Discrete database guided multi-robot kinodynamic motion planning

Rachel A. Moan, Adam Sitabkhan, and Kris Hauser
University of Illinois at Urbana-Champaign
Champaign, Illinois, USA
{rmoan2, asita2, kkhauser}@illinois.edu

***Abstract*—Multi-robot motion planning (MRMP) is a computationally challenging problem with many applications including factory automation and search and rescue operations. In this problem, robots must find safe trajectories from their starts to their goals while obeying dynamic constraints. Many MRMP planners are too slow, prone to deadlock, or ignore dynamic constraints. Multi-agent path finding (MAPF) is a discrete version of the problem which treats robots as points moving on a grid. In this work, we combine an experience-based MAPF algorithm with Model Predictive Control (MPC) to quickly reroute robots while obeying dynamics. We demonstrate preliminary results on a swapping problem in which several robots must trade places, and show that our method finds solutions where others fail.**

I. Introduction

Multi-robot motion planning (MRMP) has a wide range of applications including exploration [1], factory automation [7], construction [11], and search and rescue operations [9]. In this problem, robots must find collision free trajectories that arrive at their respective goals while obeying the robots' dynamic constraints.

One way to solve the MRMP problem is to use a sampling-based planner, such as kinodynamic RRT [13], and sample in the joint state space of all the robots. However, since the size of the search space is exponential with the number of robots, the number of samples needed to find a valid solution causes the compute time for these methods to explode. Another approach is to use precomputed motion primitives to compose safe trajectories for all robots [10].

Local MRMP methods generate plans using only the robot's immediate surrounding environment. This approach is well-suited for handling dynamic environments. They are, however, susceptible to deadlock, especially in congested scenarios or narrow passages. Optimal Reciprocal Collision Avoidance (ORCA) [12] is a local planner that avoids collisions by choosing velocities that specifically avoid the trajectories of other robots. ORCA is quick to compute and scales well with the number of agents. In very crowded scenarios, however, ORCA often chooses very small velocities that cause robots to move extremely slowly or become stuck entirely.

A third approach to solving the MRMP problem is to discretize the problem and treat robots as points moving on a grid. This is known as Multi-Agent Path Finding (MAPF), and while it does not produce dynamically feasible paths, it reduces the computational complexity of the problem significantly. Recent work has shown that *experience* can be used to speed up planning in MAPF by computing a database of solutions to various template problems and then querying that database online to avoid collisions [8] [5]. Databases have also been leveraged in the single robot case for sampling based planning [3, 2], and to solve the inverse kinematics problem [6].

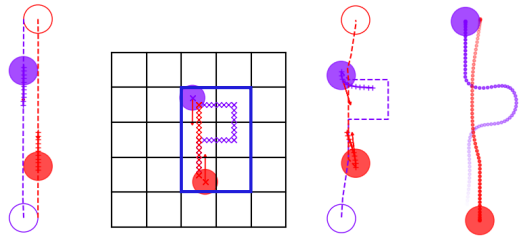


Fig. 1: Two robots (solid circles) moving toward their goals (empty circles) by following their respective guide paths (dashed lines). In (a), if the robots continue along their proposed trajectories (+ line), they will collide. (b) shows a grid placement, chosen subproblem (blue rectangle), and corresponding database solution. In (c), the robots' guide paths are rerouted using the database solution, and the proposed trajectories are updated. (d) shows the final trajectories of the robots with starts shown in solid circles and a gradient indicating start (opaque) to goal (translucent).

In this work, we utilize the experience-based MAPF solver detailed in [8] to guide planning in the continuous space. We present a framework for grid placement and conflict resolution using discrete MAPF solutions. We present an initial proof of concept of our algorithm with a comparison against ORCA.

II. Algorithm Overview

We approach MRMP as a path tracking problem. First, we compute decoupled guide paths for each robot and begin to advance the robots along these paths using MPC (Section II-B). At each iteration of the algorithm, MPC returns a trajectory for the next time horizon H. We then check these "proposed" trajectories for conflicts. When conflicts arise, the algorithm:

- Creates a discrete version of the conflict by placing a grid and assigning robots to grid cells (Section II-C1)
- Uses the planner described in [8] to get a discrete solution (Section II-C2)
- Uses this solution to reroute the guide paths and generates new trajectories for the affected robots by running MPC on the new guide paths (Section II-C2)

- Checks that all proposed plans are safe. If a proposed plan is not safe, it is rejected and the robot switches to a safe braking trajectory (Section II-D)

We execute the first Δt step of the trajectory, and then repeat until the robots reach their goals.

