AIDING VEHICLE SCHEDULING AND RESCHEDULING USING MACHINE LEARNING

JONAS WÄLTER[1], FARHAD D. MEHTA[1], XIAOLU RAO[2]
[1]HSR University of Applied Sciences Rapperswil, Switzerland
[2]Swiss Federal Railways (SBB), Switzerland

ABSTRACT

Vehicle scheduling and rescheduling are central challenges for the planning and operation of railways. Even though these problems have been the subject of many research and development over several decades, railways still – with good reason – at the end of the day rely on well-trained and experienced personnel to provide practical solutions to these problems. Over the last couple of years, novel techniques based on machine learning have been used to propose solutions to problems such as image and speech recognition that could not have been imagined previously. The aim of machine learning is to design algorithms that can improve automatically through experience. The experience possessed by traffic dispatchers is often their greatest tool. It is, therefore, not implausible that machine learning techniques could also be used to provide better automation or support to the railway scheduling and rescheduling problems. This article describes the results of a study conducted to evaluate the extent to which solutions to the scheduling and rescheduling problems could be improved using a machine learning technique called reinforcement learning. The solutions obtained using this technique are compared with solutions obtained using classical algorithmic and constraint-based search techniques. The initial results have been obtained under a simulated environment developed by Swiss Federal Railways for the public Flatland Challenge competition. This research has been ranked number 4 in this international competition. Although these initial results have been obtained under simulated conditions and using limited computational resources, they look promising compared to classical scheduling and rescheduling solutions and suggest that further work in this direction could be worthwhile

Keywords: deadlock avoidance, machine learning, multi-agent path finding, neural network, railway operation, reinforcement learning, rescheduling, scheduling, traffic management.

1 INTRODUCTION

This article summarizes the results of a study that aims to compare and summarize existing solutions and propose a novel, reinforcement learning (RL)-based solution to two fundamental challenges in the area of railway traffic management. The first challenge is multi-agent path finding (MAPF) in the context of the railway scheduling, where any number of trains (defined as agents) should be routed from their current location to the desired destination in a railway network. A route and time schedule are to be found with which all trains reach their destination in the shortest possible time. Thereby, it is also important to avoid conflicts between several trains. However, a train can unexpectedly be disturbed, for example by a malfunction. In such a situation, the second challenge is faced: the vehicle rescheduling problem (VRSP), where the malfunction causes a re-planning of routes.

The investigations are conducted using the Flatland framework, which was developed by Swiss Federal Railways (SBB) for the public Flatland Challenge competition (https://www.aicrowd.com/challenges/flatland-challenge).

2 FLATLAND ENVIRONMENT

The basis for this study is the Flatland Environment, which is provided in the form of the Python library *flatland-rl* by SBB. This library can be used to simulate an arbitrary railway network and its trains and to evaluate different approaches.

ISSN: 2058-8305 (paper format), ISSN: 2058-8313 (online), http://www.witpress.com/journals
DOI: 10.2495/TDI-V4-N4-308-320

2.1 Railway network

A railway network is built as a two-dimensional rectangular grid with any number of cells in width and height. In Fig. 1, a railway network with a height of 20 cells and a width of 35 cells is shown. A cell is identified by its coordinates in the form (*row, column*).

A cell can be empty and is then represented by trees, mountains or buildings for illustration purposes. For track cells, different cell types are valid, as shown in Fig. 2. Cells can be rotated or mirrored to obtain another cell of the same cell type. Due to the limitations in the reality of physical track switches, a cell can be exited in two directions at most when approaching from any direction.

2.2 Agent

An agent in the Flatland Environment corresponds to a train on a railway network. All agents are of the same length 1 and thus occupy only one cell at a time. Each agent has an assigned start cell and an assigned target cell. The target cell is displayed as a station in the railway network for illustration purposes. Each agent has a different but constant speed. The speed is given by the number of time steps the agent has to spend on the same cell before its next movement. There is no acceleration or braking when starting or stopping. There are also no speed restrictions on specific cells.

Depending on which actions are allowed on the current cell, an agent can move left, forward, right or stop. An agent cannot move backwards. The only exception is on a dead-end cell, where the agent is turned around and can continue in the opposite direction.

