ASSIGNMENT 1

SUBJECT CODE: CS 6300

SUBJECT: ARTIFICIAL INTELLIGENCE

LEENA KORA
EMAIL:leenak@cs.utah.edu

Unid: u0527667

Documentation and Discussion

1. A short problem description

Explanation: The problem in hand is regarding robot manipulation task. Certain task is given and we had to provide a solution to the problem using certain searches. The senario includes a set of tables and each table has number of blocks piled up on top of it. The blocks are numbered from A to Z. Maximum of 30 blocks can be present on a table and at the maximum 10 tables can be present. The problem includes moving blocks from certain tables to certain particular tables. We are required to use BFS, DFS or ASTAR searches to find the solution. It actually involves changing the given configuration into other required configuration like if the given configuration has three tables with blocks numbered 'A', 'B', 'C' on them respectively, then the required configuration can have all the three on the three table in the order 'A' , 'C' ,'B'. The blocks can be moved only one at a time from the top of the pile on one non-empty table to the top of the pile on other table which might be empty or non-empty. It includes writing a program for the robot to get the required configuration from the given configuration.

This problem is very interesting because it involves simulation of how the robot might do the job with given constraints. Figuring out how the task can done was very tricky and interesting. This problem gave an experience of how to write Aritificial Intelligent programs and also how to make AI systems act in a rational way.

2. A description of the solution.

Explanation: I started out reading the input file and storing the start and goal configuration in a data structure. I defined a class called 'node' which maintained all the required information such as the number of tables present and least cost and heuristic function associated with them. I declared a class in one file named node.h and defined or implemented it in the other file named 'node.cc'. Then the node class has the current state variable which represents it state. I then implemented BFS using queue of STL for the open list implementation. Breadth first search parses the search tree completely through breadth wise. I test all the nodes at each depth and expands them.Then I used stack of STL for depth first search with depth limit implementation. Depth first search goes to the bottom most node and them tracks up for the search. Depth first search with depthh limit goes up to the given depth level and go up for the further search. Both DFS and BFS were blind search. I used priority queue for the A-star search and overridden the '<' operator for the processing. A-star is a heuristic search algorithm. It is informed search. A-star search the goal based on certain evaluation function. The evalution function here the given problem was the summation of the least cost path along the search path and the heuristic function was the calculation of the value by adding (n-p) where 'n' is the number of blocks on the table and 'p' is the position of the block on the table.

3. A discussion of the problems I encountered in solving the problem and how I overcame them, the weaknesses and strengths of your solution, and any other interesting observations.

Explanation: In case of BFS, my program had segmentation errors at some places. I was accessing the memory location which deleted it earlier. I was using vectors of STL. I represented a table by vector of chars and a whole state by vector of vector of chars. As '=', '!=' were not defined on the data structure i defined I had to implement them. I used to have "bad-alloc" segmentation fault while dealing with vectors. I used gdb debugger for debugging and I was very carefull while using vectors in the for loops.

BFS implementation of mine is very accurate and robust. It gives better search efficiency and tracing information. DFS implementation is also very robust and correct. It gives good trace and status information. My A-star implementation i s not accurate and I tried to figure it out but beacause of time restrictions I could not.

Testing information

Explanation: I tested my program with various test cases. following are the test cases for my program.

- First I tested my BFS implementation by given the same state as start and goal state. It gave the comment as " Start state itself is the goal state".
- Then I used another example for testing my BFS, and DFS.
- Then I tested by not giving an invalid option. My program gave the output as

  "-j: unknown option
  Sorry, You have pressed an invalid option. Type executable file with '-usage' or '--help' option to get the option list back

- When the user gives the option '-g' that is 'alth' with either BFS or DFS, my program gives the comment like the following

  " Specify the option 'g' only with a-star"

- When the user gives DFS option without giving the depth limit, my program gives the message llike invalid option chosen.

Answers to additional discussion questions

1. Were the statistics for search efficiency (`-stats` option) as you expected when comparing search using BFS, DFS, and A*?

Explanation: Yes, the statistics for search efficiency were as expected when comparing search using BFA, DFS, and A*. When the A* output was compared with the BFS and DFS, it was better than the two. For some inputs the DFS gave good results than the BFS which parses all the nodes in the depth level. In some of my test cases, for some input DFS was gave better solution than BFS. But for some test cases DFS gave worse solution than BFS when the required node was in the rigth most part of the search tree.

2. Would you expect to get the same answer if you reversed the start and goal states?

Explanation: No, because the path chosen may be different and can have different intermediate states. So the statistics will differ. If the earlier goal state has fewer blocks then new goal state migth not be reached through the same intermediate nodes.

3. Is the heuristic function admissible? Is it consistent?

Explanation: Yes, it is admissible heuristic function. It does not over estimates the cost. It calculates the value based on the blocks on the tables of the goal state and their positions. No it is not consistent. Not all admissible heuristic function are consistent.

4. Is the heuristic function guaranteed to produce the shortest solution?

Explanation: Not always. Many heuristic functions suffer from a plateau effect. In such cases, A* degenerates into something approaching BFS. sometimes the heuristic functions over estimates and might not be admissible.

For our problem it gives the shortest solution compared to BFS and DFS but not guaranteed to produce the shortest solution..

5. Is there a possibility of a tie when selecting which node to expand next using the A* algorithm with the heuristic specified above? If so, would it affect the final answer?

Explanation: Yes, there is a possibility of a tie when selecting which node to expand next using A* algorithm with the given heuristic. The final solution might be same but the statistic will be different beacause of the generation of different intermediate states.

6. What issues are involved in creating a good heuristic function for this problem? How good is the default heuristic. What evidence can you provide that the alternative heuristic function (`-alth` option) as better than the default heuristic?

Explanation: Keeping track of relative positions of the blocks of start and goal state is one of the issues involved in creating a good heuristic function for our problem. Default heuristic is good and it does not over estimates the cost.

7. Sometimes there is no good evaluation function for a problem, but there ia s good comparison method: a way to tell whether one node is better than another,without assigning numerical values to either. Show that this enough to do a best-first search. Is there an analog of A*?

Explanation: Some robots have cameras in them for pattern recognition. They create a contour map of the patterns and when ever they come across the pattern again they compare the contour maps for the pattern recognition using the best first search algorithm.

I do not think we can do this using A* because it might need numerical comparisons and it needs heuristic and the least cost function both for the comparison .