Supervision Assignments in AI
Easter Term 2019
Set 1: Problem Solving by Search

Supervisor : Dionysis Manousakas
dm754@cam.ac.uk

May 6, 2019

1. (*Time dependence of performance measure, AIMA2.1*) Suppose that the performance measure of an agent's behaviour is concerned with just the first T time steps of the environment and ignores everything thereafter. Show that a rational agents action may depend not just on the state of the environment but also on the time step it has reached.

   Any sequential environment in which rewards may take time to arrive will work, because then we can arrange for the reward to be "over the horizon". E.g., suppose that in any state there are two action choices, a and b, and consider two cases: the agent is in state s at time T or at time $T-1$. In state s, action a reaches state s' with reward 0, while action b reaches state s again with reward 1; in s' either action gains reward 10. At time $T-1$, its rational to do a in s, with expected total reward 10 before time is up; but at time T, its rational to do b with total expected reward 1 because the reward of 10 cannot be obtained before time is up.
   Other examples from real life: investments whose payoff occurs after the end of life, exams where it doesnt make sense to start the high-value question with too little time left to get the answer, and so on.

2. (*Robot navigation out of maze, AIMA3.2*)Your goal is to navigate a robot out of a maze. The robot starts in the center of the maze facing north. You can turn the robot to face north, east, south, or west. You can direct the robot to move forward a certain distance, although it will stop before hitting a wall.

   a. Formulate this problem. How large is the state space?

   b. In navigating a maze, the only place we need to turn is at the intersection of two or more corridors. Reformulate this problem using this observation. How large is the state space now?

   c. From each point in the maze, we can move in any of the four directions until we reach a turning point, and this is the only action we need to do. Reformulate the problem using these actions. Do we need to keep track of the robots orientation now?

d. In our initial description of the problem we already abstracted from the real world, restricting actions and removing details. List three such simplifications we made.

a. We'll define the coordinate system so that the center of the maze is at $(0,0)$, and the maze itself is a square from $(-1,-1)$ to $(1,1)$.

   Initial state: robot at coordinate $(0,0)$, facing North.

   Goal test: either $|x| > 1$ or $|y| > 1$ where (x,y) is the current location. Successor function: move forwards any distance d; change direction robot it facing. Cost function: total distance moved.

   The state space is infinitely large, since the robots position is continuous.

b. The state will record the intersection the robot is currently at, along with the direction its facing. At the end of each corridor leaving the maze we will have an exit node. We'll assume some node corresponds to the center of the maze.

   Initial state: at the center of the maze facing North.

   Goal test: at an exit node.

   Successor function: move to the next intersection in front of us, if there is one; turn to face a new direction. Cost function: total distance moved.

   There are $4n$ states, where n is the number of intersections.

c. Goal test: at an exit node.

   Successor function: move to next intersection to the North, South, East, or West.

   Cost function: total distance moved.

   We no longer need to keep track of the robots orientation since it is irrelevant to predicting the outcome of our actions, and not part of the goal test. The motor system that executes this plan will need to keep track of the robots current orientation, to know when to rotate the robot.

d. State abstractions:

   i. Ignoring the height of the robot off the ground, whether it is tilted off the vertical
   ii. The robot can face in only four directions.
   iii. Other parts of the world ignored: possibility of other robots in the maze, the weather in the Caribbean.

   Action abstractions:

   i. We assumed all positions we safely accessible: the robot couldnt get stuck or damaged.
   ii. The robot can move as far as it wants, without having to recharge its batteries.
   iii. Simplified movement system: moving forwards a certain distance, rather than controlled each individual motor and watching the sensors to detect collisions.

