as discussed in [11]. The determination of the most efficient method for solving an MDP problem has been a subject of investigation in the literature ; for a comprehensive review, refer to [1] as well as [11, 9]. On one hand, according to [11], it has been argued that value-based algorithms, such as Value Iteration (VI) and its variants, are not as practical as policy-based approaches and are thus recommended to be avoided. On the other hand, the comparison between Policy Iteration (PI) and Policy Iteration Modified (PIM) for policy-based approaches remains unclear, although the latter appears to be more efficient [11]. Earlier findings indicated that the linear programming approach was not suitable for solving such problems, primarily due to the slow speed of the solvers [9]. Despite this, there have been limited comparative studies in the literature, and as of 2007, the question remained unresolved [10].

With the increasing performance of Linear Programming solvers, such as Gurobi or CPLEX, and advancements in parallelization possibilities, the periodic reevaluation of solving methods becomes pertinent. Consequently, in the study by [1], a comparative analysis of the performance of linear programming and policy iteration was conducted on specific Markov Decision Process (MDP) models, focusing on the expected total reward criteria.

The considered MDPs are characterized by a large state space (with a cardinality of at least 2000) and exhibit a variety of action choices (ranging from 2 to 500). Notably, all transition matrices are highly sparse, containing only 1% and 0.1% of non-zero entries.

Previous studies employs interior point methods to solve the Linear Program. They reveal that Linear Programming (LP) outperforms Policy Iteration (PI) and significantly so for particular models. It is essential to note, however, that the class of models examined by [1] is prevalent in the literature, particularly in network problems where the number of possible transitions in a given state is limited. Nonetheless, the study possesses certain limitations. Firstly, it does not consider policy iteration modified and its variants, even though these approaches may outpace standard PI in terms of speed. Secondly, the LP solving method employed in the study only provides the policy and not both policy and values, as dynamic programming does. Lastly, the generalizability of their conclusions to broader cases with less sparsity or other optimization criteria remains uncertain. The aim of this work is to find out if linear programming is still an effective tool in more general cases, and also to find under which conditions (state space and action space dimension, sparsity) it is still efficient to use dynamic programming.

2 Comparison framework

Solvers For this comparison we only consider numerical solvers to avoid any bias due to programming. Those solvers should also be easily accessible to a modeller. The choice of programming language will therefore be Python. For linear programming, we use `gurobipy` [4] which is one of the most efficient solver whose core is written in C. For stochastic dynamic programming, we consider `MDPtoolbox` [2] which is fully written in python but uses the `numpy` library which is optimized and written in C. We also consider `marmoteMDP` [7] whose code is written in C++ and then encapsulated in python.

Benchmark We divide the testing models in three different classes : random models, well known toy models and real models. Random models are built using randomly generated transition with fixed sparsity within 0.1, 0.4, 0.7 and 1.. The reward matrix is also randomly generated with a density of 1. Among well-known models, we provide *frozenLake* model [3], the *car replacement* model [6] or *four rooms* [5]. Considering real models, we implemented *Queue with Impatience* [8] and *Tandem Queue* [12].

3 Results

We calculate the average runtime of each solver over 10 experiments. VI is consistently 7 times slower than policy-oriented methods. Modified policy iteration is slower than PI in 40% of cases. Contrary to [1], the LP dual isn't always preferable, especially with fewer actions than states. LP outperforms *marmoteMDP* once and *MDPToolbox* 10 times, but is at most 30 times slower than dynamic programming. LP excels in small dimensions with Gurobi efficiency. Future work involves expanding the model range for refining these preliminary findings.

Références

[1] O. Alagoz, M.U.S. Ayvaci, and J.T. Linderoth. Optimally solving markov decision processes with total expected discounted reward function : Linear programming revisited. *Computers and Industrial Engineering*, 87 :311–316, 2015.

[2] I. Chadès, G. Chapron, M.-J. Cros, F. Garcia, and R. Sabbadin. Mdptoolbox : a multi-platform toolbox to solve stochastic dynamic programming problems. *Ecography*, 87 :916–920, 2014.

[3] Brockman et al. Openai gym. *arXiv preprint arXiv :1606.01540*, 2016.

[4] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2023.

[5] Bernhard Hengst. Hierarchical approaches. In *Reinforcement learning*, pages 293–323. Springer, 2012.

[6] Ronald A Howard. Dynamic programming and markov processes. 1960.

[7] E. Hyon and A. Jean-Marie. marmotemdp, 2023.

[8] Emmanuel Hyon and Alain Jean-Marie. Scheduling services in a queuing system with impatience and setup costs. *The Computer Journal*, 55(5) :553–563, 2012.

[9] M. L. Littman, T. L. Dean, and L. P. Kaelbling. On the complexity of solving markov decision problems. In *Eleventh Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 394–402, 1995.

[10] W. Powell. *Approximate dynamic programming : Solving the curses of dimensionality.* Wiley, 2007.

[11] M. Puterman. *Markov Decision Processes Discrete Stochastic Dynamic Programming.* Wiley, 2005.

[12] Aghasaryan Castel-Taleb Tournaire, Jin and Hyon. Factored reinforcement learning for auto-scaling in tandem queues. pages 1–7. IEEE, 2022.