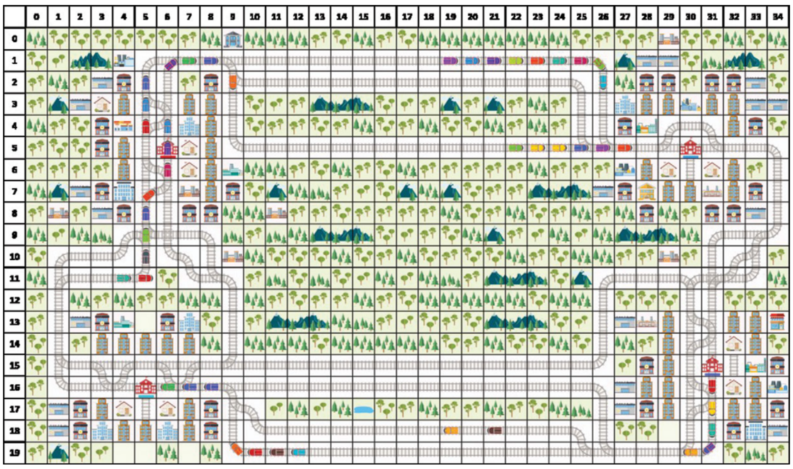

Figure 1: Railway network with a height of 20 cells, a width of 35 cells and 45 trains.

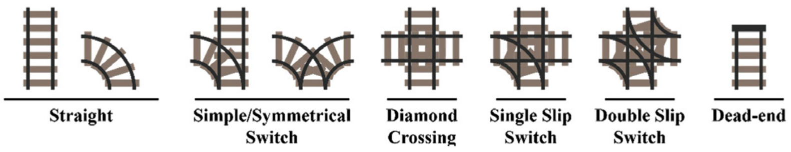

Figure 2: Valid cell types.

3 MULTI-AGENT PATH FINDING

MAPF is a fundamental problem in the field of traffic planning and is defined by Cohen et al. [1] as follows:

Given an environment and agents with assigned start and goal locations, the Multi-Agent Path Finding (MAPF) problem is to find collision-free paths for all agents from their start to their goal locations that optimize some criterion such as makespan or sum of travelled distances.

In the context of the Flatland Environment, a collision (or conflict) corresponds to the situation when an agent is moving to a cell that is already occupied by another agent. Furthermore, the definition refers to *makespan*. This is the time span until the last of all agents has reached its target cell.

A solution for MAPF consists of a schedule that specifies which agent is in which cell at what time. Finding a solution is one aspect of MAPF and the optimality of the solution is another aspect. According to Andreychuk et al. [2], finding a valid solution is feasible in a realistic setting (i.e., it has a so-called polynomial time complexity), but finding the optimal solution may not be feasible within an acceptable timeframe (i.e., the problem has the so-called 'NP-hard' complexity: it is widely assumed that the optimal solution cannot be found in polynomial time, but no proof for this currently exists).

There are different approaches to find a solution to the present MAPF problem. In the following subsections, approaches providing an optimal solution are considered first, followed by approaches that provide a suboptimal solution but within a polynomial runtime.

3.1 Linear Programming (LP)

As a first method, the mathematical approach is considered. For this purpose, the MAPF problem is formulated as a mathematical optimization model. An optimization model consists of variables, parameters, constraints and an objective function and describes a problem in mathematical form. The properties of the railway network and its agents are passed to the model as parameters and then result in an instance of the optimization problem. These parameters determine the constraints and the objective function. Finally, it is a purely mathematical task to find the corresponding variables that optimize the objective function in the desired manner (minimization or maximization) in compliance with the constraints. Over time, a variety of different so-called solvers have been developed for LP to calculate an optimal solution for such a mathematical system.

For the formulation of the MAPF problem as a mathematical optimization model, the work of Barták et al. [3] can be used as a basis and adapted to the characteristics of the Flatland Environment. In summary, the formulation can be described as follows: The presence of an agent on a cell is considered as an activity with a start time and an end time. An activity is optional, meaning it can be present or not. The cell path of an agent is finally defined by the presence of its optional activities. These activities are also used to formulate the constraints preventing the simultaneous presence of multiple agents on the same cell.

