CMPT417 individual project report

1.1 Searching in the Space-Time Domain

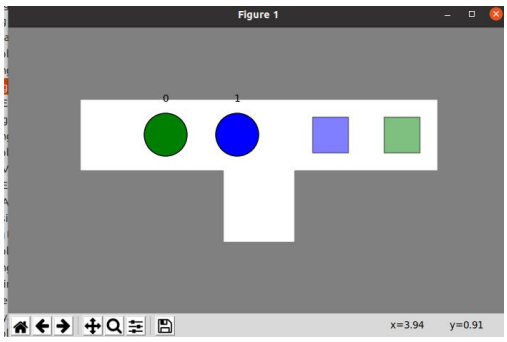
```
***Import an instance***
Start locations
@ @ @ @ @ @ @
@ 0 1 . . . @
@ @ @ . @ @ @
@ @ @ @ @ @ @

Goal locations
@ @ @ @ @ @ @
@ . . . 1 0 @
@ @ @ . @ @ @
@ @ @ @ @ @ @

***Run Independent***

 Found a solution!

CPU time (s):    0.00
Sum of costs:    6
***Test paths on a simulation***
COLLISION! (agent-agent) (0, 1) at time 3.4
COLLISION! (agent-agent) (0, 1) at time 3.5
COLLISION! (agent-agent) (0, 1) at time 3.6
COLLISION! (agent-agent) (0, 1) at time 3.7
COLLISION! (agent-agent) (0, 1) at time 3.8
COLLISION! (agent-agent) (0, 1) at time 3.9
COLLISION! (agent-agent) (0, 1) at time 4.0
COLLISION! (agent-agent) (0, 1) at time 4.1
COLLISION! (agent-agent) (0, 1) at time 4.2
COLLISION! (agent-agent) (0, 1) at time 4.3
COLLISION! (agent-agent) (0, 1) at time 4.4
COLLISION! (agent-agent) (0, 1) at time 4.5
COLLISION! (agent-agent) (0, 1) at time 4.6
```

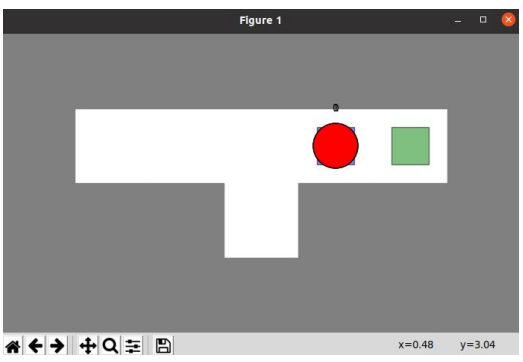
1.2 Handling Vertex Constraints

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

{
    ‘agent’: 0,
    ‘loc’: [(1,5)],
    ‘timestep’: 4
}

```
***Import an instance***
Start locations
@ @ @ @ @ @ @
@ 0 1 . . . @
@ @ @ . @ @ @
@ @ @ @ @ @ @

Goal locations
@ @ @ @ @ @ @
@ . . . 1 0 @
@ @ @ . @ @ @
@ @ @ @ @ @ @

***Run Prioritized***

 Found a solution!

CPU time (s):    0.00
Sum of costs:    7
[[(1, 1), (1, 2), (1, 3), (1, 4), (1, 4), (1, 5)], [(1, 2), (1, 3), (1, 4)]]
***Test paths on a simulation***
COLLISION! (agent-agent) (0, 1) at time 3.4
COLLISION! (agent-agent) (0, 1) at time 3.5
COLLISION! (agent-agent) (0, 1) at time 3.6
COLLISION! (agent-agent) (0, 1) at time 3.7
COLLISION! (agent-agent) (0, 1) at time 3.8
COLLISION! (agent-agent) (0, 1) at time 3.9
COLLISION! (agent-agent) (0, 1) at time 4.0
COLLISION! (agent-agent) (0, 1) at time 4.1
COLLISION! (agent-agent) (0, 1) at time 4.2
COLLISION! (agent-agent) (0, 1) at time 4.3
COLLISION! (agent-agent) (0, 1) at time 4.4
COLLISION! (agent-agent) (0, 1) at time 4.5
COLLISION! (agent-agent) (0, 1) at time 4.6
COLLISION! (agent-agent) (0, 1) at time 4.7
COLLISION! (agent-agent) (0, 1) at time 4.8
COLLISION! (agent-agent) (0, 1) at time 4.9
COLLISION! (agent-agent) (0, 1) at time 5.0
COLLISION! (agent-agent) (0, 1) at time 5.1
COLLISION! (agent-agent) (0, 1) at time 5.2
COLLISION! (agent-agent) (0, 1) at time 5.3
COLLISION! (agent-agent) (0, 1) at time 5.4
COLLISION! (agent-agent) (0, 1) at time 5.5
COLLISION! (agent-agent) (0, 1) at time 5.6
```



