CS571 Final - Jeremy Villegas 12/09/2012 very good including the design 30

**Requirements:**

· Make sure the system uses Best First/A\*.

· Make sure the system returns the solution path.

· It must be testable using g++.

**Section 1 of Your Report = Analysis Questions:**

1. How many unique states:

*32 Unique States*

2. How many unique safe states:

*20 Unique Safe States*

3. List the unsafe states:

*See attached image. Green signifies a safe state, red an unsafe. The start and goal states are also specially marked.*

4. List the actions/operators:

*Each action involves moving a select number of missionaries or cannibals between the banks of the river. Each movement must carry at least 1 passenger. These actions are defined in the generateMoves() function in the Move class.*

* Move: Boat South 2 Missionaries 0 Cannibals (S,2,0)
* Move: Boat South 0 Missionaries 2 Cannibals (S,0,2)
* Move: Boat South 1 Missionary 1 Cannibal (S,1,1)
* Move: Boat South 0 Missionaries 1 Cannibal (S,0,1)
* Move: Boat South 1 Missionary 0 Cannibals (S,1,0)
* Move: Boat North 2 Missionaries 0 Cannibals (N,2,0)
* Move: Boat North 0 Missionaries 2 Cannibals (N,0,2)
* Move: Boat North 1 Missionary 1 Cannibals (N,1,1)
* Move: Boat North 0 Missionaries 1 Cannibal (N,0,1)
* Move: Boat North 1 Missionary 0 Cannibals (N,1,0)

5. Branching factor: <= *Branching factor will always be less than or equal to 5. This depends on how many actors are on the side that you’re moving and if you’re moving into an unsafe state. Total possible moves is 10, but you cannot move the boat North if the boat is already at the North dock so that cuts the possible moves in half.*

6. Can we reach the same state in two different paths? How: *There are multiple solutions depending on how you decide to prioritize. You can always backtrack and undo a previous move and still end up solving the puzzle using a different path.*

7. Can we end up in a loop? How: *There are no forced endless loops but the player could choose to repeat the same actions causing a loop.*

**Section 2 of Your Report = Drawing the Problem Space:**

1. Draw all and only safe/legal states, and label all possible links between them with the above operators to create a graph. Mark the initial and the goal state.(Note: no node should be duplicated)

*See the attached image. There are states that are safe but not reachable. They are not included in the problem space.*

2. Mark all "dead-end" states from which you can only go forward to unsafe/illegal states.

*There are no dead-end states since you can freely move back and forth between states. i.e. If the user encounters a state that leads to an end state, they may back out by undoing the previous move.*

3. Mark the solution path(s).

*There are several solutions to the problem since you can always reach the goal state after taking several steps backwards. An example path is highlighted in the image below.*

**Section 3 of Your Report = Designing the evaluation function:**

1. What is g (actual)?

*Like FWDC, this is 1 since it takes a turn to move units across the river*

2. What is h (estimate)?

*H is the sum of the number of units still waiting to cross to the south and the number of times we have duplicated the state we are in.*

3. Give example g and h values for a couple of states in the above Problem Space.

*When the simulation starts no moves have been mad. Therefore G is 0 and H is 6. We haven’t done anything yet and there are 6 units on the north bank. The starting state is:*

*#########*

*#..MMM..#*

*#..CCC..#*

*#~~~B~~~#*

*#~~~~~~~#*

*#.......#*

*#.......#*

*#########*

*== Expansion Analysis ==*

*Move [1] M and [0] C South: OK*

*Move [2] M and [0] C South: OK*

*Move [0] M and [1] C South: Invalid*

*Move [0] M and [2] C South: Invalid*

*Move [1] M and [1] C South: OK*

*Valid Moves: 3*

*== End of Expansion Analysis ==*

*== Cost analysis ==*

*Cost of moving 1 M and 0 C South is h(n): 6*

*Cost of moving 2 M and 0 C South is h(n): 5*

*Cost of moving 1 M and 1 C South is h(n): 5*

*Best move (Cost 5): Move [2] M and [0] C to the South*

*Cumulative cost g(n): 0*

*== End of Cost Analysis ==*

*Below is the move just before the goal state is reached.*

*#########*

*#..MM...#*

*#.......#*

*#~~~B~~~#*

*#~~~~~~~#*

*#..CCC..#*

*#..M....#*

*#########*

*== Expansion Analysis ==*

*Move [1] M and [0] C South: OK*

*Move [2] M and [0] C South: OK*

*Valid Moves: 2*

*== End of Expansion Analysis ==*

*== Cost analysis ==*

*Cost of moving 1 M and 0 C South is h(n): 3*

*Cost of moving 2 M and 0 C South is h(n): 1*

*Best move (Cost 1): Move [2] M and [0] C to the South*

*Cumulative cost g(n): 40*

*== End of Cost Analysis ==*

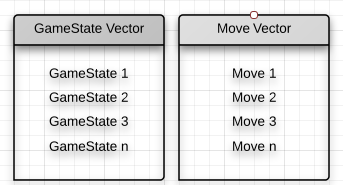
4. Defend your decision for the h. Why is it a good estimate? Does it overestimate in some cases? Explain.

*I think the H I chose is a strong choice. It will never over-estimate the cost of a state since the returned cost will always tell you the minimum amount of moves it will take for you to reach the goal state. Although this choice will not guarantee finding the best path, it will correct itself in the event that F(n) gets too big. Since the problem space is very small, either an overestimation or an under-estimation will not make much difference in the outcome.*

**Section 4 of Your Report = Program Design:**

1. What data structure did you use to store each state? Draw an example.

*Vectors were used to store the history of states and moves to display at the end.*



2. Did you isolate the calculation of g, h and f is some function so that it is easy to change them? How?

*Yes, the calculation of each of the evaluation algorithms are contained in a single function and therefore are easy to change. Since this is a relatively simple problem the calculation for G and H are very simple.*

3. Are the operations/actions hard-coded into your program? Or, can they be changed to a different set of actions easily? Explain.

*The actions are hard coded into the program but can be easily changed. The function generateMoves() should generate all of the possible moves (safe or unsafe) for the problem. The evaluation function considerMove() will throw out any additional moves that puts the player in an unsafe or invalid state.*

**Section 5 of Your Report = Implementation:**

Implement A\* using the above info and submit the program which has lots of comments.

The output from the program should be “the trace” of decisions being made by the program.

e.g. Where are we now? What are the next possibilities? What are their f values? Which one will we expand next?

At the end, the program must output the solution path.

*See bundled attachment.*

**Section 6 of Your Report = Testing:**

Test your program thoroughly and make sure it matches your expectations.

Submit a screen snapshot to prove that your program works.

*Below is a snapshot of the last few states of the program. Please keep in mind that I “reversed” the role of cannibals and missionaries to make it PG. When there are less cannibals than missionaries, the missionaries will “Convert” the cannibals to missionaries and the game will end.*

*Screenshot is below.*

