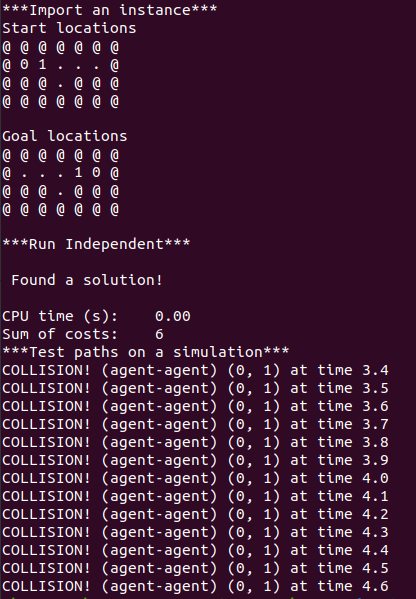
## CMPT417 individual project report

* 1. Searching in the Space-Time Domain



* 1. Handling Vertex Constraints

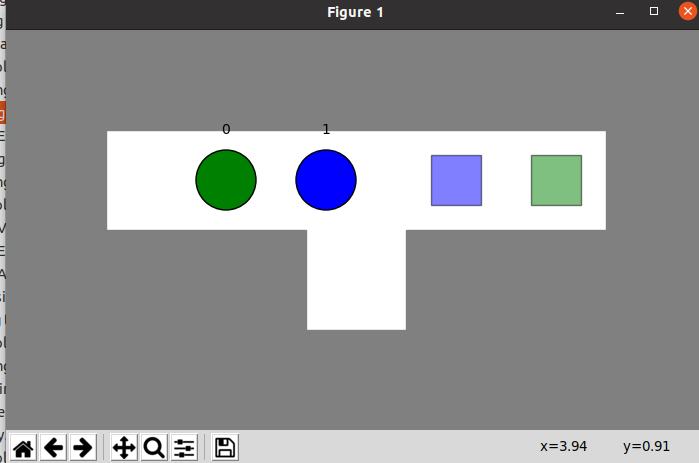
Agent 2 stay on (1, 4) for 1 timestep at time step 4, the constraint is:

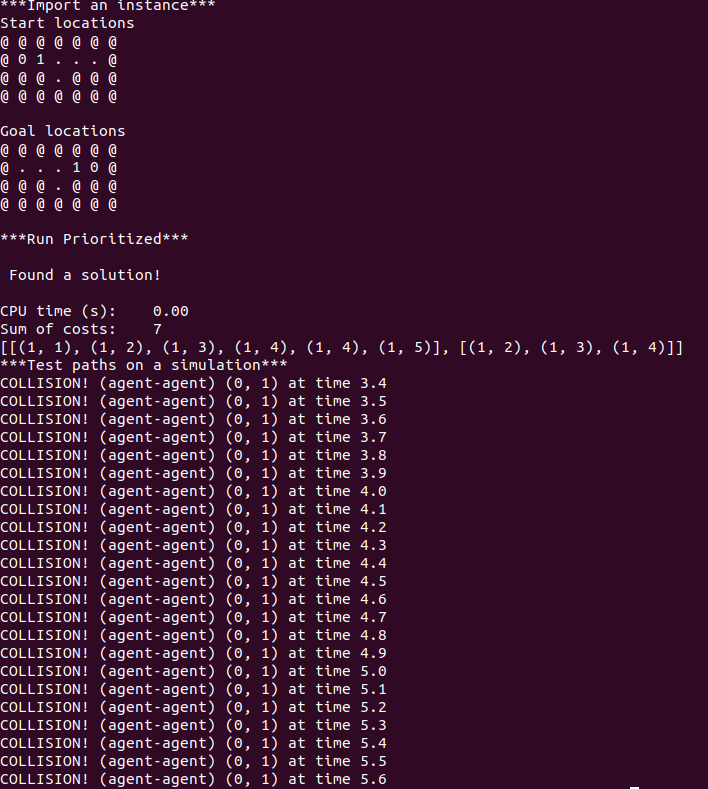
{

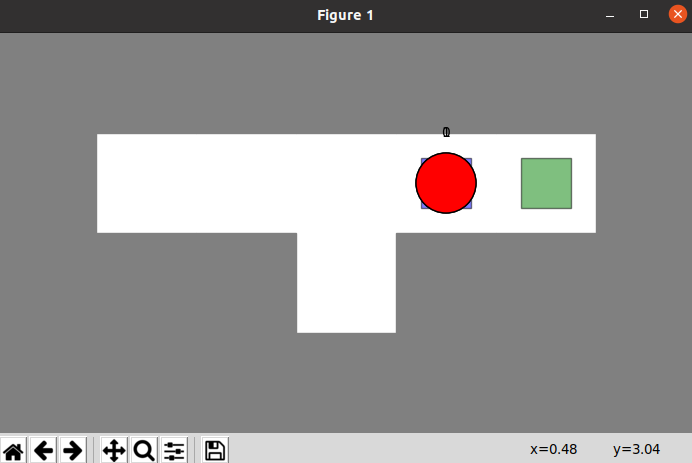
‘agent’: 0,

‘loc’: [(1,5)],

‘timestep’: 4

}





* 1. Adding Edge Constraints

constraint is:

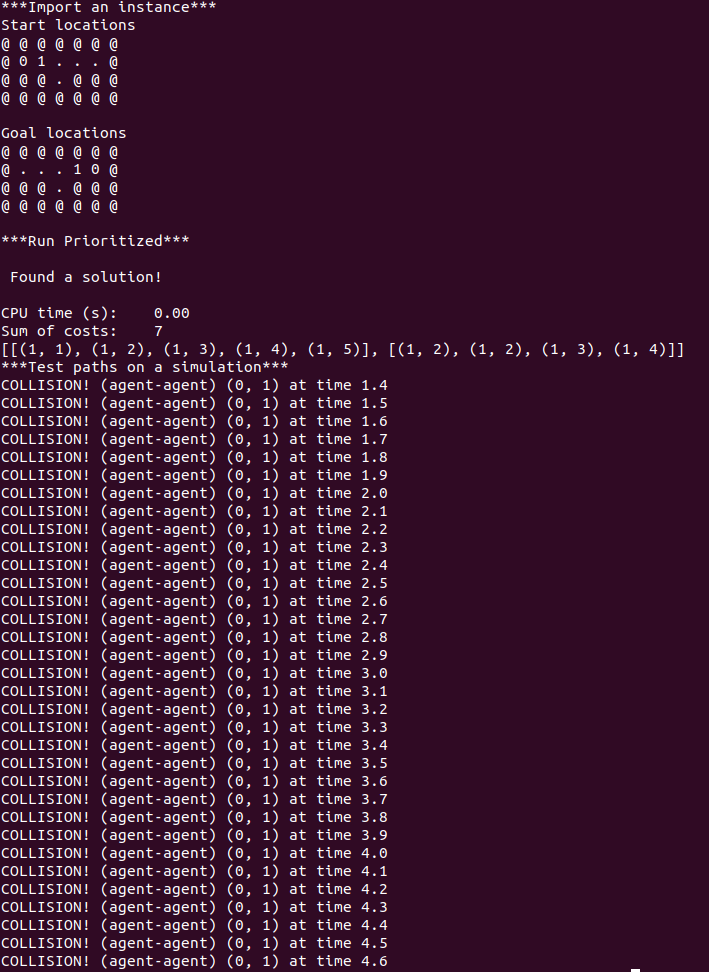
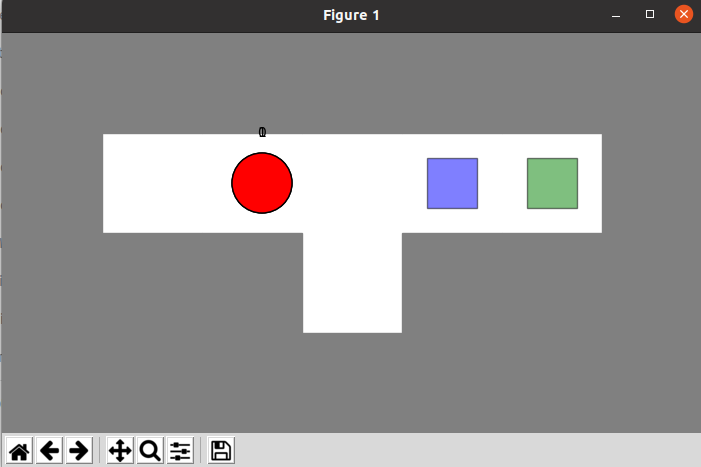
{

