24th International Congress on Modelling and Simulation, Sydney, NSW, Australia, 5 to 10 December 2021
mssanz.org.au/modsim2021

Freight train scheduling via decentralised multi-agent deep reinforcement learning

A. M. C. Bretas [a], **A. Mendes** [a], **S. Chalup** [a], **M. Jackson** [b], **R. Clement** [b] **and C. Sanhueza** [b]

[a] *The University of Newcastle, New South Wales, Australia*
[b] *Hunter Valley Coal Chain Coordinator, Broadmeadow, NSW, Australia*
Email: c3308652@uon.edu.au

Abstract: Rail traffic planning and scheduling problems have been challenging academy and industry for a few decades. Specifically, problems in the short term and real-time horizons deal with simultaneous decision-making of trains, stations and terminals. Approaches focused on decentralised decision-making have been successful in delivering real-world committed solutions. This work focuses on decentralised real-time decision-making in a closed freight rail network and applies multi-agent deep reinforcement learning (MADRL) to find efficient timetables.

We apply the MADRL model to solve the traffic decisions arising in the Hunter Valley Coal Chain (HVCC) in New South Wales, Australia. The approach uses the same simulation model currently in use for capacity planning of the system, thus allowing tests with real data. The environment is modelled as a decentralised, partially observed Markov decision process (dec-POMDP), where the train, load point, and dump station agents decide upon train movements based on local observations. The observations follow a novel state encoding strategy for rail traffic management composed of nine layers. We benefit from this strategy to apply a decentralised execution with a centralised learning approach through proximal policy optimisation.

The experiments revealed a significant performance improvement for the ten instances tested, which reproduce the challenges faced in the HVCC operations. The approach is suitable for varied levels of rail network complexity, generating efficient solutions without scaling issues. The MADRL outperformed the heuristic in use by HVCC's simulation model and a high-performance genetic algorithm in all instances, reaching performance improvements of up to 72.00% and 47.42%, respectively. Therefore, the framework with the MADRL and the simulation model allows its application with real world instances in an efficient and reliable way. These results show the method's consistency and draw a safe path towards a decentralised rail traffic management system.

Keywords: *Multi-agent deep reinforcement learning, simulation-based machine learning, decision-making, rail traffic management, train scheduling*

1 INTRODUCTION

Freight trains usually follow a tight schedule that matches the needs of train operators, stations, terminals and crews, among others. Even though medium and long-term train scheduling problems are complex tasks, efficient exact and heuristic approaches are available (Corman and Meng [2015]). However, when it comes to short term planning and real-time rail traffic management (RTM), many gaps still exist in terms of scalability, real-world commitment and decentralisation (Lamorgese et al. [2018]).

In this study, we model the RTM problem with the Hunter Valley Coal Chain (HVCC) as the baseline. The HVCC is the world's largest coal export operation, located in New South Wales (NSW), Australia. More than 87% of the coal transportation in NSW is done through railways, highlighting the importance of optimisation techniques driving the decisions across the coal chain. The HVCC starts at the Port of Newcastle (PoN) and extends to distances of up to 420 kilometres to reach the 30 existing coal mines.

Figure 1 illustrates the main rail infrastructure elements present in the HVCC. The sections have parallel lines closer to the terminals (sections between A and H). Trains leave the terminals empty and travel to a load point (LP) to load with coal, and use a balloon loop to return to the mainline and travel back to a terminal, where they use one of the dump stations to unload. LPs can be located close to the double line railroads (e.g. LP1), or at single line railroads (e.g. LPs 2 and 3). In a single line railroads, trains require a passing loop (PL) to overtake or meet-pass another train. The distance between PLs should not be too large to avoid high dwell times (or waiting times). PLs are represented in sections J-K, N-O and P-K in Figure 1.

Our approach addresses the RTM problem as defined by Corman and Meng [2015]. The method assigns feasible arrival and departure dates for the trains in each section of their routes to complete their trips, including loading/unloading activities, in order to minimise dwell times. In a feasible solution, all trains complete their trips from terminal to load point and back without a deadlock situation arising. The method described in this work uses multi-agent systems (MAS) and deep reinforcement learning (DRL) to model the problem as a decentralised system with learning capabilities.

2 LITERATURE REVIEW

This work addresses a multidisciplinary problem that carries concepts related to RTM, agent technology (AT), Markov decision process (MDP) and reinforcement learning (RL). This section provides a background with the main concepts related to these topics and discusses the related papers.

