Scalable Rail Planning and Replanning: Winning the 2020 Flatland Challenge

Jiaoyang Li,[1*] Zhe Chen,[2] Yi Zheng,[1] Shao-Hung Chan,[1]
Daniel Harabor,[2] Peter J. Stuckey,[2] Hang Ma,[3] Sven Koenig[1]
[1]University of Southern California, USA
[2]Monash University, Australia
[3]Simon Fraser University, Canada
{jiaoyanl, yzheng63, shaohung, skoenig}@usc.edu, {zhe.chen, daniel.harabor, peter.stuckey}@monash.edu, hangma@sfu.ca

Abstract

Multi-Agent Path Finding (MAPF) is the combinatorial problem of finding collision-free paths for multiple agents on a graph. This paper describes MAPF-based software for solving train planning and replanning problems on large-scale rail networks under uncertainty. The software recently won the 2020 Flatland Challenge, a NeurIPS competition trying to determine how to efficiently manage dense traffic on rail networks. The software incorporates many state-of-the-art MAPF or, in general, optimization technologies, such as prioritized planning, safe interval path planning, parallel computing, simulated annealing, large neighborhood search, and minimum communication policies. It can plan collision-free paths for thousands of trains within a few minutes and deliver deadlock-free actions in real-time during execution.

Introduction

The 2020 Flatland Challenge (Mohanty et al. 2020) is a competition set up to answer the question "How to efficiently manage dense traffic on complex rail networks?" It consists of an idealized rail planning problem. Given a map showing rail tracks and train stations (see Figure 1) and a set of m trains with start and target stations, we need to move the trains from their start stations to their target stations so that no two trains occupy the same track segment (a vertex collision) or cross each other by moving in opposite directions from adjacent track segments (an edge collision) at the same time. Trains may malfunction during execution. That is, a train may stop at a random timestep for a random duration. The reward is defined as $1 + \frac{\delta}{T_{max}} - \frac{\sum_{1 \leq i \leq m} T_i}{m T_{max}} \in [0, 1]$, where T_{max} is the given timestep limit, $T_i \in [1, T_{max}]$ is the *arrival time* of the i-th train at its target station if it reaches the target station by timestep T_{max} and T_{max} otherwise, and δ is 1 if all trains reach their target stations by timestep T_{max} and 0 otherwise.

The 2020 Flatland Challenge was organized by AIcrowd, an online AI crowd sourcing platform, and Swiss Federal Railways, Deutsche Bahn, and SNCF, three large railway network operators. It involved more than 700 participants from 51 countries making more than 2,000 submissions over 4 months. We outperformed all other entries in both rounds, including all Reinforcement Learning (RL) entries. In this paper, we focus on the key ideas of our approach for the main round (= Round 2), which gave us, for each instance, up to 10 minutes to compute the initial solution and then up to 10 seconds to generate the move commands for the trains at each timestep and asked us to maximize the accumulated reward over an infinite number of instances of increasing difficulty with an overall runtime budget of 8 hours. Details of our approach and the top RL approaches can be found in (Li et al. 2021b) and (Laurent et al. 2021).

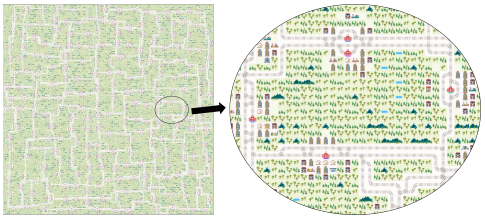

Figure 1: Flatland environment represented by a 229×229 grid map. The grey lines represent rail tracks, and the red houses on the rail tracks represent train stations.

Relationship with MAPF and Its Variants

The academic version of the 2020 Flatland Challenge is called Multi-Agent Path Finding (MAPF), which is moving multiple agents from their start vertices to their target vertices on a graph while avoiding vertex and edge collisions. At each timestep, an agent can move to an adjacent vertex or wait at its current vertex. Agents are at their start vertices at timestep 0 and remain at their target vertices after they complete their paths, where they can still collide with other agents. The task is to move all agents with a minimum sum of their travel times to their target vertices without collisions.

The 2020 Flatland Challenge has important differences to standard MAPF but is related to some MAPF variants:

1. Train movement is restricted to rails, which make up a small proportion of the map. Trains may not move backward. This requires us to take into account the orientation of the trains when planning paths, which is related to MAPF for motion planning (Cohen et al. 2019).

*Jiaoyang Li performed her research during her visit to Monash University.

