Comp3620/Comp6320 Artificial Intelligence
Tutorial 1: Search

March 7, 2013

Exercise 1 (problem formulation)

For each of the following problem, explain how states and actions can be represented, and give the initial state, goal test, successor function, and a plausible step cost function. Remember from the lectures that these elements constitutes a search problem formulation.

1. **Color a planar map using a minimum of colors in such a way that no two adjacent regions have the same color.**

   Given the number n of regions and an adjacency matrix $A[i,j], i,j \in 1,\ldots,n$, where $A[i,j]$ is true iff region i is adjacent to region j, a state is a vector C, where $C[i], i \in 1,\ldots,n$ is the color of region i or is NONE, and satisfies $A[i,j] \rightarrow C[i] \neq C[j]$.

   In the initial state $C[i] =$ NONE for all i.

   The successor function is:

   $$\text{successorFn}(C) = \{\langle \text{color}(i,c), C'\rangle \mid C[i] = NONE \text{ and } C[j] \neq c \text{ for all } j \text{ such that } A[i,j] \text{ and } C'[i] = c \text{ and } C'[j] = C[j] \text{ for } i \neq j\}$$

   The step-cost function adds a cost of one for the use of a new color, and uses a small positive cost otherwise: $g(C, \text{color}(i,c), C') = 1$ iff $C[j] \neq c$ for all j and $= \epsilon$ otherwise.

   The goal test is $goal(C)$ iff $C[i] \neq NONE$ for all i.

2. **Measure exactly 1 liter as fast as possible, using 3 jugs, measuring 12 liters, 8 liters and 3 liters, respectively, and a water tap, by filling jugs up completely with the water tap, or emptying the content of one jug into another or onto the ground.**

   A state is a vector $S[i], i = 1\ldots3$, representing the current amount of liquid in the respective jugs. In addition, we know the (fixed) jug capacities $\text{M}[i], i = 1\ldots3$.

   The successor function is:

   $$\text{successorFn}(S) = \begin{array}{l} \{\langle \text{empty}(i), S'\rangle \mid S'[i] = 0 \text{ and } S'[j] = S[j] \text{ for } j \neq i, i = 1\ldots3\} \cup \\ \{\langle \text{fill}(i), S'\rangle \mid S'[i] = \text{M}[i] \text{ and } S'[j] = S[j] \text{ for } j \neq i, i = 1\ldots3\} \cup \\ \{\langle \text{pour}(i,j), S'\rangle \mid S'[j] = \min(\text{M}[j], S[j] + S[i]), S'[i] = S[i] - S'[j] + S[j], \\ \qquad \text{and } S'[k] = S[k] \text{ for } k \neq i \text{ and } k \neq j, i, j = 1\ldots3, i \neq j\} \end{array}$$

   The goal test is $\text{goal}(S) \equiv \exists i \in 1\ldots3$ s.t. $S[i] = 1$

   The step-cost function can be equal to the amount of water moved + 1 for picking up a jug:

   - step-cost$(S, \text{empty}(i), S') = S[i] + 1$,
   - step-cost$(S, \text{fill}(i), S') = \text{M}[i] - S[i] + 1$,

- step-cost$(S, \text{pour}(i,j), S') = S[i] - S'[i] + 1$.

3. **Three missionaries and three cannibals are on the side of a river, along with a boat which can hold one or two people. Find a way of getting them all to the other side, without ever leaving missionaries outnumbered by cannibals at any place.**

   The state S represents the number of missionaries ($\text{m}(S) \in \mathbb{N}$) and canibals ($\text{c}(S) \in \mathbb{N}$) on the left, plus the position ($\text{p}(S) \in \{\text{l}, \text{r}\}$) of the boat.

   The initial state I is such that $\text{m}(I) = \text{c}(I) = 3$ and $\text{p}(I) = \text{l}$.

   Actions move the boat plus 1 or 2 people from one side to another, subject to the numbering constraints. Hence the function is:

$$\text{successorFn}(S) = \begin{array}{l} \{\langle \text{move-right}(c', m'), S' \rangle \mid \text{m}(S') = \text{m}(S) - m', \text{c}(S') = \text{c}(S) - c', \text{p}(S) = \text{l}, \text{p}(S') = \text{r} \\ 1 \leq c' + m' \leq 2, \text{m}(S') < \text{c}(S') \rightarrow m(S') = 0, \text{c}(S') < \text{m}(S') \rightarrow m(S') = 3\} \cup \\ \{\langle \text{move-left}(c', m'), S' \rangle \mid \text{m}(S') = \text{m}(S) + m', \text{c}(S') = \text{c}(S) + c', \text{p}(S) = \text{r}, \text{p}(S') = \text{l} \\ 1 \leq c' + m' \leq 2, \text{m}(S') < \text{c}(S') \rightarrow m(S') = 0, \text{c}(S') < \text{m}(S') \rightarrow m(S') = 3\} \end{array}$$

   The goal is to have no-one on the left: $goal(S) \equiv c(S) + m(S) = 0$

   The step-cost: 1 per action.

