Eric Bruck

CS 571 – Artificial Intelligence

October 17, 2011

Dr. Guillen

Missionaries and Cannibals

The cannibals and missionaries problem is a river crossing problem consisting of three missionaries, three cannibals, and a boat that can carry at most two passengers. With the initial state of everyone on one side of the river, the goal is to get all the cannibals and missionaries to the other side of the river. The constraint that this problem must satisfy is that on either side of the river, if missionaries are present, the cannibals cannot outnumber them. This seems like a simple problem but it is difficult to find the solution by hand. Many people have a hard time solving this because they do not do a formalization of the problem which leads to many repeated states. For this reason, it is a good idea to keep track of all the previously visited states. By doing so, unnecessary loops are prevented along with infinite loops.

Key:

(X Y : M N)

^

X = Number of Cannibals on left riverbank

Y = Number of missionaries on left riverbank

M = Number of cannibals on right riverbank

N = Number of missionaries on right riverbank

^ = Indicated what side of the river the boat is on

< A B >

A = Number of cannibals to ferry across the river

B = Number of missionaries to ferry across the river

Initial State:

(3 3 : 0 0)

^

Possible Actions:

1. < 1 0 >
2. < 2 0 >
3. < 0 1 >
4. < 0 2 >
5. < 1 1 >

Goal State:

(0 0 : 3 3)

^

Each state in this problem has a maximum of 5 possible actions that can be done. For some states, it may be impossible to perform the action because there are not enough missionaries or cannibals on the current side to ferry to the other. Some possible actions may lead to invalid states, so those actions are void.

Reviewing this by hand I found that all possible paths leading to the goal state will result in the same path cost of 11. For this reason Depth-First Search is the ideal search method for this problem. There is no reason to do Breadth-First Search and expand every node because this will lead to a higher time complexity. Looking at the state space further reveals symmetry in the path to the goal state. This is shown below in the state space diagram.

For this problem I used a double linked list with the Depth-First Search method. A doubly linked list is required so we can travel back up the tree if a state is reached with no possible valid child states. Only one leaf is expanded at a time for each parent node and then that route is expanded. All previously visited states are recorded so that it is never visited again, preventing loops and infinite loops. I did not implement this at first and when I ran the program it was trapped in an infinite loop.

State Space Diagram

Path Cost: 11 trips