A. Problem Definition

We consider N robots with the state of robot i denoted by $x^{(i)} \in \mathcal{X}^{(i)}$. Robots are actuated by controls $u^{(i)} \in \mathcal{U}^{(i)}$. We assume that the robots obey some dynamics given by $\dot{x}^{(i)} = f^i(x^i, u^i)$. Every agent is a circular robot with some radius r and state (x, y, θ, v) where $(x, y) \in \mathbb{R}^2$ is the position, θ is the heading, and v is the velocity. The controls for each robot are acceleration and angular velocity (a, ω).

B. Path Tracking with MPC

First, we use a single agent sampling-based planner to generate a *guide path* $L^{(i)}$ for each robot $i \in 1 \dots N$. This guide path consists of points in 2D space that begin at robot i's start and terminate at robot i's goal. Given these paths, we create a reference trajectory for each robot, $x^{(i)}_{ref}$, which is the next $K = H/\Delta t$ points on the guide path that robot i would follow from its current position.

We then use MPC to produce a feasible trajectory that follows $x^{(i)}_{ref}$ as closely as possible. The cost function for the MPC problem is given by

$$C = \sum_{i=0}^{N-1} \sum_{k=0}^{K} (x^{(i)}_{ref_k} - x^{(i)}_k)^\intercal Q (x^{(i)}_{ref_k} - x^{(i)}_k) + (u^{(i)}_k)^\intercal R u^{(i)}_k \tag{1}$$

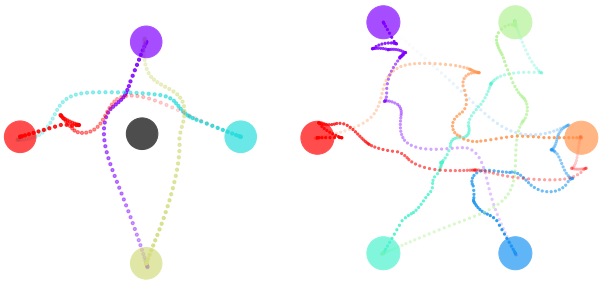
where Q is the state cost matrix and R is the control cost matrix. Note that, to further assist in collision avoidance, we can also include terms in the cost function that penalize the distance between robots and the distance to obstacles. In this work, we omit these terms and rely completely on the database solver for collision avoidance. The optimization problem is subject to the following constraints: $x^{(i)}_{k+1} = x^{(i)}_k + f^{(i)}(x^{(i)}_k, u^{(i)}_k)$, $x^{(i)}_k \in \mathcal{X}^{(i)}$, and $u^{(i)}_k \in \mathcal{U}^{(i)}$.

C. Conflict Resolution

A conflict occurs if $||x^{(i)proposed}_k - x^{(j)proposed}_k|| \leq 2r$ for any robot $i \neq j$ and $k \in 0 \dots K$. For each conflict, we first place a grid and then we provide that grid as input to the discrete solver detailed in [8], which will find a subproblem and corresponding solution.

1) Grid placement: When placing a grid, we use the following guidelines:

- The grid may not overlap with a previously placed subproblem
- All robots involved in the conflict must be contained within the grid.
- Grid indices assigned to robots may not be the same as grid indices assigned to obstacles
- Robots should be centered within their grid cell as much as possible.



(a) (b)

Fig. 2: Resulting trajectories for the 4 robot swap with an obstacle (a) and the 6 robot swap (b). Each robot must swap with its opposite. Solid circles show the starts and the gradient indicates start (opaque) to goal (translucent)

	4 robots	**4 robots obstacle**	**6 robots**
Ours	10.7	13.7	14.0
ORCA	–	19.7	–

TABLE I: Time (seconds) taken for robots to reach their goals for different problems. Dashes indicate timeout after 5 minutes.

We solve a Mixed Integer Program with the above criteria to find the best possible grid placement.

2) Database lookup and rerouting: After placing the grid and assigning robots to grid cells, we query the discrete database solver for a solution. Discrete solutions are lists of grid cell indices $p = (n, m)$ where n is the row and m is the column. The continuous space location of these points is given by $g + p \cdot h \cdot \frac{3}{2}$ where g is the origin of the grid. We reroute the guide paths to follow these discrete solutions. Since this rerouting will likely cause the robot to deviate from its original guide path, we replan using RRT* to ensure that the last point in the new solution is connected to the robot's final goal. Lastly, we update $X^{(i)proposed}$ for each affected robot i to follow the new guide path.

