Multi-Agent Motion Planning Through Stationary State Search (Extended Abstract)

Jingtian Yan, Jiaoyang Li
Carnegie Mellon University
jingtianyan@cmu.edu, jiaoyangli@cmu.edu

Introduction

We define Multi-Agent Motion Planning (MAMP) in the warehouse by an undirected graph $G = (V, E)$ and a set of agents $\mathcal{A}$. We use a four-neighbor grid map to represent G, adopting the classical MAPF grid model. Agents can possess a set of heading orientations denoted by Θ, corresponding to the four cardinal directions on G. An agent can move from vertex $v_i \in V$ to $v_j \in V$ along edge $(v_i, v_j) \in E$ if (v_i, v_j) aligns with its current orientation. Each agent a_i has a *start* $v_i^s \in V$ and a *goal* $v_i^g \in V$. We define the *path segment* as a sequence of vertices that forms a straight line. We define the *spatio-temporal profile* as a function measuring the distance an agent covers over time on a path segment, constrained by kinodynamic limits on speed and acceleration. We use a holonomic robot model commonly used in warehouses where agents can perform the following actions:

Definition 1. *(Move) For a path segment $\phi_{i,j}$, move(v_i, v_j) allows a stoped agent to move forward from v_i and stop at v_j with a kinodynamically-feasible spatio-temporal profile.*

Definition 2. *(Rotate) A rotate action lets an agent change its orientation on its current vertex.*

A *plan* is a sequence of actions that leads an agent from its start to goal. We use arrival time to indicate the time needed for an agent to arrive at its goal. We define a collision happens if the time duration during which two agents occupy the same vertex overlap. Our task is to find collision-free plans for all agents while minimizing the sum of their arrival time.

Since holonomic robots can only change their orientations through rotation. The plan of each agent is always composed of alternating between rotation and movement. Thus, the states between two actions (finishing a movement to start rotation or the reverse) show significant importance, which we refer to as *stationary states*, as the velocity of the agent at these states is zero. Our idea is to find the plan by finding the stationary states and actions that connect them. Based on this idea, we introduce the Stationary Safe Interval Path Planner (SSIPP), which searches for a kinodynamically feasible plan for individual agents. By combining SSIPP with a high-level collision solver, we propose **P**BS-**S**SIPP-**S**PS (PSS), a three-level planner to address the MAMP problem.

PBS-SSIPP-SPS (PSS)

At Level 1 of PSS, we use the MAPF-based algorithm to resolve agent collisions by finding constraints for single-agent plan finding. Specifically, we use PBS to resolve collisions through priority ordering searching. At Level 2 and Level 3, we try to find the plan for each agent while satisfying the constraints imposed by Level 1. We use SSIPP at Level 2 to search for individual agent plans, where the spatio-temporal profiles of the actions in the plan are optimized at Level 3. This section explores the details of Level 2 and Level 3.

Stationary SIPP (SSIPP)

The task of Level 2 is to find a plan for a single agent with optimal arrival time while avoiding collisions with higher-priority agents given by Level 1. Level 2 performs a Stationary SIPP (SSIPP) search on a safe interval graph. This graph associated each vertex with a set of safe intervals, which are time intervals not reserved by agents with higher priority ordering. In SSIPP, each search node contains the following information: the current vertex and orientation of the agent, the associated safe interval, and the previous action that leads the agent to the current node. We define a safe interval as stationary if the agent is in a stationary state upon reaching the associated vertex within that interval. Compared to standard SIPP (Phillips and Likhachev 2011), SSIPP uses stationary node expansion to find the stationary states and kinodynamically feasible action between them. Since agents can change their orientation only by in-place rotation, the plans always alternate between movement and rotation. At each stationary state, the agent must perform an action different from its previous action; otherwise, two identical actions can be combined into one. Accordingly, stationary node expansion includes two types: movement expansion and rotation expansion. Rotation expansion finds all neighbor nodes reachable through rotation, while movement expansion does the same for movement.

Rotation Expansion For the four-neighbor grid model, we only have four orientations. During rotation expansion, we can apply the predefined rotation speed profiles (rotate 90°, 180°, and 270°) to generate the neighboring states. We generate the new neighbor nodes based on those states.

Movement Expansion During movement expansion, we first find the vertices and safe intervals that can be reached

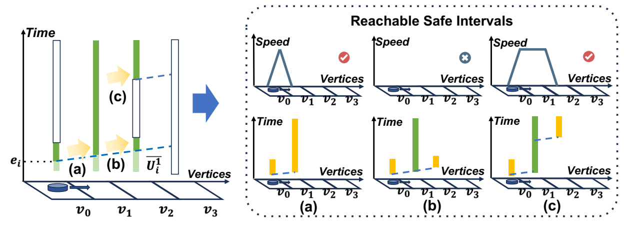

Figure 1: Illustration of the movement expansion. The shadowed strips are time intervals occupied by other agents, the green segments are safe intervals. The yellow strips denote the stationary safe intervals.

through movement from the current node. Then, for those safe intervals, we assume they are stationary safe intervals and backtrack to retrieve the safe intervals leading from the current node to it, as shown in Fig. 1 (a) (b) (c). We use Level 3 to find the kinodynamically feasible spatio-temporal profile for each stationary safe interval. If a spatio-temporal profile is found, we generate a new SSIPP node based on this stationary safe interval and the corresponding action.

