Review 10/17/2017

COSC 6368: Artificial Intelligence

**1) Best first Search and A\* [10]**

Consider the search space below, where *S* is the start node and *G1* and *G2* satisfy the goal test. Arcs are labeled with the cost of traversing them and the estimated cost to a goal (the h function itself) is reported inside nodes.

For each of the following search strategies, indicate which goal state is reached (if any) and list, *in order*, all the states *popped off of the OPEN list*. When all else is equal, nodes should be removed from OPEN in alphabetical order.

##### a) Best-First-Search (using function h only) [3]

Goal state reached: G2 [1]

States popped off OPEN: S, E, G2 [2]

##### b) A\* (using f=g+h)[4]

Goal state reached: G1 [1]

States popped off OPEN: S, A, B, G1 [3]

2

7

1

2

1

5

2

9

2

3

8

4

1

4

5

d) Assume you have 2 admissible heuristics h1(s) and h2(s) are given for a given seach problem. Is h3(s)=min(h1(s),h2(s)) also admissible? Would you prefer using h2 or using h3 in conjuction with A\*? Give reasons for your anwers[4].

Yes, h3 is admissible. If h1 and h2 always underestimate the “true” cost then the lesser of the two will certainly underestimate the true cost as well; therefore, h3 is admissible.

I will prefer h2, because h2 is always greater equal than h3 and therefore it provides a closer approximation of the true cost. As a matter of fact, h2 dominates h3, which translates into equal or better efficiency of the search, as discussed on the bottom of page 106 of our textbook.

1. Assume you apply randomized hill climbing to a minimization involving a continuous, differentiable function that has 3 minima. Will it always find the optimal solution? Give reasons for your answer! [3]

No, HC might climb down the wrong minimum depending on the chosen starting point

b) What is the “main” difference between simulated annealing and randomized hill climbing? [2]

… SA does allow downward steps…

d) Assume you apply a version of depth first search that checks for repeated states on the current path[[1]](#footnote-1), but which does not use a depth bound to the 8-puzzle. Will it always find a solution if a solution exists (assuming that there are enough computational resources)? [6]

Yes!

Because the search-tree for the 8-puzzle is finite and because the algorithm checks for loops in the current path the algorithm will sooner or later stop moving forward, backtrack, and find the solution eventually.

e) When does A\* terminate? Be precise!

2.Reinforcement Learning]

Consider the following World called ABC is given:



Give the Bellman equation for states 1 and 4 of the ABC world! [3]

U(1)= 5 + γ\*U(4) [1]

U(4)= 3 + γ\*max (U(2)\*0.3+ U(3)\*0.1+U(5)\*0.6, U(1)\*0.4+U(5)\*0.6) [2]

No partial credit!

b) Now we apply temporal difference learning, assuming the agent starts in state 2 and applies the operator sequence **w-y(ending up in state 2)-w**; assume the initial utilities are 0; what are the new utilities; also assume α=0.5 and γ=0.5)?

General Update Formula: U(s):=(1-α)\*U(s)+ α\*(R(s)+ γ\*U(s’))

U(2)=0+ 0.5(-2+0))=-1

U(3)=0+0.5\*(3+0.5\*-1)=1.25

U(2)=0.5\*-1 + 0.5\*(-2+0.5\*1,25)=-0.5+0.5\*-1.375=-1.1875

Remark; if γ would have been 1, U(2) would be greater than 1!

c) Assume you use temporal difference learning in conjunction with a random policy which choses actions randomly assuming a uniform distribution. Do you believe that the estimations obtained are a good measurement of the “goodness” of states, that tell an intelligent agent (*assume the agent is smart!!*) what states he/she should/should not visit? Give reasons for your answer! [3]

Not always; as we assume an intelligent agent will take actions that lead to good states and avoids bad states, the agent that uses the random policy might not recognize that a state is a good state if both good and bad states are successors of this state; for example,

 S2: R=+100

S1:R=-1

 S3: R=-100

Due to the agent’s policy the agent will fail to realize the S1 is a good state, as the agent’s average reward for visiting the successor states of S1 is 0; an intelligent agent would almost always go from S1 to S2, making S1 a high utility state with respect to TD-learning.

d) What role does the learning rate α play in temporal difference learning; how does running temporal difference learning with low values of α differ from running it with high values of α? [2]

It determines how quickly our current beliefs/estimations are updated based on new evidence.

e) Assume you run temporal difference learning with high values of γ—what are the implications of doing that? [2]

If γ is high the agent will more focus on its long term wellbeing, and will shy away from taking actions—although they lead to immediate rewards—that will lead to the medium and long term suffering of the agent.

### 3. Comparision of Seach Algorithms

Compare Traditional Hill Climbing/Randomized Hill Climbing and Best-first Search! What are the main differences between the two approaches?

n be the size of the search space, the number of nodes in the search tree

b the branching factor, the number of successors

m is the length of the longest path in the search tree

|  |  |  |
| --- | --- | --- |
|  | Randomized HC | Best First Search |
| The way they search | Explore a single path | Can explore multiple paths in general |
|  | Only moves forward, cannot move backward | Jumps between states; can explore multiple paths in parallel! |
| Storage | O(1) / O(m)[[2]](#footnote-2) | O(n) |
| Runtime  | O(m) but might stop prematurely, fast | O(n) in the worst case, not fast |
| Finding solutions in finite search spaces | might terminate prematurely; might go into the wrong direction and get stuck | Yes |
| Find global optimum | no | no, terminates permaturely |
| Parameter Selection | Choosing neighbor hood size and sampling rate for RHC is challenging | Straight Forward |
| Incorperating Heuristics  | Needs good state evaluation function | Need good state evaluation function  |
| Other  | RHC is a probabilistic algorithm (usually) returns different solutions in different runs | deterministic |

1. This version backtracks, if a loop in the current path is encountered. [↑](#footnote-ref-1)
2. Only if it is necessary to return the solution path, as in the 8-puzzle problem! [↑](#footnote-ref-2)