Multi-Agent Reinforcement Learning for railway rescheduling

1st Bálint Kővári
Department of Control for Transportation and Vehicle Systems
Budapest University of Technology and Economics
Budapest, Hungary
kovari.balint@kjk.bme.hu

2nd Csanád L. Balogh
Department of Control for Transportation and Vehicle Systems
Budapest University of Technology and Economics
Budapest, Hungary
csanadlevente.balogh@edu.bme.hu

3rd Szilárd Aradi
Department of Control for Transportation and Vehicle Systems
Budapest University of Technology and Economics
Budapest, Hungary
aradi.szilard@kjk.bme.hu

***Abstract*—Malfunctions, congestions, and accidents occur in every railway system from time to time, which influences the railway traffic on a given section of the system. The disturbance may cause inconvenience for several passengers and disruption in rail freight. Both the schedule and route of the affected trains must be modified to avoid further congestion and minimalize delays. The rigidity of the railway system (e.g., single tracks, vast distances without a service station, no viable alternative in case of malfunction) poses restrictions, unlike other transportation systems. Replanning schedules and train routes (called the railway rescheduling problem) is complex and demanding, even for human operators, as one must consider numerous factors. Thus, finding a satisfying solution poses a significant challenge. This paper presents a MARL-base (Multi-Agent Reinforcement Learning) solution that shows great potential for tackling this problem, even in the case of multiple connected stations.**

***Index Terms*—Traffic Control, Reinforcement Learning, Intelligent Transportation Systems, Multi-Agent Systems**

I. Introduction

A. Motivation

Rail transport has long been an essential and indispensable part of the transportation infrastructure in all countries, recently, with the growing importance of environmental awareness and sustainable development especially so. The increase in rail traffic, transport volume, and speed can significantly achieve a more sustainable transportation system. For instance, the European Union's main objectives include "greening" freight and transport by doubling rail traffic by 2050 and using new innovative data and artificial-intelligence-driven solutions in the rail sector, as formulated in the European Commission's plans for the future [1]. A more detailed discussion of such technologies can be found in [2]. Reinforcement learning (RL) is a valuable optimization method, and a promising AI-based tool for sequential decision-making problems in the transportation industry [3]. This work aims to solve the railway rescheduling problem with an RL-based approach.

B. Related work

The authors in [4] propose a neighborhood search algorithm to solve the rescheduling problem in case of metro system failures. [5] shows it can work on a real network. One can also utilize tabu search to solve the rescheduling problem, which minimizes delays by calculating optimal train sequences [6]. The alternate graph method is suitable to solve the problem as well [7]. Another option is to use the mixed linear integer programming method, as the authors suggest in [8]. OpenTrack environment can provide a base for the implementation [9]. Another viable approach would be the Monte-Carlo tree search procedure, which can find a conflict-free route in a short time [10]. Using a graph representation and a Deep Q-Network (DQN) leads to a solution as well [11], [12]. In some applications, such as a closed metro network, Q-learning can even provide an energy-optimal solution.

C. Contribution

The paper proposes a Multi-Agent Deep Reinforcement-Learning-based solution (MADRL) to the real-time rail rescheduling problem with two different state representations. The first is a two-dimensional image-like representation, and the second contains the same information as a single vector. The control policy is formulated using a CNN and a dense neural network. Given the complexity of the real-world problem, many different objectives can be utilized. Our primary focus is to avoid deadlocks (i.e., congestion that cannot be solved by simple route changes). It is also desirable that trains reach their destination as efficiently as possible, within minimal travel time. Deadlock occurs when trains running opposite each other simultaneously use the same track section. None of them can choose another route at a switching point. One of the trains would have to reverse to allow traffic to continue. The aim is to solve the problem without deadlocks and with minimum journey times. Another main objective of this work is to find ways to support generalization, so the most significant contribution is the introduction of virtual agents. Generalization is an essential property of neural networks, i.e., after solving a given problem, the same network can solve a similar but more complex issue. The following sections show how a neural network can be trained using a single station that can later solve multiple consecutive stations with virtual agents.

II. Methodology

A. Reinforcement learning

Reinforcement learning (RL) is an area of machine learning, which gained tremendous interest recently thanks to its wonderfull results in several fields [13], [14]. Unlike supervised and unsupervised methods, it does not learn from a predefined set of data but from experimentation and experience from interacting with an environment in a trial-and-error-based manner [15], [16]. RL is residing on the border of control theory and data science. While interacting with a complex environment, the agent searches for an optimal control strategy (policy). The policy is established based on the positive or negative reward it receives from the environment and the agent's current state. Since the agent's goal is to maximize the cumulated reward over the episodes. In Deep Reinforcement Learning, the neural network incorporates the policy, and the control policy development is a matter of tuning the network's weights. Figure 1 shows the RL training loop.

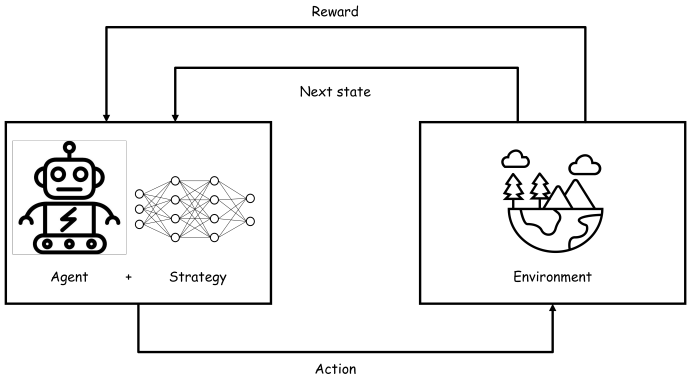

Fig. 1. RL learning loop

In MADRL, several agents participate in solving a control problem in a shared environment, i.e., several agents make decisions in parallel, working on a shared problem. The agents interact with the environment and, in many cases, with each other. Simultaneous decision-making creates a demanding learning process due to the increased dimensionality and the scalar reward system. In this paper, the learning concept allows agents to go through the learning process individually while the function approximator performs the computations for one agent at a time. This is called the independent-learner MARL approach. The core of this concept is that one Neural Network is trained, and the state representation is created from the aspect of all agents individually interacting in the environment. The representation is designed to allow agents to distinguish between walls and other agents but still handle them as a part of the environment.

B. Deep Q-Network algorithm

The learning process tunes the neural network's weights, representing the connections of neurons, which is the case in real biological neural systems. The ultimate goal is to ensure the network can select the appropriate action for a given agent based on the agent's current state. For that, the well-known Deep Q-Network algorithm is used, which tries to approximate the cumulated reward of each action in every state. This goal is achieved by utilizing the Bellman equation that allows calculating target-values for the formulated regression problem that once again is the approximation of the cumulated reward for every action in every state:

$$Q(s_t, a_t; \theta_t) = r_{t+1} + \gamma \max_a Q(s_{t+1}, a_t; \theta_t^-) \qquad (1)$$

where s_t is the current state at time t, a_t is the chosen step also at time t. γ is the discount factor, indicating the extent to which the agent wants to learn from the distant future and its emphasis on the possibility of future rewards. If the value of γ is 0, the agent does not care about future potential, and only those actions that produce an immediate good outcome receive a good value. When γ approaches 1, the agent places increasing importance on future reward values. It is an important tradeoff. The strategy needs to keep the more distant consequences of the actions in mind to the right extent. θ contains the values of the weights of the main network, while θ^- contains the values of the target network. Finally, r_{t+1} is the reward value received from the environment at time $t+1$. As can be seen, the new value depends on the current reward value and the future reward value to the extent of the discount factor. Since the current reward is always highly weighted, the net constantly moves towards the appropriate output by combining its current and past experience.

III. Environment

Modeling the environment is a significant part of the learning process since the agent learns the appropriate behavior by observing the environment and the patterns of change in it. Choosing a concise representation is recommended to make the teaching process more effective. The first state representation is two-dimensional and image-like, and the second is one-dimensional. Both models contain the layout, the destination, the position of the current decisionmaker, and all the other agents inside the same station, with a distinction based on directionality (traveling in the positive or negative direction). The station model includes two end positions (destinations), one for trains traveling in the positive direction (right) and one for trains traveling in the negative direction (left). Only the agent traveling in the appropriate direction can choose one of the two branching paths (tracks) when reaching a switch since trains are prohibited from traveling backward.

1) Action space: At each step, the agent can choose between four directions (right, left, up, down) or decide to wait. In a potential deadlock situation, frequently, the only viable option is to wait for another train to pass. Therefore the action space is augmented with a fifth waiting action, where the agent does not move in any direction. With this action space, the agent can choose an unavailable move in a given cell, which may not seem efficient initially. This design aims to teach the agents to select the correct action from the possible actions in a given position. This way, a given step

represents the same meaning regardless of the current position, which supports generalization. Thus, the current state of our action space is independent of the others and consistent with the representation. The one-dimensional representation works similarly. However, it does not contain walls. There are no invalid directions. The agent can choose to move forward (at intersections in two directions) or wait, as in the two-dimensional case.

2) Two-dimensional representation: Figure 2 highlights the different types of information indicated with different values in the representation.

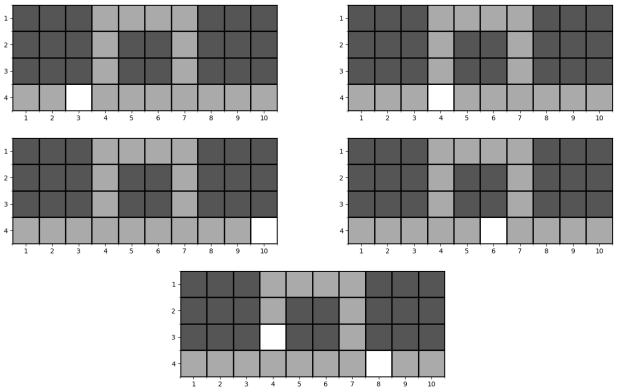

Fig. 2. Two-dimensional state representation

Grey indicates the free path, and black shows the walls. The five types of information that shall govern the formulation of a control policy are marked with white on every picture. The upper right image shows the agent's position. To help the agents determine the correct action choices, a wall is raised (so to say) in every situation to prevent the agent from traveling backward. It only appears in the representation, marked by white in the upper left picture. The left middle image contains the destination, which is also crucial information. The remaining two pictures indicate the agents traveling in positive and in negative directions, respectively.

3) One-dimensional representation: The one-dimensional representation (Fig. The one-dimensional representation (Fig. 3) contains the same valuable information without the walls being present. The first line indicates the self and the goal position, and the remaining two are the agents traveling in the same and the opposite direction, respectively. The vector consists of three parts to make the representation more separable for the agent. Based on experience, loose representation helps separate the state space, accelerates convergence, and might even improve the final results. The more compact portrayal of the environment lets us have a higher information density and a less complex state space. However, due to the spatial information, the convolutional net with the two-dimensional representation might be able to generalize over different topologies, which is not the case in the one-dimensional version. Figure 2 and Figure 3 show the same traffic situation.

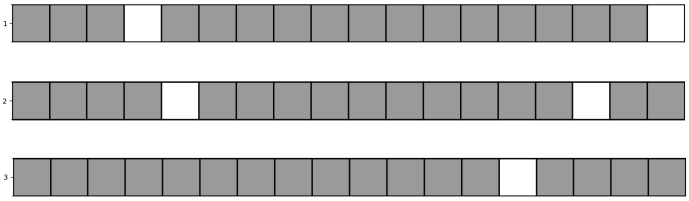

Fig. 3. One-dimensional state representation

A. Virtual agents

The main contribution of this paper is the introduction of virtual agents. One of the primary goals of our work is to support generalization, i.e., to ensure that after training on a simple problem, the neural network can solve a problem of higher complexity. In the present case, the goal was to ensure that the neural network could solve the rescheduling problem in a significant proportion of two connected stations after learning on a single station. All agents have to be aware of their station. The fundamental problem with connecting stations is the need for the agents' awareness of other stations. It makes generalization of the representation considerably more challenging and the state space extensively more complex (also dependent on the number of stations). Virtual agents are introduced to overcome this problem. If an agent can only see its own station, it can be dangerous to move to another station, as this may lead to a deadlock situation. However, it is sufficient for an agent to know whether the path to the next station is free or if continuing the journey would result in deadlock. A single agent can block the track or a combination of agents at the next station, traveling in the opposite direction, as in the scenario in Fig. 4.

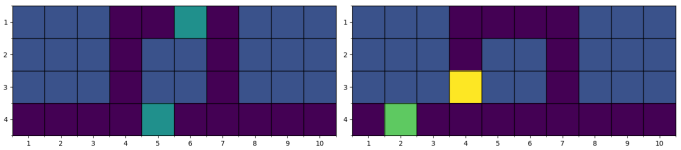

Fig. 4. Example of virtual agent

Fig. 4 shows two consecutive stations. By reaching the lower right cell of the left station, the agent is transferred to the right station, and the lower left cell of the right station leads to the left station. In this Example, the agents moving to the right (marked by blue) block all available paths. Consequently, a deadlock would occur if the yellow agent continues its journey toward the lower left corner (negative destination). Instead of including other stations, only the virtual agent (marked by green) is present in the yellow agent's representation (on the agent's own station); thus, it is sufficient to see only the current station for every agent. It is also important to note that the virtual agent occurs in the representation the same way an

ordinary agent traveling in the opposite direction would occur.

B. Reward strategy

Another critical element to successful training is the reward strategy. Agents use the reward value to get feedback on how good or bad a given step is in reaching the known goal of the process. While the state representation lets them know what the world looks like around them, the reward value indicates whether they are moving in the right direction in that world. If they receive a positive reward, they are more likely to take that step, but if the agents receive a negative reward (punishment), they are less likely to take that step again.

1) Fully competitive strategy: Agents get a negative reward for each step, so the more they take, the more negative points they accumulate. The agent that reaches its goal gets a large positive reward minus the negative amount collected for the steps taken so far. In effect, the number of steps scales the size of the final reward earned by the agent. If an agent creates a deadlock situation, it receives a large negative reward to discourage actions that result in a deadlock.

$$\text{reward} = \begin{cases} r_d, & \text{deadlock} \\ r_c - r_s, & \text{successful completion} \\ -r_s, & \text{exceeding step limit} \end{cases}$$

Where r_d is a large negative reward, r_c is a large positive reward, and r_s is a step-dependent reward, subtracted from the reward for successful completion. If agents run out of maximum steps, they receive a negative reward equal to that number of steps. Such reward strategy implements the concept of full competitive learning, i.e., every agent receives a reward independently of one another. Each agent wants to achieve their own goal without paying attention to the others or the common goal.

2) Cooperative strategy: Although initial success is achieved with the previous concept, an improvement to the reward strategy is inevitable and beneficial for obtaining the common goal. While each agent's goal is to reach their destination, the common goal is for everyone to reach the destination without disruption. Suppose an agent gets to its destination, but other agents cannot complete the episode, perhaps running into a deadlock. In that case, the shared state space includes a pattern in which an agent receives a positive reward, even though other agents blocking each other's paths made successful completion impossible. To avoid this, a fourth, neutral type of reward is introduced, which does not punish the agent that successfully completes the episode but does not reward it since the common goal is not met.

$$\text{reward} = \begin{cases} r_d, & \text{deadlock} \\ r_c - r_s, & \text{successful completion for all agents} \\ r_n, & \text{successful completion for given agent} \\ -r_s, & \text{exceeding step limit} \end{cases}$$

Where the notation is unchanged, but the neutral reward r_n is introduced.

IV. Results

This section presents the learning process and evaluation results using the methods and concepts described above. The authors illustrate the training concept and, in turn, the improvement in successful generalization ability.

A. Initial concept

In the first attempt, each episode starts by randomly choosing two agent positions and the direction of progress, also randomly. The only constraint is that the initial position cannot be a deadlock situation, in which case the agents wouldn't have a chance to solve the problem. Agent positions are always randomly selected, and a maximum step count (is also set. The first rewarding strategy described by section III-B1) is applied during the training. Ince a deadlock situation means failure to reach the goal for both agents, it is unnecessary to consider the common goal. The common goal automatically becomes unachievable together with the individual goal of both agents if deadlock occurs. Figure 5 shows the Convergence for different representations.

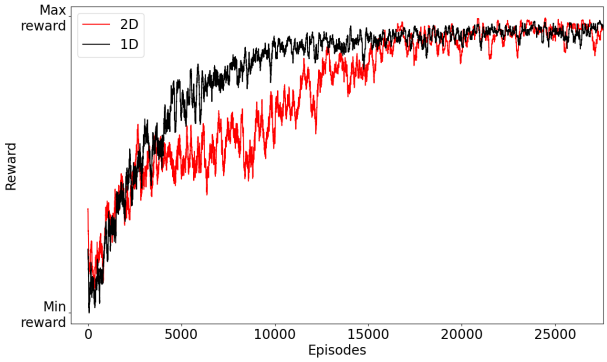

Fig. 5. Convergence of different representations

Although the convergence differs between representations, the final trained network achieved 100% efficiency in solving the original problem (One station, two agents) in both cases. The random seed is different during evaluation and training. It was anticipated that the one-dimensional version would be quicker to converge because of its simplicity, which is precisely what happened. The next step was to connect two stations with two agents on both to see how the networks could generalize. Table I shows the rate of successful completion.

TABLE I
SUCCESS RATE OF THE FIRST ATTEMPT

Representation	**Succes rate**	
	Two agents, One station	***Two agents, Two stations***
2D	100%	63.42%
1D	100%	70.26%

This paper focuses on generalization; hence the goal of the trained network is to solve the rescheduling task with

a high success rate on two stations. As seen from the table above, the agents could complete the original task without any problems, and they also found a suitable solution for two stations at a much higher rate than a random agent would. However, a higher success rate is expected. The one-dimensional representation can solve the task more efficiently, presumably due to the simpler state space. The histogram in Fig. 6 shows the number of steps required for both agents to reach their destination in two stations.

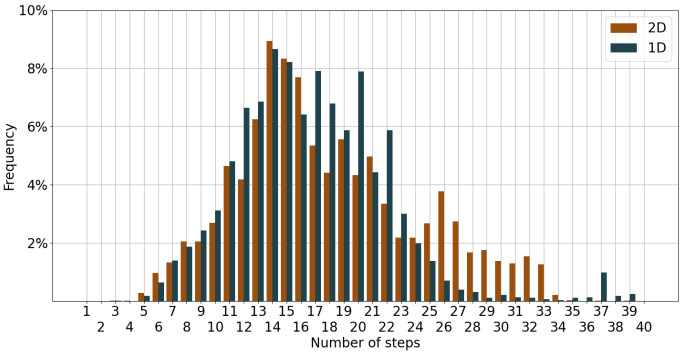

Fig. 6. Number of steps taken until completion (Initial concept)

The one-dimensional representation performs better, but not outstandingly so. The are a total of four agents in play. The connection of the stations acts as a bottleneck since only one agent can cross at a time. The fact that most scenarios are solved within 35 steps, and all of them are solved within 40 steps, lets us believe that the solution usually is of minimal or close to minimal step count. Examination of individual cases supports that too.

B. Expansion and diversification of state space

As mentioned in the previous section, since the environment is designed to support generalization, the aim is to achieve a success rate higher than 70% (as with the initial concept). Examination of failed episodes revealed the critical problem of the higher number of agents. Although two agents are at each station at the beginning, this number can quickly increase during the episode. This is problematic because it creates several situations where the state space contains unknown patterns. All agents have seen only one other agent in their station before; hence they are unprepared to meet traffic situations with 2-3 other agents, especially in cases where the required behavior is unlike any they had to apply during training. For instance, take a look at Fig. 7. In this case, the agent at coordinates (4, 4) traveling towards the right side will cause a deadlock by choosing the shorter path. With two agents present, choosing the shorter path is the optimal solution unless an agent with the opposite orientation already blocks it. Now the path is not blocked. If said agent continues, the two positive agents together will block all routes for the negative agent. The training process didn't prepare the agents for the underlying logic of the situation.

As a next step, an attempt is made to extend the state space by training the neural network using three agents and

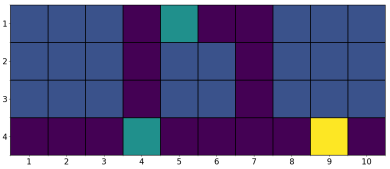

Fig. 7. Three agent conflict

evaluating the change of success by applying the new network to the same problem. A crucial difference in the concept is the following. While in the first case, there is no need for a distinction between the common goal and the goal of each agent, in the present case, two agents may run into a deadlock, making it impossible to meet the common goal. Still, the third one can evade them and achieve its own goal by reaching the destination. The higher number of agents has necessitated the introduction of a new reward strategy (described in section III-B2). The state space extension changed the two success rates to 85.56% from the original 63.42% with the image-like representation. While 86.44% from 70.26% in the case of the vector representation. The success of completion seems to lie within the sophistication of the state space, so the number of agents is also randomized in the following. At the beginning of each episode, one, two, three, or four agents are initialized with equal chance, and then, as before, each of their positions is also randomly determined. Figure 8 shows the convergence.

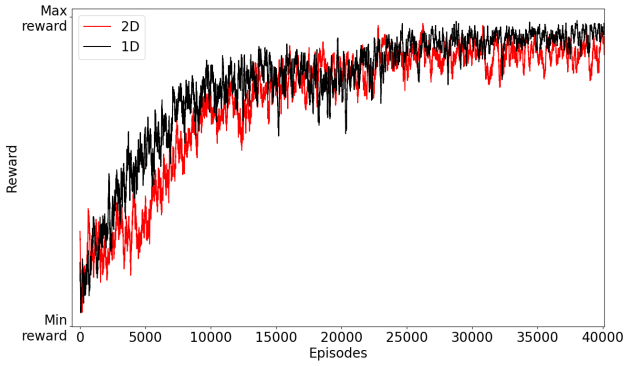

Fig. 8. Convergence of the improved concept

Again, the one-dimensional representation converges quicker, although by a smaller margin than before. In both cases, a dip is observable, with a slight fluctuation. The random number of agents can explain it, and adding diversity to the state space imposes a more significant challenge on the neural network to find adequate patterns. The two representations converge to a similar value. Although the overall convergence seems to be slower, the success rate has increased an additional 3.67% and 6.49%, as seen in Table II.

The more than 90% success rate indicates a great potential

TABLE II
COMPARISON OF SUCCESS RATES

Representation	Succes rate		
	Two agents	*Three agents*	*Varying agents*
2D	63.42%	85.56%	89.23%
1D	70.26%	86.44%	92.93%

Results of evaluation on two stations

for generalization. Figure 9 shows the number of steps to assess effectiveness. The substantive test again is two stations with two agents. This neural network is more successful than the previous ones; however, no substantial difference is observable between the number of steps taken, indicating that the efficiency is much alike. A slight difference is that the chart seems flatter, and the number of steps is more evenly distributed. A likely explanation is that the more complex traffic situations take more steps to solve, and most of the scenarios that the first network can not solve but the second can, are more complex.

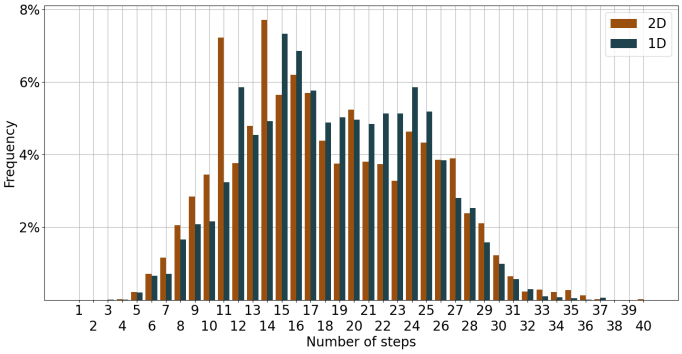

Fig. 9. Number of steps taken until completion (Refined concept)

C. Future improvements

Careful examination showed that the driving cause of unsuccessful episodes is agents taking specific steps simultaneously. For example, take two stations and one agent at both stations. The path is blocked if agents step into the connecting section between the stations. Suppose both agents are on the last cell before the connecting section. Since the path is not blocked from either direction, no virtual agents are present; hence, the agents step into the last section simultaneously, resulting in a deadlock. To resolve such conflicts, deeper communication between agents is under development so that they can negotiate the priority of passing.

V. CONCLUSION

The paper presented a solution for real-time railway rescheduling based on Multi-Agent Deep Reinforcement Learning. It provides a detailed comparison between the two different state representations. The reader can walk through the thought process that guided the improvement in concept and implementation by expanding and diversifying the initial state space. The results have demonstrated that virtual agents support generalization immensely. Agents need to know only their own station instead of the entire environment, avoiding the rapid growth in dimensionality with the increased number of stations. This paper can be considered a proof of concept study on the virtual agent concept. This concept is planned to be explored further in future studies.

ACKNOWLEDGMENT

The research was supported by the European Union within the framework of the National Laboratory for Autonomous Systems. (RRF-2.3.1-21-2022-00002)
This paper was also supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences

REFERENCES

[1] European Commission, “Sustainable and Smart Mobility Strategy—Putting European Transport on Track for the Future,’, 2021.
[2] V. De Martinis, F. Corman, “Data-driven perspectives for energy efficient operations in railway systems: Current practices and future opportunities,” Transportation Research Part C: Emerging Technologies, Vol. 95, pp. 679-697, 2018.
[3] N. P. Farazi, B. Zou, T. Ahamed, L. Barua, “Deep reinforcement learning in transportation research: A review,” Transportation Research Interdisciplinary Perspectives, Vol. 11, ISSN 2590-1982, 2021.
[4] M. Botte, C. Di Salvo, A. Placido, B. Montella, L. D’Acierno, “A neighbourhood search algorithm for determining optimal intervention strategies in the case of metro system failures,” International Journal of Transport Development and Integration, Vol. 1, Issue 1, pp. 63 - 73, 2017.
[5] M. Botte, L. D’Acierno, “Dispatching and Rescheduling Tasks and Their Interactions with Travel Demand and the Energy Domain: Models and Algorithms,” Urban Rail Transit, Vol. 4, pp. 163–197, 2018.
[6] F. Corman, A. D’Ariano, D. Pacciarelli, M. Pranzo, “A tabu search algorithm for rerouting trains during rail operations,” Transportation Research Part B: Methodological, Vol. 44, pp. 75–192, 2001.
[7] A. D’Ariano, F. Corman, D. Pacciarelli, and M. Pranzo, “Reordering and Local Rerouting Strategies to Manage Train Traffic in Real Time,” Transportation Science, Vol. 42, pp. 405-419, 2008.
[8] L. Lindenmaier, I. F. Lövétei, G. Lukács, S. Aradi, “Infrastructure Modeling and Optimization to Solve Real-time Railway Traffic Management Problems,” Periodica Polytechnica Transportation Engineering, Vol. 49(3), pp. 270–282, 2021.
[9] P. Pellegrini, G. Marlière, Grégory, J. Rodriguez, “A detailed analysis of the actual impact of real-time railway traffic management optimization,” Journal of Rail Transport Planning and Management, Vol. 6, pp. 13-31, 2016.
[10] I. F. Lövétei, B. Kővári, T. Bécsi, “MCTS Based Approach for Solving Real-time Railway Rescheduling Problem,” Periodica Polytechnica Transportation Engineering, Vol. 49(3), pp. 283–291, September 2021.
[11] M. Obara, T. Kashiyama, Y. Sekimoto, “Deep Reinforcement Learning Approach for Train Rescheduling Utilizing Graph Theory,” IEEE International Conference on Big Data, pp. 4525-4533, 2018.
[12] I. Lövétei, B. Kővári, T. Bécsi, Sz. Aradi, “Environment Representations of Railway Infrastructure for Reinforcement Learning-Based Traffic Control,” Applied Sciences, Vol. 12, 2022.
[13] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... Hassabis, D. (2015). Human-level control through deep reinforcement learning. nature, 518(7540), 529-533.
[14] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., ... Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. nature, 529(7587), 484-489.
[15] Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., ... Hassabis, D. (2021). Highly accurate protein structure prediction with AlphaFold. Nature, 596(7873), 583-589.
[16] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., ... Hassabis, D. (2017). Mastering the game of go without human knowledge. nature, 550(7676), 354-359.