CSL302 Artificial Intelligence
LAB 4

Abhishek Chowdhry
2015CSB1002

April 2017

Forward Planner using Breadth First Search

A forward planner searches through the state space graph, starting from the initial state looking for a state which satisfies the goal state description. The search complexity is defined by the branching factor which depends the number of actions available at each node. The number of actions available may be different for each node depending on whether the preconditions of a given action are satisfied in that state. Once we have a state with a valid action, the new state can be generated by adding the positive effects(predicates) of the action and removing the negative effects(predicates) of the action from that state.

So our search space is a tree. In BFS we search through this tree in such a way that we explore all the nodes at a particular depth before moving onto the next level. As a result the Goal state which we get from this search is optimal(considering path length 1 for each edge). But this strategy also explored many unwanted states due to which the time complexity increases exponentially with depth. Due to these reasons the solution length which we got from this strategy was minimum but it explored a large number of unwanted states. Due to this reason it is not able to solve 5.txt and 6.txt even in 2-3 hours.

The time taken, solution length and the number of nodes expanded for different initial state and goal state descriptions is given below:

1.txt or 2.txt (Description-1)

Search Time: 0.026 seconds
Solution Length: 10
Number of nodes expanded: 112

3.txt or 4.txt (Description-2)

Search Time: 0.083 seconds
Solution Length: 14
Number of nodes expanded: 614

Forward Planner using AStar Search

In this search algorithm the forward planner uses the AStar algorithm for searching through the state space graph. If a large number of actions are available at a particular node, then it chooses that action which leads to a state with the minimum value of the sum of heuristic function and the actual path cost till the given state. So in this way it explores the most relevant states first(due to heuristic) before exploring the irrelevant states. For this a good heuristic is needed. My heuristic is described below:

The heuristic takes as input the goal state and the state at which the heuristic value is to be found. Then it compares the two states predicate by predicate and takes those predicates which are present in one state and absent in the other. A total of 5 predicates types are possible out of which I have considered only the types on(x,y) and clear(z) because they are the most significant ones (because once they are equal in both the states, then the goal state is not far away from the current state). Then I have assigned them weights: 10 to on(x,y) and 1 to clear(z), and I have taken up their sum as my heuristic value. So my heuristic is a short and effective heuristic but it is not admissible(the question doesn't says anything about admissibility of the heuristic). As a result there is a significant decrease in the number of nodes expanded and the time taken but the solution length obtained is not optimal for all cases.

The time taken, solution length and the number of nodes expanded for different initial state and goal state descriptions is given below:

1.txt or 2.txt (Description-1)

Search Time: 0.020 seconds
Solution Length: 10
Number of nodes expanded: 13

3.txt or 4.txt (Description-2)

Search Time: 0.022 seconds
Solution Length: 16
Number of nodes expanded: 22

5.txt or 6.txt (Description-3)

Search Time: 0.168 seconds
Solution Length: 30
Number of nodes expanded: 160

Goal Stack Planner

For this part we implemented the goal stack planner for solving the given problem. Goal Stack Planning integrates the advantages of both forward planning and backward planning. An action is added to the plan only if all its preconditions are satisfied. If any precondition is not satisfied, then we add a relevant action for that precondition and push its preconditions on the stack and repeat the same process. Since there are more than one relevant actions for some predicates, we have to choose the relevant action wisely.

Since Goal Stack planning tries to choose only the relevant action starting from the goal state, the time taken by goal stack planning is small compared to the other two search strategies. But it does not aways give the optimal path. It only gives the optimal when the predicates are pushed in the stack in a serializable order(Sussman's Anomaly). Since the order is not known beforehand, it doesn't always give the optimal solution.

The time taken, solution length and the number of nodes expanded for different initial state and goal state descriptions is given below:

1.txt or 2.txt (Description-1)

Search Time: 0.019 seconds
Solution Length: 14

3.txt or 4.txt (Description-2)

Search Time: 0.020 seconds
Solution Length: 26

5.txt or 6.txt (Description-3)

Search Time: 0.022 seconds
Solution Length: 36

Conclusion

From the above observations we can conclude that:

- Forward Planning with Breadth First Search is complete and optimal because because it searches through all the states and gives the best possible solution. But it expands a large number of irrelevant nodes and hence takes the longest time.
- Forward Planning with AStar Search is complete but not optimal because it tries to expand the relevant nodes before the irrelevant ones and hence finds the solution quickly with the expansion less number of nodes but the solution length may not be minimum. We had used an inadmissible heuristic which made our AStar suboptimal, but if we use an admissible heuristic, AStar Search will be optimal. So AStar Search finds the solution quickly with the expansion of less number of nodes but the solution is suboptimal.
- Goal Stack Planning is not complete as in some cases it may not find a solution even if one exists(although in our case it solves all the .txt files). Also Goal Stack Planning is not optimal(as explained above- Sussman's Anomaly). But since it tries to always choose the relevant action starting from the goal state, whenever it finds the solution, it takes very less time compared to the other two search strategies.