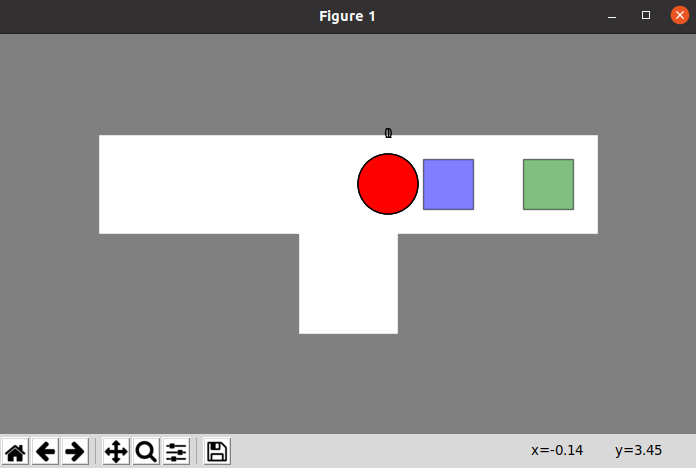
‘agent’: 1,

‘loc’: [(1,2),(1,3)],

‘timestep’: 1

}





* 1. Handling Goal Constraints

Agent 0 and Agent 1 didn’t wait for timestep 10 and start the next cycle immediately as long as they get their goal positions.

What I made to the goal test condition is that: it will check the constraint table first, if the constraint table is empty or the maximum timestep of the constraint table is smaller than the current timestep, then we are done, which means the agent ‘truely’ find it’s goal, and can get its path immediately, otherwise, it does not ‘truely’ find it’s goal, which means there might exist an timestep that larger than the current timestep that the agent is prohibited to stay on the goal position. So I made a for loop to look for the last one that have greater timestep and is at goal location from the constraint table.

* 1. Designing Constraints

constraint is:

{

‘agent’: 1,

‘loc’: [(1,4)],

‘timestep’: 2

}

{

‘agent’: 1,

‘loc’: [(1,2)],

‘timestep’: 2

}

{

‘agent’: 1,

‘loc’: [(1,3)],

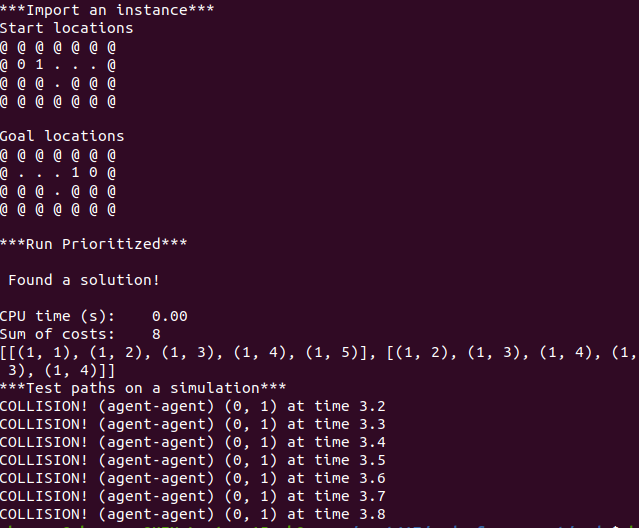
‘timestep’: 2

}

solution: [[(1, 1), (1, 2), (1, 3), (1, 4), (1, 5)], [(1, 2), (1, 3), (2, 3), (1, 3), (1, 4)]]

