Bounded Suboptimal Path Planning with Compressed Path Databases

Shizhe Zhao¹, Mattia Chiari², Adi Botea³, Alfonso E. Gerevini⁴, Daniel Harabor⁴, Alessandro Saetti⁵, Peter J. Stuckey¹

¹Monash University, ²University of Brescia, ³Eaton

Introduction

What are CPDs?
- Precompute all-pair first moves: $O(n^2)$
- Compress: $\ll O(n^2)$
- Search-free path extraction

Motivation of this work:
- Existing approaches have achieved good compression and hard to improve.
- Trade-off between space and suboptimality.

Compress First move matrix by:

Method: Compute Centroids

<table>
<thead>
<tr>
<th>Type</th>
<th>Mean</th>
<th>min</th>
<th>25%</th>
<th>50%</th>
<th>75%</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>fwd</td>
<td>0.79</td>
<td>0.75</td>
<td>0.79</td>
<td>0.81</td>
<td>0.85</td>
<td>0.85</td>
</tr>
<tr>
<td>rev</td>
<td>0.78</td>
<td>0.77</td>
<td>0.80</td>
<td>0.81</td>
<td>0.84</td>
<td>0.84</td>
</tr>
</tbody>
</table>

Method: Reverse CPD

Advantages of reverse-CPD:
- get more space reduction from the centroids idea
- forward-CPD throws non-centroids columns - may get same number of entries after compression
- reverse-CPD throws non-centroids rows - always reduce the size
- faster path extraction
- forward-CPD look-up in multiple row
- reverse-CPD look-up in target row - less cache missing - single look-up may faster
- and one look-up may extract multiple continuous moves - needs less number of look-up

Result

Size reduction ratio:

<table>
<thead>
<tr>
<th>Type</th>
<th>Mean</th>
<th>min</th>
<th>25%</th>
<th>50%</th>
<th>75%</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>fwd</td>
<td>0.79</td>
<td>0.75</td>
<td>0.79</td>
<td>0.81</td>
<td>0.85</td>
<td>0.85</td>
</tr>
<tr>
<td>rev</td>
<td>0.78</td>
<td>0.77</td>
<td>0.80</td>
<td>0.81</td>
<td>0.84</td>
<td>0.84</td>
</tr>
</tbody>
</table>

Speed up ratio:

<table>
<thead>
<tr>
<th>Type</th>
<th>Mean</th>
<th>min</th>
<th>25%</th>
<th>50%</th>
<th>75%</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>fwd</td>
<td>0.79</td>
<td>0.75</td>
<td>0.79</td>
<td>0.81</td>
<td>0.85</td>
<td>0.85</td>
</tr>
<tr>
<td>rev</td>
<td>0.78</td>
<td>0.77</td>
<td>0.80</td>
<td>0.81</td>
<td>0.84</td>
<td>0.84</td>
</tr>
</tbody>
</table>

Idea

- For each node, only store first-move data for a subset of grid nodes $C$, called centroid
- First-move matrix: $N \times N \rightarrow N \times |C|
- Each node $i$ belongs to a centroid $C(i)$
- Centroid path (cp for short)
  \[cp(s, t) = \text{shortestPath}(s, C(t)) + \text{+ shortestPath}(C(t), t)\]
- Bounded suboptimal: $|t, C(t)| \leq \delta$

Contribution

- Order-of-magnitude reductions in pre-processing
- Several factors improvement in database size
- Faster look-up
- Suboptimality cost per path is substantially smaller than the guaranteed upper bound

Method: Encode 'illegal' move

Encoding $S$ to $SW$ allows us to compress the rectangle region to $SW$.
When look-up column-3 symbols, we know $SW$ is illegal, thus we can decode it to a valid move $S$. 