LSRP*: Scalable and Anytime Planning for Multi-Agent Path Finding with Asynchronous Actions (Extended Abstract)

Shuai Zhou[2*], **Shizhe Zhao**[1], **Zhongqiang Ren**[1,3†]

[1]UM-SJTU Joint Institute, Shanghai Jiao Tong University, China
[2]SHIEN-MING WU School of Intelligent Engineering, South China University of Technology, China
[3]Department of Automation, Shanghai Jiao Tong University, China
davidzhou718@gmail.com, {shizhe.zhao,zhongqiang.ren}@sjtu.edu.cn

Introduction

Multi-Agent Path Finding (MAPF) finds collision-free paths for multiple agents on a graph, assuming all agents move synchronously with equal action duration. This may limit real-world use, where agents often have different speeds or edge traversal times. To bypass this assumption on synchronous actions, a few variants of MAPF were studied, such as Continuous-Time MAPF (Andreychuk et al. 2022), MAPF with Asynchronous Actions (MAPF-AA) (Ren, Rathinam, and Choset 2021), MAPF$_R$ (Walker, Sturtevant, and Felner 2018). Most of these algorithms seek an optimal or bounded sub-optimal solution at the cost of limited scalability as the number of agents grows. Our prior work (Zhou, Zhao, and Ren 2025) developed an unbounded sub-optimal planner Loosely Synchronized Rule-based Planning (LSRP) for MAPF-AA. LSRP can handle a large number of agents but often finds poor quality solutions due to its unbounded sub-optimality. In this work, we introduce LSRP*, an anytime variant of LSRP that can keep improving solution quality till a given runtime budget depletes.

Anytime algorithms that can iteratively improving solution quality exist for MAPF, e.g., LaCAM (Okumura 2023). However, developing such an algorithm for MAPF-AA is non-trivial. MAPF-AA adopts duration conflict, where an agent occupies both its departure and arrival vertices throughout the duration of an action, and a conflict arises when two agents occupy the same vertex during overlapping intervals. The planner must account for the varying time dimensions of different agents and avoid duration conflicts, resulting in a significantly more complex state space than in MAPF. Fig 1 demonstrates an example, goal location of agents A_1, A_2 are vertices D, C respectively. Fig 1 (a) and (b) shows a plan with costs of 100 and 78, respectively. In both cases, the same joint vertex $v = (\mathrm{B}, \mathrm{E})$ is reached, where plan (a) has a smaller cost to v ($(10, 40)$ vs. $(13, 45)$). If we apply pruning in classical MAPF, which is purely based on the cost at joint vertices, plan (b) would be discarded, missing a better solution.

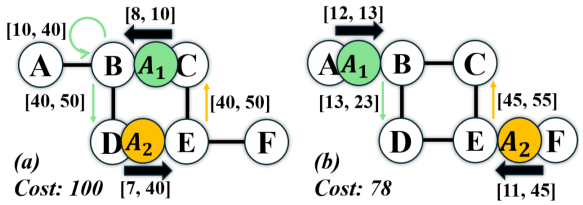

Figure 1: Goal locations of agents A_1, A_2 are D, C respectively. Black arrows denote current actions, with brackets showing their departure and arrival times; colored arrows represent each agent's subsequent path to its goal. (a) and (b) show two possible partial plans during the search, both reaching the same joint vertex (B, E). In (a), A_1 waits at vertex B to avoid a duration with A_2, resulting in a cost of 100. In (b), both agents reach the joint vertex later but incur a smaller total cost of 78.

To address this, LSRP* employs the successor generation and pruning strategy of LSS (Ren, Rathinam, and Choset 2021) to systematically explore the state space, ensuring both optimality and completeness. Experiments show that LSRP* achieves up to 25× greater scalability than existing baselines and can reduce solution costs by up to 40% within a reasonable time budget.

Method

LSRP* searches in a time-augmented join state space with three components:

- *Search Tree*: The search node n of a search tree represents the joint action (either move or wait) that all agents are performing. Let $I_{curr}(n)$ denote the set of agents who have the minimum action finish time (either move or wait). All possible joint actions of the agents in $I_{curr}(n)$ define the children of n in the search tree.
- *Action Tree*: All joint actions of agents in $I_{curr}(n)$ are generated by an action tree AT. Each depth of AT is associated with the a specific agent in $I_{curr}(n)$, and the max depth of AT is the size of $I_{curr}(n)$. The children of a tree node are generated by enumerating all possible collision-free actions that consider all determined actions from ancestors. Fig 2(b) illustrate an example.

*Shuai Zhou conducted this research during his internship at UM-SJTU Joint Institute at Shanghai Jiao Tong University.

†Corresponding Author.

