Finding Near-optimal Solutions in Multi-robot Trajectory Planning

Michal Čáp¹, Peter Novák², Alexander Kleiner³

Abstract—We deal with the problem of planning collision-free trajectories for robots operating in a shared space. Given the start and destination position for each of the robots, the task is to find trajectories for all robots that reach their destinations with minimum total cost such that the robots will not collide when following the found trajectories. Our approach starts from individually optimal trajectory for each robot, which are then penalized for being in collision with other robots. The penalty is gradually increased and the individual trajectories are iteratively replanned to account for the increased penalty until a collision-free solution is found. Using extensive experimental evaluation, we find that such a penalty method constructs coordinated non-colliding trajectories for robots operating in a shared space. Given the start and destination position for each of the robots, the task is to find trajectories that are too close to trajectories of other robots are penalized. Starting from trajectories that disregard collisions with other robots, the collisions between robots’ trajectories are gradually being penalized with increasing severity so that they are finally forced out of collision as the penalties tend to infinity. Using extensive experiments, we demonstrate that this heuristic approach tends to generate near-optimal trajectories that are of significantly lower cost than the trajectories generated by prioritized planning and reactive techniques.

I. INTRODUCTION

An important problem in multi-robotics is the coordination of trajectories of individual robots. That is, given \( n \) circular robots, together with their starting and destination positions, we are interested in finding a set of individual trajectories \( \pi_1, \ldots, \pi_n \) that do not collide with each other, i.e., \( \forall i,j \forall t : |\pi_i(t) - \pi_j(t)| > d_{\text{sep}} \), where \( d_{\text{sep}} \) is the required separation distance – usually the sum of radii of the two robots, while at the same time the overall costs (e.g., sum of trajectory lengths) is minimized.

It is known that path coordination of circular vehicles among polygonal obstacles is NP-hard [4]. While the problem is relatively straightforward to formulate as a planning problem in the Cartesian product of the state spaces of the individual robots, the solutions are difficult to find using standard search techniques because the joint state-space grows exponentially with the number of robots.

Prioritized planning [2] is a heuristic approach based on the idea of sequential planning for the individual robots. This prioritizes the higher-priority robots as moving obstacles and plans its trajectory to avoid them. While fast, prioritized planning is incomplete and often fails to find a solution even if one exists. Further, the resulting trajectories are typically noticeably suboptimal.

The techniques of mathematical optimization such as the penalty-based approach [3] have been also studied in the context single-robot and multi-robot trajectory generation. In particular, a distributed penalty-based method has been used to solve the multi-robot rendezvous problem [1].

We use similar distributed penalty-based approach to find coordinated non-colliding trajectories for multiple robots and propose the \( k \)-step penalty method that can be seen as a generalization of prioritized planning approach. The algorithm performs a series of single-robot path planning queries in a dynamic environment such that the trajectories that are too close to trajectories of other robots are penalized. Starting from trajectories that disregard collisions with other robots, the collisions between robots’ trajectories are gradually being penalized with increasing severity so that they are finally forced out of collision as the penalties tend to infinity. Using extensive experiments, we demonstrate that this heuristic approach tends to generate near-optimal trajectories that are of significantly lower cost than the trajectories generated by prioritized planning and reactive techniques.

II. PENALTY-BASED METHOD

We propose a penalty-based approach that attempts to mitigate the low success rate and low solution quality of prioritized planning, but in the same time retain its tractability. We combine the idea of decoupled planning as used in prioritized planning with iterative increasing of penalty assigned to each robot in collision.

The requirement on minimal separation between trajectories of a pair of robots \( i,j \) is modelled by a penalty function assigning penalty to each part of the trajectory of robot \( i \) that gets closer to the trajectory of robot \( j \) than the required separation distance \( d_{\text{sep}} \). The penalty function has the form

\[
\Omega_{ij}(\pi_i, \pi_j) = \int_0^{\infty} \omega_{ij}(|\pi_i(t) - \pi_j(t)|) \, dt,
\]

where \( \pi_i(t) \) is a trajectory of robot \( i \) and \( \omega_{ij}(d) \) is a bump function:

\[
\omega_{ij}(d) = \begin{cases} \frac{1}{\pi_1} \cdot e^{-\frac{1}{(d/d_{\text{sep}})^2}} & \text{for } d < d_{\text{sep}} \\ 0 & \text{otherwise} \end{cases}
\]

Algorithm 1 exposes the \( k \)-step Penalty Method (PM) algorithm that replans the trajectory of each robot exactly \( k \)-times. The algorithm starts by finding a cost-optimal trajectory for each robot using \( w = 0 \), i.e., while ignoring interactions with other robots. Then, it gradually increases the weight \( w \) and thus the penalties start to be taken into account. After each increase of the weight coefficient, one of the robots is selected and its trajectory is replanned to account for the increased penalty. The trajectory is planned by performing space-time search in time-extended roadmap using A*.

Algorithm 1

1. Agent Technology Center, Dept. of Computer Science, Faculty of Electrical Engineering, CTU in Prague
2. Algorithmics, EEMCS, Delft University of Technology
3. iRobot Inc., Pasadena
Algorithm 1: k-step Penalty Method

1. \textbf{Algorithm PM}(k)
2. \textbf{for } i \leftarrow 1 \ldots n \textbf{ do } \pi_i \leftarrow \text{Replan}(i, 0)
3. \textbf{for } i \leftarrow 1 \ldots n(k-2) \textbf{ do }
4.     w_i \leftarrow \tan\left(\frac{n(k-2) + i}{2}\right)
5.     \pi_i \leftarrow \text{Replan}(i, w_i)
6. \textbf{for } i \leftarrow 1 \ldots n \textbf{ do } \pi_i \leftarrow \text{Replan}(i, \infty)
7. \textbf{if } \forall j \Omega_{ij}(\pi_i, \pi_j) = 0 \textbf{ then return } \langle \pi_1, \ldots, \pi_n \rangle
8. \textbf{else report failure}

9. \textbf{Function} Replan(r, w)
10. \textbf{return} trajectory \pi for robot r that minimizes $c(\pi) + w \sum_{j \neq r} \Omega_{ij}(\pi, \pi_j)$

Fig. 1. Experimental environments

Fig. 2. Results. The plot shows: Success rate: the percentage of instances successfully solved by PM($k = 3, 20, 10$) and PP. Suboptimality: average suboptimality of solution generated by PM($k = 1, \ldots, 100$) and PP on instances where optimum was known. Time out of goal: average difference in solution quality generated by PM($k = 1, \ldots, 100$), PP, and ORCA on instances with 10 robots in Scenario B.

In future, we plan to focus on the investigation of theoretical properties of the method on our problem.

REFERENCES


