The Analysis of Flatland Challenge Winners' Multi-agent Methodologies

BumKyu Choi, Jong-Kook Kim†[1]
Dept. of Electric and Electronics Engineering, Korea University
qjarb3411@korea.ac.kr, jongkook@korea.ac.kr

ABSTRACT

Scheduling the movements of trains in the modern railway system is becoming essential and important. Swiss Federal Railway Company (SBB) and machine learning researchers began collaborating to make a simulation environment and held a Flatland challenge. In this paper, the methodologies of the winners of this competition are analyzed to achieve insight and research trends. This problem is similar to the Multi-Agent Path Finding (MAPF) and Vehicle Rescheduling Problem (VRSP). The potential of the attempted methods from the Flatland challenge to be applied to various transportation systems as well as railways is discussed.

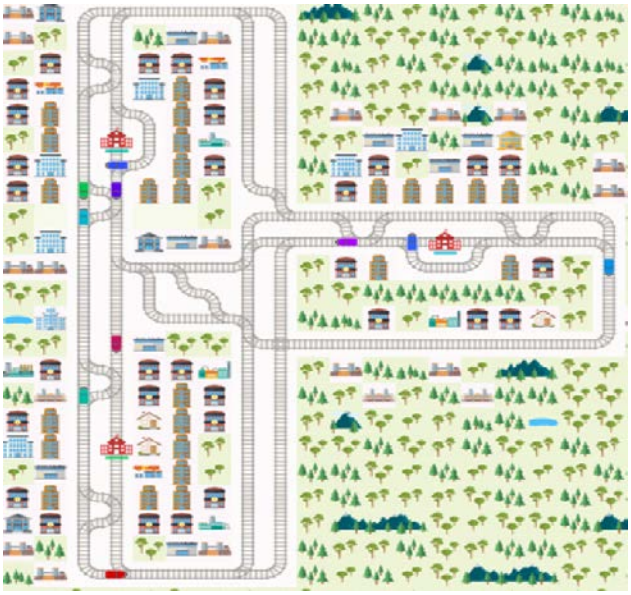
1. INTRODUCTION

In the real world, there is traffic everywhere; on roads, on railways, and on subways. In Switzerland, there are more than 10,000 trains travelling with 1.25million passengers and 200,000 tons of freight to various destinations. The research group at Swiss Federal Railway Company (SBB) developed a 2-dimensional railway traffic simulator called Flatland [1] which consists of physics simulation and traffic management system (TMS) [2] in an attempt to develop an intelligent way of controlling this massive train traffic. This train traffic control problem is a version of the vehicle rescheduling problem (VRSP) [3] where the vehicle routes are replanned or rescheduled due to the dynamic change in the environment (e.g., accidents, malfunction).

To this end, a public competition called the Flatland challenge was held on AIcrowd [4]. Participants of this challenge attempted various techniques in the area of Operation Research (OR) [5], Reinforcement Learning (RL), or combinations of both OR and RL. The purpose of the challenge is to guide all trains to arrive at their target destination in the total minimum amount of travel time. The challenge was held in 2019 and 2020. It is becoming an annual event where the 2021 challenge will be held this year. Techniques from OR were mainly used in the first year, and for the second year, RL methods were encouraged. We expect that novel learning methods will be attempted this year. In this paper, we briefly introduce the Flatland environment, the Flatland problem, baseline methods, and analyze the winning methods in 2019 and 2020.

2. FLATLAND

2.1 Flatland Environment



(Figure 1). Visualization of a Flatland simulation environment with three train stations and ten trains

Flatland [1] is a cell-oriented 2-dimensional grid and discrete-time simulation environment. Flatland generates various fully connected railways with different target stations and schedules for trains. Only one train can occupy a railway tile and the type of tile determines the possible movements of a train. A train can stop or move in up to two directions, depends on the underlying tile. The problem to solve in this environment is to find the optimal policy for all trains to reach their target station as fast as possible. Therefore, it is natural to specify this environment as a cooperative multi-agent

[1] Corresponding author

system (MAS) [6] and Multi-Agent Path Finding Problem (MAPF) [7] with trains as agents. Flatland simulates the vehicle rescheduling problem (VRSP) [2] by setting a malfunction rate, determining the probability of malfunctioning at each step for trains. One of the challenging situations in Flatland is a deadlock. It happens when two or more trains are trying to move in the opposite direction and toward each other in a single railway. Deadlock becomes a crucial issue as the scale of the environment increase. Therefore, avoiding deadlock is the critical problem of Flatland.