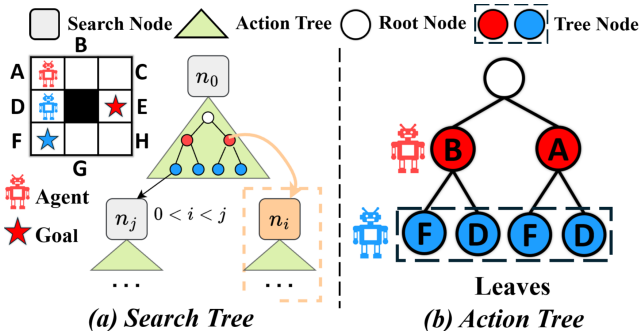

Figure 2: A toy example showing the search tree of LSRP* and the action tree of the initial search node. (a) The orange search node has actions planned by LSRP, and all gray search nodes only have the actions determined by action tree expansion.(b) A fully expanded action tree of the initial search node n_0.

- *Rule-based Planner*: The search space increases exponentially with the number of agents, so LSRP* explores the search tree in a depth-first manner due to the memory constraint. However, this may cause the search to take a large number of iterations to find a feasible solution. To resolve this issue, LSRP* employs LSRP to quickly find feasible solutions. Fig 2(a) illustrate an example.

Remark 1. *Leaf nodes of an action tree is equivalent to* neighbors *in LSS, which implies that LSRP* inherits all theoretical properties of LSS, e.g., optimality. The differences are: 1. LSS searches in a best-first manner, while ours is depth-first; 2. The search tree of LSRP* also includes successors that are non-leaf nodes of an action tree, which are generated by LSRP. Additionally, the rule-based planner can be replaced by other variants of LSRP, e.g. LSRP-SWAP (denote as LSRP*-SWAP) for better performance.*

Experiment Results

We evaluate LSRP* and LSRP*-SWAP on four maps. For each map, we run 20 instances with varying number of agents N and compare against three baselines: (1) LSRP and LSRP-SWAP, (2) CCBS (Andreychuk et al. 2022), where we do the same modification as (Zhou, Zhao, and Ren 2025) and choose it over LSS due to its better scalability. We evaluate scalability and solution quality of LSRP* and LSRP*-SWAP under runtime limits of 30 and 60 seconds, respectively. As shown in Fig. 3, LSRP*-SWAP scales best, reaching the benchmark's upper agent limit across all maps, especially on sparse graphs like Room64 and Den312d. In cluttered environments such as Warehouse and Random32, LSRP*-SWAP still performs competitively and outperforms other methods including LSRP*. For solution quality (Fig. 4), we use sum of costs (SoC) and compare LSRP*-SWAP against LSRP-SWAP and CCBS on instances that are successfully solved by both planners. LSRP*-SWAP consistently produces results up to 40% cheaper than LSRP-SWAP and is close in quality to CCBS, with differences not exceeding 40%. With many agents (e.g., 1000 agents), the advantage of LSRP*-SWAP decreases as the time budget becomes insufficient for further optimization.

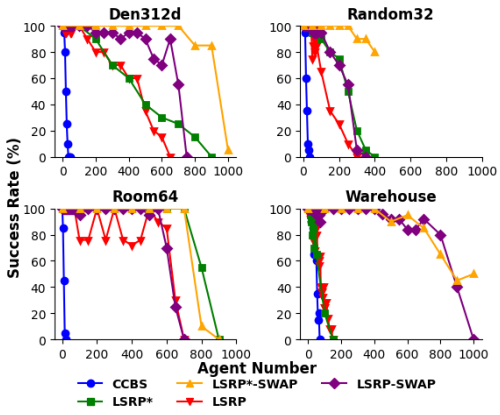

Figure 3: Success Rate Results

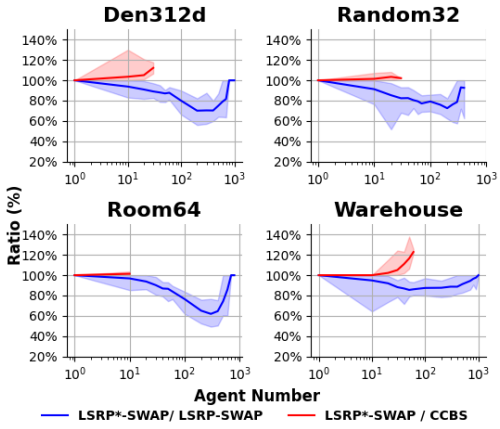

Figure 4: Solution Cost Ratio Results: solution cost that LSRP* finds against baseline LSRP-SWAP or CCBS.

References

Andreychuk, A.; Yakovlev, K.; Surynek, P.; Atzmon, D.; and Stern, R. 2022. Multi-agent pathfinding with continuous time. *Artificial Intelligence*, 305: 103662.

Okumura, K. 2023. LaCAM: Search-Based Algorithm for Quick Multi-Agent Pathfinding. In *AAAI*.

Ren, Z.; Rathinam, S.; and Choset, H. 2021. Loosely synchronized search for multi-agent path finding with asynchronous actions. In *2021 IROS*, 9714–9719. IEEE.

Walker, T. T.; Sturtevant, N. R.; and Felner, A. 2018. Extended Increasing Cost Tree Search for Non-Unit Cost Domains. In *IJCAI*, 534–540.

Zhou, S.; Zhao, S.; and Ren, Z. 2025. Loosely Synchronized Rule-Based Planning for Multi-Agent Path Finding with Asynchronous Actions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(14): 14763–14770.