2. Trains enter the environment over time and leave it after reaching their target stations, which is related to online MAPF (Svancara et al. 2019).
3. We are asked to move as many trains as possible (instead of all trains) to their target stations before the given timestep limit, which is related to MAPF with deadlines (Ma et al. 2018).
4. Trains break down randomly while moving and remain stationary at the breakdown rail segment for a number of timesteps, which is related to MAPF with delay probabilities or stochastic travel times (Cáp, Gregoire, and Frazzoli 2016; Ma, Kumar, and Koenig 2017; Wagner and Choset 2017; Li et al. 2019; Coskun and O'Kane 2019; Atzmon et al. 2020; Street et al. 2020).

Approaches

The 2020 Flatland Challenge is a MAPF problem at its core. The first three differences to standard MAPF can be addressed by small modifications of existing MAPF algorithms. The last difference, namely the malfunctions, can be handled during execution. We therefore first plan collision-free paths under the assumption that no malfunctions occur and then handle malfunctions once they occur.

Planning Collision-Free Paths

We use *Prioritized Planning* (PP) (Silver 2005), a popular MAPF algorithm, to generate the initial collision-free paths. PP first sorts all trains according to a priority ordering and then, from the highest priority train to the lowest priority train, plans a shortest path for each train while avoiding collisions with the already planned paths. For efficiency, we use *Safe Interval Path Planning* (SIPP) (Phillips and Likhachev 2011), an advanced version of A*, to plan each path.

Although PP can find collision-free paths rapidly, its solution quality is far from optimal. We therefore use Large Neighborhood Search (LNS) (Shaw 1998) to improve the solution quality. We follow Li et al. (2021a) by using PP to generate an initial solution and repeating a neighborhood search process to improve the solution quality until an iteration limit is reached. In each iteration, we select a subset of trains and replan their paths using PP. The new paths need to avoid collisions with each other and with the paths of the other trains. We adopt the new paths if they reduce the sum of travel times of the solution. As the competition provides 4 CPUs for evaluation and an overall runtime limit of 8 hours, we run 4 LNSes in parallel and use simulated annealing to determine the iteration limit for each instance.

Although SIPP runs significantly faster than A*, it is still slow when there are thousands of trains to schedule because, as the paths of more trains are planned, SIPP has to plan paths that avoid collisions with more existing paths, resulting in its runtime growing rapidly. We therefore used a *lazy planning* scheme where we planned paths only for some of the trains in the beginning, then let the trains move, and planned paths for the rest of trains during execution. Although delaying planning for some trains may delay their departure times, which in turn may delay their arrival times, lazy planning has two benefits: (1) It avoids pushing too many trains into the environment at once, which can prevent severe traffic congestion; and (2) planning during execution takes into account the influence of the malfunctions that have already happened or are happening.

Recovering from Malfunctions

When trains malfunctions during execution, deadlocks could happen if all trains stick to their original paths. *Minimum Communication Policies* (MCP) (Ma, Kumar, and Koenig 2017) avoid the deadlocks by stopping some trains to maintain the ordering in which trains visit each rail segment. However, MCP can stop trains unnecessarily. We therefore developed a *partial replanning* mechanism to avoid such unnecessary waits. When train A encounters a malfunction at some timestep, we collect all switches and crossing rail segments that train A is going to visit in the future and then collect the trains who are going to visit at least one of these switches or rail segments after train A. We use PP to replan the paths of these trains one at a time and terminate when either new paths have been planned for all of these trains or the runtime limit of 10 seconds has been reached.

Results

Our software was implemented in C++ and is available at https://github.com/Jiaoyang-Li/Flatland. On the leaderboard, our basic approach, namely PP with A* plus MCP, solved 349 instances within 8 hours with a score of 282.56. When we replaced A* with SIPP, our approach solved 3 more instances with a score of 285.359. LNS then improved the score to 289.102, and partial replanning further improved it to 291.873. Eventually, with the help of lazy planning, we solved 362 instances and reached our highest score of 297.507, which was 1.160 and 24.168 higher than the scores of the teams in the second and third place, respectively. The largest instance for which we managed to move all trains to their target stations before the timestep limit contains 3,256 trains, and we solved it in 704 seconds (which includes both the planning and simulation time) for a reward of 0.802, even though the planning time of our software was probably much smaller than 704 seconds because, when we ran the evaluation on our local server, 70% of the 8-hour runtime budget was spent on the simulator. Our MAPF-based approach also significantly outperformed the RL approaches: The highest score obtained by the RL approaches was 214.15, a score that was already reached by our approach after 15 minutes.