2.2 Evaluation Metric

Each train agent receives a combined return consisting of a combination of local reward r_l^i and a global reward signal r_g for evaluation. A penalty signal r_p^i is received if the move is illegal, but set to 0 in the competition. If all agents have reached their destination at time step t, a global reward $r_g = 1$ is given to every agent. Both α and β are set to 1.0

$$r_i(t) = \alpha r_l^i(t) + \beta r_g(t) + r_p^i(t) \in [-\alpha - 2, \beta] \quad (1)$$

$$g_i = \sum_{t=1}^{T} r_i(t) \quad (2)$$

$$S = \sum_{j=0}^{N} \sum_{i=0}^{M_j} g_i^j \quad (3)$$

$$S_{MNR} = \frac{1}{N} S = \frac{1}{N} \sum_{j=0}^{N} \sum_{i=0}^{M_j} g_i^j \quad (4)$$

$$S_{TNR} = \sum_{j=0}^{N} (\sum_{i=0}^{M_j} g_i^j + 1.0) \quad (5)$$

g_i is a cumulative return of each agent. S is the accumulated total normalized reward in total N environments with M_i agents for each configuration. For evaluation, in the challenge, it is worth noting that evaluation metrics are slightly different on each round and each year. The mean normalized return score for all evaluation episodes S_{MNR}, and the total normalized return score S_{TNR} ,which is added +1.0 value for each episode to make the value positive, are both used. Besides, the rate of trains that has arrived is provided as an auxiliary evaluation metric.

3. BASELINE METHODS

The Flatland challenge documentation [8] provides several baseline methods. These methods are for the 2020 challenge. In this section, we briefly explain Reinforcement Learning (RL) methods and Technical methods.

i) RL methods

There are from simple single-agent learning algorithm to RLlib framework [9] based algorithm.

- *Dueling Double DQN (DDDQN)*

Dueling Double Deep Q-Network (DDDQN) is the value-based algorithm which is the combination of Double Q-Network (DDQN) [10] which has a separate target network, and dueling network [11].

- *PPO/ Centralized Critic PPO (CCPPO)*

Proximal Policy Optimization (PPO) [12] is an actor-critical algorithm and Centralized Critic PPO (CCPPO) is the combination of PPO and Transformer [13] which is for attention mechanism.

- *Imitation Learning: MARWIL, Ape-X DQFD*

For bootstrapping the RL process and improvement with well-performing previous OR methods, the idea of imitation learning which imitates the already sufficient is proposed. Pure imitation learning MARWIL [14] and Hybrid or Mixed Learning Ape-X DQFD [15] are categorized in imitation learning.

ii) Technical methods

There are some technical approaches for the Flatland observation and agents.

- *Combined Observation*

Combined Observation is the way of combining multiple observations and can be applied with any algorithms.

- *Frame Skipping and Action Masking*

Frame Skipping and Action Masking are technical methods to filter out unnecessary cells or actions at specific time steps when deciding the system's efficiency.

4. PROPOSED METHODS

In this section, we analyze the winners' methodologies for each of the two years (2019, 2020). The competition consists of three rounds (warm-up, round-1, and round-2). The one who wins in the second round becomes the final winner. Analysis are based on winners' presentation 2019 (AMLD conference [16]), and 2020 (NeurIPS conference [17]).

■ 2019 CHALLENGE

Challenge in 2019 top solutions mainly used analytical method Operation Research (OR) [5]. OR method has been used as an approach for the vehicle rescheduling problem (VRSP) for decades. Therefore, it was natural to apply OR to the Flatland environment in the first 2019 challenge. Done rate and normalized reward were used as evaluation metrics and the done rate was a priority metric for ranking.

- *First Place: Mugurelionut*
  Done rate: 0.990
  Normalized reward: -11.320

There are two main components of this solution. First is (re)generating agent paths. The second is updating agent paths to avoid deadlock after malfunction occurring. A time-expanded graph for each simulation environment is used and each agent should have a feasible path on it previously. The operation runs in parallel with multiple threads each of which generates the permutation of the agents. A new possibly shortest path is found by using a variant of A* pathfinding

methods. If the best set is found, start the next run with it as the initial set. As a way of scoring a set of paths, they set the main objective as the number of agents reaching the destination. For Tie-breaking, they use an equation (6) which is a summation of T for all agents. T is time to reach the destination and E is the hyperparameter (from 0.25 to 4) and set to 1 which performs best.

$$\sum_{i=1}^{N} T^{E} \quad (6)$$

With detailed simulation settings and rules, they apply the shortest path algorithm for all single agents.

