Towards Unstructured MAPF: Multi-Quadruped MAPF Demo

Rishi Veerapaneni*, Nikhil Sobanbabu*, Guanya Shi, Jiaoyang Li, Maxim Likhachev

Carnegie Mellon University
{rveerapa, nsobanba, guanyas, jiaoyanl, mlikhach}@andrew.cmu.edu

Abstract

Multi-Agent Path Finding (MAPF) in its most broad perspective focuses on finding collision free paths for general teams of agents in a shared environment. Theoretically, MAPF methods could solve a variety of multi-agent problems. However, MAPF research primarily focuses on simplified warehouse domains, i.e., gridworld with discrete spaces, discrete timesteps, and point-mass agents without kinematic constraints. Thus, the perception of MAPF is tied closely to gridworld and its assumptions, which limits its attractiveness to more broad domains. However, there are several ways to extend MAPF methods past these classical assumptions. To this end, our demo shows how MAPF techniques can be used to plan for a team of quadrupeds. Our system plans in continuous space, in continuous time, with realistic footprints, and incorporates dynamics constraints. See https://youtu.be/TLEVkN4ywgU for additional details.

1 Introduction

Multi-Agent Path Finding (MAPF) focuses on finding collision-free paths for a team of agents in a shared workspace. The crown applications of MAPF are automated warehouse robotic systems which contain 100s-1000s of planar robots that need to transport items between different locations. Modern MAPF methods are extremely capable and can plan for 1000s of these agents in seconds. However, MAPF methods require many simplifying assumptions. In particular, most MAPF methods require a discretized world, discretized timesteps, point-mass agents, and no kinematic constraints. Thus, the perception of MAPF is that it is constrained to warehouses and not applicable to other complex robotics systems with fewer assumptions. To that end, our demo shows how to use MAPF for a team of quadrupeds. In particular, we plan in continuous space, continuous time, realistic footprints, and incorporate kinematic constraints for a team of heterogeneous quadrupeds. We note that we are not the first to relax these assumptions and that prior work, in particular dB-CBS (Moldagalieva et al. 2024), does so. Our demo seeks to emphasize these advancements with a realistic demo with complex agents.

2 Related Works

Classic MAPF Formulation Multi-Agent Path Finding (MAPF) is the problem of finding collision-free paths for a group of N agents, that takes each agent i from its start location s_i^{start} to its goal location s_i^{goal}. In traditional 2D MAPF, the environment is discretized into grid cells, and time is broken down into discrete timesteps. Agents are allowed to move in any cardinal direction or wait in the same cell. A valid solution is a set of agent paths without vertex collisions (two agents at the same location at the same timestep) and edge collisions (two agents swapping locations). Thus, according to this construction, classical MAPF has the following restrictions:(1) Discretized locations, (2) Discretized timesteps, (3) Point-mass agents, and (4) No dynamics or kinematic constraints.

Variations on MAPF Assumptions There are several works that reduce the above assumptions. Large Agents MAPF (LA-MAPF) specifically focuses on MAPF with agents with non-point-mass footprints (Li et al. 2019). LA-MAPF specifically introduces two different constraints in the context of Conflict-Based Search (Sharon et al. 2015). Several works have focused on MAPF with continuous time with non-point-mass footprints. Continuous time CBS (CCBS) (Andreychuk et al. 2019) used Safe-Interval Path Planning (SIPP) (Phillips and Likhachev 2011) to plan for disk agents in continuous time on a 2^k connected grid. Discontinuity-Bounded CBS (dB-CBS) (Moldagalieva et al. 2024) uses CBS with a combination of heuristic search and optimization solvers to plan for a set of heterogeneous agents in continuous space, time, different footprints, and kinematic constraints.

Handling Execution Uncertainty Even after finding a collision-free MAPF solution, how do we actually execute these plans given that agents are not perfectly modeled and can have execution imperfections? In particular, there are two types of imperfections: (1) Spatial uncertainty and (2) Temporal uncertainty.

