CS486: Artificial Intelligence

Homework 1 (15 pts)

Search

Due 6 September 2017 @ 1630

Instructions

This is an individual assignment; however, *you may receive assistance and/or collaborate without penalty, so long as you properly document such assistance and/or collaboration in accordance with DAW.*

Answer the questions below and submit a hardcopy with DAW coversheet and acknowledgment statement to your instructor by the due date.

Problem 1: Class III Resupply

Welcome to the Forward Arming and Refueling Point (FARP), Lieutenant! You've just arrived at your first Quartermaster assignment and one of your 92Fs (Petroleum Supply Specialist) needs your help: she has an M978A4 Fuel Servicing Truck full of JP-8 (2,500 gallon capacity), but the tanker's dispenser gauge is broken. A new one is on order, but in the meantime, she needs to dispense exactly **one** gallon of fuel for the CO's driver. She has two fuel containers with 8 and 3 gallon capacities, respectively, but there are no measurement marks on them, so she can only know how much is in a container if it is full or has been previously filled with some known quantity(ies) of fuel. With each step, she can fill one of the containers to capacity, pour from one container to the other (up to the remaining capacity of the receiving container), or pour fuel from one container back into the tanker. She asks you how she can most efficiently perform the POL measurement, minimizing the number of steps required.

a. Describe a state encoding for this problem.

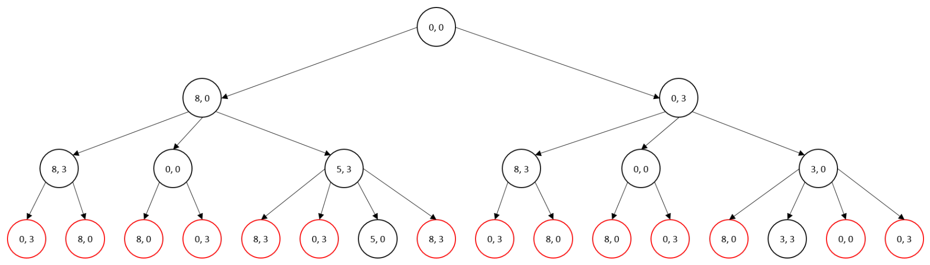
   A tuple (x, y) of two values, the first of which is the number of gallons in the 8-gallon container, and the second is the number of gallons in the 3-gallon container.

b. Provide the formal specification of the problem (start state, goal state(s), transitions, costs) in sufficient detail to apply a search strategy to it.

   - start state: $(0, 0)$

- goal states: $(1, y), (x, 1)$
- transitions:
  - fill:
    $(x, y) \to (8, y)$
    $(x, y) \to (x, 3)$
  - empty:
    $(x, y) \to (0, y)$
    $(x, y) \to (x, 0)$
  - transfer:
    $(x, y) \to (\max(0, x - (3 - y)),\ \min(3, y + x))$
    $(x, y) \to (\min(8, x + y),\ \max(0, y - (8 - x)))$
  - All costs are 1.

c. Suppose you used breadth-first *tree search*. If the start state of the search tree is level 0, how many states are present at level 3?



16

d. How many states are present at level 3 using breadth-first *graph search*?

2

e. Continue building the tree from level 3 using breadth-first graph search. What is the solution to your 92F's problem?

(a) fill $y \to (0, 3)$
(b) transfer y to $x \to (3, 0)$
(c) fill $y \to (3, 3)$
(d) transfer y to $x \to (6, 0)$
(e) fill $y \to (6, 3)$
(f) transfer y to $x \to (8, 1)$

Problem 2: Kitchen Patrol

The mess hall has a pile of P potatoes to be eaten and B potato-processing bots. At any time step, a bot is either a *chopper* or a *devourer*; all begin as choppers. At a given time step, a chopper can *chop*, *idle*, or *transform*. If it chops, it will turn one potato into one pile of fries. If it is idle, it will do nothing. If it transforms, it will do nothing that time step, but it will be a devourer in the next time step. Devourers can only *devour* or *transform*. If D devourers devour, they will consume exactly D^2 piles of fries in that time step—but only if at least that many piles exist; if there are fewer piles, nothing will be devoured. If a devourer transforms, it will do nothing that time step, but it will be a chopper in the next one. The goal is to have no potatoes or fries left while using the minimum number of time steps.

a. Describe a *minimal* state space representation, i.e., what is the smallest set of values you must use to differentiate states in order to use an *uninformed* search strategy to solve this problem? You may describe this in terms of the variables used above (P, B, D) or define your own (e.g., F for piles of fries, T for time steps).

