COMP3702 Assignment 1 Report

Team Pied Piper

Alec Bassingthwaighte - s4395541
Sunny Lee - 43905513
Qishi Zheng - 43759453

Introduction

For this project, the task required a simplified prototype implementation of a robotic box mover to discover optimal paths to move boxes inside a fixed work space, with the theme being building refurbishment. The requirement of the robot itself was that it must be able to push two-dimensional boxes from one location to their goal positions. Within the given room space there were multiple obstacles, some moveable and others static, such that the robot could not move them.

The robot was defined as a two-dimensional point in the room and could only push one obstacle at any given time. To push a box or moveable obstacle, the following requirements existed:

1. The robot must first place itself, such that at least three-quarters of its length coincides with a side of the box or obstacle
2. Once in the given position, the box/obstacle follows the motion of the robot as long as at least three-quarters of the robot's length coincides with the side of the box or obstacle
3. If the robot does not satisfy these requirements, the robot and the box or obstacle remain in place
4. The robot can only move one box or moveable obstacles at a time.

The resulting program was required to output a collision-free path for the robot to push the moving boxes to their respective goal locations. A collision-free path means that the robot, all boxes and all moveable obstacles do not collide with one another, nor with any static obstacles.

1. [5 pts] Please define the agent design problem for this movers prototype.

The agent reads a problem specification, containing the initial location of the robot, all movable boxes, movable obstacles and static obstacles and determines a path that the robot has to make to push all the moveable boxes and movable obstacles to get the former into their designated end position.

2. [10 pts] Please describe your search method at the conceptual level (i.e., pseudo code and what abstract data structure is used for the container). If you use sampling-based method, please describe the strategy you apply or develop for each of its four components. Otherwise, please describe the details of your discretization method.

This solution implemented the Probabilistic Roadmap (PRM) method to build a state graph, itself using BFS (breadth-first search), the Rapidly exploring Random Trees (RRT) method as well as the A* search method to find the optimal path for a robot to reach a box.

PRM is a state graph which only captures the “important parts” of the state space. In this context, random samples were taken from the state space in the form of vertices and added to the state graph of the room if it lies in the previously determined C free space. Between vertices, random samples were again taken and used to form edges between them if no collision with a box or obstacle exists between the given vertices. As a result of this, a “roadmap” of possible paths was created for the robot to move around.

If a path between the robot’s initial location and goal exists, then a BFS search is performed to plan a path for the robot in the state graph. With branching factor as b and the depth of the shallowest goal node d, then the time complexity and memory required for this algorithm are both O(b^d).

Compared to a Depth-First Search (DFS), it was decided to use BFS because although the search itself requires exponential space, it will always find the minimum step path, a step being a step between two vertices.

After this, the A* search method is used to plan the paths of boxes which are to be moved. An A* search takes an initial node and a goal node. Nodes are backwards linked containers that stores the information of its X and Y coordinate, its parent coordinate, its neighboring coordinates, and its F cost. The initial node is added to a priority queue called “opened” where the queue prioritizes the node with the lowest F-cost. F-cost is the sum of movement cost from initial node to current node and the estimated cost from current node to goal node. In the event where there are multiple nodes with the same lowest F-cost, the newest added node is prioritized. Once the prioritized node is popped out of the “opened” list, its neighbouring nodes are discovered and added into the “opened” list if the nodes themselves have not been added to the “opened” list before and also if it is not being occupied by an obstacle. For each of the neighboring nodes we also check if it is the goal node. If it is the goal node, a path that connects from the initial node to the goal node is returned by backtracking through the nodes by getting their parent coordinates. Otherwise, the parent node is added to the “closed” list which keeps track of all the nodes that has been popped out of “opened” list before.

3. [12.5 pts] Which class of scenarios do you think your program will be able to solve well? Please explain your answer.

The implementation of this robotic mover has an advantage with small configuration space, as since the PRM method and RRT methods have an increasing computational complexity as the amount of nodes increase, smaller configuration space will naturally run much faster. There is an element of randomness given how the PRM method works in taking random samples of the state space. As a result, even with the same scenario the program may find paths quickly while at other times will take much longer if the initial space is poor.

To cover the inconsistency of PRM and RRT, A* algorithm was implemented to plan the path of moving boxes. With the implementation of A* algorithm, the program excels at finding the most cost effective paths which means the path that the robot takes will usually be very efficient. This is due to the nature of A* algorithm where the selected nodes to explore are based on a priority that prioritizes nodes with the lowest F cost. This also means the output path is the same every time which will consistently produce the most efficient result.

4. [12.5 pts] Under what situation do you think your program will fail? Please explain your answer.

Given a problem with large C space, the program may struggle when performing the BFS due to the exponential space requirement. If there are enough vertices in a state graph, the program can run out of memory thus giving no result at all.

Furthermore, for the A* algorithm, Manhattan Distance is chosen as the calculations for approximating heuristics. This means that it will be very inefficient on searching for a valid path if the path from initial node to goal node is filled with obstacles. This is most prominently seen when the path starts off going opposite direction of the goal node. This is because A* algorithm will prioritize going towards the goal node only to discover a dead end.

Additionally, due to how A* algorithm is implemented, if no valid path exists for the box to move to its goal position, it will take until every single node has been explored to return a null path error. Which means the program will not only fail, it also will take up a large amount of time before failing.

The program will also fail under the situation where there requires a prioritization in which moving box to push first. Because the algorithm used by the program does not account for moving boxes blocking the path of other moving boxes to their goal. This also leads to the problem of moving obstacles blocking the path of a moving path to its goal. Because the implemented A* algorithm does not have a method of checking if a moveable obstacle is blocking a path, the program will also fail or return no valid path found if there are any moveable obstacles blocking the path.

In conclusion, the created program is capable of performing most of the tasks specified in the problem specification with the application of multiple search algorithms to achieve this. Using PRM, RRT, BFS and A* algorithms which compliments each other’s inconsistencies, the program is able to maximise its ability to search for a solution to the problem specification.