Dealing with spatial uncertainty is relatively simple if the agent's controller has bounded error; we can inflate the agent's footprint during MAPF planning to incorporate the bounded error. We can test an agent's controller ahead of time to measure its error. We choose to handle temporal uncertainty by using a Temporal Planning Graph (TPG) that encodes temporal dependencies between agents (Hönig et al. 2016). A TPG is more flexible as it does not require modifying the MAPF solver/solution but instead requires agents to be able to communicate to each other during execution.

*These authors contributed equally.

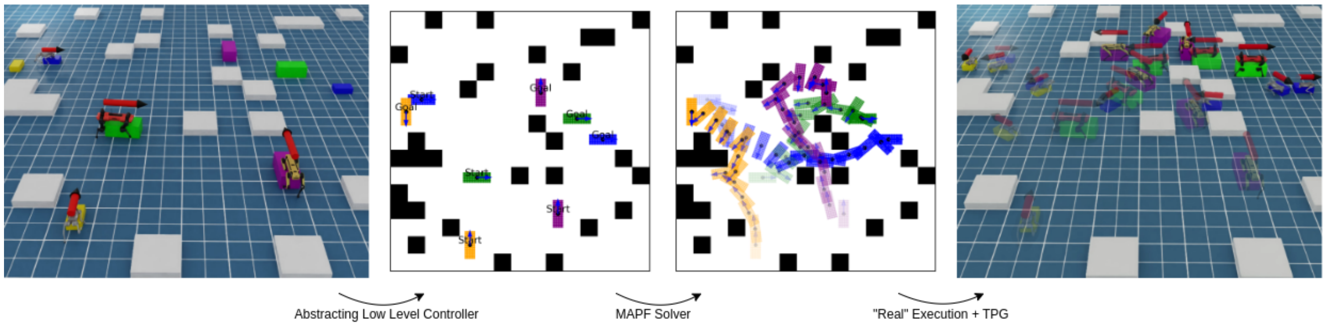

Figure 1: We visualize the pipeline for MAPF with quadrupeds (see Section 3).

3 MAPF for Multiple Quadrupeds

This section describes the technical details of our demo for planning and executing for multiple quadrupeds in a shared planar environment.

Modeling the Quadruped We model the quadruped as a rectangle given the existence of robust Reinforcement Learning (RL) policy training pipelines for velocity tracking tasks in legged robots. This low-level RL policy enables the quadruped to track a 2D twist command consisting of linear translational velocity (v_x, v_y), and angular velocity (w_z). Further, we incorporate heterogeneity in our framework with three different quadrupeds: Unitree Go2, Anybotics Anymal C, and Boston Dynamics' Spot. However, we noticed that the pre-trained locomotion policies for the Unitree Go2 and Anymal C were poor in tracking high velocities and high acceleration trajectories. Hence, we trained custom policies for these two robots with extra regularization rewards and higher velocity command ranges.

Given a sufficiently performant velocity controller, we can abstract away the quadrupeds 12 DoF leg joints (4 legs x 3 DoF per joints) but instead only represent the quadruped with a position (x, y) and a heading θ. A quadruped additionally has its own kinematic dynamics/constraints (e.g., rotating takes longer than moving forward) based on the velocity controller. Finally, we simplify the footprint of the quadruped to be a rectangle. Given the (x, y, θ) state and rectangular footprint per quadruped, the MAPF planner finds a set of collision-free paths per quadruped, where each path is a sequence of (x, y, θ) waypoints.