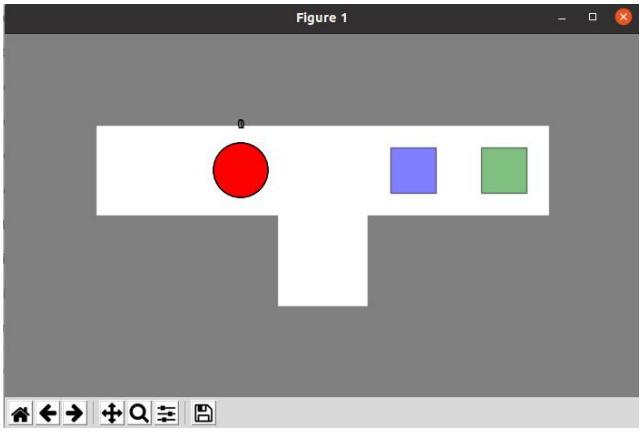


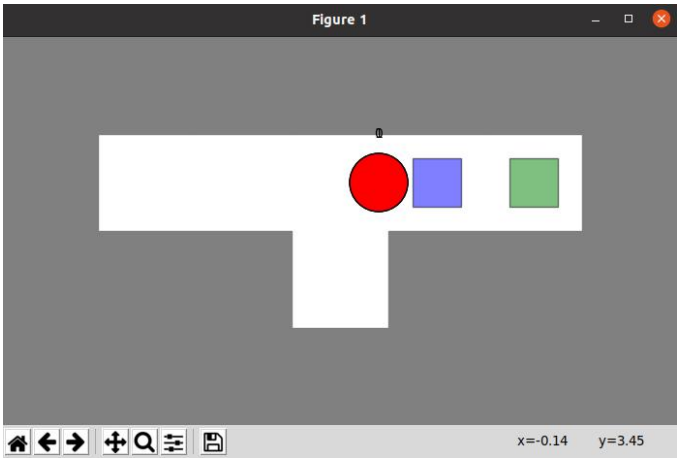
1.3 Adding Edge Constraints

constraint is:

```
{
    'agent': 1,
    'loc': [(1,2),(1,3)],
    'timestep': 1
}
```

```
***Import an instance***
Start locations
@ @ @ @ @ @ @
@ 0 1 . . . @
@ @ @ . @ @ @
@ @ @ @ @ @ @

Goal locations
@ @ @ @ @ @ @
@ . . . 1 0 @
@ @ @ . @ @ @
@ @ @ @ @ @ @

***Run Prioritized***

 Found a solution!

CPU time (s):    0.00
Sum of costs:    7
[[(1, 1), (1, 2), (1, 3), (1, 4), (1, 5)], [(1, 2), (1, 2), (1, 3), (1, 4)]]
***Test paths on a simulation***
COLLISION! (agent-agent) (0, 1) at time 1.4
COLLISION! (agent-agent) (0, 1) at time 1.5
COLLISION! (agent-agent) (0, 1) at time 1.6
COLLISION! (agent-agent) (0, 1) at time 1.7
COLLISION! (agent-agent) (0, 1) at time 1.8
COLLISION! (agent-agent) (0, 1) at time 1.9
COLLISION! (agent-agent) (0, 1) at time 2.0
COLLISION! (agent-agent) (0, 1) at time 2.1
COLLISION! (agent-agent) (0, 1) at time 2.2
COLLISION! (agent-agent) (0, 1) at time 2.3
COLLISION! (agent-agent) (0, 1) at time 2.4
COLLISION! (agent-agent) (0, 1) at time 2.5
COLLISION! (agent-agent) (0, 1) at time 2.6
COLLISION! (agent-agent) (0, 1) at time 2.7
COLLISION! (agent-agent) (0, 1) at time 2.8
COLLISION! (agent-agent) (0, 1) at time 2.9
COLLISION! (agent-agent) (0, 1) at time 3.0
COLLISION! (agent-agent) (0, 1) at time 3.1
COLLISION! (agent-agent) (0, 1) at time 3.2
COLLISION! (agent-agent) (0, 1) at time 3.3
COLLISION! (agent-agent) (0, 1) at time 3.4
COLLISION! (agent-agent) (0, 1) at time 3.5
COLLISION! (agent-agent) (0, 1) at time 3.6
COLLISION! (agent-agent) (0, 1) at time 3.7
COLLISION! (agent-agent) (0, 1) at time 3.8
COLLISION! (agent-agent) (0, 1) at time 3.9
