Cache-Efficient Dynamic Programming MDP Solver

Jaël Champagne Gareau, Guillaume Gosset, Éric Beaudry, Vladimir Makarenkov

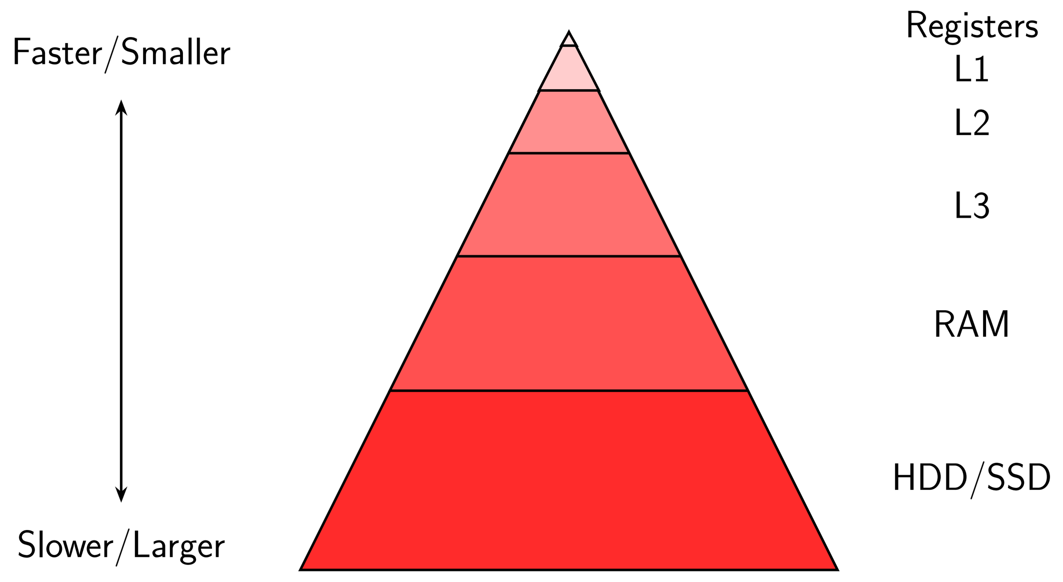
Centre de Recherche en Intelligence Artificielle (CRIA), Département d'Informatique, Université du Québec à Montréal (UQAM)

Introduction

- Markov Decision Processes (MDPs) are used to model problems of decision-making under uncertainties.
- MDPs can be solved with different approaches:
  - Dynamic Programming (e.g., Value Iteration (VI) and Policy Iteration);
  - Heuristic search (e.g., LRTDP and LAO*);
  - Prioritized methods (Prioritized VI (PVI) and Topological VI (TVI)).

Computer Architecture

- One way of improving speed is to consider modern computer architectures:
  - e.g., **Memory hierarchy**, Thread/Data Level Parallelism (SIMD, GPU), etc.
- In Machine Learning (ML), taking these elements into account lead to a speedup of many orders of magnitude.
- In MDP planning, these elements have been much less considered.



Type	Size	Transfer Rate (GB/s)	Latency (ns)
L1	64 Ko/cores	2000	1
L2	1 Mo/cores	1000	3
L3	32 Mo	600	12
DDR5	8-128 Go	50	90
SSD (NVMe)	250 Go-4 To	7	80 000

Cache-Efficient Memory Representation of MDPs

- CSR-MDP is inspired by the *Compressed Sparse Row* repr. of graphs.
- It has minimal wasted memory (no pointers, no memory padding).
- By being packed tightly in memory, we ensure that most memory inside loaded cache lines is useful for the current computation.
- This representation simplifies an SIMD (e.g., SSE, AVX) implementation.
- Most solving algorithms can be used with MDPs stored in CSR-MDP format.

