Lookahead Pathology in Real-Time Path-Finding

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

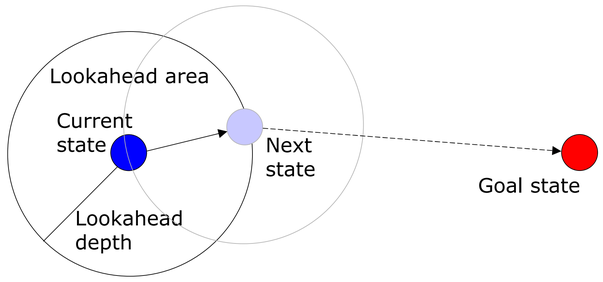
Mitja Luštrek (mitja.lustrek@ijs.si)
Jožef Stefan Institute, Department of Intelligent Systems, Ljubljana, Slovenia

Introduction

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

Setting

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



- Two types of experiments:
  - On-policy: start state → goal state, heuristic updated
  - Off-policy: randomly selected states, one move, heuristic not updated
- Degree of pathology: number of lookahead depths where error is larger than at the previous depth
- 1,000 problems (map, start state, goal state)

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

First Explanation

- Many pathological states

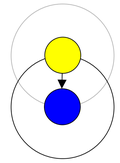
Degree of pathology	0	1	2	3	4	≥ 5
Off-policy (problems %)	95.7	3.7	0.6	0.0	0.0	0.0

- First explanation apparently not correct
- Why the large difference between on-policy and off-policy?

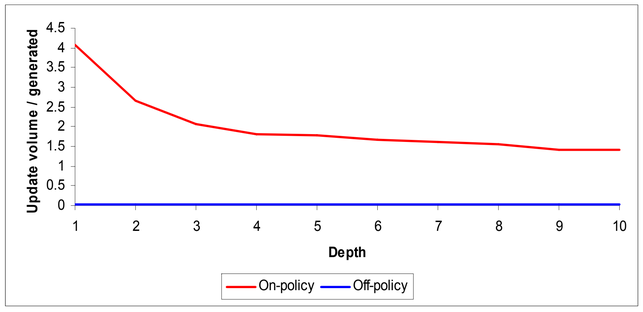
Second Explanation

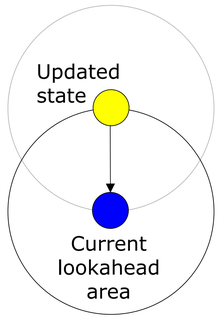
- Smaller lookahead depths benefit more from the updates to the heuristic ⇒
  ⇒ depths closer ⇒ larger more likely worse than smaller
- First test: on-policy, ignore updates when measuring error ⇒
  ⇒ less pathology

Degree of pathology	0	1	2	3	4	≥ 5
No updates (problems %)	79.8	14.2	4.5	1.2	0.3	0.0



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



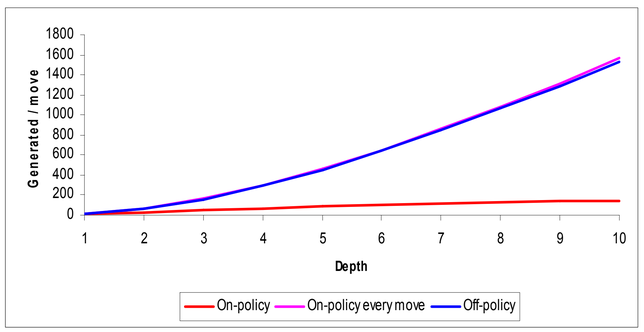


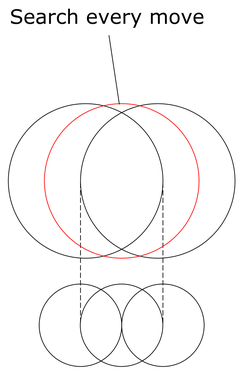
Third Explanation

- 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

Degree of pathology	0	1	2	3	4	≥ 5
Every move (problems %)	86.9	9.0	3.3	0.6	0.2	0.0

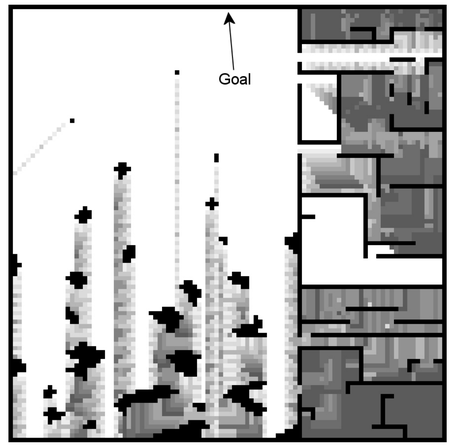
- Second test: observe number of states generated per move when searching every move ⇒
  ⇒ steeper increase than normally





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 ($7.6 \cdot 10^6$ pairs)

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