- *Second Place: CkUa(Team)*
  Done rate: 0.960
  Normalized reward: -14.620

This team approached with graph-based path search. They represent every railroad cell as nodes and possible movement as edges. In the beginning, minimal possible path to every target from every cell, and orientation pair are pre-calculated. At every time step, whether there are any malfunctions or not is checked. When it occurs, reschedule all started trains, or partially reschedule if it is not entirely available. For non-started trains, keep checking a good path. Similar to the first-place team solution, a sequential space-time data structure is used. With this structure, new trains can check if some cells will be available at the exact time step that is crucial to multi-agent environments. To find trains' path, they use the A* algorithm with a heuristic function (7).

$$f(\mathrm{x}) = g(x) + h(x) \quad (7)$$

$g(\mathrm{x})$ is a sum of waiting time and found travel time in seconds to the current vertex. $h(x)$ is minimal possible travel time to the target.

We omitted other participants because their methods similar to the above methods.

■ 2020 CHALLENGE

Leaderboard tracks are separated as RL Track and Non-RL Track in the 2020 challenge. Although RL Track participants were expected to achieve relatively low performance compared to Non-RL, the competition host encouraged the use of reinforcement learning and awarded separately due to its potential of general applicability to unseen problem instances. In contrast to the 2019 challenge, the total normalized return was used as the priority evaluation metric.

A. Non-RL Track (OR)

- *First Place: An_Old_Driver(Team)*
  Total reward: 297.507
  Done rate: 0.986

They used a combined approach to the planning method. Combination of Prioritized Planning (PP) [18], Large Neighborhood Search (LNS), Minimum Communication Policy (MCP) [19], and Partial Re-planning is the solution. First, they planned paths for all agents with the PP method. PP plans the path sequentially from the first agent to the last agent. The shortest paths are planned by an algorithm called Safe Interval Path Planning (SIPP) [20]. To improve the quality of the resulting initial solution, re-plan the paths of the agents in the neighborhood. The neighborhood is classified by three criteria (Station-based neighbor: have the same target, Intersection-based neighbor: pass through the same intersection, Agent-based neighbor: have the longest path with blocking agent). To handle deadlock caused by malfunctions, MCP is used. MCP prevents deadlocks by keeping the ordering of the agents to visit each location. Partial re-planning methods are finally added for more effective processing. After collecting the intersections that the malfunction agent is going to traverse, Re-plan the paths of the agents who are going to traverses these intersections. The combination of the above methods suitable for environmental characteristics ensured high performance.

B. RL Track

On the RL Track, winners focused on the main factors of reinforcement learning, such as the algorithm itself or representation of observation space and reward shaping.

- *First Place: JBR_HSE(Team)*
  Total reward: 214.15
  Done rate: 0.785

For training the multi-agent, they used both global observation and tree observation, similar to provided tree observation in the baseline documentation [8]. From the observations, common and local tree features are extracted. Agents move forward and determine the action only at the intersections. If the agent is in a deadlock, the episode ends. The reward is shaped with three components (8). ΔD_{min} is the shortest distance to the target, $is_{succeed}$ and $is_{deadlock}$ are binary indicators for the agent's succeed and deadlock. α, β, γ the hyperparameters are set to 0.01, 10, -5.

$$\mathrm{r} = \alpha * \Delta D_{min} + \beta * is_{succeed} - \gamma * is_{deadlock} \quad (8)$$

With an actor-critic algorithm, Proximal Policy Gradient(PPO) [11], agents are trained. Both actor and critic architecture are separated into Two (Common Features Net and Tree Features Net) and then combine via a fully connected(FC) layer. It is worth noting that there are Action block and Communication block in the Actor architecture. Passing the output from the FC layer and from the Multi-head attention layer [13] which contains features of neighbors, Action block outputs action distribution. Meanwhile, inputting output from FC layer to Communication block generates outcome message vector of an agent. The critic architecture, same with basic PPO, outputs scalar value function. Finally, to avoid deadlock, they trained and used the Scheduler model which estimates the probability the train will

reach its target successfully. Good policy is learned by training the scheduler iteratively with trained and fixed PPO agent.