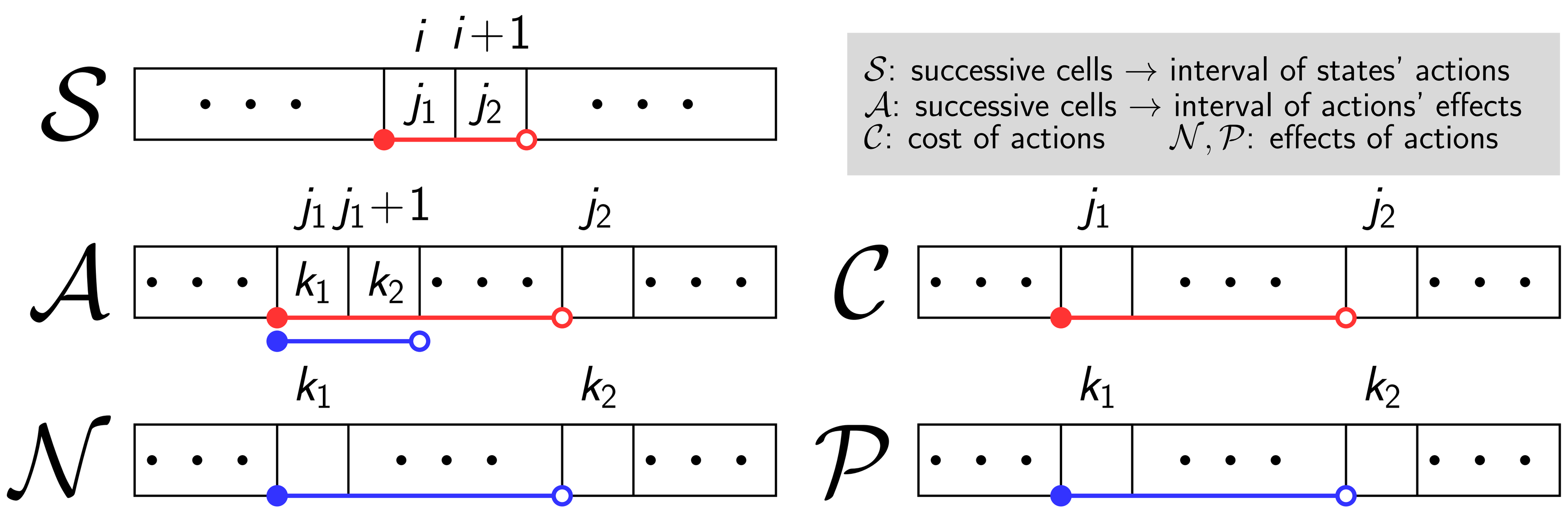
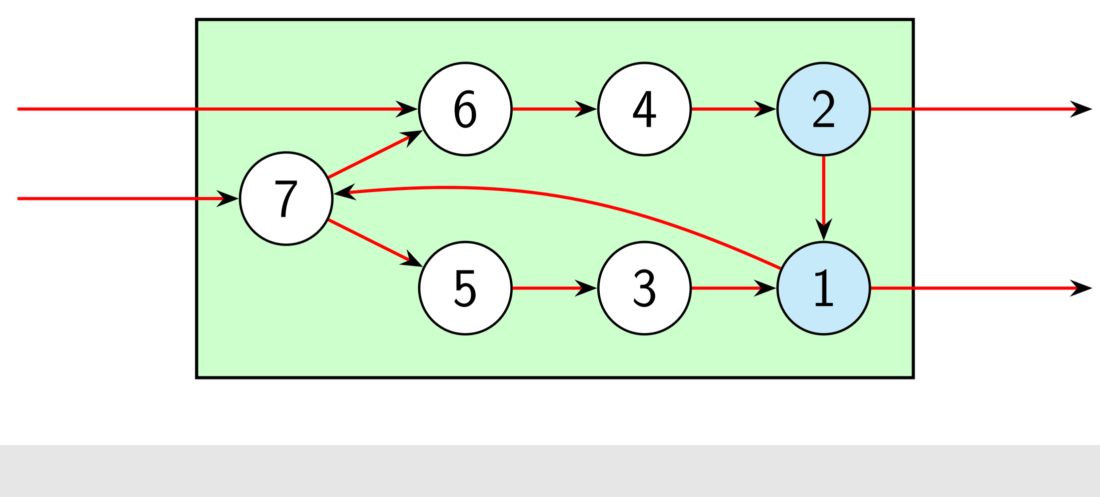