2.1 Background

In AI, problems are often modelled using agents that observe their environment and act on behalf of its goals (Russell [2010]). Multi-agent systems (MAS) contain multiple agents acting simultaneously, either pursuing the same goal (cooperative MAS) or individual goals (non-cooperative MAS). According to Bazzan and Klügl [2014], the complex decisions arising in traffic and transportation problems can be addressed by MAS approaches, as those are inherently scalable and decentralised.

Reinforcement learning RL stands for a group of machine learning methods that generates knowledge by trial, error and reward evaluation. In a single-agent RL application, an agent interacts with its environment by choosing actions that might better respond to a given observation. After a few episodes exploring different action outcomes, it learns a policy that unveils the best observation-action pairs, given an initial goal (Russell [2010]). A popular RL algorithm is the Q-learning method (Watkins [1989]). It is a value-based method that learns a utility function by building a Q-table of the expected return of observed state-action pairs.

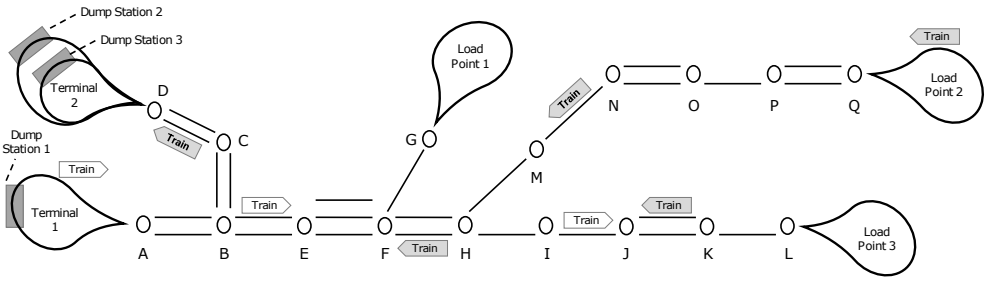

Figure 1. Example of the HVCC rail network main elements.

RL methods such as Q-learning are an efficient option for solving sequential decision problems. However, the method fails to decide upon unseen states. A powerful alternative to overcome this issue is the policy function approximation through artificial neural networks (ANN), which originates the concept of Deep Reinforcement Learning (DRL). DRL has been largely applied since the success of Google Deepmind's project AlphaGo and the advent of Deep Q-Networks (DQN) (Mnih et al. [2015]). Proximal Policy Optimisation (PPO) (Schulman et al. [2017]) is another DRL method that has been popular due to its ability to handle high-dimensional continuous and discrete state spaces. The combination of RL and deep neural networks opened a promising research branch with many significant results (Arulkumaran et al. [2017])

Multi-agent learning According to Panait and Luke [2005], multi-agent learning (MAL) is the application of machine learning techniques to problems with multiple agents. Multi-agent deep reinforcement learning (MADRL) applications arise when dealing with a MAS where the agents learn through a DRL algorithm.

The actions of all the agents compose together a joint action $a_k = [a_k^1, \cdots, a_k^N]$, $a_k \in A, a_k^n \in A^n$ and their policies $\pi^n : S \times A^n \rightarrow [0, 1]$ compose a joint policy π that influences the state transitions. Since the agent does not have full access to the complete state of the environment, the problem can be classified as a partially observed Markov decision problem (POMDP). When the agents interact with a different part of the environment, a decentralised POMDP (dec-POMDP) problem arises. This approach suits distributed problems, where agents are independent from each other and act simultaneously.

In a POMDP, the next state of the environment results from the actions of all agents. The simultaneous learning of multiple agents can change the state transition probabilities and generate a non-stationary environment, representing a convergence challenge. Some cooperative applications allow the definition of joint action learners (JAL) (Claus and Boutilier [1998]). This type of agent replaces two or more agents that have access to each other's actions. It learns from the joint action of the original agents, reducing learning concurrence.

2.2 Related work

The number of papers applying MAS and RL to the RTM and related problems has increased as the methods became more popular. For a summary of the papers that apply agent technology, we refer the reader to Bretas et al. [2021]. In this section, we focus on the papers applying RL to the RTM problem.

Parvez Farazi et al. [2021] brings a comparative review on DRL methods applied to seven different domains related to traffic and transportation, including rail transportation. Obara et al. [2019] used a similar centralised approach for the passenger train rescheduling problem. The authors applied graph theory to model temporal activities of the rail network and a DQN agent to learn how to change the graph and optimise arrival and departure dates in all nodes. The main outcome reveals that the method generates positive changes in the timetable in 57% of the instances tested. Ning et al. [2019] presents an application of DQN to minimise the average total delay for a high-speed timetable rescheduling problem. The model has a centralised dispatcher agent that receives an encoded timetable as the state and acts by defining the departure sequence of the trains in each station. The authors in Ying et al. [2021] focus on the minimisation of total passenger and operating cost. They build an optimisation framework based on an MDP for centralised decision-making on metro services scheduling and train composition. The model applies an ANN to represent the decision space and a mask scheme to facilitate the incorporation of the operational constraints. Experiments with real data from London's metro system show that the method outperforms evolutionary heuristics in solution quality, overall cost and computational efficiency.