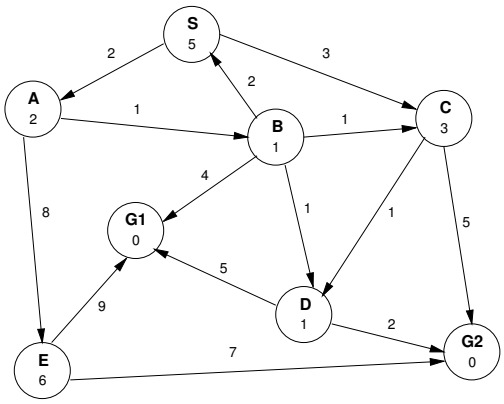
   As an aside, here is solution sequence:

state(c,m,p)	*action*	*comment*
$(3,3,\text{l})$	move-right$(2,0)$	move 2 canibals
$(1,3,\text{r})$	move-left$(1,0)$	move 1 canibal back
$(2,3,\text{l})$	move-right$(2,0)$	move 2 canibals
$(0,3,\text{r})$	move-left$(1,0)$	move 1 canibal back
$(1,3,\text{l})$	move-right$(0,2)$	move 2 missionaries
$(1,1,\text{r})$	move-left$(1,1)$	move 1 canibal & 1 missionary back
$(2,2,\text{l})$	move-right$(0,2)$	move 2 missionaries
$(2,0,\text{r})$	move-left$(1,0)$	move 1 canibal back
$(3,0,\text{l})$	move-right$(2,0)$	move 2 canibals
$(1,0,\text{r})$	move-left$(1,0)$	move 1 canibal back
$(2,0,\text{l})$	move-right$(2,0)$	move 2 canibals
$(0,0,\text{r})$	goal!	

Exercise 2 (search strategies)

Consider the search space below, where S is the initial state and $G1$ and $G2$ both satisfy the goal test. Arcs are labelled with the cost of traversing them and the estimated cost to a goal is reported inside nodes.

For each of the following search strategies: breadth-first, depth-first, iterative deepening, uniform cost, greedy search, and A*, indicate the goal state reached (if any) by graph-search and list, in order, all the states expanded — recall that a state is expanded when it is removed from the frontier. Everything else being equal, nodes should be removed from the frontier in alphabetical order. Assume that regardless of the strategy, the goal-test is performed when a node is dequeued from the fontier, and that newly generated nodes are not added to the frontier if they have already been explored or are on the frontier.



Breadth-first: finds G2. S(0), A(1), C(1), B(2), D(2), E(2), G2(2)

Depth-first: finds G1. S(0), A(1), B(2), D(3), G1(4)

Iterative deepending: finds G2. S(0), S(0), A(1), C(1), S(0), A(1), B(2), E(2), C(1), D(2), G2(2)

Uniform cost: finds G2. S(0), A(2), B(3), C(3), D(4), G2(6)

Greedy: finds G1. S(5), A(2), B(1), G1(0)

A*: finds G2. S(0+5=5), A(2+2=4), B(3+1=4), D(4+1=5), c(3+3=6) G2(6+0=6)

Exercise 3 (heuristics)

Consider the problem of moving k knights from k starting squares to k goal squares on an unbounded chess board, subject to the rule that no two knights can land on the same square at the same time.

1. **Each action consists of moving exactly one knight.**

   (a) **What is the maximum branching factor in this state space?**
   (i) $\mathbf{8}k$ **(ii)** $\mathbf{9}k$ **(iii)** 8^k **(iv)** 9^k

   (b) **Suppose h_i is an admissible heuristic for the problem of moving knight i to the goal g_i by itself. Which of the following heuristics are admissible:**
   (i) $\min_{i=1}^{k} h_i$ **(ii)** $\max_{i=1}^{k} h_i$ **(iii)** $\sum_{i=1}^{k} h_i$

   (c) **Which of these is the best heuristic?**

2. **Now suppose that each action consists of moving up to k knights simultaneously. Answer (a) (b) (c) again.**

1. Each action consists of moving exactly one knight

   (a) (i) ($8k$) is the correct answer. We choose one knight among k, and each has at most 8 possible moves as show in the following table:

	1		2	
8				3
		X		
7				4
	6		5	

(b) All of (i), (ii), and (iii) are admissible. All knights will have to move, and even if there was no conflict preventing them to land on the same square, knight i would take at least h_i to move to its final position. Hence $\sum_{i=1}^{k} h_i$ is admissible. Since $\min_{i=1}^{k} h_i \leq \max_{i=1}^{k} h_i \leq \sum_{i=1}^{k} h_i$ then the min and max are also admissible.