sum of path lengths: 8

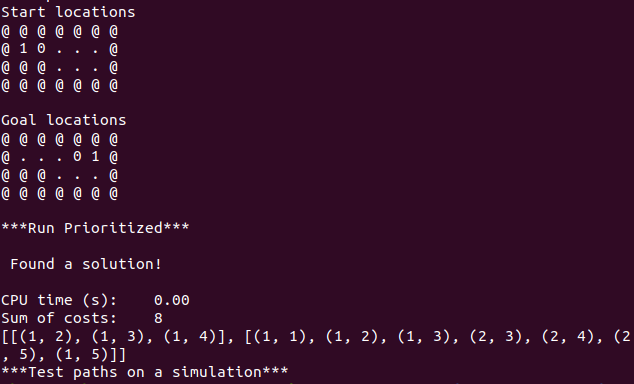
2.1 Adding Vertex Constraints

There still exist collusions, agent 0 and agent 1 exchange their positions at timestep 3, which is an edge collusion

2.2 Adding Edge Constraints

There’s no more collusion in exp2\_1

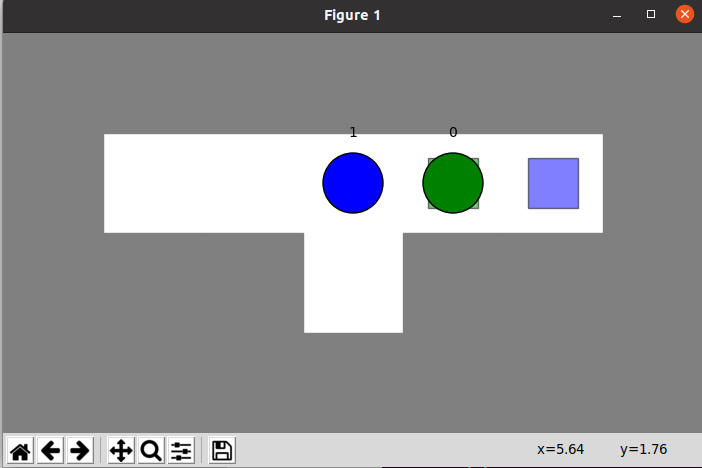
2.3 Adding Additional Constraints

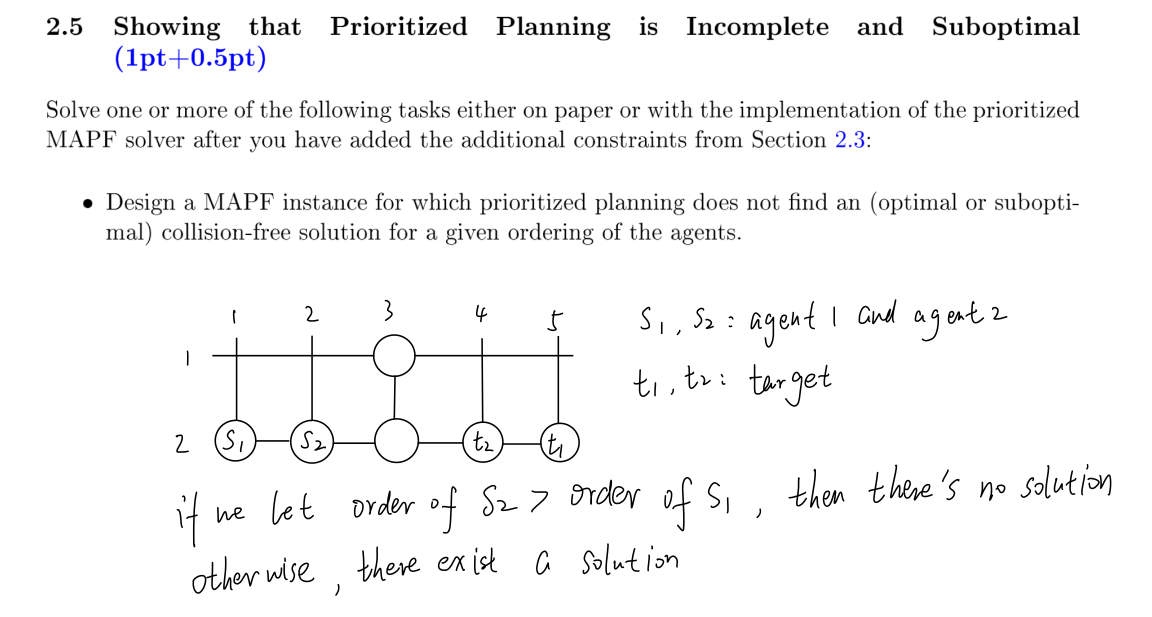


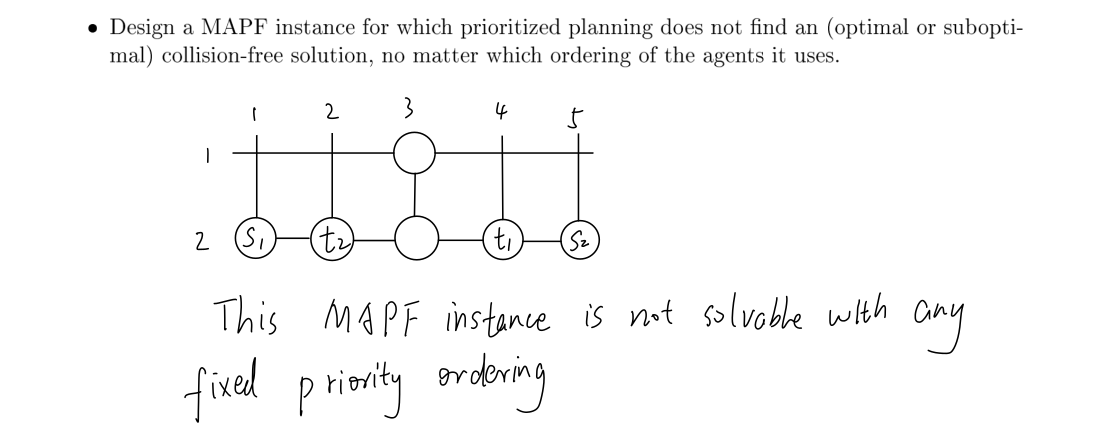
2.4 Addressing Failures

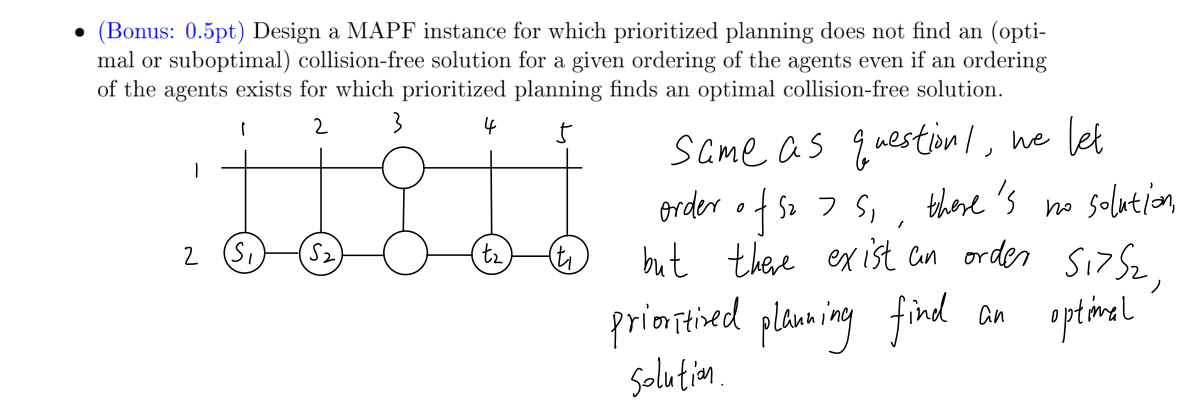
It didn’t report “no solutions” first: after agent 0 arrives its goal, agent 1 wait at (1,4) and it comes with a picture as following shows. However, it can only stop agent 1 from acrossing agent 0 in limited timestep. I think that’s probably because the timestep of additional constraint is limit. So I came up with a tricky idea that set each goal location to “@” in the map as long as the corresponding agent arrives, after I did that, the agent 1 can wait forever.

