15-381 Spring 2007 Assignment 1 Solutions

Out: January 23rd, 2007
Due: February 6th, 1:30pm Tuesday

Many people lost 5 points for failing to put their andrew ID on their assignments. This was clearly listed in the instructions for the homework.

Many people lost 5 points for failing to follow the collaboration policy. This was clearly listed in the instructions for the homework.

1. The missionaries and cannibals problem is as follows. Three missionaries and three cannibals are on one side of a river, along with a boat. The boat can hold one or two people (and obviously cannot be paddled to the other side of the river with zero people in it). The goal is to get everyone to the other side, without ever leaving a group of missionaries outnumbered by cannibals. Your task is to formulate this as a search problem.

   (a) Define a state representation. (2 points)
   There are many possibilities. One example is:
   Represent the missionaries by M and the cannibals by C. Let the boat be B. Each state can be represented by the items on each side, e.g. $Side_1\{M, M, C, C\}, Side_2\{M, C, B\}$.

   (b) Give the initial and goal states in this representation. (1 point)
   Initial state: $Side_1\{M, M, M, C, C, C, B\}, Side_2\{\}$
   Goal state: $Side_1\{\}, Side_2\{M, M, M, C, C, C, B\}$

   (c) Define the successor function in this representation. (2 points)
   A set of missionaries and/or cannibals (call them $Move$) can be moved from $Side_a$ to $Side_b$ if:
   - *The boat is on $Side_a$.*
   - *The set $Move$ consists of 1 or 2 people that are on $Side_a$.*
   - *The number of missionaries in the set formed by subtracting $Move$ from $Side_a$ is 0 or it is greater than or equal to the number of cannibals.*
   - *The number of missionaries in the set formed by adding $Move$ to $Side_b$ is 0 or it is greater than or equal to the number of cannibals.*

   (d) What is the cost function in your successor fuction? (1 point)
   Each move has unit cost.

   (e) What is the total number of reachable states? (2 points)
   16:
   $Side_1\{M, M, M, C, C, C, B\}, Side_2\{\}$
   $Side_1\{\}, Side_2\{M, M, M, C, C, C, B\}$
   $Side_1\{M, M, M, C, C, B\}, Side_2\{C\}$
   $Side_1\{M, M, M, C, C\}, Side_2\{C, B\}$
   $Side_1\{M, M, M, C, B\}, Side_2\{C, C\}$
   $Side_1\{M, M, M, C\}, Side_2\{C, C, B\}$
   $Side_1\{M, M, C, C, B\}, Side_2\{M, C\}$
   $Side_1\{M, M, C, C\}, Side_2\{M, C, B\}$
   $Side_1\{M, C, B\}, Side_2\{M, M, C, C\}$

$Side_1\{M, C\}, Side_2\{M, M, C, C, B\}$
$Side_1\{C, C, C, B\}, Side_2\{M, M, M\}$
$Side_1\{C\}, Side_2\{M, M, M, C, C, B\}$
$Side_1\{C, C, B\}, Side_2\{M, M, M, C\}$
$Side_1\{C, C, B\}, Side_2\{M, M, M, C\}$
$Side_1\{M, M, M\}, Side_2\{C, C, C, B\}$
$Side_1\{C, B\}, Side_2\{M, M, M, C, C\}$

The last one is only reachable through the goal state, but it is still technically reachable (e.g. if you are just exploring the state space instead of searching for a goal).

These two are not reachable because the preceding state must have had more cannibals than missionaries on one side of the river:

$Side_1\{C, C, C\}, Side_2\{M, M, M, B\}$
$Side_1\{M, M, M, B\}, Side_2\{C, C, C\}$

2. Consider a state space where the start state is number 1 and the successor function for state n returns two states, numbers $2n$ and $2n + 1$.

   (a) Draw the portion of the state space for states 1 to 15. (2 points)

   Level 1: 1

   Level 2: 2 3

   Level 3: 4 5 6 7

   Level 4: 8 9 10 11 12 13 14 15

   (b) Suppose the goal state is 13. List the order in which nodes will be visited for breadth first search, depth-limited search with limit 3, and iterative deepening search. (4 points)

   Breadth-first search: 1 2 3 4 5 6 7 8 9 10 11 12 13

   Depth-limited search: 1 2 4 8 9 5 10 11 3 6 12 13

   Iterative-deepening search: 1 1 2 3 1 2 4 5 3 6 7 1 2 4 8 9 5 10 11 3 6 12 13

   (c) Would bidirectional search be appropriate for this problem? Why or why not? (2 points)

   Yes. The predecessor function for a node is floor(n/2). The branching factor for searching backward from the goal is 1. The forward branching factor is 2. Since both factors are small and similar, bidirectional search could work. Actually, we will probably be better off just doing the backward search alone since it's branching factor is only 1.

3. Give a description of a search space in which best-first search performs worse than breadth-first search. How many nodes are visited by each strategy in your domain? (8 points)

   Consider a state space in which we have 2 paths from start to goal, and a bad heuristic. The first path is of length n where for each transition the heuristic-estimated cost is 1. The second path has length 1, and estimated-cost $n - 1$. Best-first search will find the first path because at every step the lowest cost step is along path 1 (cost 1 instead of $n - 1$), visiting n nodes. Breadth-first search will find path 2 in either the first or second step as it explores all paths of length 1 first, visiting 1 or 2 nodes.

