CSCI 5511 ARTIFICIAL INTELLIGENCE 1

Prof. Nikolaos Papanikolopoulos

PROGRAMMING ASSIGNMENT
MISSIONARY AND CANNIBAL SOLVER

Manish Kumar Keshri
ID: 5417554
keshr005@umn.edu

Due Date: 12/07/2017

Problem Statement:

There are 15 Missionary, 15 Cannibal on left side of the river. There is a boat which can carry 6 people at most. The goal is to safely transfer all the missionary and cannibals to the right side of the river. By safely we mean that at any instant like the left side, right side or even on the boat the no. of cannibal should not be greater than the no. of cannibals.

Problem Implementation Description

A* implementation with evaluation function given by,

f(n)= g(n)+h(n)

where,

g(node) = 1 + g(parent node of the given node)

h(node) = (no. of missionaries at the given node + no. of cannibals at the given node)/boat size

g(intial_node)=0

here, node represents the state of missionaries and cannibals at the left bank of river and boat position

1. The input condition is taken as the first node and pushed in a queue.
2. A recursive function to pop out a node from the queue and check if it is the goal node or not. If it is not the goal node, we mark it as visited. If it is the goal node, program is terminated with a call to print moves.
3. At the given node, generate all possible next nodes and push only those nodes in the queue which are previously unvisited and is a valid state.
4. For generating next state, generate all combinations of number of missionaries and number of cannibals such that both are not zero and number of missionaries is more than or equal to cannibal in the boat.
5. Validity check sees if missionaries are outnumbered on the either side of the river.
6. Sort the queue so that whenever a new node is popped out of the queue, it has smallest f value among all nodes in the queue. Quick sort is implemented so that the sorting is faster.
7. To print the path, we back track from the goal node to its parent node and subsequently to the initial node.

Program Functions:

solution**:** function to initialise the first node and call other functions to solve and display the result

print_path: print the different states of solution and boat movement

m_c_sol: recursive function to solve the problem and generate next nodes

sort_states: quick sort implementation to sort the states based on their f-value

partition**:** partition the array, helps in quick sort

isgoal: to check if the given node is a goal node or not

Isvalid: to check the validity of a node

Program variables:

s_n: a structure to store attributes of a node like cost function, state id, parent state id etc.

all_states: stores all non-visited and valid generated nodes
path: stores the sequence of states that led to the goal node
arr_boat: to store the number of missionaries, number of cannibals and boat position
count: to assign state id to a non-visited and valid generated node
goal_state: to define the goal node condition

Coding Guidelines

1) Definitions of each variable
2) Definitions of each function
3) Comments describing each function and variable
6) Proper indentation for readability.

Files Included

1) mxc.lisp : Main source code file
2) mxc_ReadMe.txt : Includes the guidelines to run the source code
3) mxc_Output.txt : Sample output for 15 Missionary, 15 Cannibal, 6 Boat Size

Deficiencies

1. The recursive implementation of sequence of moves occupies a lot of memory in the stack of RAM.
2. Though quick sort implementation of sorting the queue is O(nlogn) in time complexity where n is the number of nodes, time can be significant for larger value of missionaries and cannibals and smaller boat size because the nodes generated at each step will be huge in number.