3.2 Constraint-based search (CBS)

Another optimal approach is the CBS, presented by Sharon et al. [4]. CBS makes use of a constraint tree and nodes. A constraint node contains a valid or invalid MAPF solution, a set of constraints and the total cost of the solution (e.g., makespan).

In summary, CBS works as follows: First, an initial solution is produced consisting of the shortest paths from the start position to the target position for all agents. The root node of the constraint tree is built from this initial solution and an empty constraint set. This solution usually has conflicts between agents. From the root node, the first conflict between two agents (A1 and A2) is used to form two new child nodes. The first child node is given a constraint that prohibits agent A1 from performing the action that leads to the conflict. Then, a new shortest path is determined for agent A1, with respect to the constraint. This results in a new solution. The same is done for the second child node and agent A2. If a solution is free of conflicts, the optimal solution is found. Otherwise, the process is repeated with the next node. For this, the node with the lowest solution cost is selected as the next node.

During the CBS algorithm, the shortest path with respect to the given constraints must be determined for an agent. For this purpose, the well-known A* search algorithm is very well suited. As the required A* heuristic, the actual distance from the current position to the target position in the railway network is used.

3.3. Operator Decomposition & Independence Detection (OD+ID)

Another approach is a kind of cooperative path finding consisting of the two components Operator Decomposition (OD) and Independence Detection (ID), presented by Standley [5].

3.3.1 Operator Decomposition

OD is based on the well-known A* search algorithm and uses nodes containing the current positions of all agents. The initial node contains the start positions of all agents. Then, the first agent in the sequence is considered. For each valid action of this agent, a new node is created with the new corresponding position of the agent. If an action causes a collision, the respective node is discarded. Between all valid nodes, a heuristic is used to select which node is considered next. With the next node and the next agent, this process is repeated until all positions match the target positions resulting in an optimal solution.

3.3.2 Independence Detection

The MAPF problem can be solved optimally with OD. However, instead of solving the entire problem with OD, the problem can be divided into smaller problems which are then solved with OD. ID can be used to build such sub-problems as follows: First, each agent is assigned to a separate group and its shortest path from the start position to the target position is considered the solution for this group. Afterwards, the solutions of all groups are checked against each other for conflicts. The two groups that cause the first conflict are considered for the next steps: It is tried to find a conflict-free but equivalent partial solution for one of the two groups. If this is not successful, the two groups are merged and the solution for the new group is searched using OD. Then, the solutions of all groups are checked again for conflicts. This procedure is repeated until no more conflicts occur and thus an optimal solution is found.

3.4 Optimal Anytime Algorithm (OAA)

The MAPF problem can be solved optimally with the OD+ID approach. A modified version of that approach is the OAA, presented by Standley and Korf [6]. The OAA is used to limit the runtime of OD+ID using a time limit. In this approach, the modified OD+ID algorithm is called repeatedly with increasing parameters producing an ever-improving solution. Thus, the algorithm can be aborted anytime and then provides the best solution for that time. The name of the algorithm is misleading because it is not the optimal solution but the best solution so far at the time of abort.

3.5 Prioritized Planning (PP)

As an alternative approach, greedy algorithms can be considered as well. The greedy algorithm developed in the course of this study is covered in the literature [7] under the designation PP. The procedure is straightforward: First, all agents are put into a sequence according to certain criteria. Then, the first agent is considered according to this order and its shortest path is determined. In the sense of greedy, this path is final and will not be changed anymore. In the next step, the second agent in the sequence is considered. For this agent, the shortest path is determined but without any conflicts with the previous agent. This path is again final and unchangeable. In the same way, the shortest conflict-free paths are determined for all remaining agents in the sequence.

The result is fully dependent on the chosen planning order. There are different ways to prioritize the agents based on criteria:

- Identification number
- Length of shortest path
- Time step of first conflict
- Number of conflicts

The list is not complete, and other criteria could be explored. The criteria can be applied in ascending or descending order. In addition, the criteria can be used either individually or combined as multi-level prioritization. For example, the agents can be sorted first by shortest path length, then by first conflict and finally by identification number.

