Dr. Eick

Artificial Intelligence

COSC 6368

Midterm Exam

Tuesday, October 24, 2017



*Name:*

*Student id:*

1. A\* & Best-first Search (13 points)
2. Hill Climbing, SA and EC (13 points)
3. Reinforcement Learning (13 points)
4. Designing a State Evaluation Function (8 points)
5. Games and Adversarial Search (9 points)

Point Total (out of 56):

Number Grade:

The exam is “open books and notes”, but no computers and cell phones allowed; you have 75 minutes to complete the exam. Write all your answers on this document (you can use back sides!).

**1) Best first Search and A\* [13]**

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) [2]

Goal state reached: G2

States popped off OPEN: S, E, G2

##### b) A\* (using f=g+h)[5]

Goal state reached: G1

States popped off OPEN: S, E, A, B, D, G1 [1 error: 1 point only!]

D

1

8

DDD

3

3

2

 B

2

 C

6

 E

1

7

2

1

2

1

1

9

A 3

G1

 0

G2

 0

**S**

4

2

3

8

4

1

4

5

c) Assume you run A\* with a state evaluation function h that is precise; that is, it gives the exact cost h(s) of reaching a goal state from s. What can be said about the efficiency of A\* in this case? Which nodes are expanded by A\* when this particular evaluation function h is used? [4]

Very efficient [2]; does not do any unnecessary[[1]](#footnote-2) state expansions as it only expands states along the optimal path from the initial state to a goal state [2]

d) When does A\* terminate? Be precise! [2]

When the goal state is expanded/popped from the open list **No credit for other answers!**

**2) SA, Hill Climbing, and EC [13]**

a) Compare randomized hill climbing and simulated; annealing what are the main differences between the two search strategies? [3]

SA moves down/stays at the same level RHC always goes up [2]

RHC employs a steepest ascent approach by picking the best solution in the neighborhood whereas SA just samples one solution[1]/

b) What **advantages** you see in using hill climbing strategies over more sophisticated strategies such as A\* or best-first search? [3]

fast [1]; no overhead from having to use more complex data structures such as trees[0.5], very low storage requirement—do not run out of storage[1] can be run multiple times with different initializations in the same time as more complicated search strategies need for a single run[1], easy to implement[0.5]; at most 3 points; there may be other answers that deserve credit. .

c) Evolution computing relies on Darwinian evolution and survival of the fittest. What does this mean? How is the survival of the fittest in evolutionary computing search strategies accomplished—what do they do to simulate Darwinian evolution? [4]

Better solutions participate in higher proportions in the breeding of the next generation/fitter individual have a higher chance to be selected for the breeding of the next generation. As a results genetic material that is responsible for a solution to be “good” becomes more prevalent, leading to improved population fitness [2]

The employ probabilistic parent selection, such as roulette wheel selection, in which fitter individuals are selected with a higher probability to participate in producing offspring. [2] **If they describe a particular selection mechanism, such as tournament or roulette wheel selection in their answer this is also fine!**

d) What role do crossover[[2]](#footnote-3) operators play in evolutionary computing systems? [3]

It merges information from two parent genotypes. By mating individuals with different but desirable features; it is capable to produce an offspring that combines these two features[2] that is, crossover is an exploitation operator that searches all possible combinations of features[1], but does not introduce anything new[1] **At most 3 points!**

**3) Reinforcement Learning [13]**

Consider the following World called ABC is given:



1. Give the Bellman equation for states 2 and 3 of the ABC world! [3]

U(2)= -2+γ\* max(U(1),U(4)) [1.5]

U(3)= -9+γ\*max (U(2),U(4)) [1.5]