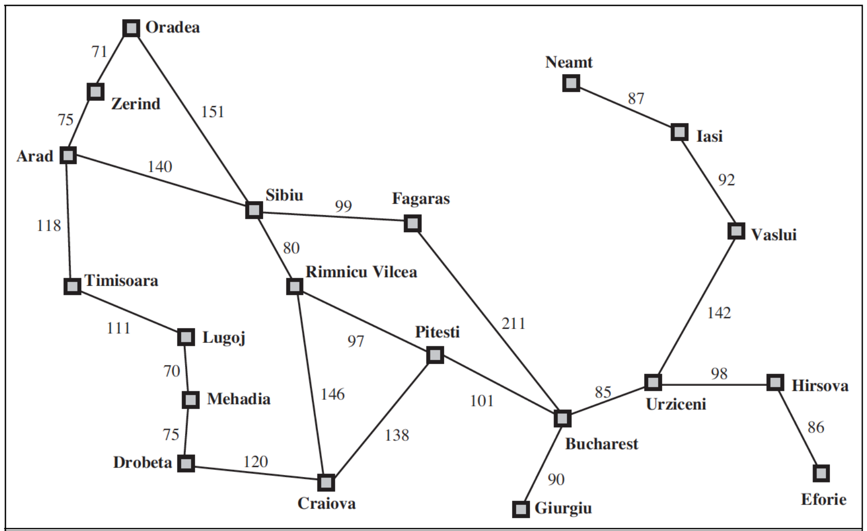

Figure 3.2 A simplified road map of part of Romania.

3. (*Friends meeting, AIMA3.3*) Suppose two friends live in different cities on a map, such as the Romania map shown below. On every turn, we can simultaneously move each friend to a neighboring city on the map. The amount of time needed to move from city i to neighbor j is equal to the road distance $d(i, j)$ between the cities, but on each turn the friend that arrives first must wait until the other one arrives (and calls the first on his/her cell phone) before the next turn can begin. We want the two friends to meet as quickly as possible.

   a. Write a detailed formulation for this search problem. (You will find it helpful to define some formal notation here.)

   b. Let $D(i, j)$ be the straight-line distance between cities i and j. Which of the following heuristic functions are admissible?

      (i) $D(i, j)$

      (ii) $2D(i, j)$

      (iii) $D(i, j)/2$.

   c. Are there completely connected maps for which no solution exists?

   d. Are there maps in which all solutions require one friend to visit the same city twice?

a. State space: States are all possible city pairs (i, j). The map is not the state space.

   Successor function: The successors of (i, j) are all pairs (x, y) such that Adjacent(x, i) and Adjacent(y, j).

   Goal: Be at (i, i) for some i.

   Step cost function: The cost to go from (i, j) to (x, y) is $max(d(i, x), d(j, y))$.

b. In the best case, the friends head straight for each other in steps of equal size, reducing their separation by twice the time cost on each step. Hence (iii) is admissible.

c. Yes: e.g., a map with two nodes connected by one link. The two friends will swap places forever. The same will happen on any chain if they start an odd number of steps apart. (One can see this best on the graph that represents the state space, which has two disjoint sets of nodes.) The same even holds for a grid of any size or shape, because every move changes the Manhattan distance between the two friends by 0 or 2.

d. Yes: take any of the unsolvable maps from part (c) and add a self-loop to any one of the nodes. If the friends start an odd number of steps apart, a move in which one of the friends takes the self-loop changes the distance by 1, rendering the problem solvable. If the self-loop is not taken, the argument from (c) applies and no solution is possible.

4. (*True/False, AIMA3.14*) Which of the following are true and which are false? Explain your answers.

   a. Depth-first search always expands at least as many nodes as A^* search with an admissible heuristic.
   b. $h(n) = 0$ is an admissible heuristic for the 8-puzzle.
   c. A^* is of no use in robotics because percepts, states, and actions are continuous.
   d. Breadth-first search is complete even if zero step costs are allowed.
   e. Assume that a rook can move on a chessboard any number of squares in a straight line, vertically or horizontally, but cannot jump over other pieces. Manhattan distance is an admissible heuristic for the problem of moving the rook from square A to square B in the smallest number of moves.

a. *False*: a lucky DFS might expand exactly d nodes to reach the goal. A^* largely dominates any graph-search algorithm that is guaranteed to find optimal solutions.

b. *True*: $h(n) = 0$ is always an admissible heuristic, since costs are nonnegative.

c. *False*: A^* search is often used in robotics; the space can be discretized or skeletonized.

d. *True*: depth of the solution matters for breadth-first search, not cost.

e. *False*: a rook can move across the board in move one, although the Manhattan distance from start to finish is 8.

