# The League of Robot Runners: Competition Goals, Designs, and Implementation [*]

**Shao-Hung Chan[1], Zhe Chen[2], Teng Guo[3], Han Zhang[1], Yue Zhang[2], Daniel Harabor[2], Sven Koenig[1], Cathy Wu[4], Jingjin Yu[3]**

[1] University of Southern California, USA. [2] Monash University, Australia.
[3] Rutgers University, USA. [4] Massachusetts Institute of Technology, USA
{shaohung, zhan645, skoenig}@usc.edu, {zhe.chen, yue.zhang, daniel.harabor}@monash.edu,
{gt286, jingjin.yu}@cs.rutgers.edu, cathywu@mit.edu

## Abstract

Multi-Agent Path Finding (MAPF) is an important practical problem found in many application settings, from logistics and transportation to robotics and automation. In this paper, we introduce the goals, designs and implementations of the League of Robot Runners (LoRR) [1], a competition to foster and advance this research area. LoRR aims to identify the core challenges for solving MAPF, develop suitable benchmark instances, evaluate algorithmic performance and track the state-of-the-art. The competition provides participants with a standardised system to develop, evaluate, and compare algorithmic techniques. Submissions, solutions and problem instances are all open sourced, to lower barriers, promote dissemination and enable further advancements.

## Introduction

Multi-Agent Path Finding (MAPF), an important problem for many new and emerging industrial applications. Research on MAPF and closely related problem variants has grown exponentially in recent years, including solvers, visualisations, and problem instances. Currently, there exist many different models of the problem and algorithmic solutions. Often these works place different emphasis on various parts of the problem, leading to a diversity of perspectives and solution techniques. For example, MAPF can be modelled as a planning problem and solved by algorithms that aim for optimal or near-optimality solutions. It can also be modelled from a robotics perspective, where execution considerations are considered the foremost priority. Although each of these perspectives is valid, the divergence makes it difficult for practitioners, esp. new entrants to the growing field, to have a clear picture of the main challenges in the area, the currently leading techniques and what is considered state-of-the-art on those topics. Thus, LoRR aims to lower the barrier for practitioners to enter this area by addressing the following goals in its design and implementation:

**Identifying core challenges**: MAPF is often solved with a simplified model, which causes a disconnection between the abstract model and real executions when deploying solvers into real applications. Therefore, the first goal of LoRR is to identify the challenges that cause the most "disconnections". In the first round of LoRR, we identify two main challenges: (1) *Turn actions:* Robots are often modelled as unit action cost and free turning cost in MAPF, which is disconnected from real robot motions. The first challenge we identified is turn actions, which are frequent actions in real applications and account for the most increase in the achieved real execution cost (Zhang et al. 2023). (2) *Online lifelong problem:* Typical MAPF solvers often solve a single-shot problem offline while enough planning time is given upfront. We then identify the problem to be *online lifelong*, where the time that agents wait during planning is considered and frequent replanning is needed to adapt typical solvers into practice.

**Standardised benchmarks**: MAPF solvers are often tested using a common benchmark (Stern et al. 2019), which misses some practical applications. Moreover, people often generate individual benchmarks for different variants, making tracking and reusing challenging. Thus, our second goal is to develop standardised benchmarks with high degrees of reproducibility. We consider different domains, map layouts, and task distributions. This allows practitioners to readily identify difficult yet meaningful MAPF challenges.

**Common API**: Working with different problem models, languages, and programming habits, it is generally time-consuming to evaluate/compare/reuse open-source MAPF implementations due to interface variations. We developed a standardised planning and visualisation system allowing users to easily build, run, and visualise existing and new planners through the provided system.

**Tracking Progress**: Evaluating solver performance poses unique challenges due to implementation efforts. Our third goal is let participants to simply compare their algorithms against others through an online evaluation system. In addition, we archive and open-source the code implementations for each round, which helps practitioners to understand, and be inspired by these state-of-the-art techniques.

## Problem Model

We follow most of the settings of a classical MAPF model that agents are moving on a 4-connected grid map and time is discredited into unit-size timesteps except that at each timestep, each agent: (1) has a facing direction and (2) can

[*]Lex order, organising committee then chairs

[1]Competition website: https://www.leagueofrobotrunners.org/
Video demo: https://bit.ly/league-of-robot-runners-demo-video

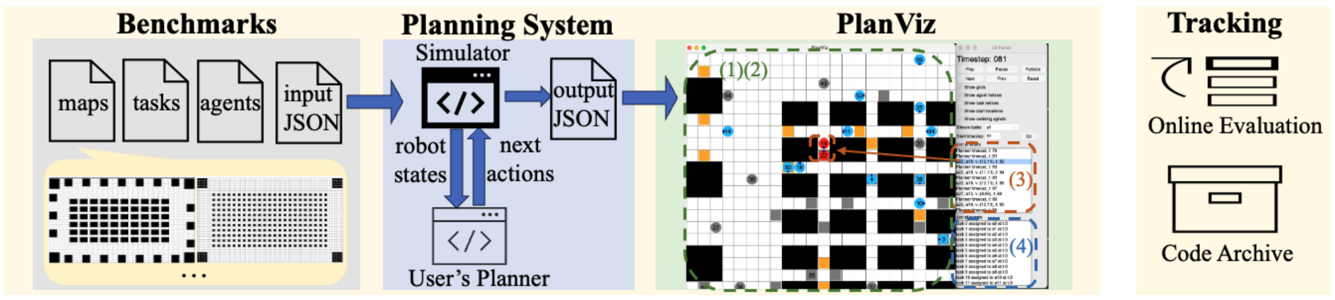

Figure 1: System Overview. Each component is explained in each section.