COLLISION! (agent-agent) (0, 1) at time 4.0
COLLISION! (agent-agent) (0, 1) at time 4.1
COLLISION! (agent-agent) (0, 1) at time 4.2
COLLISION! (agent-agent) (0, 1) at time 4.3
COLLISION! (agent-agent) (0, 1) at time 4.4
COLLISION! (agent-agent) (0, 1) at time 4.5
COLLISION! (agent-agent) (0, 1) at time 4.6
```





1.4 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.

1.5 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

```
***Import an instance***
Start locations
@ @ @ @ @ @ @
@ 0 1 . . . @
@ @ @ . @ @ @
@ @ @ @ @ @ @

Goal locations
@ @ @ @ @ @ @
@ . . . 1 0 @
@ @ @ . @ @ @
@ @ @ @ @ @ @

***Run Prioritized***

 Found a solution!

CPU time (s):    0.00
Sum of costs:    8
[[(1, 1), (1, 2), (1, 3), (1, 4), (1, 5)], [(1, 2), (1, 3), (1, 4), (1,
 3), (1, 4)]]
***Test paths on a simulation***
COLLISION! (agent-agent) (0, 1) at time 3.2
COLLISION! (agent-agent) (0, 1) at time 3.3
COLLISION! (agent-agent) (0, 1) at time 3.4
COLLISION! (agent-agent) (0, 1) at time 3.5
COLLISION! (agent-agent) (0, 1) at time 3.6
COLLISION! (agent-agent) (0, 1) at time 3.7
COLLISION! (agent-agent) (0, 1) at time 3.8
```

2.2 Adding Edge Constraints

There’s no more collusion in exp2_1

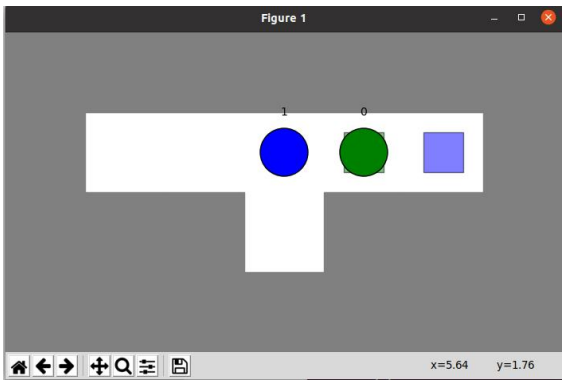
2.3 Adding Additional Constraints

```
Start locations
@ @ @ @ @ @ @
@ 1 0 . . . @
@ @ @ . . . @
@ @ @ @ @ @ @

Goal locations
@ @ @ @ @ @ @
@ . . . 0 1 @
@ @ @ . . . @
@ @ @ @ @ @ @

***Run Prioritized***

 Found a solution!

CPU time (s):    0.00
Sum of costs:    8
[[(1, 2), (1, 3), (1, 4)], [(1, 1), (1, 2), (1, 3), (2, 3), (2, 4), (2
, 5), (1, 5)]]
***Test paths on a simulation***
```

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.