I set a limit value max\_timestep to solve the problem. Considering that the map may not be rectangular, I set the max\_timestep to be len(max(my\_map)) \* len(my\_map). And after that, it report “no solution”

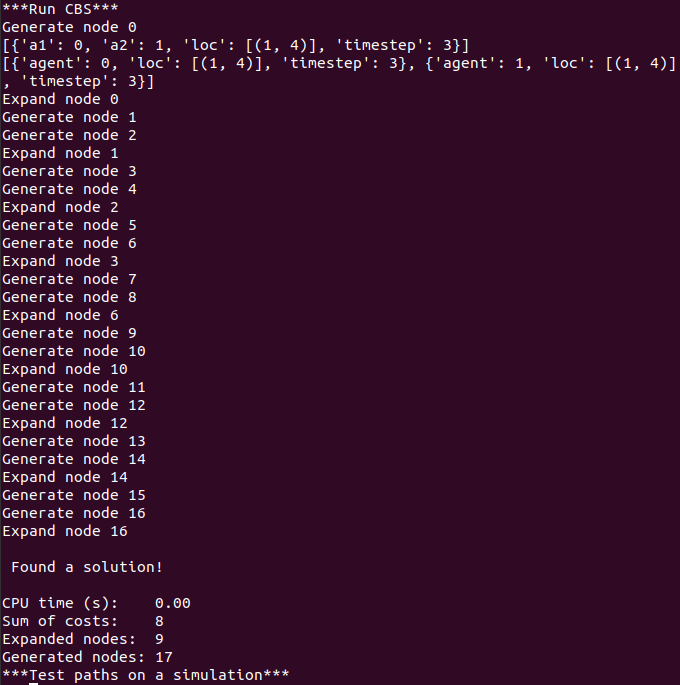






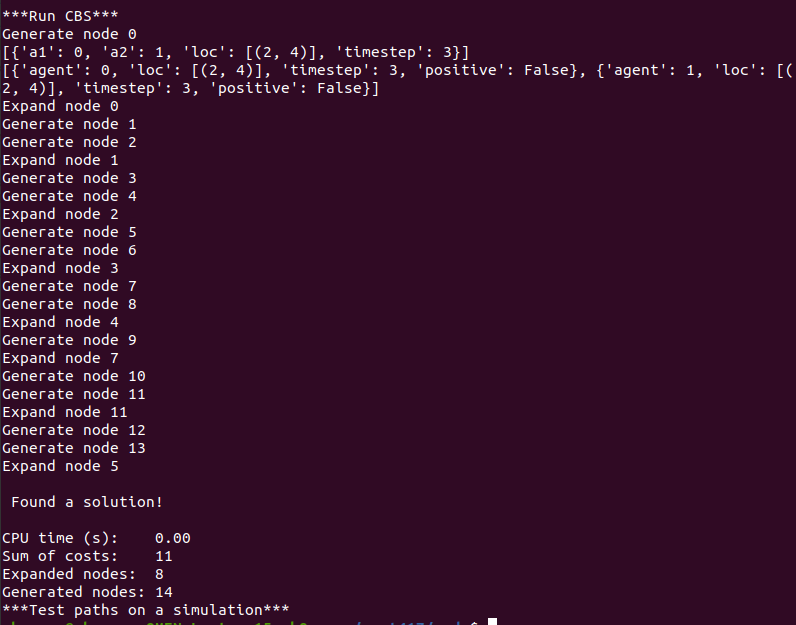


3.3 Implementing the High-Level Search



4.3 Adjusting the High-Level Search

Test for exp4.txt



1. Benchmark

I set 11 agents in the map Berlin\_1\_256.map and use three algorithm to run it, the result are as follow. The agent I set is:

11

1 1 255 255

1 76 234 210

212 144 112 227

194 110 229 60

229 60 158 129

203 218 123 180

177 208 239 28

208 36 109 120

102 57 136 203

141 174 158 148

222 65 172 227

And the map is looks like this:



result:

Prioritized:

CPU time (s): 11.87

Sum of costs: 2383

CBS with standard splitting:

CPU time (s): 7.96

Sum of costs: 2383

Expanded nodes: 5

Generated nodes: 9

CBS with disjoint splitting

CPU time (s): 8.17

Sum of costs: 2383

Expanded nodes: 4

Generated nodes: 7CPU time (s): 11.87

Sum of costs: 2383