5. (*Iterative deepening, AIMA3.18*) Describe a state space in which iterative deepening search performs much worse than depth-first search (for example, $O(n^2)$ vs. $O(n)$).

Consider a domain in which every state has a single successor, and there is a single goal at depth n. Then depth-first search will find the goal in n steps, whereas iterative deepening search will take $1 + 2 + 3 + \cdots + n = O(n^2)$ steps.

6. *(BFS/DFS/Uniform-cost, AIMA 3.21)* Prove each of the following statements, or give a counterexample:

   a. Breadth-first search is a special case of uniform-cost search.
   b. Depth-first search is a special case of best-first tree search.
   c. Uniform-cost search is a special case of A^* search.

a. When all step costs are equal, $g(n) \propto \mathit{depth}(n)$, so uniform-cost search reproduces breadth-first search.

b. Breadth-first search is best-first search with $f(n) = \mathit{depth}(n)$; depth-first search is best-first search with $f(n) = -\mathit{depth}(n)$; uniform-cost search is best-first search with $f(n) = g(n)$.

c. Uniform-cost search is A^* search with $h(n) = 0$.

7. (*n-queens, AIMA4.4*) Generate a large number of 8-puzzle and 8-queens instances and solve them (where possible) by hill climbing (steepest-ascent and first-choice variants), hill climbing with random restart, and simulated annealing. Measure the search cost and percentage of solved problems and graph these against the optimal solution cost. Comment on your results.

Coding.

8. (*n-queens, AIMA3.5*) An efficient incremental formulation of n-queens problem is as follows:

- States: All possible arrangements of n queens ($0 \leq n \leq 8$), one per column in the leftmost n columns, with no queen attacking another.
- Actions: Add a queen to any square in the leftmost empty column such that it is not attacked by any other queen.

Explain why the state space has at least $\sqrt[3]{n!}$ states and estimate the largest n for which exhaustive exploration is feasible. (Hint: Derive a lower bound on the branching factor by considering the maximum number of squares that a queen can attack in any column.)

The formulation puts one queen per column, with a new queen placed only in a square that is not attacked by any other queen. To simplify matters, well first consider the n-rooks problem. The first rook can be placed in any square in column 1 (n choices), the second in any square in column 2 except the same row that as the rook in column 1 ($n-1$ choices), and so on. This gives $n!$ elements of the search space.
For n-queens, notice that a queen attacks at most three squares in any given column, so in column 2 there are at least $(n-3)$ choices, in column at least $(n-6)$ choices, and so on. Thus the state space size $S \geq n(n-3)(n-6)\ldots$. Hence we have

$$\begin{aligned} S^3 &\geq n \cdot n \cdot n \cdot (n-3) \cdot (n-3) \cdot (n-3) \cdot (n-6) \cdot (n-6) \cdot (n-6) \cdot \cdots \\ &\geq n \cdot (n-1) \cdot (n-2) \cdot (n-3) \cdot (n-4) \cdot (n-5) \cdot (n-6) \cdot (n-7) \cdot (n-8) \cdot \cdots \\ &= n! \end{aligned}$$

or $S \geq \sqrt[3]{n!}$.

9. *(A*, AIMA3.23)* Trace the operation of A^* search applied to the problem of getting to Bucharest from Lugoj using the straight-line distance heuristic. That is, show the sequence of nodes that the algorithm will consider and the f, g, and h score for each node.

