Lookahead Pathology in Real-Time Path-Finding

Mitja Luštrek (mitja.lustrek@ijs.si)  Jožef Stefan Institute, Department of Intelligent Systems, Ljubljana, Slovenia

Vadim Bulitko (bulitko@ualberta.ca)  University of Alberta, Department of Computing Science, Edmonton, Alberta, Canada

Introduction
• Real-time path-finding ⇒
  ⇒ incomplete search methods ⇒
  ⇒ suboptimal actions
• Deeper lookahead believed to produce better actions
• Sometimes the opposite is true: pathology

Hypothesis 1
• Many pathological states
  Degree of pathology  0  1  2  3  4  ≥ 5
  On-policy (problems %) 38.5 15.1 20.3 17.0 7.6 1.5
• Hypothesis 1 apparently not correct
• Why the large difference between on-policy and off-policy?

Setting
• Path-finding in grid world
• Algorithm: LRTS [Bulitko & Lee 06]

Pathology Observed
Degree of pathology  0  1  2  3  4  ≥ 5
On-policy (problems %) 38.5 15.1 20.3 17.0 7.6 1.5

Hypothesis 2
• Smaller lookahead depths benefit more from the updates to the heuristic ⇒
  ⇒ depths closer ⇒ larger more likely worse than smaller
• First test: on-policy, ignore updates when measuring error ⇒
  ⇒ less pathology

<table>
<thead>
<tr>
<th>Degree of pathology</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>≥ 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>No updates (problems %)</td>
<td>79.8</td>
<td>14.2</td>
<td>4.5</td>
<td>1.2</td>
<td>0.3</td>
<td>0.0</td>
</tr>
</tbody>
</table>

• Second test: observe volume of updates to the heuristic ⇒
  ⇒ smaller volume at smaller depths

<table>
<thead>
<tr>
<th>Degree of pathology</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>≥ 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Every move (problems %)</td>
<td>86.9</td>
<td>9.0</td>
<td>3.3</td>
<td>0.6</td>
<td>0.2</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Hypothesis 3
• Fewer searches performed at larger lookahead depths ⇒
  ⇒ depths closer ⇒ larger more likely worse than smaller
• First test: on-policy experiment, search every move ⇒
  ⇒ less pathology

<table>
<thead>
<tr>
<th>Degree of pathology</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>≥ 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Every move (problems %)</td>
<td>86.9</td>
<td>9.0</td>
<td>3.3</td>
<td>0.6</td>
<td>0.2</td>
<td>0.0</td>
</tr>
</tbody>
</table>

• Second test: observe number of states generated per move when searching every move ⇒
  ⇒ steeper increase than normally

Towards a Remedy
• Adaptive lookahead on an example map:
  • optimal depth per start state:
    • 48% less travel than best fixed depth
  • optimal depth per move:
    • additional reduction
    • 14% in travel
    • 43% in computation per move

• Need to know optimal depths!
• Expensive to pre-compute for every state pair (7.6 · 10^6 pairs)

• State abstraction [Bulitko et al. 05]:
  • 0.004% state pairs pre-computed
  • 33% less travel than best fixed depth