- *Second Place: Ai-team-flatland(Team)*
  Total reward: 181.497
  Done rate: 0.881

Unlike the above first-place team, this team's main focus is on representing the environment, rather than the RL algorithm itself. They represent the environment with a simple cell graph which represents only the intersection node and edge representing connections. With the graph representation of the environment, off-policy and on-policy baseline RL models (In section 3) have experimented. To input the observation vector, the priority, and the conflict-info features are mainly used. In the learning process, they designed the reward in detail. (if agent is in the target: +10, deadlock: -10, not following priority: -5, missing the path: -3, stopping on switch: -3, stopping: -1, regular step: -0.5, avoided deadlock: -0.35). A notable fact is that they achieved a higher done rate than the first-place team.

Participants were usually unable to focus on all environment elements and instead focused on designing the algorithm model or environment itself. However, it was generally agreed that the efficiency and computational speed of the environment should be upgraded (e.g., 2019 first-place winner commented that converting Python code to C++ can be considered). To optimize in more large-scale maps and general applicability, using graphical methods including GNN [21] with RL approaches are expected to be the key methods for the 2021 challenge and the next.

5. CONCLUSION

In this paper, we introduced the railway simulation environment called Flatland and the Flatland challenge. Various methodologies from Operation Research (OR) and Reinforcement Learning (RL) are introduced and discussed. The analysis of the winners' methods will help guide researchers to the solutions for Multi-Agent Path Finding (MAPF) and Vehicle Rescheduling Problem (VRSP) and can be helpful to (potential) participants of the challenge.

REFERENCES.

[1] Mohanty, Sharada, et al. "Flatland-RL: Multi-Agent Reinforcement Learning on Trains." *arXiv preprint arXiv:2012.05893* (2020).

[2] De Souza, Allan M., et al. "Traffic management systems: A classification, review, challenges, and future perspectives." *International Journal of Distributed Sensor Networks* 13.4 (2017): 1550147716683612

[3] Li, Jing-Quan, Pitu B. Mirchandani, and Denis Borenstein. "The vehicle rescheduling problem: Model and algorithms." *Networks: An International Journal* 50.3 (2007): 211-229.

[4] Links: https://www.aicrowd.com

[5] Frederick, S., and Gerald LIEBERMAN. "Introduction to Operation research." (2001).

[6] Panait, Liviu, and Sean Luke. "Cooperative multi-agent learning: The state of the art." *Autonomous agents and multi-agent systems* 11.3 (2005): 387-434.

[7] Stern, Roni, et al. "Multi-agent pathfinding: Definitions, variants, and benchmarks." *arXiv preprint arXiv:1906.08291* (2019).

[8] Links: https://flatland.aicrowd.com/intro.html

[9] Liang, Eric, et al. "RLlib: Abstractions for distributed reinforcement learning." *International Conference on Machine Learning*. PMLR, 2018.

[10] Van Hasselt, Hado, Arthur Guez, and David Silver. "Deep reinforcement learning with double q-learning." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 30. No. 1. 2016.

[11] Wang, Ziyu, et al. "Dueling network architectures for deep reinforcement learning." *International conference on machine learning*. PMLR, 2016.

[12] Schulman, John, et al. "Proximal policy optimization algorithms." *arXiv preprint arXiv:1707.06347* (2017).

[13] Vaswani, Ashish, et al. "Attention is all you need." *arXiv preprint arXiv:1706.03762* (2017).

[14] Wang, Qing, et al. "Exponentially Weighted Imitation Learning for Batched Historical Data." *NeurIPS*. 2018.

[15] Pohlen, Tobias, et al. "Observe and look further: Achieving consistent performance on atari." *arXiv preprint arXiv:1805.11593* (2018).

[16] Links: https://www.youtube.com/watch?v=rGzXsOC7qXg

[17] Links: https://www.youtube.com/watch?v=wDKbL7CuHpQ

[18] Silver, David. "Cooperative Pathfinding." *Aiide* 1 (2005): 117-122.

[19] Ma, Hang, TK Satish Kumar, and Sven Koenig. "Multi-agent path finding with delay probabilities. " *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 31. No. 1. 2017.

[20] Phillips, Mike, and Maxim Likhachev. "Sipp: Safe interval path planning for dynamic environments. " *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011

[21] Battaglia, Peter W., et al. "Relational inductive biases, deep learning, and graph networks." *arXiv preprint arXiv:1806.01261* (2018).