3.6 Interim conclusion

There are several approaches to solve the MAPF problem. It is necessary to figure out which approach provides the best possible solution in the shortest possible time. The performance of the different methods is examined and compared in Section 5. The collection of the described approaches does not claim to be complete. There are other optimal and suboptimal algorithms for MAPF, which are not discussed in this paper.

4 VEHICLE RESCHEDULING PROBLEM

The VRSP is another fundamental problem in the field of traffic planning and is defined by Li et al. [8] as follows:

The vehicle rescheduling problem (VRSP) arises when a previously assigned trip is disrupted. A traffic accident, a medical emergency and a breakdown of a vehicle are examples of possible disruptions that demand the rescheduling of vehicle trips. The VRSP can be approached as a dynamic version of the classical vehicle scheduling problem (VSP) where assignments are generated dynamically.

In the context of the Flatland Environment, a disruption is a malfunction that occurs to a random agent at a random time and blocks the agent for a random number of time steps. There are existing and novel approaches to solve the VRSP.

4.1 Rescheduling

A first approach to solve the VRSP is already contained in the name of the problem: rescheduling. It is an attempt to apply the MAPF algorithms to the VRSP as well. The principle is simple: First, an initial solution for all agents is determined using an MAPF algorithm. Then, the simulation is started with all the agents following their determined paths. This continues until a malfunction occurs. Since the original solution is no longer feasible due to the malfunction, a new solution must be determined from the current positions of the agents. This can again be achieved using an MAPF algorithm. The described process is repeated continuously until all agents reach their target positions.

However, it was found that the described approach cannot work completely, at least not with the characteristics of the Flatland Environment: With an optimal or very good solution for an MAPF problem, the trains usually follow each other very closely in the railway network. Thus, there is a certain probability that a deadlock will occur during the reaction time after the occurrence of a malfunction. Under these circumstances, rescheduling using existing MAPF algorithms is not appropriate. For this reason, the rescheduling approach using these algorithms is not considered further.

4.2 Complete path reservation (CPR)

Rescheduling is susceptible to deadlocks caused by malfunctions. Thus, a robust scheduling approach is wanted which can eliminate this circumstance as described by Zuo [9]: 'For an uncertain scheduling problem, the goal of robust scheduling is to generate a suboptimum scheduling scheme that is not sensitive to stochastic disturbances, i.e., robust scheduling emphasizes on the stability of scheduling schemes.' In the course of this study, such an approach was elaborated and given the name CPR. In contrast to the approaches discussed so far, the agents are not controlled centrally, but each agent decides on its own in each step which action to perform next.

4.2.1 Procedure

The concept of reservation is introduced as the basis for this approach. Each track cell can be reserved by agents in the different directions. Thereby, a cell can be reserved by multiple agents in the same direction. However, it is not allowed to reserve a cell in one direction if the cell is already reserved in the opposite direction.

In the procedure, one agent after the other is processed. Each agent in the sequence must attempt to find the shortest path from its start position to its target position, for which all cells are not yet reserved in the opposite direction. Only if this succeeds, the cells of this path can be reserved in the corresponding directions and the respective action can be executed. Otherwise, the agent is not allowed to execute the action.

The procedure is illustrated with an example in Fig. 3: Agent 1 finds a non-reserved path to its target position and reserves the corresponding cells. For agent 2, there is no non-reserved path to its target position because some necessary cells are already reserved in the opposite direction by agent 1. In contrast, agent 3 finds a non-reserved path again. Thereby, some of the necessary cells are already reserved by agent 1 but in the same direction.

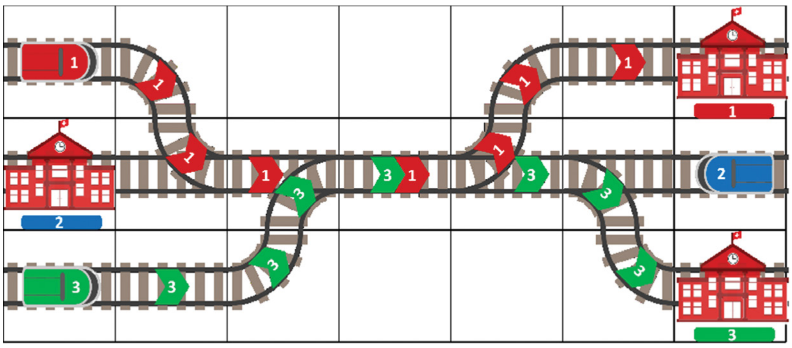

Figure 3: Complete path reservation with three agents on a partial railway network.

4.2.2 Agent prioritization

As described in the procedure, each agent acts independently and tries to reserve a complete path. Nevertheless, there is an order in which the agents can do this. In Fig. 3, the default prioritization with ascending agent numbering is used. Instead, a different prioritization could be used, where agent 2 is first to act. In this case, agent 2 could reserve a complete path, but the other two agents could not, resulting in a different outcome. Thus, the result is fully dependent on the chosen agent prioritization.

4.3 Reinforcement learning

Due to the development in the field of artificial intelligence in recent years, various novel methods have emerged in computer science. A promising approach from the field of machine learning is the so-called RL. RL is the third area of machine learning methods besides supervised and unsupervised learning. The basic idea of RL is that a software agent autonomously learns what it should ideally do in which situation. For this purpose, the learning behaviour in nature is simulated.

4.3.1 Introduction

The scheme of RL is shown in Fig. 4. The main components are the agent, the environment and the action, which are already known in the context of the Flatland Environment. The learning takes place in a cycle between the agent (train) and the environment: The agent receives a state describing the current situation of the environment. Based on this state, the agent must decide which action to take. With the help of its learned knowledge, the agent selects an action and executes it in the environment. This changes the environment, and the new state is passed on to the agent again. In addition, the agent receives a reward from the environment indicating whether the previous action was good or not. The higher the reward, the better the action was. Thus, the agent learns which action has led to which reward in which state. This learning cycle can then be continued for any period to deepen the knowledge of the agent.

4.3.2 Knowledge

While learning, the agent builds up its knowledge. The method used for this is called Deep Q-Learning with the help of a neural network consisting of several layers. The first layer is

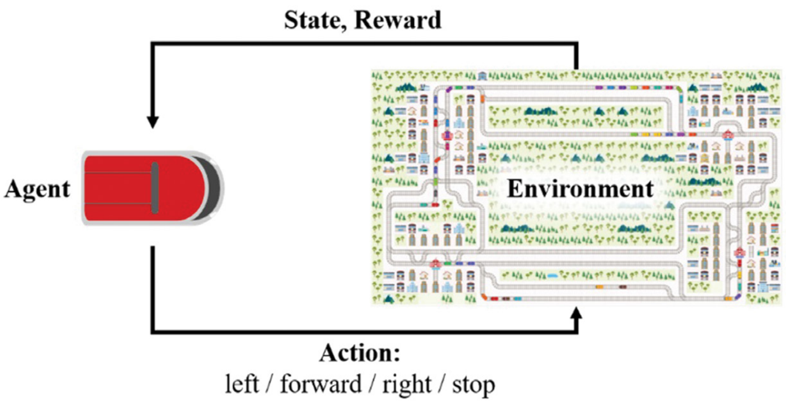

Figure 4: Reinforcement learning in the context of the Flatland Environment.

the input layer that directly corresponds to the state. The last layer is the output layer containing a value as quality for each action. The action with the highest value is meant to be the best action for the given state.

A function is specified which converts the state (input layer) into the action qualities (output layer). The parameters of this function are optimized with each learning cycle. To achieve a better result, any number of so-called hidden layers can be used as intermediate layers between the input layer and the output layer.

4.3.3 State

There are several ways to observe and represent the state of an environment. One approach is the *global observation*, which aims to reflect all available information of the environment in the state. A multidimensional array is built as a state, which contains all properties of the individual cells and agents. However, this approach quickly results in an amount of data for the state, which cannot be handled by usual workstations.

In contrast, an approach with *local observation* does not cover the entire railway network but only the interesting part of it. It is observed what the track sections ahead of the agent look like and what is happening on them. The information about the sections (e.g., distances, number of agents, properties of the agents, possible conflicts) then form the state. With this approach, the state size is reduced while the important information is retained.

4.3.4 Enhancement: Output mapping

In the course of this study, several enhancements for the RL approach were elaborated. A first enhancement achieves a reduction of the output size of the neural network. The smaller the output size, the easier the decision should be. So far, the output layer has directly corresponded to the four different actions of the agent. But an agent can exit a cell in a maximum of two directions. Thus, an agent can only choose between two different movement actions or the *stop* action.

With this knowledge, the output of the neural network can be adapted. The output no longer corresponds directly to the actions. Instead, the output now expresses whether the track more to the left, more to the right or a stop should be selected. This output is then mapped to the actual action.

4.3.5 Enhancement: Decisions only

For each application of the neural network, the computation needs a certain amount of time. This enhancement aims to use the neural network only for real decisions and thus reducing the number of applications.

An analysis of the rail networks revealed that in most situations only one action is really relevant for an agent. If an agent is on a straight track and not just before an intersection, it can only continue or stop. However, stopping makes no sense because the aim is to reach the target as quickly as possible. A real decision is only required if the agent is located directly at an intersection with two possible exits (*fork*) or in front of a cell where another track section joins its current track (*join*). There is also a situation where both conditions apply.

With this knowledge, the procedure is adapted so that the neural network is only applied in case of a real decision. Otherwise, the agent simply performs the only action that makes sense. Thus, the number of applications of the neural network is reduced and the agent no longer must learn trivial decisions but can focus on the real decisions.

4.3.6 Enhancement: State partitioning

A further enhancement is related to the state consisting of a local observation. With local observation, both possible exit directions are considered, and a partial observation is made for each. So far, both partial observations have been concatenated into a one-dimensional array representing the state which is then passed to the neural network. However, the neural network does not know that the first half and the second half of the state have the same structure and represent two partial observations.

As an enhancement, the two partial observations are instead combined into a two-dimensional array. The neural network can now be applied separately to both partial observations, resulting in one single quality for the respective action. This reduces both the input size and the output size of the neural network. Thus, the same knowledge can be applied to a mirrored situation as well which has not been the case so far.

4.3.7 Enhancement: Deadlock avoidance

An additional risk in railway networks is the occurrence of deadlocks with several trains blocking each other. So far, the detection of deadlocks has been left to RL: The agent should learn what action would cause a deadlock and avoid it. This detection requires a lot of learning, and there are alternative approaches to handle this task. However, complete deadlock detection cannot be done in polynomial time, according to Gawrilow et al. [10].

We, therefore, use the CPR technique described earlier in section 4.2 as a heuristic to detect deadlocks and shorten learning times in the following way: To check whether an action leads to a deadlock, it is tried to find a non-reserved path to the target position. If this is successful, the action certainly does not lead to a deadlock. If not, the action is assumed to lead to a deadlock. This approach, although not complete, is designed to be conservative. Any deadlock situation is correctly detected as a deadlock, but some deadlock-free situations are falsely also detected as deadlocks.

It should be noted that there are other heuristics for deadlock avoidance as well. For instance, one could similarly adapt other algorithms for MAPF.

5 RESULTS

Various approaches to the two fundamental problems MAPF and VRSP are discussed in this paper. All approaches were evaluated and compared using 20 different test cases provided by the Flatland Challenge.

5.1 Multi-agent path finding

The results of the various MAPF algorithms are listed and compared in Table 1. The used test cases differ in the number of track cells and the number of agents. For all algorithms, the runtime (in seconds) is listed. A test was aborted after 10 minutes with a timeout. For the suboptimal algorithms, the difference (*diff*) to the optimal solution concerning the makespan (*opt. mksp*) is additionally shown.

For the implementation of LP, the Python library *mip* with the included solver *Coin-or branch and cut (CBC)* was used. The alternative library *PuLP* led to similar results. Only 20% of all test cases could be solved using LP within the time limit. The OD+ID approach seems to be much better suited in the given context, as around 80% of all test cases could be solved within a short time. The CBS approach achieved the best performance among the optimal approaches with a timeout rate of only 5%. Nevertheless, there are situations where this algorithm takes a long time.

The OAA approach is a suboptimal version of the OD+ID algorithm, which keeps running until it is aborted and then returns the best solution found so far. However, a timeout occurred in six test cases. In four of these test cases, an optimal solution had already been found within the time limit but not yet recognized as optimal. A better alternative is the suboptimal PP approach, which achieves the best results with respect to the runtime. Moreover, the quality of the solutions lies in a tolerable distance to the optimal solutions.

5.2 Vehicle rescheduling problem

The results of both VRSP approaches are listed and compared in Table 2. For testing the VRSP approaches, more complex test cases were used. A maximum number of time steps (*max. steps*) are given before a test is aborted. If all agents (100%) reached their target within this limit, the number of used steps is stated. If not, the percentage of agents that reached their target within the limit is stated.

The CPR approach works well for simpler test cases, but the results get noticeably worse for more complex test cases. Across all test cases, an average of 85.75% of all agents had reached their target within the step limit. For the RL approach, local observation was used and enhanced with output mapping, real decisions and state partitioning. Regarding deadlocks, detection by CPR was used to reduce the learning time. The learning took place on randomly generated railway networks with a height and width of 35 cells each and 80 agents. This configuration corresponds to the properties of smaller test cases. The acquired knowledge was applied to the test cases and resulted in an average success rate of 72.16%.

The success rate of RL is significantly lower compared to the CPR approach. However, it must be noted that the possibilities of RL are far from being exhausted. The achieved success rate is very promising, considering that the exploration of RL was limited by the given hardware and time. To further improve the performance, the learning process could be repeated with more complex railway networks and with better hardware.

A more detailed description of the methods used, and the results can be found in [11].

Table 1: Comparison of the MAPF algorithms using different test cases.

Test	Track cells	Agents	LP runtime	CBS runtime	OD+ID runtime	Opt. mksp	OAA		PP	
							Runtime	diff	Runtime	diff
0.0	37	1	0.35225	0.00006	0.00005	8	0.00005	(=)	0.00008	(=)
0.1	36	1	0.28387	0.00004	0.00006	11	0.00006	(=)	0.00010	(=)
1.0	37	3	9.58017	0.00024	0.00056	10	0.00128	(=)	0.00027	(=)
1.1	36	3	115.14047	0.00013	0.00013	11	0.00012	(=)	0.00024	(=)
2.0	66	5	Timeout	0.00110	0.00104	11	0.00168	(=)	0.00053	(=)
2.1	63	5	Timeout	0.00023	0.00020	11	0.00024	(=)	0.00042	(=)
3.0	222	10	Timeout	0.02175	0.01082	28	0.01600	(=)	0.00443	(=)
3.1	183	10	Timeout	0.00139	0.00358	26	0.00603	(=)	0.00216	(=)
4.0	210	10	Timeout	0.01252	Timeout	30	Timeout	(=)	0.00748	(=)
4.1	212	10	Timeout	Timeout	364.95296	29	Timeout	+4	0.00518	+4
5.0	429	15	Timeout	0.01129	0.12862	49	0.11072	(=)	0.01192	(=)
5.1	481	15	Timeout	0.02905	0.06104	53	0.18168	(=)	0.01502	(=)
6.0	694	10	Timeout	0.05049	0.05177	64	0.10445	(=)	0.02240	(=)
6.1	703	10	Timeout	0.01685	0.07653	61	0.16499	(=)	0.01988	(=)
7.0	1,648	40	Timeout	0.85441	Timeout	75	Timeout	(=)	0.25005	(=)
7.1	1,659	40	Timeout	0.88645	0.39294	79	Timeout	+1	0.36779	(=)
8.0	1,001	10	Timeout	0.00293	0.00358	82	0.00351	(=)	0.01746	(=)
8.1	1,007	10	Timeout	0.00478	0.00564	130	0.00559	(=)	0.02567	(=)
9.0	4,721	50	Timeout	30.76673	Timeout	147	Timeout	(=)	1.63163	(=)
9.1	4,686	50	Timeout	6.75903	Timeout	165	Timeout	(=)	1.68559	(=)

Table 2: Comparison of the VRSP approaches using different test cases.

Test	Track cells	Agents	Max. steps	CPR		RL	
				Steps	Agents (%)	Steps	Agents (%)
0.0	140	1	600	379	100.00	412	100.00
0.1	90	1	600	527	100.00	Max.	90.00
1.0	190	3	600	594	100.00	Max.	98.75
1.1	250	3	600	Max.	93.75	Max.	85.00
2.0	212	5	720	627	100.00	Max.	90.00
2.1	231	5	720	627	100.00	Max.	88.75
3.0	374	10	960	Max.	88.75	Max.	56.25
3.1	415	10	960	Max.	88.75	Max.	73.75
4.0	399	10	960	752	100.00	Max.	98.75
4.1	496	10	960	Max.	85.00	Max.	75.00
5.0	693	15	1,120	Max.	91.25	Max.	76.25
5.1	544	15	1,120	Max.	92.50	Max.	73.75
6.0	1,346	10	1,760	Max.	75.00	Max.	75.00
6.1	1,203	10	1,760	Max.	68.00	Max.	57.00
7.0	1,378	40	1,600	Max.	84.00	Max.	55.00
7.1	1,444	40	1,600	Max.	88.00	Max.	67.00
8.0	1,602	10	1,760	Max.	65.00	Max.	51.00
8.1	1,552	10	1,760	Max.	52.00	Max.	53.50
9.0	2,514	50	2,560	Max.	64.50	Max.	46.00
9.1	2,936	50	2,560	Max.	78.50	Max.	32.50
				Average:	85.75	Average:	72.16

6 CONCLUSION

In this study, various approaches for resolving both MAPF and the VRSP were investigated with respect to railway traffic.

Regarding MAPF, various approaches, including LP, CBS, OD+ID, OAA and PP, are discussed, tested and compared. It was shown that the theoretically optimal solutions (such as LP, CBS, OD+ID) were not suitable for use in a real-time traffic management system. Instead, suboptimal algorithms like PP that provide fast solutions are more appropriate.

Regarding VRSP, the CPR approach was able to perform best in the evaluation. Surprisingly, RL was almost able to keep up with it and thus demonstrated the potential of this novel approach. RL is still a very young and little researched area. There are many more optimization possibilities, and the results could be improved with additional effort and computational power. In conclusion, this study has shown that RL is a novel but promising technique that could be used to aid vehicle scheduling and re-scheduling.

REFERENCES

[1] Cohen, L., Wagner, G., Kumar, T.S., Choset, H. & Koenig, S., Rapid randomized restarts for multi-agent path finding solvers, 2017.

[2] Andreychuk, A., Yakovlev, K., Atzmon, D. & Stern, R., Multi-agent pathfinding with continuous time, 2019.

[3] Barták, R., Švancara, J. & Vlk, M., A scheduling-based approach to multi-agent path finding. *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 748-756, 2018.

[4] Sharon, G., Stern, R., Felner, A. & Sturtevant, N.R., Conflict-based search for optimal multi-agent pathfinding. *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, pp. 563-569, 2012.

[5] Standley, T., Finding optimal solutions to cooperative pathfinding problems. *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, pp. 173-178, 2010.

[6] Standley, T. & Korf, R., Complete algorithms for cooperative pathfinding problems. *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pp. 668-673, 2011.

[7] Cáp, M., Novák, P., Kleiner, A. & Selecký, M., Prioritized planning algorithms for trajectory coordination of multiple mobile robots. *IEEE Transactions on Automation Science and Engineering*, **12**, pp. 835-849, 2015.

[8] Li, J.-Q., Mirchandani, P.B. & Borenstein, D., The vehicle rescheduling problem: model and algorithms. *Networks: An International Journal*, **50**, pp. 211-229, 2007.

[9] Zuo, X., An immune algorithm based robust scheduling methods. *Handbook of Research on Artificial Immune Systems and Natural Computing: Applying Complex Adaptive Technologies*, pp. 124-140, 2009.

[10] Gawrilow, E., Klimm, M., Möhring, R.H. & Stenzel, B., Conflict-free vehicle routing: load balancing and deadlock prevention. *EURO Journal on Transportation and Logistics*, **1**, pp. 87-111, 2012.

[11] Wälter, J., *Existing and novel approaches to the vehicle rescheduling problem*, HSR University of Applied Science Rapperswil: Switzerland, https://eprints.hsr.ch/855/, 2020.