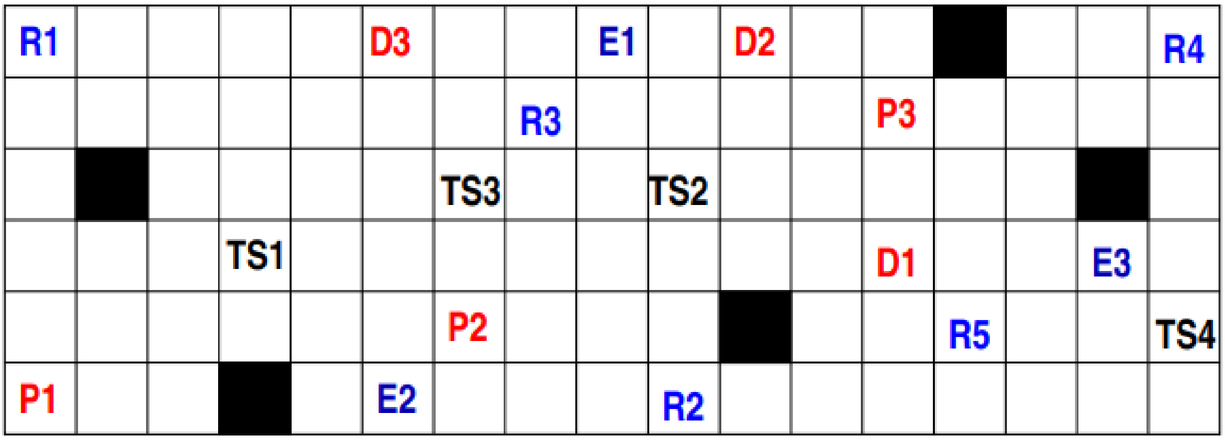
AIFA Assignment 1 Report

Problem Statement

[Multi Path Finding Agent] 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 represent a location where no robot can move. Rest of the cells are considered as normal cells. Let there be k number of robots and r number of tasks. The details of robot location and tasks are provided as per the following table.

Robot	Location		Task	Location	
	Init	Final		pick-up	deliver
Robot-1	R1	E1	Task-1	P1	D1
Robot-2	R2	E2	Task-2	P2	D3
Robot-3	R1	E2	Task-3	P1	D3
. . .	. . .	. . .	. . .	. . .	. . .

Assume that a robot can move at most one cell (either vertically or horizontally) at a time step, a normal cell can be occupied by at most one robot. Source, destination, temporary storage locations can accommodate multiple robots simultaneously. Our target here is to develop a work schedule that minimizes the time to complete all tasks. You need to develop both optimal as well as heuristic algorithms.



Approach for the problem
At first, taking input as a number of robots, their location points and end points, number of pickup points, their location points and their destination points. In the order of the input, or any order, maintain a fixed order with indexing.
Our first task is to choose the nearest pickup location for a particular robot.

Method for choosing nearest pickup points for the robots
As we have the robot locations and pickup locations.

- Function Robot_Pickup does the task. Create a data type with a variable containing 3 values
- For each element have 3 values, store the value of minimum distance between a particular robot and a pickup location in 1st one, store the index value of the robot in 2nd one, store the index value of pickup location in the 3rd value.
- Now, after filling for all robots with all pickup locations. Sort the values and obtain the minimum paths for all robots with particular pickup location.
- Note that, while obtaining the minimum path for a robot to a particular pickup location, avoid the problem of allocating 1 robot to 2 or more pickup and 1 pickup to 2 or more robots by comparing the index values stored in each variable.

Now, depending on the number of robots and number of pickup points, we have 3 cases.

1. Number of robots equal to Number of pickup locations.
2. Number of robots greater than Number of pickup locations.
3. Number of robots less than Number of pickup locations.

Case 1 : Number of robots equal to Number of pickup locations

- For this case, number of variables required for calculating the minimum paths is (number of robots)*(number of pickups).
- After finding the minimum paths, there will not be any robots or pickup locations which are left behind.
- For every robot, there is a particular pickup location, eventually a destination location and an end location.

Case 2 : Number of robots greater than Number of pickup locations

- Here, as the number of robots are greater than the number of pickup locations. Some robots are left behind without having pickup locations.
- So, after finding the distance of paths for each robot and pickup points and taking the minimum paths for respective robot and pickup location.

- These robots have a pickup location, eventually a destination location and an end location.
- The left out robots will eventually go to the end points directly.

Case 3 : Number of robots less than Number of pickup locations

- Here, as the number of robots are less than the number of pickup locations, robots have to take a repetition of going from pickup to destination and again to another pickup and to its destination.
- After finding the minimum paths, for an equal number of robots, those number of pickup points will be choosed. For these, they will be reaching the destination points respective to pick up locations.
- Then again the robots from destination points will be checked for paths with the remaining pick up locations. Again minimum paths were taken and robots are allotted with new pickup points.
- They will be reaching the destination points. Again from there to other pick up points to destination points, until there will be no pick up points left.
- Then finally the robots will reach the respective end points.

As we know which robot will go to which pickup, now we need to find the path and store it.

Method for finding shortest path between two points by using A* search

- First we check whether the given points are valid points or not.
- Create a closed list and make it empty which means no cell has been included yet.
- Create a class to keep the information of cells. The information needed are its parent cell(which it came from), cost from source(h), cost from destination(heuristic function(f)), total cost.
- Create a storage class for storing cells and its details.
- Create an open list and insert the source in the list.
- Remove the cell in the open list and add it to the closed list.
- Open the cells connecting to the cell that is removed.
- Check the cell whether it is the destination or not.
- If it is then the set of cells in the closed list will be the path. If not then repeat the above three processes.
- Use heuristic function(f) to determine the cell that needs to be taken first in the open list. Choose a cell with minimum total cost given by g = h+f.
- By using the tracePath() function we will get the path from storage class.

