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

In this paper, we introduce the goals, designs and implementations of the League of Robot Runners (LoRR) [1], a competition to foster and advance research area in the area of Multi-Agent Path Finding. LoRR aims to: (i) identify the core challenges for solving MAPF; (ii) develop suitable benchmark instances; (iii) evaluate algorithmic performance in the area and; (iv) track the state-of-the-art. The competition provides a standardised system to develop, evaluate, and compare algorithmic techniques. Submissions, solutions and problem instances from the competition are 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. The problem asks us to coordinate a team of moving agents, from start to target, collision-free. Research in this area has grown exponentially in recent years, with many different models and many algorithmic solutions having been proposed. These works often place emphasis on different parts of the problem, leading to a diversity of perspectives. For example, MAPF can be modelled as a planning problem and solved with optimality guarantees. It can also be modelled from a robotics perspective, where execution considerations are considered the foremost priority. Although each approach 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. LoRR aims to lower barrier for practitioners to enter this area by addressing the following goals in its design and implementation:

(G1) Identifying core challenges: MAPF is often solved using a simplified model of operations (Stern et al. 2019). This allows researchers to focus on main combinatorial difficulties of the problem: the interactions between agents. However these simplifications can lead to a disconnect, between computed solutions and the execution environments where those solutions must be deployed. Therefore, the first goal of LoRR is to identify the challenges that cause the most "disconnections". In its inaugural year (2023) LoRR identifies two main challenges:

- (1) *Turn actions:* In MAPF, robots are often modelled as rotationally invariant agents with unit action costs. This model is disconnected from many applications, where turning actions can substantially increase achieved execution costs (Zhang et al. 2023).
- (2) *Online lifelong problem:* In MAPF, the problem is one-shot and solved entirely offline. Yet real applications are *lifelong* and *online*: agents receive new tasks upon arrival and they must be constantly planned and replanned, so as to maximise a throughput objective.

(G2) Standardised benchmarks: MAPF solvers are evaluated on well known benchmark sets (Stern et al. 2019), but these problems are not reflective of practical applications. Researchers interested in more realistic settings must therefore generate new problems ad-hoc, often for a specific experiment or problem setting. These problems are not collected or tracked, making further evaluation and comparison extremely challenging. Thus our second goal is to develop new standardised benchmarks, which more closely model real applications. We consider different domains, map layouts, and task distributions. These problems offer a variety of meaningful challenges for the community and facilitate direct comparisons between competing techniques.

(G3) Common API: It is generally time-consuming to evaluate/compare/reuse open-source MAPF implementations, as each solver has unique conventions, models, formats and programming styles. We propose a standardised planning and visualisation system, with common conventions, formats and APIs, which allow users to build, run, evaluate and compare new and existing planners.

(G4) Tracking Progress: Establishing solver performance poses unique challenges: there are many potential competitors and experimental setups vary significantly from one paper to the next. Our fourth goal is is to develop an online evaluation system that can act as a common denominator. We further archive and open-source the submitted implementations and best solutions, which helps practitioners build on successful ideas and track progress over time.

*Lex order, organising committee then chairs

[1]Competition website: https://www.leagueofrobotrunners.org/ Video demo: https://youtu.be/Y-v3h_27PXk

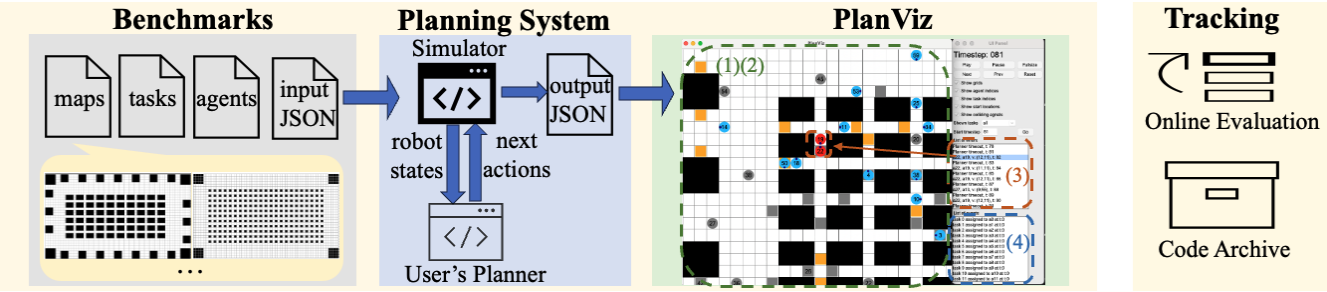

Figure 1: System Overview. Each component is explained in each section.

Problem Model

Similar to classical MAPF, our problem is situated on a 4-connected grid with unit-cost actions and time-steps. Different from classical MAPF, each agent has a facing direction and must execute one of three possible actions: move forward, wait in place or turn 90°. We consider the problem to be *online* (solvers are subject to timeout limits and agents wait on 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 problem instances using two types of maps: those drawn from classical MAPF benchmarks (Stern et al. 2019) (`games`, `city` and `random` layouts) and newly generated maps from two industrial-inspired domains: `fulfilment` (robots pick orders in a warehouse) and `sortation` (robots move packages in a mail sortation centre).

Each problem instances comprises a map, an agent file (specifying initial configurations) and a task file (specifying goal locations, according to various distributions). Problem sizes range from 100 agents up to 10,000 agents, depending on the type of challenge. We also develop and open source problem generators, which can be used to create new instances with user-specified input parameters (e.g., different map layouts, team sizes and agent/task distributions).

Planning System

Our system has two components: a simulator and a planner. The **simulator** tracks the positions of agents on the map and decides which agent is assigned what task. Each timestep, the simulator calls the planner, asking for the next action of each agent. The simulator checks if the set of planned actions are valid (i.e., collision-free) and, if so, updates the position of each agent – otherwise the agents wait in place. The **planner** is an interface between the simulator and a user-implemented solving technique. There are only two functions for users to define: `initialise`, which facilitates map preprocessing, and and `plan`, which returns a next set of actions for agents to execute. Both functions are subject to time limits. In case the `plan` function does not return before the time limit is reached the simulation will proceed to the next timestep, and all agents will wait in place. The planner and simulator are both written in C++. We also provide planner bindings for Python, as an example of how to integrate implementations written in other languages.

Input/Output: Our system takes benchmark problems as input and produces corresponding output file. We record planned and executed actions, planning errors (collisions), and other simulation events. File formats are standardised and well-documented. In addition, output files can be used with PlanViz, our tool for offline analysis and visualisation.

PlanViz

Although there are many tools for visualising multi-agent plans, few of them help users to contrast and explore planning and execution. We develop PlanViz to help fill this gap.

PlanViz helps 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) current and future task assignments. For (3), to understand collisions, we label affected agents (all agents or user-selected pairs) and jump to one timestep before the collision occurs. We show the planned action and the executed action (from the simulator). For (4), to understand task assignments, we label all currently assigned, future assigned and finished locations. This helps participants to better understand algorithm performance for completing tasks. In addition, PlanViz can also visualise solutions from MAPF Tracker (Shen et al. 2023), a system that records best-known solutions to classical MAPF problems. We provide tools to convert from that format.

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. We include code for each entry on the final leaderboard and any submission that produces a best solution for any evaluation instance. We also open-source all best-known solutions. We believe these (code and benchmark) archives can foster further research on the topic of MAPF: by establishing state-of-the-art performance, disseminating best-known results and lowering barriers into the area. The latter is especially important for newcomers, so they can quickly get started with a high-quality 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.