Acknowledgments

This abstract is a short version of (Li et al. 2021b). The research at the University of Southern California was supported by the National Science Foundation (NSF) under grant numbers 1409987, 1724392, 1817189, 1837779, and 1935712 as well as a gift from Amazon. The Research at Monash University was partially supported by the Australian Research Council (ARC) under grant numbers DP190100013 and DP200100025 as well as a gift from Amazon. The research at Simon Fraser University was supported by the Natural Sciences and Engineering Research Council (NSERC) under grant number RGPIN-2020-06540.

References

Atzmon, D.; Stern, R.; Felner, A.; Sturtevant, N. R.; and Koenig, S. 2020. Probabilistic Robust Multi-Agent Path Finding. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 29–37.

Cáp, M.; Gregoire, J.; and Frazzoli, E. 2016. Provably Safe and Deadlock-Free Execution of Multi-Robot Plans Under Delaying Disturbances. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 5113–5118.

Cohen, L.; Uras, T.; Kumar, T. K. S.; and Koenig, S. 2019. Optimal and Bounded-Suboptimal Multi-Agent Motion Planning. In *Proceedings of the International Symposium on Combinatorial Search (SoCS)*, 44–51.

Coskun, A.; and O'Kane, J. M. 2019. Online Plan Repair in Multi-robot Coordination with Disturbances. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 3333–3339.

Laurent, F.; Schneider, M.; Scheller, C.; Watson, J. D.; Li, J.; Chen, Z.; Zheng, Y.; Chan, S.; Makhnev, K.; Svidchenko, O.; Egorov, V.; Ivanov, D.; Shpilman, A.; Spirovska, E.; Tanevski, O.; Nikov, A.; Grunder, R.; Galevski, D.; Mitrovski, J.; Sartoretti, G.; Luo, Z.; Damani, M.; Bhattacharya, N.; Agarwal, S.; Egli, A.; Nygren, E.; and Mohanty, S. P. 2021. Flatland Competition 2020: MAPF and MARL for Efficient Train Coordination on a Grid World. *CoRR* abs/2103.16511.

Li, J.; Chen, Z.; Harabor, D.; Stuckey, P. J.; and Koenig, S. 2021a. Anytime Multi-Agent Path Finding via Large Neighborhood Search: Extended Abstract. In *Proceedings of the International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 1581–1583.

Li, J.; Chen, Z.; Zheng, Y.; Chen, S.-H.; Harabor, D.; Stuckey, P. J.; Ma, H.; and Koenig, S. 2021b. Scalable Rail Planning and Replanning: Winning the 2020 Flatland Challenge. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, (to appear).

Li, J.; Zhang, H.; Gong, M.; Liang, Z.; Liu, W.; Tong, Z.; Yi, L.; Morris, R.; Pasareanu, C.; and Koenig, S. 2019. Scheduling and Airport Taxiway Path Planning under Uncertainty. In *Proceedings of the AIAA Aviation Forum*.

Ma, H.; Kumar, T. K. S.; and Koenig, S. 2017. Multi-Agent Path Finding with Delay Probabilities. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 3605–3612.

Ma, H.; Wagner, G.; Felner, A.; Li, J.; Kumar, T. K. S.; and Koenig, S. 2018. Multi-Agent Path Finding with Deadlines. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 417–423.

Mohanty, S. P.; Nygren, E.; Laurent, F.; Schneider, M.; Scheller, C.; Bhattacharya, N.; Watson, J. D.; Egli, A.; Eichenberger, C.; Baumberger, C.; Vienken, G.; Sturm, I.; Sartoretti, G.; and Spigler, G. 2020. Flatland-RL: Multi-Agent Reinforcement Learning on Trains. *CoRR* abs/2012.05893.

Phillips, M.; and Likhachev, M. 2011. SIPP: Safe Interval Path Planning for Dynamic Environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 5628–5635.

Shaw, P. 1998. Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems. In *Proceedings of the International Conference on Principles and Practice of Constraint Programming (CP)*, 417–431.

Silver, D. 2005. Cooperative Pathfinding. In *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE)*, 117–122.

Street, C.; Lacerda, B.; Mühlig, M.; and Hawes, N. 2020. Multi-Robot Planning Under Uncertainty with Congestion-Aware Models. In *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 1314–1322.

Svancara, J.; Vlk, M.; Stern, R.; Atzmon, D.; and Barták, R. 2019. Online Multi-Agent Pathfinding. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 7732–7739.

Wagner, G.; and Choset, H. 2017. Path Planning for Multiple Agents under Uncertainty. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 577–585.