Heuristic function
It is the minimum distance between two points in the grid. For two points p1,p2 the value of f is the sum of absolute difference of x & y coordinates of p1,p2.

We need to store all the paths of robots for calculation of time and may be useful for further operations like minimizing time, conflict checking and solving etc.

As we got the paths we need to check whether collisions are happening or not as a normal cell cannot occupy more than one robot.

Method for checking collision/conflict between the robots

- We will check two paths whether they have a collision or not, by checking the points where the robots are present which are at a particular time.
- Like this we check for all paths by taking any two paths.
- We use a conflict function to check for collisions whether the point is a collision or source point or destination point or temporary storage point.
- if the point is a collision, then it prints the position of collision and which robots are colliding.
- The conflictCheck function will check whether the point is a source point or destination point or temporary storage point.

To resolve conflicts

- Where there is a conflict/collision we keep a wall/barrier in the grid for that specific robot and we apply the search method and find its new path. And we check for conflicts again and before that we remove the wall placed in the grid.
- Time elapsing on a robot involved in a collision by making it stop for a time step before the collision point.

Example
Sample Input :

```
Enter number of robots and pick up locations : 3 4
Enter Robot locations :
0 0
5 9
1 7
```

```
Enter Source/pick up locations :
4 6
5 0
1 12
4 16
Enter Destination/drop locations :
3 12
0 10
0 5
5 2
Enter End locations :
0 8
5 5
3 15
Enter number of Temporary storage locations : 3
Enter Temporary storage locations :
3 3
2 9
2 6
```

Output :

```
r(0,0) -> p(5,0) The destination cell is found
The Path is -> (0,0) -> (1,0) -> (2,0) -> (3,0) -> (4,0) -> (5,0)

r(5,0) -> d(0,10) The destination cell is found
The Path is -> (5,0) -> (4,0) -> (3,0) -> (2,0) -> (1,0) -> (0,0) -> (0,1) -> (0,2) ->
(0,3) -> (0,4) -> (0,5) -> (0,6) -> (0,7) -> (0,8) -> (0,9) -> (0,10)

r(5,9) -> p(4,6) The destination cell is found
The Path is -> (5,9) -> (4,9) -> (4,8) -> (4,7) -> (4,6)

r(4,6) -> d(3,12) The destination cell is found
The Path is -> (4,6) -> (3,6) -> (3,7) -> (3,8) -> (3,9) -> (3,10) -> (3,11) -> (3,12)

r(1,7) -> p(1,12) The destination cell is found
The Path is -> (1,7) -> (1,8) -> (1,9) -> (1,10) -> (1,11) -> (1,12)

r(1,12) -> d(0,5) The destination cell is found
The Path is -> (1,12) -> (0,12) -> (0,11) -> (0,10) -> (0,9) -> (0,8) -> (0,7) -> (0,6)
-> (0,5)
```

```
r(3,12) -> p(4,16) The destination cell is found
The Path is -> (3,12) -> (3,13) -> (3,14) -> (3,15) -> (3,16) -> (4,16)

r(4,16) -> d(5,2) The destination cell is found
The Path is -> (4,16) -> (4,15) -> (4,14) -> (4,13) -> (4,12) -> (4,11) -> (3,11) ->
(3,10) -> (3,9) -> (3,8) -> (3,7) -> (3,6) -> (3,5) -> (3,4) -> (3,3) -> (3,2) -> (4,2) ->
(5,2)

r[0] (0,10) -> e[0] (0,8) The destination cell is found
The Path is -> (0,10) -> (0,9) -> (0,8)

r[1] (5,2) -> e[1] (5,5) The destination cell is found
The Path is -> (5,2) -> (4,2) -> (4,3) -> (4,4) -> (4,5) -> (5,5)

r[2] (0,5) -> e[2] (3,15) The destination cell is found
The Path is -> (0,5) -> (0,6) -> (0,7) -> (0,8) -> (0,9) -> (0,10) -> (0,11) -> (0,12)
-> (1,12) -> (1,13) -> (1,14) -> (2,14) -> (3,14) -> (3,15)

Path of the robot 1
(0,0) (1,0) (2,0) (3,0) (4,0) (5,0) (4,0) (3,0) (2,0) (1,0) (0,0) (0,1) (0,2) (0,3) (0,4)
(0,5) (0,6) (0,7) (0,8) (0,9) (0,10) (0,9) (0,8)  time - 22

Path of the robot 2
(5,9) (4,9) (4,8) (4,7) (4,6) (3,6) (3,7) (3,8) (3,9) (3,10) (3,11) (3,12) (3,13) (3,14)
(3,15) (3,16) (4,16) (4,15) (4,14) (4,13) (4,12) (4,11) (3,11) (3,10) (3,9) (3,8) (3,7)
(3,6) (3,5) (3,4) (3,3) (3,2) (4,2) (5,2) (4,2) (4,3) (4,4) (4,5) (5,5)  time - 39

Path of the robot 3
(1,7) (1,8) (1,9) (1,10) (1,11) (1,12) (0,12) (0,11) (0,10) (0,9) (0,8) (0,7) (0,6) (0,5)
(0,6) (0,7) (0,8) (0,9) (0,10) (0,11) (0,12) (1,12) (1,13) (1,14) (2,14) (3,14) (3,15)
time - 27
Total Time: 88

no conflict found
```

Group Members

1. **Medicharla Venkata Sri Ram - 19MA20027**
2. **Ayush Dongre - 19MA20008**
3. **Abhinek Kumar Pandey - 18HS20047**