D. Safety Guarantees

It is possible that there will not be a valid grid placement for a given conflict. In this case, we rely upon the concept of contingency plans to guarantee the safety of the robots' chosen trajectories. Grady et al. [4] proposes a safe asynchronous multi-robot planning framework in which robots choose plans for themselves that cannot result in collisions in the future. We adopt a centralized version of their algorithm that guarantees safety.

III. Preliminary Experiments

We demonstrate our algorithm in a classic example in which robots are in a circular formation, and each robot must swap with its opposite. We demonstrate with 4 robots, 6 robots, and 4 robots with an obstacle. Table I shows the results on these scenarios for both ORCA and our planner. These scenarios are particularly challenging for ORCA since they are extremely congested. When all of the robots meet in the middle, ORCA begins to choose extremely small velocities and gets stuck, whereas our planner has access to a database that can quickly reroute the robots. Fig. 2 shows the resulting trajectories from our algorithm for two scenarios.

References

[1] Wolfram Burgard, Mark Moors, Dieter Fox, Reid Simmons, and Sebastian Thrun. Collaborative multi-robot exploration. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 1, pages 476–481. IEEE, 2000.

[2] Constantinos Chamzas, Zachary Kingston, Carlos Quintero-Peña, Anshumali Shrivastava, and Lydia E. Kavraki. Learning Sampling Distributions Using Local 3D Workspace Decompositions for Motion Planning in High Dimensions. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1283–1289, 2021. doi: 10.1109/ICRA48506.2021.9561104.

[3] David Coleman, Ioan A. Şucan, Mark Moll, Kei Okada, and Nikolaus Correll. Experience-based planning with sparse roadmap spanners. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 900–905, 2015. doi: 10.1109/ICRA.2015.7139284.

[4] Devin K Grady, Kostas E Bekris, and Lydia E Kavraki. Asynchronous distributed motion planning with safety guarantees under second-order dynamics. In *Algorithmic Foundations of Robotics IX: Selected Contributions of the Ninth International Workshop on the Algorithmic Foundations of Robotics*, pages 53–70. Springer, 2010.

[5] Shuai D Han and Jingjin Yu. Ddm: Fast near-optimal multi-robot path planning using diversified-path and optimal sub-problem solution database heuristics. *IEEE Robotics and Automation Letters*, 5(2):1350–1357, 2020.

[6] Kris Hauser. Learning the problem-optimum map: Analysis and application to global optimization in robotics. *IEEE Transactions on Robotics*, 33(1):141–152, 2016.

[7] Jiaoyang Li, Andrew Tinka, Scott Kiesel, Joseph W Durham, TK Satish Kumar, and Sven Koenig. Lifelong multi-agent path finding in large-scale warehouses. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11272–11281, 2021.

[8] Rachel A Moan, Courtney McBeth, Marco Morales, Nancy M Amato, and Kris Hauser. Experience-based Multi-Agent Path Finding with Narrow Corridors. In *Robotics Science and Systems (RSS)*, 2024.

[9] Jorge Pena Queralta, Jussi Taipalmaa, Bilge Can Pullinen, Victor Kathan Sarker, Tuan Nguyen Gia, Hannu Tenhunen, Moncef Gabbouj, Jenni Raitoharju, and Tomi Westerlund. Collaborative multi-robot search and rescue: Planning, coordination, perception, and active vision. *Ieee Access*, 8:191617–191643, 2020.

[10] Indranil Saha, Rattanachai Ramaithitima, Vijay Kumar, George J Pappas, and Sanjit A Seshia. Automated composition of motion primitives for multi-robot systems from safe LTL specifications. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1525–1532. IEEE, 2014.

[11] Ashley Stroupe, Terry Huntsberger, Avi Okon, Hrand Aghazarian, and Matthew Robinson. Behavior-based multi-robot collaboration for autonomous construction tasks. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1495–1500. IEEE, 2005.

[12] Jur Van den Berg, Ming Lin, and Dinesh Manocha. Reciprocal velocity obstacles for real-time multi-agent navigation. In *2008 IEEE international conference on robotics and automation*, pages 1928–1935. Ieee, 2008.

[13] Dustin J Webb and Jur Van Den Berg. Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics. In *2013 IEEE international conference on robotics and automation*, pages 5054–5061. IEEE, 2013.