

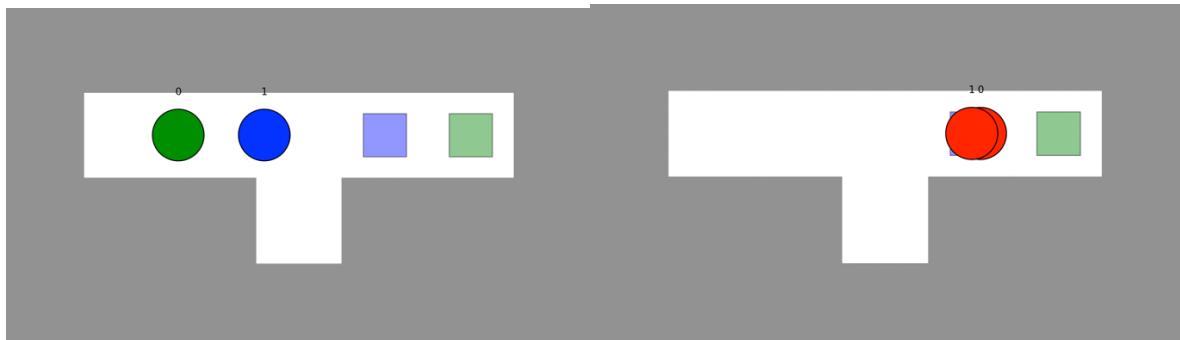
417/827 Project Report

S M Shahjiban Munjoreen

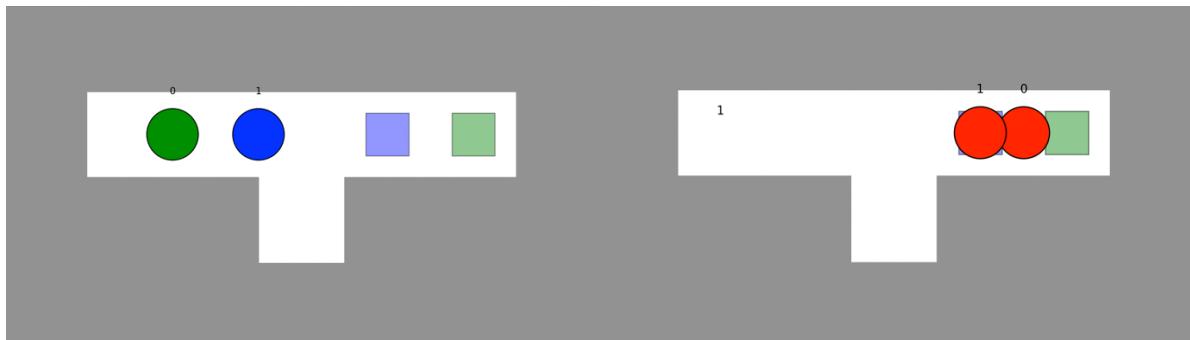
301323565

1 Task 1: Implementing Space Time A*

1.1 Searching in the Space-Time Domain



1.2 Handling Vertex Constraint



Waits for one time step before getting to the goal

1.4 Handling goal constraint

```
# Constraint for Task 1.2
constraints = [{'agent': 0, 'loc': [(1,5)], 'timestep': 4}]

# Constraint for Task 1.3
constraints = [{'agent': 0, 'loc': [(1,2),(1,3)], 'timestep': 1}]

# Constraints for Task 1.4
constraints = [{'agent': 0, 'loc': [(1,5)], 'timestep': 10},
#                 {'agent': 0, 'loc': [(1,3),(1,4)], 'timestep': 5},
#                 {'agent': 0, 'loc': [(1,3),(1,4)], 'timestep': 7},
#                 ]

# Constraints for Task 1.5
constraints = [{'agent': 1, 'loc': [(1,3)], 'timestep': 2},
#                 {'agent': 1, 'loc': [(1,2)], 'timestep': 2},
#                 {'agent': 1, 'loc': [(1,3),(1,4)], 'timestep': 2}]
```

1.5 Designing constraints

```
Iqras-MacBook-Pro:code iqra$ python3 run_experiments.py --instance instances/exp1.txt --solver Prioritized
[olver Prioritized
***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), (2, 3), (1, 3), (1, 4)]]]
***Test paths on a simulation***
```

Agent 1 (blue) move into the little nook and lets agent 0 pass by without any collision.
Success!!

2 Task 2: Implementing Prioritized Planning

2.4 Addressing Failures

```
if curr['time_step']>10:  
    print("couldnt be solved in time :( ")  
    return None
```

Added a timestep of 10 as a condition. Therefore, if there are more than 10 steps, it says the problem could not be solved in time

2.5 Showing that Prioritized Planning is Incomplete and Suboptimal

For exp2_3 the program provides so solution given the correct order but in the event of a reversed order, we are provided with a solution

For the plan provided, the program provides us with no result regardless of the order even though a solution exists

```
4 7  
@ @ @ @ @ @ @  
@ @ .... @ @ @  
@ @ @ . @ @ @  
@ @ @ @ @ @ @  
2  
1 2 1 4  
2 3 2 3
```

For test_4 we are provided with a solution but there is a collision in it, so we are not able to get a collision free answer

3 Task 3: Implementing Conflict-Based Search (CBS)

3.3 implementing the high-level search

```
****run CBS****
Generate node 0
[{"a1": 0, "a2": 1, "loc": [(2,4)], "timestep": 3}]
[{"agent": 0, "loc": [(2, 4)], "timestep": 3}, {"agent": 1, "loc": [(2, 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 4
Generate node 7
Generate node 8
Expand node 5
Generate node 9
Generate node 10
Expand node 6
Generate node 11
Generate node 12
Expand node 8
Generate node 13
Generate node 14
Expand node 10
Generate node 15
Generate node 16
Expand node 11
Generate node 17
Generate node 18
Expand node 13
Generate node 19
Generate node 20
Expand node 16
Generate node 21
Generate node 22
Expand node 18
Generate node 23
Generate node 24
Expand node 24
Generate node 25
Generate node 26
Expand node 3
No collision
***test paths on a simulation***
```

4 Task 4: Implementing CBS with disjoint splitting

4.3 adjusting the high-level search

```
Generate node 0
[{'a1': 0, 'a2': 1, 'loc': [(2,4)], 'timestep': 3}]
[{'agent': 0, 'loc': [(2, 4)], 'timestep': 3}, {'agent': 1, 'loc': [(2, 4)], 'time
step': 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 6
Generate node 7
Generate node 8
Expand node 8
Generate node 9
Generate node 10
Expand node 10
Generate node 11
Generate node 12
Expand node 9
No collision
***test paths on a simulation***
```