**No partial credit!**

1. Now we apply temporal difference learning, assuming the agent starts in state 3 and applies the operator sequence **ne-y(ending up in state 3)-ne-y(ending up ins state 3)**; what are the final utilities of state 3 and 4—give each step of the computation? Assume the initial utilities are 0; also assume α=0.5 and γ=0.5)? [5]
2. U(3)=0+0.5\*(-9+0.5\*0)=-4.5
3. U(4)=0+0.5\*(+3+0.5\*-4.5)=0.5\*0.75=0.375
4. U(3)= 0.5\*-4.5\*+ 0.5\*(-9 + 0.5\*0.375)=-2.25-4.4≈--6.7
5. U(4)= 0.5\*0.375 + 0.5\*(3+0.5\*-6.7)= 0.1875+0.5\*-0.335≈0

**One major error: 0-2.5 points**

**Two major errors: 0 points**

**Correct Formulas but computation errors: 3-4 points**

1. Describe the approach temporal difference learning uses to determine which states are “good” or ‘bad’. How is temporal difference learning ultimately able to learn paths to good state in initially unknown worlds? [5+2 extra points]

It uses exploration in the state space at hand and the obtained feedback during the exploration is used to learn the utility of states and/or actions[1]. As elucidated by Richard Sutton, the core idea of TD learning is that one adjusts predictions to match other, more accurate, predictions about the future.[[3]](https://en.wikipedia.org/wiki/Temporal_difference_learning#cite_note-RSutton-1988-3)[2]. State/operator utilities are updated using the difference between its currently value and the current reward and utilities of states visited in the future[1**] At most 3+1 points!**

Initially TD-learning learns that a particular state is good and then propagates the notion of goodness to predecessors, predecessor of predecessors, etc. of good states, ultimately learning paths to good states in a state space[2+ 1 extra]

**Feel free to give partial credit here if they something that matches the above answer!**

1. **Finding a Good State Evaluation Function for Heuristic Search [8]**

Assume a **non-finite maze** is given in which a robot has the task to move from an initial position (0,0) to a goal position. The available operators are north, south, east, and west that move the robot one field in the indicated direction. However, sometimes the robot cannot move in a particular direction e.g. if the robot faces a wall in the north it cannot move north; e.g. in the figure below being in position (2,0) the robot cannot move north.

(0,0) (0,1) (0,2) (0,3) (0,4)…

(1,0) (1,1) (1,2) (1,3) (1,4)…

(2,0) (2,1) (2,2) (2,3) (2,4)…

(3,0) (3,1) (3,2) (3,3) (3,4)…

(4,0) (4,1) (4,2) (4,3) (4,4)…

……………………………..

Assume that best-first search is used for this particular problem, and there are two goal states (500,500) and (900,900); moreover, your strategy should work for different mazes that differ in the walls placed. Design a state evaluation function h that searches the space efficiently and always finds a path to a goal state if such a path exists (be aware of the fact the search space is non-finite, that that there might walls that might prevent you from moving into a particular direction; moreover, a particular goal position might not be reachable from (0,0)).

h(x,y):= min(|x-500|+|y-500|, |x-900|+|y-900|) [5]

**Can give up to 2 points to sub-optimal heuristics h, as long they make any sense!**

Characterize how the search space is searched when your proposed h function is used. Moreover, comment if your proposed h-function is admissible.

Searches expanding the state that is the closest to one of the two goal states [2]

It is admissible, as Manhattan distance is the smallest number of steps the agent has to travel to the nearest goal state, assuming that there are no walls in the search space [1]

**5) Games and Adversarial Search [9]**

a) For most game with complete information, such as Go and chess, it is not feasible to construct the complete search tree; how do game-playing programs cope with this challenge? [4]

**Should mention that a state evaluation functions is developed for non-terminal nodes, and that the game tree is then grown to a certain depth; finally alpha-beta is applied to this tree to select the best move--see text book.**

b) Assume you try to develop a search strategy for games with incomplete information where the available moves depend on a dice role, such as Backgammon or Monopoly. Give a brief description how min-max/alpha beta search can be generalized to deal with the uncertainty that future moves depend on dice rolls. [5]

**Should explain how chance nodes are used to generalize alpha-beta for probabilistic game---see text book.**

1. Expanding states that are not visited in the optimal solution. [↑](#footnote-ref-2)
2. Also sometime call recombination operator in the literature. [↑](#footnote-ref-3)