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.

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](image)

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

<table>
<thead>
<tr>
<th>Domain</th>
<th>TVI vs VI</th>
<th>eTVI vs TVI</th>
<th>eiTVI vs TVI</th>
<th>eiTVI vs VI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layered (var. states)</td>
<td>2.4988</td>
<td>1.4306</td>
<td>1.3955</td>
<td>1.9965</td>
</tr>
<tr>
<td>Layered (var. layers)</td>
<td>1.8054</td>
<td>1.4549</td>
<td>0.9774</td>
<td>1.4220</td>
</tr>
<tr>
<td>SAP</td>
<td>1.3999</td>
<td>1.3725</td>
<td>1.7440</td>
<td>2.3937</td>
</tr>
<tr>
<td>Wetfloor</td>
<td>1.3810</td>
<td>1.7788</td>
<td>1.8935</td>
<td>3.3147</td>
</tr>
<tr>
<td>Average</td>
<td>1.6285</td>
<td>1.6018</td>
<td>1.3119</td>
<td>2.1014</td>
</tr>
</tbody>
</table>

Table 2: Cache metrics obtained on the Layered domain

<table>
<thead>
<tr>
<th>Solver</th>
<th>Cache-Refs</th>
<th>Cache-Misses</th>
<th>Miss Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>TVI</td>
<td>2.87G</td>
<td>0.860G</td>
<td>29.96</td>
</tr>
<tr>
<td>eTVI</td>
<td>2.39G</td>
<td>0.413G</td>
<td>17.28</td>
</tr>
<tr>
<td>eiTVI</td>
<td>1.59G</td>
<td>0.328G</td>
<td>20.62</td>
</tr>
</tbody>
</table>

References


Online Material

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

![QR code](image)

Acknowledgments

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).