To achieve this, we use a breadth-first search on safe intervals along the movement orientation. We use an example to illustrate this process as shown in Fig. 1. We begin by initializing the root interval using the interval of the current node and push it to the open list. During each iteration, we pop an interval from the open list and expand it by assuming the agent moves one vertex forward. In our case, we first expand the interval at v_0 denoted as $[lb_0, ub_0)$, where lb_0 and ub_0 are the lower and upper bounds for the safe interval. As the agent moves from v_0 to v_1, a new interval $[lb_0+t_{min}, \infty)$ is generated at v_1, where t_{min} is the minimum time required for this movement. Since the kinodynamic constraints are considered in Level 3, we can use relaxed kinodynamic constraints to expedite this expansion process without compromising the guarantee of completeness. Specifically, we estimate t_{min} as the time the agent takes to move at maximum speed. For safe intervals at v_1 with a lower bound smaller than ub_0, we identify and assume the overlap between these intervals and $[lb_0 + t_{min}, \infty)$ as stationary safe intervals. We get the safe intervals shown in Fig. 1 (a) in our example. Then, we call Level 3 to find the spatio-temporal profile that travels within those safe intervals while satisfying the kinodynamic constraints. If a feasible spatio-temporal profile is found, we generate a new SSIPP node using the new stationary interval and this profile. In the next iteration, we continue to expand the safe intervals at v_1. This search process proceeds recursively until no stationary safe intervals can be found.

Spatio-Temporal Profile Solver (SPS)

Given the path segment and its associated temporal constraints from Level 2, the task of Level 3 is to find a spatio-temporal profile that is kinodynamically feasible and finish in the shortest time. We use two solvers in our method: Binary Acceleration Solver and Bézier-Curve Solver. Notably, this framework is adaptable to other SPS.

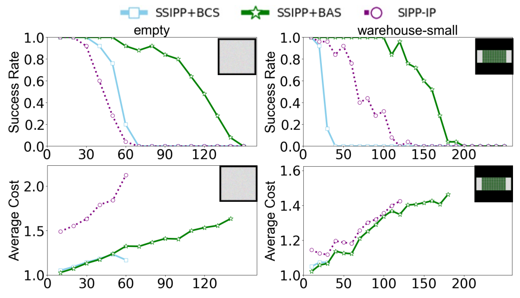

Figure 2: Success rate and solution cost across all maps. The solution cost is averaged only over scenarios where the planner successfully generates a solution.

Binary Acceleration Solver (BAS) This solver assumes that the agent begins by waiting at the first vertex of the path segment for a specified waiting time. Then, it always moves with its maximum acceleration until it reaches its maximum speed, and only decelerates with maximum deceleration to stop at the last vertex. The objective is to minimize the waiting time. BAS is an incomplete but fast method.
Bézier-Curve Solver (BCS) We borrow BCS from Level 3 of (Yan and Li 2024). BCS models the spatio-temporal profile using the scaled Bézier curve, which can approximate any continuous function within its feasible range with sufficient control points. BCS is a complete but slow method.

Empirical Evaluation

We compare PSS with SIPP-IP (Ali and Yakovlev 2023). SIPP-IP also uses PBS for solving collisions and uses motion primitives for low-level single-agent plan searches. All agents share the same kinodynamic constraints, with speed limits of $[0, 2]$ $grid/s$ and acceleration limits of $[-0.5, 0.5]$ $grid/s^2$. Fig. 2 presents the *success rate* and *solution quality* on a 32×32 empty map and a 161×63 warehouse map. We evaluate solution quality by dividing the total arrival time of all agents by the sum of the best possible arrival times for each agent. PSS with BAS shows a better success rate than both PSS with BCS and SIPP-IP. At the same time, both PSS with BCS and PSS with BAS outperform SIPP-IP in terms of solution quality.

Acknowledgements

The work was supported by NSF Grant 2328671.

References

Ali, Z. A.; and Yakovlev, K. 2023. Safe Interval Path Planning with Kinodynamic Constraints. In *Proceedings of the AAAI*, volume 37, 12330–12337.

Phillips, M.; and Likhachev, M. 2011. SIPP: Safe interval path planning for dynamic environments. In *Proceedings of the ICRA*, 5628–5635.

Yan, J.; and Li, J. 2024. Multi-Agent Motion Planning with Bézier Curve Optimization under Kinodynamic Constraints. *IEEE Robotics and Automation Letters*, 9(3): 3021–3028.