MAPF Solver Given a set of start (x, y, θ) positions, goal (x, y, θ) positions, and rectangular footprints, the MAPF solver need to find collision-free (x, y, θ) paths while satisfying the agent's kinematic/dynamic constraints. We use Conflict-Based Search (CBS) with a few modifications for practicality at the expense of optimality/bounded-suboptimality or completeness (Sharon et al. 2015). First, since we want to plan in continuous space and time, we use a Rapidly-Exploring Random Tree (RRT) single-agent planner (LaValle and Kuffner 2001). The RRT planner works in continuous space and time while avoiding obstacles, constraints, and incorporating the dynamic constraints. Second, since our agents have large footprints, we do not have edge conflicts but only deal with vertex conflicts. Since we work with continuous space and time, we detect collisions via a finely discretized collision checker. Third, given a detected collision between two rectangles a_1, a_2 at $(x_1^t, y_1^t, \theta_1^t)$ and $(x_2^t, y_2^t, \theta_2^t)$ respectively, we apply a simplified vertex constraint. We detect a collision point in the overlap between the agents and apply a constraint that each agent needs to avoid overlapping with the collision point for a time range. We note that using this time range constraint means that CBS is not complete. Finally, since we are mainly interested in finding feasible paths as opposed to finding minimal time paths, we use greedy-CBS which sorts the high-level queue via (#num conflicts, sum of time) (Barer et al. 2014).

Executing the MAPF solution Given a collision-free set of paths, we execute them in Isaac Sim using our trained velocity tracking RL policies. The MAPF solution is first transformed into a TPG, which encodes the action dependencies across agents. Each quadruped in Isaac Sim follows this TPG, executing actions in sequence to reach the designated waypoints. To ensure smooth transitions between waypoints, the trajectory is refined using a Bézier spline. At each timestep, a proportional controller in the body frame computes the velocity commands for the RL policy based on the robot's current position and the next intermediate waypoint within the spline.

Results Figure 1 illustrates the planning and execution process of a four-agent system navigating from their start positions to the corresponding goal locations without any collisions. We further evaluated our pipeline on maps with 5-, 8-, and 12-agent systems. The average execution times observed were 113.32 ± 4.79, 198.60 ± 12.44, and 312.86 ± 23.76 seconds, respectively. These results were obtained over three independent trials for each configuration, with each trial using randomized, heterogeneous agents. This shows how MAPF can be used to plan relatively long horizon trajectories.

Future Goals This system is in a developmental state currently. Our target by September (so before the ICAPS conference) is to have a GitHub repository that cleanly allows researchers to focus on different components. In particular, we would like to make it a type of MAPF benchmark that MAPF researchers can use to develop and evaluate new algorithms. We are additionally working on deploying this on a real-world system. Overall, our goal with this demo is to show how MAPF research can be used for more complex robot systems, and potentially serve as a tool for future research.

References

Andreychuk, A.; Yakovlev, K.; Atzmon, D.; and Stern, R. 2019. Multi-Agent Pathfinding with Continuous Time. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, 39–45.

Barer, M.; Sharon, G.; Stern, R.; and Felner, A. 2014. Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem. In *Seventh Annual Symposium on Combinatorial Search*.

Hönig, W.; Kumar, T. S.; Cohen, L.; Ma, H.; Xu, H.; Ayanian, N.; and Koenig, S. 2016. Multi-agent path finding with kinematic constraints. In *Twenty-Sixth International Conference on Automated Planning and Scheduling*.

LaValle, S. M.; and Kuffner, J. J. 2001. Rapidly-exploring random trees: Progress and prospects. *Algorithmic and computational robotics*, 303–307.

Li, J.; Surynek, P.; Felner, A.; Ma, H.; Kumar, T. K. S.; and Koenig, S. 2019. Multi-Agent Path Finding for Large Agents. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01): 7627–7634.

Moldagalieva, A.; Ortiz-Haro, J.; Toussaint, M.; and Hönig, W. 2024. db-CBS: Discontinuity-bounded conflict-based search for multi-robot kinodynamic motion planning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 14569–14575. IEEE.

Phillips, M.; and Likhachev, M. 2011. SIPP: Safe interval path planning for dynamic environments. In *2011 IEEE International Conference on Robotics and Automation*, 5628–5635.

Sharon, G.; Stern, R.; Felner, A.; and Sturtevant, N. R. 2015. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219: 40–66.