Figure 1: CSR-MDP memory representation scheme

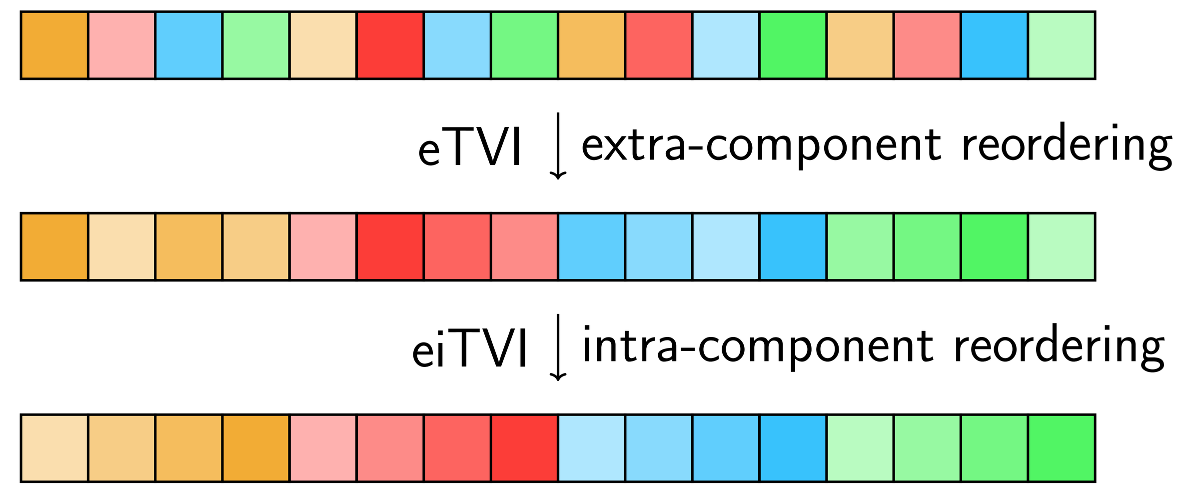
eTVI and eiTVI: Reordering states to improve cache-performance

- Both techniques improve cache performance by matching the order of states in memory and the Bellman sweeps' states consideration order.
- eTVI: Reorders states to make each SCC contiguous in memory.
  - Since TVI solves each SCC one-by-one and only considers each of them once, making each of them contiguous in memory minimizes the number of cache-misses.
- eiTVI: Also reorders states such that the order *within* an SCC match the order of states in the Bellman sweep inside the SCC.
  - Making these orders match increases the amount of useful data in each loaded cache line.
  - We should use an order that maximizes state-values propagation.
  - We propose an order given by a reversed BFS from the outward border states of the SCC.



Example of eTVI/eiTVI

- Assume each state takes 16 bytes and each SCC contains four states.
- With TVI: each SCC is spreaded across four cache lines.
- With eTVI: each SCC is contained in a single cache line.
- With eiTVI: each cache line is read in order.



Results

Table 1: Speedup factors when comparing VI, TVI, eTVI and eiTVI

Domain	TVI vs VI	eTVI vs TVI	eiTVI vs eTVI	eiTVI vs TVI
Layered (var. states)	2.4988	1.4306	1.3955	1.9965
Layered (var. layers)	1.8054	1.4549	0.9774	1.4220
SAP	1.3999	1.3725	1.7440	2.3937
Wetfloor	1.3810	1.7788	1.8635	3.3147
Average	1.6285	1.6018	1.3119	2.1014

Table 2: Cache metrics obtained on the Layered domain

Solver	Cache-Refs	Cache-Misses	Miss Percent
TVI	2.87G	0.860G	29.96
eTVI	2.39G	0.413G	17.28
eiTVI	1.59G	0.328G	20.62

References

[1] Jaël Champagne Gareau, Éric Beaudry, and Vladimir Makarenkov, 'Cache-efficient memory representation of Markov decision processes', *Proceedings of the Canadian Conference on Artificial Intelligence*, (2022).

[2] Peng Dai, Mausam, Daniel Weld, and Judy Goldsmith, 'Topological value iteration algorithms', *Journal of Artificial Intelligence Research*, 42, 181–209, (2011).

[3] Anuj Jain and Sartaj Sahni, 'Cache efficient value iteration using clustering and annealing', *Computer Communications*, 159, 186–197, (2020).

[4] Mausam and Andrey Kolobov, *Planning with Markov Decision Processes: An AI Perspective*, number 1, Morgan & Claypool, 2012.

[5] David Wingate and Kevin D Seppi, 'Cache performance of priority metrics for MDP solvers', in *AAAI Workshop - Technical Report*, volume WS-04-08, pp. 103–106. AAAI Press, (2004).

[6] David Wingate and Kevin D Seppi, 'P3VI: A partitioned, prioritized, parallel value iterator', in *Proceedings of the Twenty-First International Conference on Machine Learning, ICML 2004*, pp. 863–870, (2004).

[7] David Wingate and Kevin D Seppi, 'Prioritization methods for accelerating MDP solvers', *Journal of Machine Learning Research*, 6, 851–881, (2005).

Online Material

The paper, presentation slides, C++ code, test instance generators and supplementary materials are available by scanning the following QR code:

Acknowledgments

Fonds de recherche Nature et technologies Québec

We acknowledge the support of the *Natural Sciences and Engineering Council of Canada* (NSERC) and the *Fonds de recherche du Québec — Nature et technologies* (FRQNT).