(c) The best heuristic is the sum as it dominates the other two.

2. Each action conists of moving up to k knights

(a) (iv) (9^k) is the correct answer. The setting is equivalent to moving all knights in parallel, where each knight has at most 9 moves, where the 9th move is a NoOp.

(b) only min and max are admissible. The knight that was going to take the largest number of moves to get to its goal position will still take that number of moves. Hence if h_i is an admissible heuristic for each knight, then $max_{i=1}^{k} h_i$ is admissible, and since min is less than max, so it is. But because several knights can move in parallel, there is no guarantee that the sum will be admissible.

(c) The best heuristic is the max if we're trying to find the optimal solution. On the other hand, a non-admissible heuristic may help finding a good solution quickly.

Exercise 4 (game playing)

1. **Approximately how many possible games of tic-tac-toe (sequences of moves) are there? Ignore symmetry and early wins.**

2. **Draw the whole game tree starting from an empty board down to depth 2 (one X and one O on the board). Exploit symmetry.**

3. **Define X_n as the number of rows, columns, or diagonals with exactly n X's and no O. Define O_n similarly for O. All non-terminal positions are evaluated using the following evaluation function: $Eval(s) = 3X_2(s) + X_1(s) - (3O_2(s) + O_1(s))$. Mark on the tree the evaluations of all the positions at depth 2.**

4. **Run the minimax algorithm to choose the best starting move.**

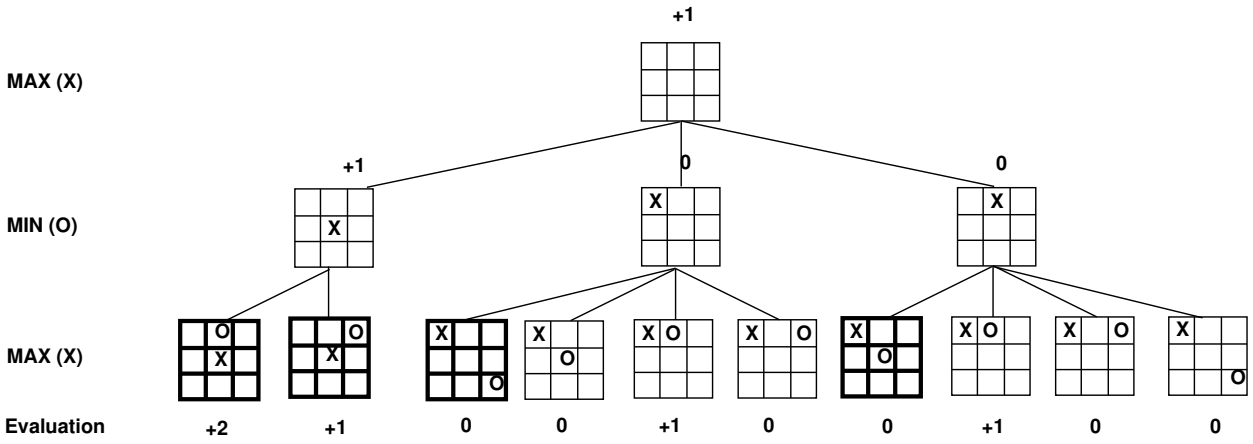
5. **Which nodes at depth 2 would not be evaluated if alpha-beta pruning was applied, assuming that the nodes are generated in the optimal order for alpha-beta pruning?**

1. There are at most 9! sequences of tic-tac-toe ($n^2!$ for an $n \times n$ board).

2. See figure below

3. See figure below. The best starting move is the center position.

4. Alpha beta will only explore the depth 2 nodes in bold. After the second bold node has been explored, the root node of the tree has a lower bound of 1. Hence, upon generating a max node n at depth 2 that has value less than 1, we can prune its siblings: the value of the min node parent of n will be less than 1, and the root node will never choose to move to that min node as it already has a better move with value 1.



Recommended exercises from the book:

3.3, 3.6/b-c, 3.9, 3.10, 3.11, 3.13, 3.14, 3.23, 3.36, 5.3, 5.7, 5.8, 5.12, 5.16, 5.18, 5.21