4. Trace the operation of the A* search algorithm applied to the problem of getting to Bucharest from Oradea, using the map in the text (Figure 3.2). Use the straight-line-distance heuristic values given in Figure 4.1. (8 points)

Priority Queue:

Step	*Expand (Pop)*	*City*	*g*	*h*	*f*
0		*Oradea*	*0*	*380*	*380*
1	*Oradea*	*Sibiu*	*151*	*253*	*404*
		Zerind	*71*	*374*	*445*
2	*Sibiu*	*Rimnicu Vilcea*	*231*	*193*	*424*
		Fagaras	*250*	*176*	*426*
		Zerind	*71*	*374*	*445*
		Arad	*291*	*366*	*657*
3	*Rimnicu Vilcea*	*Fagaras*	*250*	*176*	*426*
		Pitesti	*328*	*100*	*428*
		Zerind	*71*	*374*	*445*
		Craiova	*377*	*160*	*537*
		Arad	*291*	*366*	*657*
4	*Fagaras*	*Pitesti*	*328*	*100*	*428*
		Zerind	*71*	*374*	*445*
		Bucharest	*461*	*0*	*461*
		Craiova	*377*	*160*	*537*
		Arad	*291*	*366*	*657*
5	*Pitesti*	*Bucharest*	*429*	*0*	*429*
		Zerind	*71*	*374*	*445*
		Craiova	*377*	*160*	*537*
		Arad	*291*	*366*	*657*
6	*Bucharest*	*goal reached!*			

5. What algorithms are the following special cases equivalent to? EXPLAIN YOUR ANSWER.

   (a) Local beam search with $k = 1$. (2 points)

   In local beam search with $k = 1$, we would randomly generate 1 start state. At each step, we would generate all the successors, and retain the 1 best state. This is equivalent to hill-climbing.

   (b) Local beam search with one initial state and no limit on the number of states retained. (2 points)

   In this case, we start at the initial state and generate all successor states (no limit on how many). If one of those is a goal, we stop. Otherwise, we generate all successors of those states (2 steps from the inital state), and continue. This is equivalent to breadth-first search.

   (c) Simulated annealing with $T = 0$ at all times (and omitting the termination test). (2 points)

   If T is infinitesimally small, the probability of accepting an arbitrary neighbor with lower value approaches 0. This means that we choose a successor state randomly k times, and move to that state if it is better than the current state. This is equivalent to first-choice hill climbing.

   (d) Genetic algorithm with population size $N = 1$. (2 points)

   The selection step necessarily chooses the single population member twice, so the crossover step does nothing. If we think of the mutation step as selecting a successor at random, there is no guarantee that the successor is an improvement over the parent. This can be seen a random walk.

6. Column Jump is a game played on a grid with balls of different colors. The inital board has some configuration of colored balls with at least one empty space. The objective is to remove all but one ball from the board. Balls are removed when they are jumped according to the following rules. If a ball of one color jumps over a different colored ball to an empty space, the jumped ball is removed from the board. Additionally, if multiple balls of the same color are in a line, they can be jumped and removed together (by a different colored ball, provided that an empty space is on the other side of the line). You can play the game at http://www.2flashgames.com/f/f-354.htm to get a sense of the rules. Be sure to play in Remover mode instead of Color Life mode.

   In this problem, we will consider a version of this game, and you will implement search algorithms to solve it. Our version can use grids of various sizes with different numbers of colors. You will be provided with several example input and output files for testing your implementation. The input format will

be a text file. The first line of the file is the size of the (square) grid. The second line is the number of colors. This is followed by characters representing the colors in each location of the grid, with 0 representing an empty space. For example, an input file might look like this:

```
4
3
1 2 2 1
2 1 3 2
3 1 3 2
0 2 3 0
```

Your output should contain the series of moves required to solve the puzzle. There should be one move on each line, where a move is represented by two locations on the grid with (row,column) numbers. The grid locations follow this convention:

```
(1,1) (1,2) (1,3) (1,4)
(2,1) (2,2) (2,3) (2,4)
(3,1) (3,2) (3,3) (3,4)
(4,1) (4,2) (4,3) (4,4)
```

So, the output file for the input given above looks like this:

```
(2,1) (4,1)
(1,4) (4,4)
(1,1) (1,4)
(4,2) (1,2)
(4,4) (4,2)
(4,1) (4,3)
(4,3) (1,3)
(1,4) (1,1)
```

(a) How many different possible game states are there for the 7x7 version of this puzzle? (2 points)
Acceptable answers:
7^{49}
$7^{49} - 1$ *(excluding the empty board)*
$7^{49} - 1 - 6^{49}$ *(also excluding all full boards)*

(b) Define a state representation. (2 points)
There are many possibilities. One is a matrix with the same format as the input file: one entry for every space on the board, with zeros for empty spaces and a different number for each color.

(c) Give two admissible heuristics for this problem. (5 points)
Some possibilities:

- *The number of different colors on the board minus one.*
- *min(non-empty columns, non-empty rows)*
- *A fixed zero.*
- *Number of lone balls (surrounded by spaces).*

(d) Implement depth-first search to solve the Column Jump puzzle. (15 points)

(e) Implement A* search to solve the Column Jump puzzle. Describe the heuristic you are using for A*. (15 points)

(f) Compare the performance of depth-first search and A* search on the examples provided (testExample1.txt, testExample2.txt, testExample3.txt) in terms of running time and solution length. (5 points)
testExample1.txt has an optimal solution length of 11 moves.
testExample2.txt has an optimal solution length of 8 moves.

testExample3.txt has an optimal solution length of 9 moves.

Your A implementation should have found these optimal lengths. DFS might not have found optimal solutions.*

(g) How would you expect the performance of iterative-deepening search to compare to depth-first search in this domain? (3 points)

IDS could find optimal solutions in cases where DFS does not.

(h) Would bidirectional search be useful in this domain? Why or why not? (3 points)

No, bidirectional search would not be useful because the branching factor of backward search is much higher than forward search in this domain. There are many possible goal states (any board with a single ball of any color in any position) and many states that could lead to those goal states, most of which are not reachable from the start state.