Decentralized Multi-agent Path Planning

Puneet Singhal, Hadi Salman, Sha Yi, Yifan Ding, Zhening Yang

Abstract— **Multiagent path planning can be divided into two categories: centralized and decentralized. Traditional centralized path planners can suffer when the number of robots increases, or when no perfect communication can be maintained between the robots and a central control base. These limitations motivated us to look into this poblem and present a solution that tackles these limitations. Specifically, we considered M* algorithm, which is a centralized multiagent path planning algorithm, and we implement a decentralized version of it that can scale up to any number of robots, and requires only local communication between robots near each other, without the need to commnuicate with a central computer.**

I. Introduction

Planning in joint configuration space becomes difficult as we increase the number of robots. Glenn et al. Wagner et. al [1] proposed a new algorithm, M*, that is based on the idea that planning in joint configuration space of a set of robots is only necessary when they are strongly coupled, which is often not true if the robots are well separated in the workspace. The inherent problem with M* is when working with large number of robots in a cluttered environment as the interaction between robots is strong. In this case the centralized (vanilla) M* will be computationally slow. We, therefore, propose the decentralized version, where each robot follows their individual policy (A* with Manhattan distance as heuristic). But when a set of robots come in close proximity, we use the computation available with each robot to start multiple M* searches. This reduces the dimensionality of the system. We will compare the performance of decentralized M* with decentralized priority planner.

II. Motivation

Multi-agent planning is widely used in many fields, like warehouse robots, multi-agent robotic assembly. Among various approaches of multi-agent planning, decentralization could increase efficiency of planning considerably because each agent plans for itself without time-consuming coordination from central node. Decentralized way could be suffer from collision of agents. Thus we intend to implement an effective method to solve the multi-agent path planning problem without collision based on belief space and M* algorithm.

[1]All the authors are with Carnegie Mellon University, Pittsburgh, PA 15213, USA (psinghal, hadis, shayi, yifand1, zheningy) @andrew.cmu.edu

III. Implementation and Hardware

The implemention of algorithm base on simulated environment, Gazebo. Open source and off-the-shelve simulated robots, Husky robots used as agents. We utilized Python and ROS environment to integrate the whole system.

IV. mstar

M star is a multi-robot path planner that initially plan for each robot separately, and only couple sets of robots after they have been found to interact, thus minimizing the dimensionality of the search space (avoid joint planning from when unnecessary). It is an implementation of subdimensional expansion startegy that dynamically generates low dimensional search spaces embedded in the full configuration space. M star is complete and finds minimal cost paths [1]. In our project, we build upon Mstar and implement it in a decentralized fashion, while using parts of the original code as black boxes.

V. Architecture

Our multi-agent system includes planner node, control node, path database, and collision detection node. The pseudo code of our algorithm is shown in Algorithm 1. Our system architecture is shown in Fig. 1.

A. Path Planner

The path planner node takes input from coordinator node of start and goal pairs of the robot to be planned, and also waypoints of other robots if there exists path history in the database.
The planner will plan with M* algorithm if the waypoint of other robots is empty, and plan with priority planner if there is existing path for other robots.

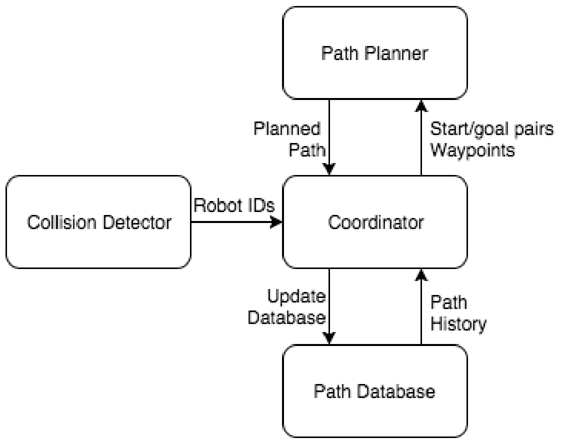

Fig. 1. System Architecture

Algorithm 1 Decentralized M*

```
1: Given: N robots with starting and goal positions
2: Run A* search for each of the N robots
3: Start executing the plan of each robot
4: while GOAL_NOT_REACHED do          ▷ parallel thread for each robot
5:     if DETECT_COLLISION then
6:                         ▷ assume M robots are in collision
7:         Run M* for the colliding robots
8:         Update the plans for these M robots
9:     Execute next step of the plan of the robot
```

B. Coordinator

Coordinator node is the central controller of each robot. When the collision detector sends out the robots in collision range, the coordinator will query the path database for existing paths and pass to the planner node. After getting the latest planned path, path database will be updated. For each of the robot, the permutation of robot IDs passed to the planner is different. Whenever one of the robot finishes planning, it will publish its result and all the other robot will follow this plan. In this case, we are utilizing the random restart idea [2].

C. Collision Detector

Collision detector node will keep checking the distance between current robot and all other robots. Whenever a collision is detected, coordinator will be notified.

D. Path Database

The path database keeps record of paths of all the robot encountered, and update with the newest path if a duplicated path comes in.

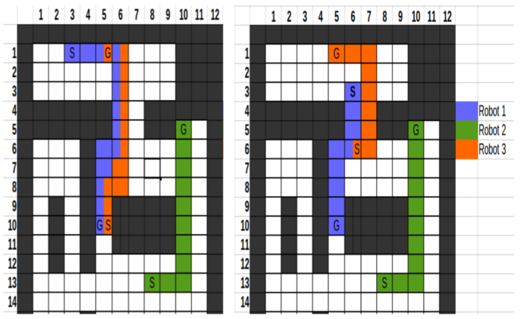

Fig. 2. Decentralized planner for 3 robots

VI. Results

We have tested the performance of decentralized planning algorithm in two situations. 1. with three robots in a cluttered environment using M* for planning at collisions, 2. with 5 robots in an empty environment using priority planning at collisions. We will share the results of each experiments as below and the video could be found here and there.

Fig. 2 (left) shows the individual path generated by A* algorithm. We can see that robot 1 and robot 3 paths overlap as indivual policy does not take care of other robots. When the collision is detected between robot 1 and robot 3, M* planner generates path which prevents collision (shown in right part of fIG. 2).

No. of Robots	Algorithm	Time	Path Length
2	M*	0.00124	20
2	Priority	0.0289	18
4	M*	0.0196	23
4	Priority	0.0330	20
9	M*	0.0701	24
9	Priority	0.0149	19

VII. Conclusion

In this project, we implemented a decentralized version of a global multi-agent path planning algorithm (M*). We compare our algorithm with a simple prioritized A* planner that we implemented. We tested our implementation on a multi-agent system in Gazebo composed of N Husky robots. Both the decentralized M* and the prioritized A* algorithm gave similar results in terms of the length of the path returned by the planners, and the time it takes to find a plan.

References

[1] Glenn Wagner and Howie Choset. M*: A complete multirobot path planning algorithm with performance bounds.

[2] Glenn Wagner and Howie Choset. Subdimensional expansion for multirobot path planning. *CoRR*, abs/1706.02794, 2017.