The sequence of queues is as follows:
L[0+244=244] M[70+241=311], T[111+329=440] L[140+244=384],
D[145+242=387], T[111+329=440] D[145+242=387], T[111+329=440],
M[210+241=451], T[251+329=580] C[265+160=425], T[111+329=440],
M[210+241=451], M[220+241=461], T[251+329=580] T[111+329=440],
M[210+241=451], M[220+241=461], P[403+100=503], T[251+329=580],
R[411+193=604], D[385+242=627] M[210+241=451], M[220+241=461],
L[222+244=466], P[403+100=503], T[251+329=580], A[229+366=595],
R[411+193=604], D[385+242=627] M[220+241=461], L[222+244=466],
P[403+100=503], L[280+244=524], D[285+242=527], T[251+329=580],
A[229+366=595], R[411+193=604], D[385+242=627] L[222+244=466],
P[403+100=503], L[280+244=524], D[285+242=527], L[290+244=534],
D[295+242=537], T[251+329=580], A[229+366=595], R[411+193=604],
D[385+242=627] P[403+100=503], L[280+244=524], D[285+242=527],
M[292+241=533], L[290+244=534], D[295+242=537], T[251+329=580],
A[229+366=595], R[411+193=604], D[385+242=627], T[333+329=662]
B[504+0=504], L[280+244=524], D[285+242=527], M[292+241=533],
L[290+244=534], D[295+242=537], T[251+329=580], A[229+366=595],
R[411+193=604], D[385+242=627], T[333+329=662], R[500+193=693],
C[541+160=701]

10. *(Consistent/admissible heuristics, AIMA3.29)* Prove that if a heuristic is consistent, it must be admissible. Construct an admissible heuristic that is not consistent.

A heuristic is consistent iff, for every node n and every successor n' of n generated by any action a,
$$h(n) \le c(n, a, n') + h(n').$$
One simple proof is by induction on the number k of nodes on the shortest path to any goal from n. For $k = 1$, let n' be the goal node; then $h(n) \le c(n, a, n')$. For the inductive case, assume n' is on the shortest path k steps from the goal and that $h(n')$ is admissible by hypothesis; then
$$h(n) \le c(n, a, n') + h(n') \le c(n, a, n') + h^*(n') = h^*(n),$$
so $h(n)$ at $k + 1$ steps from the goal is also admissible.

11. (*Online DFS, AIMA4.14*)Like DFS, online DFS is incomplete for reversible state spaces with infinite paths. For example, suppose that states are points on the infinite two-dimensional grid and actions are unit vectors $(1, 0)$, $(0, 1)$, $(1, 0)$, $(0, 1)$, tried in that order. Show that online DFS starting at $(0, 0)$ will not reach $(1, 1)$. Suppose the agent can observe, in addition to its current state, all successor states and the actions that would lead to them. Write an algorithm that is complete even for bidirected state spaces with infinite paths. What states does it visit in reaching $(1, 1)$?

Since we can observe successor states, we always know how to backtrack from to a previous state. This means we can adapt iterative deepening search to solve this problem. The only difference is backtracking must be explicit, following the action which the agent can see leads to the previous state.
The algorithm expands the following nodes:
Depth 1: $(0,0),(1,0),(0,0),(1,0),(0,0)$
Depth 2: $(0,1),(0,0),(0,1),(0,0),(1,0),(2,0),(1,0),(0,0),(1,0),(1,1),(1,0),(1,1)$.

12. (*AIMA4.1*) Give the name of the algorithm that results from each of the following special cases:

a. Local beam search with $k = 1$.

b. Local beam search with one initial state and no limit on the number of states retained.

c. Simulated annealing with $T = 0$ at all times (and omitting the termination test).

d. Simulated annealing with $T = \infty$ at all times.

a. Local beam search with $k = 1$ is hill-climbing search.
b. Local beam search with one initial state and no limit on the number of states retained, resembles breadth-first search in that it adds one complete layer of nodes before adding the next layer. Starting from one state, the algorithm would be essentially identical to breadth-first search except that each layer is generated all at once.
c. Simulated annealing with $T = 0$ at all times: ignoring the fact that the termination step would be triggered immediately, the search would be identical to first-choice hill climbing because every downward successor would be rejected with probability 1.
d. Simulated annealing with $T = \infty$ at all times is a random-walk search: it always accepts a new state.