MAPF Project

Task 0: The optimal path for the agent is chosen by A\* search with no constraints hence leading to agents’ collision at timestep 3.4 – 4.6

![Text

Description automatically generated]()

Task 1: Implementing Space-Time A\*

* 1. running command *python run\_experiments.py --instance instances/exp1.txt --solver Independent* for exp1 file yields a similar result as seen in Task 0 but with A\* search in Space-time Domain instead of just space.

![Text

Description automatically generated]()

* 1. running command *python run\_experiments.py --instance instances/exp1.txt --solver Prioritized* with *constraints = [{'agent': 0,'loc': [(1,5)],'timestep': 4}]* after implementing build\_constraint\_table and is\_constrained functions. We can see a change in agent 0’s behavior at time 4 as the constraint prevents it from going to loc = (1,5) causing it to wait at loc = (1,4).

Text

Description automatically generated

* 1. Setting timestep 10 for agent 0 causes it to be active even after reaching its goal with the agent waiting at its goal till time = 10 and then being constrained to leave (1,5) and collide with agent 1.

New added goal test conditions included checking if the child loc is outside the bounds of the map and checking if the child loc might be constrained at the next timestep.

Text

Description automatically generated

* 1. For collision free paths, we must force agent 1 to move to loc = (2,3) at timestep 2, this is only possible by setting up some specific constraints. The list of constraints I came up with are:
     + {'agent':1, 'loc':[(1,2),(1,2)], 'timestep':1} => prevents agent 1 from waiting at 1,2 at t = 1 and forces it move to the only available option (1,3) (avoid collision with agent 0 moving to (1,2) at t = 1)
     + {'agent':1, 'loc':[(1,3),(1,2)], 'timestep':2} => prevents agent 1 from moving back to 1,2 at t = 2
     + {'agent':1, 'loc':[(1,3),(1,3)], 'timestep':2} => prevents agent 1 from waiting to 1,3 at t = 2
     + {'agent':1, 'loc':[(1,4)], 'timestep':2} => prevents agent 1 from moving to 1,4 at t = 2

This way we don’t have add constraints to agent 0, hence keeping sum of paths low.

Text

Description automatically generated

Task 2: Implementing Prioritized Planning

2.4 Running command *python run\_experiments.py --instance instances/exp2\_3.txt --solver Prioritized* causes the problem to be stuck in an infinite loop since the priorities between agent 0 and 1 are switched causing agent 1 to have greater priority than 0. This affects the path planning since making 0 reach its goal is no longer important to us, agent 1 is given the shortest path and therefore blocking 0 from ever reaching its goal.

Hence, the problem being stuck in a loop as *len(open\_list)* will always be greater than 0.

Adding time constraint should stop the problem from being stuck in a loop.

2.5 (Subproblem 1 & 3)

If we look at this instance, in the given case where agent 1 is given priority over agent 0. We can spot that there is a collision free path for 1 to reach its goal. But the program is unable to find any solution for the given priorities. However, if we switch priorities the program is able to find a solution with 1 taking the obvious path.

|  |  |
| --- | --- |
|  | Icon  Description automatically generated with medium confidence |

Task 3: Implementing Conflict-Based Search (CBS)

3.3 Testing High-level CBS with command *python run\_experiments.py --instance instances/exp2\_1.txt --solver CBS.* A total of 15 nodes are expanded for the CBS on standard splitting.

|  |  |
| --- | --- |
| Text  Description automatically generated | Text  Description automatically generated |

Task 4: Implementing CBS with Disjoint Splitting

4.3 Testing High-level CBS with command *python run\_experiments.py --instance instances/exp4.txt --disjoint --solver CBS.* A total of 7 nodes are expanded for CBS on disjoint splitting.

Text

Description automatically generated