either perform 90° turns, wait or forward to the next empty cell. We consider the problem to be *online* (solvers have a tight timeout limit and agents wait when timeout) and *lifelong* (a new task appears when an agent finishes its current task). The objective is maximising the total number of tasks finished over a given time horizon.

## Benchmarks

We create our online lifelong problem instances from classical MAPF benchmarks (Stern et al. 2019) and two industrial-inspired domains. Each instance includes a map file, an agent file specifying initial configurations, and a task file containing tasks. Maps feature a diverse range of layouts and domains. For task generation, we employ various task distribution methods like random and distance-based.

**Classical Domains:** Given a map, we generate problem instances from the largest connected component on the map with user-specified numbers of agents and tasks.

**New Benchmarks:** We develop new maps on two realistic problem domains, called **Fulfillment** (where robots pick up and deliver orders in a warehouse) and **Sortation** (where robots sort and move orders in a mail sortation centre). We also develop generators to generate these maps and the associated problem instances with user-specified parameters, including map layouts, team size and task assignments.

## Planning System

The planning system consists of a simulator and a planner. Users need to implement their own algorithms in the planner. The planning system is written in C++, but users can implement bindings for other programming languages. We have provided a Python binding as an example.

**Simulator:** The simulator iteratively calls the planner for the MAPF plan. It includes a validation component to check if a MAPF plan is valid (i.e., does not contain conflicts or runs into obstacles) and executes the plan based on its feasibility. It also includes a task assignment component to decide what happens to an agent after it reaches a target.

**Planner:** The interface between the simulator and the planner is designed to be simplistic and easy to understand. Specifically, a user needs to implement the `initialise` function for map pre-processing and the `plan` function for planning, both subject to some given time limits. In case the `plan` function does not terminate in time, the simulation will proceed to the next timestep (with the `plan` function running), and all agents will wait for the current timestep.

**Input/Output:** The planning system reads input from the benchmark generator and produces an output file that records the plans, planning errors, and simulation events. Both the input and output files are standardised and well-documented. In addition, the output can be used by PlanViz for visualisation, which will be explained in the next section.

## PlanViz

Although there are many tools for visualising a MAPF plan, few of them help understand the errors of the plan, such as collisions between agents and invalid actions. Thus, we design PlanViz to visualise both the MAPF plans (either one-shot or lifelong) and the errors occured.

PlanViz contains a visualisation window and a UI panel for controlling what to show in the visualisation window. PlanViz can visualise: (1) locations (and movements) of agents at each timestep, (2) paths of agents from their start locations to locations at the planning horizon, (3) collisions between pairs of agents, and (4) task assignment. For (3), we label the colliding agents (either all collisions or some pairs of colliding agents specified by the user) and jump to one timestep before the collision occurs.For (4), we label tasks as unassigned, currently assigned, assigned, and finished. This helps participants to better understand algorithm performance for completing tasks. In addition, PlanViz can also visualise paths from MAPF tracker (Shen et al. 2023) for classical MAPF problems. We provide a simple format transform between these two projects.

## Evaluation and Code Archives

LoRR utilises an online evaluation platform that allows participants to submit and evaluate implementations at any time before the competition deadline. A leaderboard is dynamically updated to track the progress of the competition.

At the end of the competition, we collect all the implementations submitted and open-source these implementations as a code archive. It includes the implementations for each entry shown on the leaderboard and the implementations that produce the best solution on each benchmark instance, together with the evaluation results. We believe the code archive will not only foster further research in the field but also provide a valuable resource for newcomers to quickly get started with high-quality MAPF solver.

## References

Shen, B.; Chen, Z.; Cheema, M. A.; Harabor, D. D.; and Stuckey, P. J. 2023. Tracking Progress in Multi-Agent Path Finding. *ArXiv*, abs/2305.08446.

Stern, R.; Sturtevant, N.; Felner, A.; Koenig, S.; Ma, H.; Walker, T.; Li, J.; Atzmon, D.; Cohen, L.; Kumar, T.; et al. 2019. Multi-agent pathfinding: Definitions, variants, and benchmarks. In *SOCS*, volume 10, 151–158.

Zhang, Y.; Harabor, D.; Le Bodic, P.; and Stuckey, P. J. 2023. Efficient Multi Agent Path Finding with Turn Actions. In *Proceedings of the International Symposium on Combinatorial Search*, volume 16, 119–127.