CS486: Artificial Intelligence

Homeworks 2 & 3 (30 pts)
Search
Due 28 September 2018 @ 1600

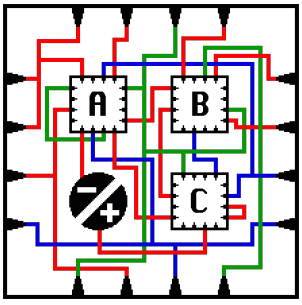
Instructions

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: Search Considerations

The picture below is a recursive maze—the A, B, and C boxes contain exact duplicates of the entire picture (and so on for the inner A/B/C's in those). The goal is to get between the top-level + and - (or vice versa); it is not sufficient to find a path from the top-level + to a - at some nested level.



a. (2 pts) At a minimum, what information does a state have to contain to formulate this as a search problem for our algorithms?

There are many possible answers, but they must contain a) some way of representing the level in the maze, b) the location in the current level, and c) the path so far. With respect to the level, not only must they track the current level, but they also need d) the previous levels. This could

be a stack that tracks both the current level as the top value and the previous levels as the remaining values, or maybe it's incorporated in the path information. +0.5 for each of these components.

b. (1 pt) What is the state space size for this search graph?

Infinite. +1 for correct answer, +0.5 for attempt.

c. (3 pts) What is an estimated branching factor for this search graph? (Note: Here, we are not looking for a "right" answer, *per se*. We'd just like you to explain a reasonable way you might calculate this, and then give the result of your calculation. Show your work.)

Again, there are many possible answers. If you consider each port along the edge of the maze as a decision node, then all three nested mazes share the "incoming" decision nodes represented by the entry ports of the outer square, and each nested maze (A, B, C) has a unique set of "outgoing" decision nodes, resulting in 64 decision nodes, total. Considering each in turn (labeling the nodes North, East, South, and West, and numbering them from 0 to 3 from left to right or top to bottom), we get the following branch counts:

Incoming:

	0	1	2	3
N	3	1	4	1
E	1	1	2	2
S	4	2	3	1
W	3	3	2	3

A Outgoing:

	0	1	2	3
N	3	0	2	1
E	4	0	0	1
S	1	3	1	1
W	1	0	2	0

B Outgoing:

	0	1	2	3
N	1	0	1	1
E	0	0	4	1
S	0	1	0	0
W	1	0	1	0

C Outgoing:

	0	1	2	3
N	4	0	0	1
E	0	2	1	1
S	0	0	1	0
W	0	1	0	1

From here, we can either establish an upper bound for the branching factor (4) or an average branching factor (79 branches/64 nodes ≈ 1.23 or 79 branches/43 non-zero nodes ≈ 1.84). Other reasonable calculations should be accepted.

d. (5 pts) Based on your answers above, identify which algorithm(s) we have covered in the course (depth-first, breadth-first, iterative deepening depth-first, uniform cost, greedy, and A^*) would be appropriate to use to find the shortest path to the goal and which one(s) would not. If you found more than one to be appropriate, discuss which one you would use. *Justify all answers*, discussing the relevant factors in the decisions.

- DFS: inappropriate because state space is infinite
- BFS: appropriate, but exponential ($O(b^d)$) space requirement
- IDFS: **best choice because of balance between space and optimality**
- UCS: appropriate, but same problem as BFS; path cost should incorporate penalty for exploring deeper levels
- GBFS and A*: inappropriate—no good heuristics

+0.5 each correct inappropriate/appropriate; +1 best selection or +0.5 for at least specifying one of the appropriate algorithms

Problem 2: Constraint Satisfaction Problems

a. (1 pt) Is the Missionaries and Cannibals problem a CSP? Why or why not?

No! It feels like one because the problem describes constraints that limit actions from some states, but CSPs only care about finding a goal state (an assignment of values to variables that satisfy all constraints). The Missionaries and Cannibals problem is a planning problem—you are trying to find the sequence of steps to the solution.
+1 for correct answer.

b. Learn about the *battleship puzzle* (`https://en.wikipedia.org/wiki/Battleship_(puzzle)`)—especially the "Rules" section.

i. (5 pts) Define a possible set of variables and their domains for this CSP. (Note: While the examples in class all used explicit representation for variables and domains, you may use an implicit representation for these definitions, if you wish.)

Two options come to mind:

1. The locations on the board—$(0, 0)$ through $(9, 9)$ could be the variables X_{ij} (similar to the N-queens problem) with domain $D = \{top\_end, bottom\_end, left\_end, right\_end, body, sub\}$.

2. The ship segments could be variables: B_{11} through B_{14} for the battleship, C_{11} through C_{13} for cruiser 1, C_{21} through C_{23} for cruiser 2, and so on for the three destroyers and four submarines. More mathematically: $T_{k,s}$ where T is the ship type $\{B, C, D, S\}$, k is which ship of that class it is $\{1, \ldots, 4\}$, and s is the segment number for that ship $\{1, \ldots, 4\}$. The domains for all of these variables would be the grid locations, $D = \{X_{ij}\} \forall i, j \in \{0, \ldots, 9\}$.

ii. (5 pts) Based on your choice above, choose *one* of the possible puzzle constraints to define using *either* implicit or explicit constraint representation.

Any valid solution is fine, but this question is probably easier to answer using option 2 above.

Note that the first option has 100 variables to assign, and each variable must start with the full range of possible domain values (except for the locations that are pre-filled by the puzzle). Many constraints need to be defined to prevent using the wrong type of end (e.g., a top or bottom end on a horizontal ship) or too many ends in a ship, or a submarine as a non-submarine segment, etc. That doesn't even include the segment count constraints for each row and column, non-adjacent ship constraints, non-diagonal ship constraints, etc. Many of these are very complicated to describe and check.

The second option has 20 variables to assign. The values assigned to any one particular ship must either have the same i value and be contiguous in j values or vice versa. For any square in the puzzle that is given, we can eliminate that square's location from the domain of any ship segment that wouldn't match (unary constraints). All ship segments must have different values (alldiff($T_{k,s}$))and no ship segment from any one ship can be adjacent to any segment from any other ship. Adding up all of the segments with the same i value is an easy way to check the row constraints, and likewise for the column constraints with the j values.

iii. (5 pts) Would you choose iterative improvement with min-conflicts heuristic or backtracking search with forward checking (constraint propagation) and MRV? Justify your answer.

Backtracking search is most appropriate here. These puzzles are designed such that there is only *one* correct answer. The chance that you will randomly come across it using iterative improvement is very small. It is better to use a methodical approach here.
+5 for correct answer and justification; consider justification of wrong answer for partial credit

Problem 3: Adversarial Search (3 pts)

We saw in class that the effectiveness of α-β pruning is affected by the order in which moves are evaluated. All of our examples have been using "static ordering." Research and describe a "dynamic ordering" technique you could use to enhance the effectiveness of α-β pruning for chess. Cite your source(s) appropriately.

Consider the source and the quality of the description. I expect most cadets to receive full credit for this one.