AI SMPS 2024 Week 6 Algorithms

Prepared by S. Baskaran

Beam Search

We will use the following version of Beam Search in assignments and exams because it is assignment friendly and prevents infinite loops.

Here, **sort$_h$** sorts from best to worst **h**-values.

```
BEAM-SEARCH(S, w)
OPEN ← S : []
N ← S
do  bestEver ← N
    if OPEN contains goal node
        return that goal node
    else neighbours ← MOVE-GEN(OPEN)
        OPEN ← take w (sort_h neighbours)
        N ← head OPEN        ▷ best in new layer
while h(N) is better than h(bestEver)
return bestEver
```

```
MOVE-GEN(OPEN)
neighbours ← []
for each X in OPEN
    neighbours ← neighbours ++ MOVE-GEN(X)
return neighbours     ▷ the list preserves duplicates
```

(**take** n LIST) returns at most n values from the beginning of LIST.

$[o, u, t] =$ **take** 3 $[o, u, t, r, u, n]$

$[a, t] =$ **take** 3 $[a, t]$

$[a] =$ **take** 3 $[a]$

$[] =$ **take** 3 $[]$

WA*(S, w)

```
1  Use A* algorithm with the following changes:
2        f(M) ← g(M) + w ∗ h(M)    ▷ Line 16
3        f(X) ← g(X) + w ∗ h(X)    ▷ Line 27
```

SMGS(S)

1. SMGS (Sparse-Memory Graph Search) is a memory optimized version of A*.
2. There is no change to OPEN list.
3. CLOSED list is split into two disjoint sets: BOUNDARY nodes (unrestricted memory), and KERNEL nodes (a fixed size memory).
4. KERNEL memory is periodically cleared to make way for new KERNEL nodes.

Where,

$\mathcal{I}$ = set of search interior nodes, i.e., nodes whose lowest-cost paths have been found,

Pred(k) = set of predecessor nodes of k, i.e., set of nodes that can make a transition into node k in the **underlying (state space) graph**.

$$\text{KERNEL}(\mathcal{I}) = \left\{ k \;\middle|\; k \in \mathcal{I},\ \forall p\big(\ p \in \text{Pred}(k) \ \Rightarrow\ p \in \mathcal{I}\ \big) \right\}$$

$$\text{BOUNDARY}(\mathcal{I}) = \mathcal{I} \setminus \text{KERNEL}(\mathcal{I})$$

$$\text{BOUNDARY}(\mathcal{I}) = \left\{ b \;\middle|\; b \in \mathcal{I},\ \exists p\big(\ p \in \text{Pred}(b)\ \wedge\ p \notin \mathcal{I}\ \big) \right\}$$

A*(S)

```
1   default value of g for every node is +∞
2   parent(S) ← null
3   g(S) ← 0
4   f(S) ← g(S) + h(S)
5   OPEN ← S : []
6   CLOSED ← empty list

7   while OPEN is not empty
8       N ← remove node with lowest f value from OPEN
9       add N to CLOSED

10      if GOAL-TEST(N) = TRUE
11          return RECONSTRUCT-PATH(N)

12      for each M in MOVE-GEN(N)
13          if g(M) > g(N) + k(N, M)
14              parent(M) ← N
15              g(M) ← g(N) + k(N, M)
16              f(M) ← g(M) + h(M)

17              if M is in OPEN
18                  continue
19              else if M is in CLOSED
20                  PROPAGATE-IMPROVEMENT(M)
21              else add M to OPEN        ▷ M is new

22  return empty list
```

PROPAGATE-IMPROVEMENT(M)

```
23  for each X in MOVE-GEN(M)
24      if g(X) > g(M) + k(M, X)
25          parent(X) ← M
26          g(X) ← g(M) + k(M, X)
27          f(X) ← g(X) + h(X)

28          if X is in CLOSED
29              PROPAGATE-IMPROVEMENT(X)
30  return
```