COL 864 - AI for Cognitive Robot Intelligence
Homework 2

Nilaksh Agarwal, 2015PH10813

April 5, 2020

Planning Algorithm

Our object space is defined as:

$$\{'apple','orange','banana','table','table2','box','fridge','tray','tray2','cupboard'\}$$

- {'apple', 'orange', 'banana', 'box', 'tray', 'tray2'} are objects the robot can pick up
- {'table', 'table2', 'box', 'fridge', 'tray', 'tray2', 'cupboard'} are surfaces for other objects to be placed on them
- {'fridge', 'cupboard','box'} are enclosures, i.e., objects can be placed inside them
- {'fridge', 'cupboard'} are state objects which can be opened and closed by the robot

Our action space is defined as:

$$\{'moveTo','pick','drop','changeState','pushTo'\}$$

- {moveTo, object} - moves robot close to object. Here object can be any object in the world
- {pick, object} - picks the specified object. Here object can be any object the robot can pick up
- {drop, destination} - drops a grabbed object to destination object. Here destination cab be any surface object or enclosure.
- {changeState, object, state} - changes the state of an object (open or close). Here object can be any state object.
- {pushTo, object, destination} - pushes object close to the destination object. Here object can be any object the robot can pick up and destination cab be any surface object or enclosure.

A state is defined by:

- state'grabbed' - object currently grabbed by the robot
- state'fridge' - fridge state in (Open/Close)
- state'cupboard' - cupboard state in (Open/Close)
- state'inside' - consists of pairs of objects (a,b) where object a is inside object b
- state'on' - consists of pairs of objects (a,b) where object a is on top of object b
- state'close' - list of objects close to the robot

So, the domain of this planning problem is the set of all possible states, and the actions that can transform a state to the other.

Planning Strategies

Forward Planning

Here, we start from the initial state and perform actions to traverse the state space to find a state which satisfies the goal constraints. In a default BFS search, there are approximately 70 possible actions that the robot can take (some might not be allowed in the current state). Hence, the branching factor is around 70 for a default BFS forward planner.

This default BFS planner is quite slow, with a lot of goals/worlds being unable to find a plan within the 120 second deadline.

This state space search operates without much planning and it has quite a high branching factor. However, Forward planning is sound (if a plan is returned, it will indeed be a solution) and it is complete (if a solution exists, it will be found).

Backward Planning

Here, we start from the goal constraints and work backwards towards the initial state. Since the goal constraints are not a complete state, we take them to be the ensemble of all possible combination of states which satisfy those constraints. We define inverse actions and work from this incomplete state back to our goal state. Generally, we perform a lifted backward search, where some variables are only assigned at the end of the plan, remaining unknowns throughout.

For backward search, we define a heuristic function to signify the distance to our initial state, and use this in our A* search.

Backward planning, since it unifies only actions which can satisfy the current goal constraints, have significantly lesser branching factor. However, due to the incompleteness of the goal and all subsequent state spaces, it is difficult in some cases to find a path from this inexact goal state.

Improvements to Planner

Since as per the Github Repo, a basic task was to come up with a planner that generates a plan under 60 seconds for all world-goal pairs, the following modifications were suggested:

- Use a dictionary to store the visited states of my BFS (reduce it to constant time). This is done through converting the state dictionary to a json string (after sorting) which results in a significant speedup
- Instead of going over all possible actions for our BFS search for every node, we only consider the subset of actions involving the objects present in the goal (as well as the enclosures they might be inside). This results in a significant reduction in the branching factor (<0 for most goals/worlds)

World/Goals	Goal0	Goal1	Goal2	Goal3
World0	0.01	0.17	0.01	0.13
World1	0.03	0.08	0.08	0.29
World2	0.10	0.11	0.09	0.16
World3	0.17	0.54	0.06	0.43
World4	0.01	0.34	0.05	0.06

Table 1: Timings for best planner (sec)

References

[1] Russel, S., Norvig, P., et al. Artificial intelligence: a modern approach. 1995. *Cited on 20* (1994), 90020–9.