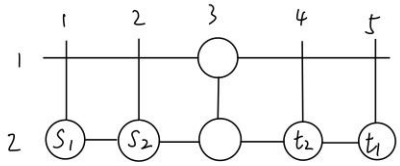
I set a limit value max_timestep to solve the problem. Considering that the map may not be rectangular, I set the max_timestep to be len(max(my_map)) * len(my_map). And after that, it report “no solution”



2.5 Showing that Prioritized Planning is Incomplete and Suboptimal (1pt+0.5pt)

Solve one or more of the following tasks either on paper or with the implementation of the prioritized MAPF solver after you have added the additional constraints from Section 2.3:

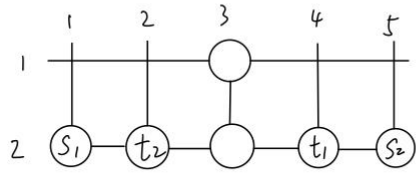
- Design a MAPF instance for which prioritized planning does not find an (optimal or suboptimal) collision-free solution for a given ordering of the agents.



s_1, s_2 : agent 1 and agent 2
t_1, t_2 : target

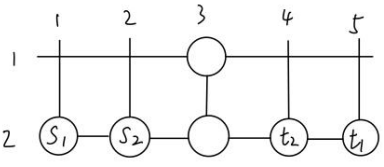
if we let order of s_2 > order of s_1, then there's no solution
otherwise, there exist a solution

- Design a MAPF instance for which prioritized planning does not find an (optimal or suboptimal) collision-free solution, no matter which ordering of the agents it uses.



This MAPF instance is not solvable with any fixed priority ordering

- (Bonus: 0.5pt) Design a MAPF instance for which prioritized planning does not find an (optimal or suboptimal) collision-free solution for a given ordering of the agents even if an ordering of the agents exists for which prioritized planning finds an optimal collision-free solution.



Same as question 1, we let order of $s_2 > s_1$, there's no solution, but there exist an order $s_1 > s_2$, prioritized planning find an optimal solution.

3.3 Implementing the High-Level Search

```
***Run CBS***
Generate node 0
[{'a1': 0, 'a2': 1, 'loc': [(1, 4)], 'timestep': 3}]
[{'agent': 0, 'loc': [(1, 4)], 'timestep': 3}, {'agent': 1, 'loc': [(1, 4)]
, 'timestep': 3}]
Expand node 0
Generate node 1
Generate node 2
Expand node 1
Generate node 3
Generate node 4
Expand node 2
Generate node 5
Generate node 6
Expand node 3
Generate node 7
Generate node 8
Expand node 6
Generate node 9
Generate node 10
Expand node 10
Generate node 11
Generate node 12
Expand node 12
Generate node 13
Generate node 14
Expand node 14
Generate node 15
Generate node 16
Expand node 16

 Found a solution!

CPU time (s):    0.00
Sum of costs:    8
Expanded nodes:  9
Generated nodes: 17
***Test paths on a simulation***
```

4.3 Adjusting the High-Level Search

Test for exp4.txt

```
***Run CBS***
Generate node 0
[{'a1': 0, 'a2': 1, 'loc': [(2, 4)], 'timestep': 3}]
[{'agent': 0, 'loc': [(2, 4)], 'timestep': 3, 'positive': False}, {'agent': 1, 'loc': [(
2, 4)], 'timestep': 3, 'positive': False}]
Expand node 0
Generate node 1
Generate node 2
Expand node 1
Generate node 3
Generate node 4
Expand node 2
Generate node 5
Generate node 6
Expand node 3
Generate node 7
Generate node 8
Expand node 4
Generate node 9
Expand node 7
Generate node 10
Generate node 11
Expand node 11
Generate node 12
Generate node 13
Expand node 5

 Found a solution!

CPU time (s):    0.00
Sum of costs:    11
Expanded nodes:  8
Generated nodes: 14
***Test paths on a simulation***
```

5. 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