Multi Agent Path Planning

Project By :

1. Sanskar Singh - 19CH10047
2. Praveen Kumar Koram - 17HS20028

Index:

Problem:

[Multi Agent Path Finding] Consider a warehouse and a set of robots which pick up items from designated places, deliver those in desired locations, and finally the robots go to the end location. Assume that the warehouse is represented as a 2-D grid of size n × m. Each location (Li) on the warehouse floor can be denoted by a pair of integers Li = (xi, yi). Each cell on the

grid can be categorized in one of the following ways (see diagram below) - source location (P1-P3), destination location (D1-D3), temporary storage location (TS1-TS4), obstacle (black square), normal (rest of the cells). Source & destination denote pick-up and drop locations respectively. Temporary storage location denotes the place where robots can keep some items. Obstacle presents a location where no robot can move. Rest of the cells are considered as normal cells.

Initial thought process:

Initial inference from the problem statement was that it is similar in nature to the single agent path finding problem that we had discussed earlier in class. This problem, however, is greatly higher in difficulty as multiple agents need to function, being aware of the position of the other agents functioning concurrently. The search space is exponentially larger as every branch can expand due to a move by any robot agent.

Due to the large search space, it is important to implement an algorithm that is sufficiently fast to solve these problems. Such algorithms are often needed in facilities like amazon, where such bots run to deliver objects to their specific aisles so it is important that they perform well in real time.

Setup and Project Description:

Language of choice: Python

Algorithms used:

- Space-Time A* Algorithm
- Prioritized Planning
- Conflict Based Search

Project Files:

- indepedendant.py: includes code to find independent paths of each robot, acting as if paths of other robots do not conflict.
- prioritized.py: includes code to solution using prioritized planning; which is incomplete and suboptimal
- runexpiremts.py: includes code to generate results and to run experiments on the code files

- cbs.py: includes code to solution using conflict based search which is complete and finds the optimal solution.
- visualise.py: includes code to visualise the findings in terms of a short animation
- single_agent_planner.py: includes code to plan the path of a single agent using space time a star algorithm.

Constraints:

There are some key differences in this problem from the traditional problem of single agent path finding. The main difference being some constraints in the motion of a particular bot introduced due to the motion of all other bots over the given terrain. These constraints are explained below:

1.) Vertex Constraints:

A bot cannot occupy a cell which is already occupied by another bot moving in the plane.

2.) Edge Constraints:

Two bots in consecutive gridpoints cannot simply switch places in the next time step.

Space Time A* Algorithm:

Conventional A* algorithm searches over the distance considerations only but in this problem it is crucial the algorithm is aware of the time considerations as well.

There are going to be constraints which are dependent on the timestep of the situation, hence the time variable needs to be taken into account.

Each node in the map which is visited by the robot needs to have a temporal variable associated with it.

As we search over the space, we can see if the robot is allowed to be in a cell at a given moment in time. If it is not permissible for the robot to be in some position. That branch in the search space can be pruned.

Hence this modified A* algorithm searches over the search space keeping in mind constraints.

Independent Planning:

While implementing independent planning. The constraints introduced due to multiple robots on the motion of each robot are ignored. The path of every robot is planned as if other bots do not exist. This algorithm returns a state space in which all the robots have a path to their goal which might intersect at a given time. The problem after this implementation is to resolve this configuration such that no paths collide.

Prioritized Planning:

Every robot has a goal destination. In prioritized planning the paths of robots are prioritized in order. The path of the robot having the highest priority is planned independently from others. When planning the path of the robots which follow; the first robot is treated as a dynamic obstacle. What this means is that the plane already has static obstacles, but the first moving robot can be treated as an obstacle whose position changes with every time step. As the priority list of robots are explored, the number of such dynamic obstacles increases. Prioritized planning is suboptimal and incomplete. It is incomplete because it starts with a priority order of bot planning, which almost always results in a suboptimal solution to the problem. What is worse is that it is possible that the algorithm never returns a solution as the initial priority order can be so that the planning of bots further down the line becomes impossible.

Conflict Based Search: