Intelligent Railway Capacity and Traffic Management Using Multi-Agent Deep Reinforcement Learning

Stefan Schneider[1,*], Anne Roets[2], Ciprian Stirbu[3], Farhad Safaei[1], Faten Ghriss[3], Jan Wülfing[1], Mehmet Güral[1], Nima Siboni[3], Roman Liessner[1], Thomas Hustache[3], Thomas Lecat[3], Umashankar Deekshith[2], Valerii Markin[3], Victor Le[3], Wissam Bejjani[3], Michael Küpper[1], and Irene Sturm[1]

Abstract— **A more attractive future railway system needs to offer more capacity in the railway network and improve quality and punctuality. A fundamental centerpiece of future digitized railway network operations is automated and optimized planning and dispatching. The sector initiative "Digitale Schiene Deutschland" (DSD) develops a holistic and intelligent Capacity & Traffic Management System (CTMS) that can automatically plan and continuously optimize railway traffic at scale. Both, planning and dispatching tasks, are highly complex and, today, require human expertise and oversight.**

Our main contribution is a multi-agent deep reinforcement learning approach at the core of the envisioned CTMS, which learns from interaction with a realistic, microscopic railway simulation. Our results demonstrate that the proposed approach flexibly solves planning and re-scheduling tasks in the realistic setting of a medium-sized part of the German railway network. It exhibits response times and scaling properties that make it a promising candidate for future applications in railway operations at scale.

I. Introduction

Moving traffic from other modes of transport to rail will significantly contribute to reducing carbon emissions and help fight climate change. To accommodate more railway passengers and freight traffic, the railway system must offer significantly more capacity and quality to its customers. With over 33 000 route kilometers, Germany has Europe's largest railway network. This network is shared by dense traffic of roughly 40 000 trains rides per day, containing mixed traffic of regional, long-distance, and freight trains with vastly different characteristics [1]. This poses challenges for both capacity and reliability.

Constructing timetables that maximize the existing network's capacity by distributing train traffic is a highly complex task. Today, this requires many months of preparation by human experts. Similarly, the quality of railway operations today depends on constant oversight by thousands of human dispatchers and signallers. To keep up the flow of traffic, they monitor parts of the network and need to quickly react to disruptions such as malfunctions or unforeseen delays, coordinating with other dispatchers and signallers of nearby areas to reroute trains and adjust schedules.

With growing capacity needs and even denser traffic, today's largely manual planning and dispatching processes quickly reach their limits. Hence, the sector initiative "Digitale Schiene Deutschland" (DSD) has the goal to create a fundamentally modernized railway system that optimizes the capacity in the rail network and improves quality and efficiency [2]. A core part of DSD's vision is a Capacity & Traffic Management System (CTMS), which can be understood as a central planning and dispatching machine over all planning horizons. CTMS is able to create microscopic schedules at all planning stages with a level of detail that enables the automated operation of trains and the automated control of digital interlocking. Thereby, CTMS seamlessly integrates the processes of railway planning and operations that are separated today. CTMS also supports a train-centric moving block safety logic, which will further increase the capacity on the railway network where implemented [3].

Developing the envisioned CTMS requires overcoming many challenges. CTMS relies on an accurate, up-to-date model of the railway infrastructure and vehicles. It needs a fine-grained and realistic simulation to evaluate impact and feasibility of different planning and operations decisions at scale. Finally, CTMS needs an optimization component to produce microscopic, executable schedules that maximize capacity, quality, and efficiency under a large number of constraints and with a complex optimization function. In this paper, we focus on the optimization component of CTMS.

Inspired by the success of Deep Reinforcement Learning (RL) in recent years, both in real-time continuous control and in strategic planning [4], we propose a Multi-Agent Deep Reinforcement Learning (MARL) approach. We formalize the corresponding Partially Observable Markov-Decision Process (POMDP) to match the real-world requirements of railway capacity and traffic management. Our approach learns from interaction with a realistic railway simulation, flexibly adapting to various scenarios and generalizing to unseen situations. In doing so, it can produce and continuously adjust schedules for planning as well as dispatching purposes. Our approach is scalable thanks to the multi-agent architecture, where the schedules of a high number of trains can be processed in parallel while considering global objectives. While training MARL models is time-consuming, trained models can produce solutions quickly, promptly reacting to disruptions or changed requirements.

In the current, still prototypical implementation, the ap-

All authors contributed equally to this work.

[1] DB InfraGO AG, EUREF-Campus 17, 10829 Berlin, Germany

[2] DB Systel GmbH, Jürgen-Ponto-Platz 1, 60329 Frankfurt, Germany

[3] InstaDeep Ltd, 5 Merchant Square, London, W2 1AY, UK

[*] Corresponding author. Email: stefan.sh.schneider@deutschebahn.com

proach already handles and coordinates dense railway traffic in a medium-sized corridor. We demonstrate its capabilities on a part of the German railway network with 29 stations including a large central station. We show its ability to construct initial timetables as well as to react to disrupted tracks by automatically rerouting (directly and indirectly) affected trains and adjusting their schedules.

II. Related Work

Early studies applying RL to transportation problems, e.g., by controlling traffic signals [5], have demonstrated that RL can effectively reduce congestion in dynamic transportation environments. Due to the use of tabular single-agent RL, such initial approaches are still limited to simple, small-scale problems. Most recent approaches leverage Deep RL, which, unlike tabular RL, supports large and continuous state spaces. Farazi et al. survey such Deep RL approaches for transportation, including railway [6], where scheduling and routing is very challenging and constrained by the limited amount of tracks that are shared by diverse train types.

The surveyed work [6] as well as the very recently published work by Gorsane et al. [7] propose single-agent Deep RL approaches that focus on rescheduling trains in railway operations. Gorsane et al. [7] work on an abstract event graph representing the schedule of all trains. Their approach iteratively modifies this event graph and the corresponding schedule by choosing from a predefined set of actions, e.g., to re-order trains and adjust timings. While a heuristic is needed to create the initial schedule, their RL approach can effectively reduce overall delay thanks to its global view of the schedule and its restricted action space. In contrast, our proposed MARL approach can produce and adjust railway schedules from scratch, allowing more degrees of freedom without requiring a feasible starting solution. Our MARL approach can also overcome potential scalability limits of single-agent RL approaches by distributing control over multiple parallel actors for each train [8].

Most related to our approach, Laurent et al. propose a MARL approach for railway transportation [9]. To some degree, our approach builds on theirs in terms of POMDP design, e.g., limiting the complexity of large state spaces in railway networks by constructing tree-based local observations ahead of each train. However, Laurent et al. focus on a simplified, grid-based railway simulation. While this simplification makes the problem more tractable and easier to compute, it significantly reduces realism and real-world applicability especially in future railway systems, where maximizing the capacity of networks is crucial. Framing problems realistically, thereby ensuring real-world applicability, is a core challenge of RL for transportation [10]. Therefore, our approach integrates a realistic, microscopic simulation environment, which provides continuous observations, supports realistic driving physics and speed profiles, and enables various train types and safety systems as well as other highly relevant features of a real-world railway system.

III. Problem Statement

In today's railway systems, planning and operations are strictly separated processes, involving different departments (planners vs. dispatchers), business rules, time scales (months in advance vs. in real-time), and objectives. Strongly simplified, the goal of planning processes is to generate a timetable that accommodates requests for train runs (by train operators), while the goal of processes in operations is to execute a pre-planned schedule. Although business processes at both stages may continue to differ in the future, we propose a problem formulation that addresses both problems and therefore contributes to a more seamless connection between the two domains. Among others, this will allow to better include expected operational constraints at the planning stage, which will lead to schedules that are easier to execute. In this section, we outline our joint problem formulation, indicating the problem parameters, the variables controlled by our approach, and the optimization goal.

A. Problem Parameters

There are three main problem parameters, given as input: 1) The railway network to consider, 2) the train runs requested on this network, and 3) the current operational state of trains and switchable elements in the railway network.

We consider the *railway network* at a microscopic level of individual tracks and points connecting these tracks. Commonly, points connect three tracks and switch between two options. Tracks may have stopping locations at certain positions, e.g., at platforms inside a train station. At the border of the considered railway network, trains can transition from/to other areas of control through "handover gates".

Train runs define at which handover gate a train arrives in the railway network, which stops to traverse in which order, and at which outgoing handover gate the train is expected to leave the network. Stops in a schedule can be specified as a group of alternatives, e.g., all platforms within a given train station, or a precise stopping location at an exact position.

Finally, the *operational state* specifies which trains are currently in the railway network, their train types, positions, velocities, and progress within the train runs at a given time. Additionally, it contains the current positions of all points and any restrictions in the railway network, e.g., non-traversable tracks or malfunctioning points. Any change of the operational state during real-time operations can trigger a re-calculation of the schedule. For planning tasks, a stable operational state is assumed.

B. Problem Variables and Goal

Based on the aforementioned input, the CTMS has the task to produce different types of information: On the one hand, it has to produce all information required to execute train runs, such as train routes from handover via stops to handover, velocities, and details of commercial stops. On the other hand, it needs to produce all necessary requests to the digital interlocking for setting points (and other network elements) and granting movement permissions to the trains. All commands have to take into account the rules of the safety logic and other constraints such as speed limits. Safety-technically, the CTMS is not responsible for guaranteeing collision-free execution of railway traffic on the network. This task is performed by a separate system (the interlocking), in our case the Advanced Protection System (APS) [3]. Nevertheless, CTMS shall observe the rules of the safety system in order to produce executable commands.

Fulfilling the aforementioned task entails routing trains and, especially in dense traffic, coordinating between trains. Such coordination is very challenging for railway traffic since the number of available tracks in the railway network and their connectivity are highly limited. Wrong navigation decisions can lead to a cascade of effects that result in congestion or even deadlocks much later and faraway from the original cause. Similarly, disruptions of a single track (e.g., by a fallen tree) can have far-reaching effects on the entire schedule. In comparison, traffic management for cars or drones is much easier since they can easily evade small obstacles without blocking all following traffic.

IV. Proposed MARL Approach

To solve the aforementioned problem of railway capacity and traffic management (Sec. III), we propose a Multi-Agent Reinforcement Learning (MARL) approach. Our MARL approach controls each train individually, taking relevant information into account and issuing commands to steer trains through the railway network to complete their requested train runs. In contrast to our MARL approach, a typical single-agent RL approach would consider and control all trains at once, which easily leads to scalability issues [8].

The MARL approach interacts with a (simulated) railway environment by observing its current state, issuing commands (called actions), and receiving feedback through rewards. A main challenge in applying MARL to our problem is designing the underlying Partially Observable Markov Decision Process (POMDP), which defines the corresponding observations, actions, and rewards. In the following, we describe the considered simulated railway environment, our POMDP design, and the underlying RL algorithm.

A. Simulated Railway Environment

To train and evaluate our MARL approach, we utilize a simulation environment that is being developed at Deutsche Bahn in the context of DSD. It models the railway system with a high degree of detail, enables target features of the future digitized railway system, and is tailored to be used for training and executing MARL models.

Simulated level of detail: The simulated railway environment is microscopic in the sense that it models the network including all relevant field elements, down to individual point switches. It also allows fine-grained temporal control down to single seconds: Rather than just simulating predefined events (as it would be the case for an event-based simulation), it can simulate per-second train movements and allows interaction with the environment by issuing commands. Therefore, at any given time step the simulation yields an accurate picture of the operational situation in the rail network. This feature is crucial for the decision-making by our MARL approach.

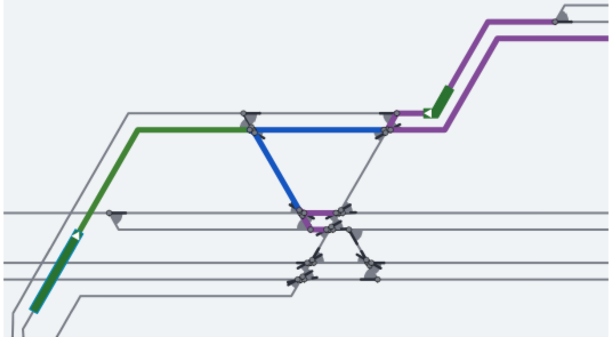

Fig. 1. Visualization of local, tree-based observations for the left train.

Simulated dynamics: The environment accurately simulates train physics, integrating between the discrete time steps without loss of precision. Unlike other available railway simulations that operate on a coarse-grained grid [11], our environment simulates with much higher precision (down to millimeters).

Realistic and artificial data: The simulation allows importing real railway networks, schedules, and other operationally relevant data. It also supports generation and use of artificial, yet realistic data, which is useful for MARL training.

Modular architecture: The simulation environment is built in a modular way that allows interaction with our MARL approach as well as other optimization approaches. It is also modular with respect to the simulated railway safety logic. In principle, it can accommodate different version of block-based safety systems as well as the future, train-centric safety logic of DSD's target picture [3].

B. Partially Observable Markov Decision Process (POMDP)

To solve the railway scheduling problem with MARL, we design the corresponding POMDP, taking information about train positions, velocities, and planned routes of all trains into account. In most real-world problems the full environment state with all underlying details is not explicitly known by the RL approach, i.e., it is only partially but not fully observable [12]. Similarly, observing the full environment state (entire railway network, all trains and switches, etc.) would be not scalable for our approach. Hence, it relies on partial information that is relevant and realistically available.

Formally, a POMDP is defined by the tuple (O, A, R, P, γ), where O are the available observations of the overall state, A the actions, and R the reward function, each designed by us. P is the transition function that describes the environment dynamics, i.e., formalizing what happens when applying a certain action in a specific situation, and is dictated by the (simulated) railway environment. The discount factor γ is relevant for the reward and explained later on. We formulate the observations, actions, and reward of our POMDP as follows:

Observations O: For each train, the MARL model receives information reflecting its partial perspective of the overall environment state in local, tree-based observations. These local observations contain information on the surrounding of the train, including track information, switch statuses, and positions of nearby trains, encapsulated in a tree structure. Fig. 1 illustrates an example tree structure for the train in the bottom left corner (tree depth 0 shown in green, depth 1 in blue, depth 2 in purple). Tree observations encode spatial relationships and track choices efficiently. However, their scalability is challenged by the exponential increase in observation size with tree depth, highlighting the need for a balance between depth and manageability of representation. Complementing the local observations, we designed observations for distant trains, summarizing corresponding data like their current track numbers, speeds, and destinations.

Actions A: The MARL model controls each train individually. For each train, it has two primary dimensions of control: 1) Adjustments to train speed and 2) switching upcoming points in the railway network ahead of the train. As many MARL algorithms require discrete actions, we discretize the speed control into a set of predefined velocities.

Reward R: The reward function is designed to incentivise completing the requested train runs. It gives a binary, positive reward of +1 for each stop reached by the train as well as for arriving at the final destination. The underlying RL algorithm tries to achieve rewards quickly, which motivates the MARL model to steer trains to their stops and destination as quickly as possible. This emphasis on early rewards and completing train runs quickly can be controlled by discount factor γ.

C. APEX DQN Algorithm

Our MARL approach leverages Distributed Prioritized Experience Replay, also known as APEX DQN algorithm [13]. APEX DQN improves over the popular Deep Q-Learning (DQN) algorithm [14] by its distributed architecture and prioritized experience replay, facilitating efficient learning in complex, multi-agent environments.

Our MARL approach controls each train individually, yet the model leverages a shared neural network for each train. Using a shared neural network, experiences are collectively processed, allowing for the rapid dissemination of insights and strategies across all trains.

V. Evaluation

A. Evaluation Setup

1) Evaluation Scenarios: We evaluate our MARL approach across different situations on parts of the Magdeburg railway network (Fig. 2). First, we consider dense traffic on a small network section in Sec. V-B. In Sec. V-C we then investigate varying traffic on the full Magdeburg network, also evaluating the impact of a disrupted track. Finally, Sec. V-D analyzes the scalability of our approach.

2) Baseline Comparison: We compare our MARL approach against a *greedy baseline*. The greedy approach always runs each train at maximum allowed speed. If routing decision have to be made, it chooses one of several options to reach the destination at random. This means that trains can individually reach their goals quickly and reliably. However, it does not explicitly consider coordination between trains, which is a key challenge in railway scheduling.

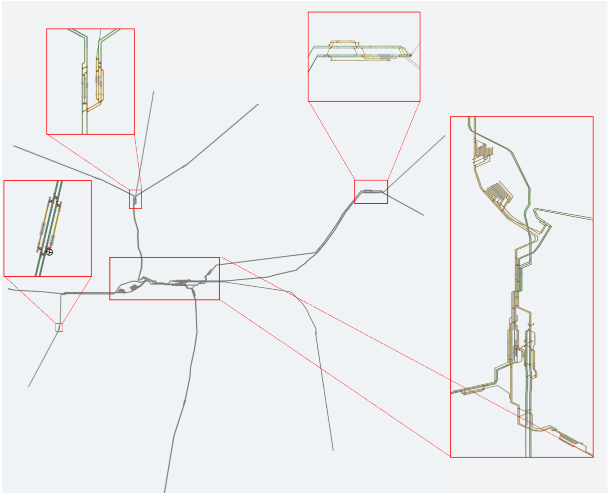

Fig. 2. Magdeburg railway network used for evaluation, consisting of 29 stations, 354 stopping locations, 619 junctions, and 894 bidirectional tracks with a total length of 343 kilometers.

TABLE I

Success rates of our MARL approach vs. a greedy baseline.

Evaluation Scenario	MARL Approach	Greedy Baseline
Small Network (Sec. V-B)	100%	69%
Large Network (Sec. V-C)		
Without Disruption	94%	14%
With Disruption	94%	16%
Scalability (Sec. V-D)		
19 Trains	100%	32%
41 Trains	99%	10%

3) Training Setup: Our MARL approach was trained for 13 million environment time steps, where each step corresponds to 20 s of simulated time. The approach was trained up-front on the different parts of the network in randomly generated situations. The overall training time was around 72 hours on an AWS cluster with 45 CPUs and up to 115 GB of RAM. While training is time-consuming, the trained MARL approach can create new railway schedules rapidly and generalizes to new and unseen situations.

B. Traffic Management on a Small Network

First, we evaluate our approach on a small part of the Magdeburg network containing one station with dense traffic (top left in Fig. 2). Specifically, we consider 100 different scenarios, each with 10 active trains with varying start positions and requested train runs. In each case, the trains need to stop at the station at one of the platforms. We are interested in how well our approach handles these dense traffic scenarios, controlling train routing and timing to distribute trains and avoid congestion and deadlocks. We measure and evaluate the success rate of completed train runs (in percent), where the requested stops and destinations have been reached.

As shown in Table I and Fig. 3, our MARL approach solves these scenarios perfectly, reaching 100% success rate.

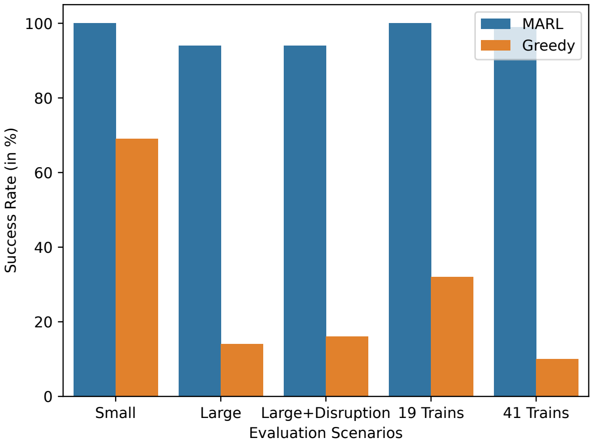

Fig. 3. Overview of evaluation results: Success rate of our MARL approach vs. a greedy baseline across all evaluation scenarios.

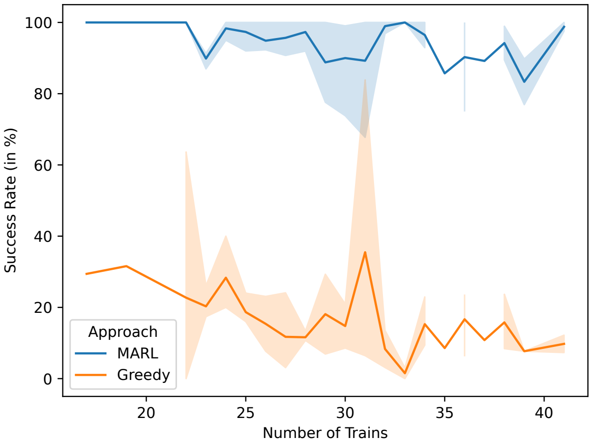

Fig. 4. Success rates of our MARL vs. a greedy approach on scenarios with increasing number of trains.

In contrast, the greedy baseline only achieves 69% success rate. The remaining train runs were not fully completed, as trains did not reach their requested stops or destinations due to congestion or deadlock.

C. *Traffic Management on a Large Network*

Following the dense scenarios on the small network, we now evaluate our approach on the entire Magdeburg network (Fig. 2). On this large network, we created 72 different scenarios with 17 to 41 trains and six distinct traffic classes of freight traffic, regional traffic, and long distance traffic. Each traffic class is associated with a different train type, characterized by specific technical attributes such as maximum speed, acceleration and deceleration capabilities. Traffic management on this larger network with more trains is much more challenging, because a) more trains and network elements (points) have to be controlled and b) more farsighted coordination between faraway trains and across longer time horizons is required.

1) Without Disruption: On average, our proposed MARL approach still manages to complete 94% of all requested train runs on the full Magdeburg network (Table I and Fig. 3). While the results are not perfect yet, the approach is already able to coordinate trains across relatively long distances and time horizons in almost all cases. In contrast, the greedy baseline performs very poorly and only completes 16% of the requested train runs on average. Because the greedy approach steers all trains along any of the available paths without waiting for or coordinating with other trains, a large majority of trains ends stuck in congestion and deadlock.

2) With Disruption: Next, we consider the same large Magdeburg network and the same 72 scenarios but introduce a disruption. Particularly, we assume that a heavily used track at the center of the network is blocked (e.g., by a fallen tree or a malfunction). Now, the task is to react to this disruption and reroute affected trains such that they can still fulfill their requested train runs – a use case where automation is highly desirable in today's railway systems.

The greedy approach is barely affected by the disruption because it fails to complete most requested train runs even without disruption. However, the schedules produced by our MARL approach without disruption (Sec. V-C.1) need to be adjusted significantly to avoid the blocked track. Our evaluation shows that the MARL approach successfully learns to reroute affected trains and still complete 94% of train runs successfully, even after such a major disruption (Table I and Fig. 3). In fact, the MARL approach not only reroutes trains whose original path was directly affected by the disrupted track. It also reroutes other, nearby trains to make way for the directly affected trains and to avoid congestion around the disruption. To illustrate the produced schedules of the MARL approach before and after disruption, we publish a video with visualizations of the corresponding schedules[1].

Note that the MARL approach has been trained with various disruptions but not explicitly with the track blocked here. Hence, it learned from its training experience and successfully generalized to this disruption that was likely never seen during training. This indicates that the approach, after diverse training, also handles unforeseen situations well.

D. *Scalability Analysis*

Finally, we investigate the scalability of our approach by evaluating its success rate on the large Magdeburg network with an increasing number of trains. Our MARL approach maintains a success rate of 100% for up to 22 trains. Beyond that number, the success rate slowly declines towards 90% (Fig. 4). The figure shows high variance in success rate, which is likely due to varying scenario complexity. The complexity not only grows with an increasing number of trains but is also determined by other factors, such as relative train positions and the specifics of the requested train runs. We plan to further investigate how to measure and control problem complexity in future work.

[1] The video still needs to be released by DSD and will be available for the final, camera-ready version of this manuscript.

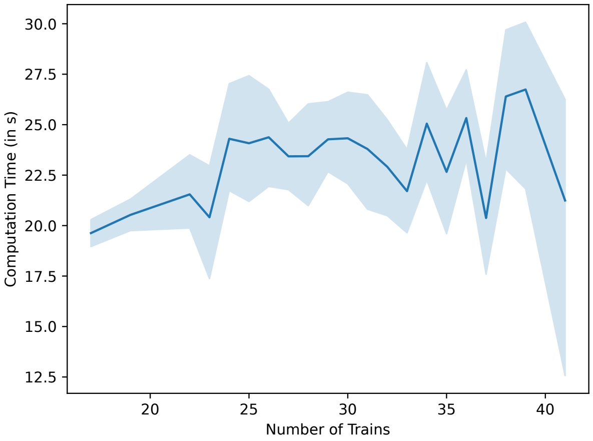

Fig. 5. Computation times for an increasing number of trains.

While the greedy baseline can reliably complete single, isolated train runs (not shown in Fig. 4), it does not coordinate between trains (as mentioned in Sec. V-C). This leads to success rates of only 32% for 19 trains on average. The lack of coordination by the greedy baseline aggravates for larger numbers of trains, which leads to an average success rate of only 10% for 41 trains.

In addition to the success rate, we also investigate the computation times of our MARL approach for increasing numbers of trains. Fig. 5 shows the average times needed to compute a complete schedule with growing numbers of trains on the large Magdeburg network. While there is high variance, the figure roughly shows a linear increase in computation time with the number of trains. Again, the variance is likely caused by varying scenario complexity and also partly by fluctuating utilization of the underlying hardware. The computed schedules represent scenarios of $42\,\mathrm{min}$ of (simulated) train movements on average. Computation times for these schedules are consistently below $30\,\mathrm{s}$, which is a massive improvement over today's manual processes. Furthermore, our MARL approach is significantly faster than common Operations Research approaches using mathematical optimization, where processing time grows exponentially with increasing problem size [7].

VI. Conclusion

We present a MARL approach for planning and dispatching railway traffic. Our approach was tested in a microscopically simulated railway environment that represents features of a future digitized railway system. The results show that our proposed MARL approach is reliably able to construct schedules on small but heavily utilized railway networks. On larger networks, where the challenge of coordinating trains at short and long range is much more pronounced, the MARL approach produces promising though not yet perfect results across a wide range of situations.

To further improve the proposed MARL approach, future research could tackle the POMDP design (Sec. IV-B), e.g., by constructing a more explicit reward for cooperation among trains. Performance might be further improved by fundamentally optimizing the underlying RL algorithms or neural network architectures.

Still, our findings indicate that the presented approach is able to cope with the problem's complexity and scales well. Importantly, our analysis suggests a very favourable linear relationship between problem size and computing time. Overall, our results show that our proposed MARL approach has the potential to enable a new level of automation in railway planning and dispatching, ultimately leading to more capacity and quality in the railway system.

Acknowledgment

This research was conducted in the context of the sector program "Digitale Schiene Deutschland" (DSD) established by Deutsche Bahn [2]. It is a part of the development of a Capacity & Traffic Management System (CTMS), which will optimize the planning and control of rail traffic to fully exploit the features and benefits of digitized and automated railway operation.

References

[1] "DB InfraGo Profile," https://www.dbinfrago.com/web-en/profile-12541196, accessed: 2024-03-18.

[2] "Digitale Schiene Deutschland," https://digitale-schiene-deutschland.de/en, accessed: 2024-03-18.

[3] "Advanced Protection System (DSD Initiative)," https://digitale-schiene-deutschland.de/en/Advanced-Protection-System, accessed: 2024-03-18.

[4] H.-n. Wang, N. Liu, Y.-y. Zhang, D.-w. Feng, F. Huang, D.-s. Li, and Y.-m. Zhang, "Deep reinforcement learning: a survey," *Frontiers of Information Technology & Electronic Engineering*, vol. 21, no. 12, pp. 1726–1744, 2020.

[5] B. Abdulhai, R. Pringle, and G. J. Karakoulas, "Reinforcement learning for true adaptive traffic signal control," *Journal of Transportation Engineering*, vol. 129, no. 3, pp. 278–285, 2003.

[6] N. P. Farazi, B. Zou, T. Ahamed, and L. Barua, "Deep reinforcement learning in transportation research: A review," *Transportation research interdisciplinary perspectives*, vol. 11, p. 100425, 2021.

[7] R. Gorsane, K. G. Mestiri, D. T. Martinez, V. Coyette, B. Makhlouf, G. Vienken, M. T. Truong, A. Söhlke, J. Hartleb, A. Kerkeni, *et al.*, "Reinforcement learning based train rescheduling on event graphs," in *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2023, pp. 874–879.

[8] E. Liang and R. Liaw, "Scaling multi-agent reinforcement learning," *Berkeley Artificial Intelligence Research Blog*, 2018, https://bair.berkeley.edu/blog/2018/12/12/rllib/.

[9] F. Laurent, M. Schneider, C. Scheller, J. Watson, J. Li, Z. Chen, Y. Zheng, S.-H. Chan, K. Makhnev, O. Svidchenko, *et al.*, "Flatland competition 2020: Mapf and marl for efficient train coordination on a grid world," in *NeurIPS 2020 Competition and Demonstration Track*. PMLR, 2021, pp. 275–301.

[10] A. Haydari and Y. Yılmaz, "Deep reinforcement learning for intelligent transportation systems: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 1, pp. 11–32, 2022.

[11] S. Mohanty, E. Nygren, F. Laurent, M. Schneider, C. Scheller, N. Bhattacharya, J. Watson, A. Egli, C. Eichenberger, C. Baumberger, *et al.*, "Flatland-rl: Multi-agent reinforcement learning on trains," *arXiv preprint arXiv:2012.05893*, 2020.

[12] G. Dulac-Arnold, D. Mankowitz, and T. Hester, "Challenges of real-world reinforcement learning," *arXiv preprint arXiv:1904.12901*, 2019.

[13] D. Horgan, J. Quan, D. Budden, G. Barth-Maron, M. Hessel, H. van Hasselt, and D. Silver, "Distributed prioritized experience replay," in *International Conference on Learning Representations*, 2018.

[14] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.