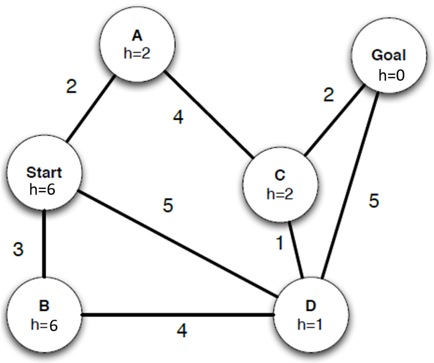
   (P, F, D). We need to know the number of potatoes and piles of fries remaining to detect a goal state. The number of piles of fries cannot be determined from the number of potatoes remaining, and vice versa. We must know how many choppers and how many devourers we have in order to determine what states we can transition to. Knowing either one of these values allows us to calculate the other, since the total number of bots, B, is fixed. Our choices of which bots are idle/transforming/chopping/devouring do not need to be encoded in the state—the combinations of those choices are the *actions* that determine the state we transition to.

b. Using the variables above, *write and explain* an expression for the resultant state space size.

   $\frac{1}{2}(P+1)(P+2)(B+1)$.

   If there are P potatoes remaining, then there can be no piles of fries. Ignoring devourers for a moment, this yields one state: $(P, 0)$. If there are $P-1$ potatoes remaining, there could be one pile of fries, or no piles of fries (if it was devoured). This yields two states: $(P-1, 1)$ or $(P-1, 0)$. If there are $P-2$ potatoes remaining, there could be 0, 1, or 2 piles of fries, depending on how many have been devoured. This yields three states.... This results in an arithmetic sequence: $1 + 2 + \ldots + (P+1) = \frac{1}{2}(P+1)(P+2)$. Each of these states can be combined with anywhere from 0 to B devourers, so we multiply by the $B+1$ different devourer states.

Problem 3: Search Strategy Behavior



a. In each of the following *graph search* strategies, specify the order in which states are expanded and the path returned by the graph search. In all cases, assume ties resolve in such a way that states with earlier alphabetical order are expanded first.

   (i) Depth-first search.
   Expansion order: Start-A-C-D-B.
   Path returned: Start-A-C-D-Goal.

   (ii) Breadth-first search.
   Expansion order: Start-A-B-D.
   Path returned: Start-D-Goal.

   (iii) Uniform cost search.
   Expansion order: Start-A-B-D-C.
   Path returned: Start-A-C-Goal.

   (iv) Greedy best-first search with the heuristic h as shown on the graph.
   Expansion order: Start-D.
   Path returned: Start-D-Goal.

   (v) A* search with the same heuristic.
   Expansion order: Start-A-D-C.
   Path returned: Start-A-C-Goal.

   Expansion means the node (or path containing the node as the last one in the path) is removed from the frontier and its neighbors are generated.

b. For each of the following strategies, state when the *goal test* should be performed—*before* a node is placed into the frontier (fringe) or *after* it is removed from the frontier (fringe).

(i) Depth-first search. after

(ii) Breadth-first search. before

(iii) Greedy best-first search. after

(iv) Uniform-cost and A* search. after

Basically, any time you are using a priority queue, you might put something on later that is better than something seen earlier, so you must only do the goal check after removing something from the structure. For DFS, suppose you are expanding nodes at a level left-to-right; you must allow the search algorithm to go deeper down a left-most path before allowing it to select a goal at the current level.

Problem 4: Heuristic Properties

a. Is the heuristic for the graph in Problem 3 consistent? If not, adjust the heuristic values to fix it.

No. Multiple options. One is $h(A) = 4$ and $h(B) = 5$.

b. Is the heuristic for the graph in Problem 3 admissible? If not, adjust the heuristic values to fix it.

Yes, it's admissible.

c. If $f(s)$, $g(s)$, and $h(s)$ are all admissible heuristics, which of the following are also guaranteed to be admissible heuristics:

(i) $f(s) + g(s) + h(s)$

(ii) $f(s)/6 + g(s)/3 + h(s)/2$

(iii) $\min(f(s), g(s), h(s))$

(iv) $\max(f(s), g(s), h(s))$

(v) $f(s)/3 + g(s)/3 + h(s)/3$

(vi) $f(s) \cdot g(s) \cdot h(s)$

(vii) $\min(f(s), g(s) + h(s))$

(viii) $\max(f(s), g(s) + h(s))$

In order to guarantee that a function of admissible heuristics is still admissible, the expression must be less than or equal to the max of the heuristics. Sums and products to not satisfy this, so i, vi, and viii all fail immediately. Answers iii and iv are correct because the max of a collection of admissible heuristics is admissible as is the min of an admissible heuristic and anything else. Answer iv is an average, so the value can never be greater than any of the values used to produce it. Answer ii is a weighted average, so it will always be smaller than the values producing it as well.