A decentralised decision-making approach takes place in the work of Khadilkar [2019] for linear bi-directional railways with passenger trains. The author presents a Q-learning algorithm with a decentralised execution with centralised learning framework to solve the RTM problem in scheduling and rescheduling scenarios. In a MAS with a team of train agents, each train decides for itself based on local observations and using the state-action table as a common knowledge base. The local observation is an encoded vector that informs the level of occupation in the surrounding sections. The local observation includes the b sections behind the train, the train's current section and the f sections in front of it. The action is a binary variable that determines if the train moves ahead or stays in the current section. The approach suits large scale applications since the state vector encoding strategy do not vary with the number of trains and rail network sizes.

3 METHODOLOGY

The Hunter Valley Coal Chain Coordinator (HVCCC) conducts medium and long term planning analysis using an simulation model produced internally, named *enhanced Whole of Coal Chain* (eWoCC). eWoCC is a

discrete-event and agent-based simulation model that reproduces many aspects of the HVCC with a high level of detail. The model is implemented in the simulation software Anylogic[1], and was the starting point to the development of the intelligent agents that compose the MAS described here.

3.1 The intelligent agents

The MAS in this work comprises three types of agents: train agents (TA), load point (LP) agents and dump station (DS) agents. Each TA represents a train travelling in the network that has to decide whether to move to the next section $a = 1$, or keep its current location $a = 0$, based on a local observation of the current state of the environment. Table 1 details the framework to encode the environment information to support the agent's decision. The LP and DS agents monitor the occupation and the flow of trains around their balloon loops, and act by recommending a decision to a train. Since all the agents are part of a cooperative MAS, we model TA and LP/DS agents as JALs when they are physically close. Thus, when a train is less than f sections from its next destination (LP or DS), the agents perform a single joint action in the discrete action space $AS = \{0, 1\}$.

The occupation state layer holds the most important role in the observation encoding. It is relevant to all decisions taken by the trains in each part of the network. Analogously, some state layers as the Y-junction occupation and the conflict detection are more relevant when the train approaches a Y-junction and when another train is approaching, respectively.

The occupation S_r of each resource r assumes a value between 0 and R, calculated by Equation 1. It is an adaption from Khadilkar [2019] for problems that consider trains travelling in the same section and different sizes for trains and tracks. $S_r = 0$ indicates that the resource r has no associated section (e.g. next resource after the last section of the route). Q_r is a binary parameter that detects empty resources. N_r is the number of parallel tracks in the section, and τ_r represents the number of those tracks currently occupied. w indicates whether there is enough space for the current train to fit behind another train in the next section ($w = 0.5$), or if the next section is fully occupied ($w = 1$). Finally, R is a threshold for the maximum occupation value. We use $R = 3$ for all experiments – which represent low, medium and high occupation.

$$S_r = Q_r(R - \min(R - 1, \lfloor N_r - w\tau_r \rfloor)) \tag{1}$$

Reward calculation Our approach uses immediate rewards and delayed rewards, given only at the end of an episode and calculated by Equation 2 and Equation 3, respectively. Since this is a cooperative MAS, both are global rewards assigned to the whole team of agents.

$$I^t_{RW} = M^t/M_{tot} + T^t/T_{tot} \tag{2}$$

$$D_{RW} = -1 + (FCFS^{DT}_{1TS}/RL^{DT}) \tag{3}$$

The immediate rewards are given after each decision step. Its objective is to guide the agents towards a feasible solution. If all agents have an action $a = 0$ in the time step t, then $I^t_{RW} = 0$. Otherwise, it assumes the normalised number of sections travelled ($M^t/2M_{tot}$) by all trains plus the normalised number of train trips completed ($T^t/2T_{tot}$). M_{tot} and T_{tot} represent the total number of sections and trips in a given instance, respectively. M^t and T^t are the values achieved for both metrics at the end of the episode. Feasible solutions have all trains completing their trips to the LPs and back. Thus, $M^t/M_{tot} = 1$ and $T^t/T_{tot} = 1$ by the end of the episode. An episode ends in an infeasible solution when it reaches the maximum number of allowed steps. In this case, I_{RW} would be less than 2.

The delayed reward D_{RW} reflects the main objective to minimise the total dwell time of all trains. The model applies Equation 3 to calculate the D_{RW} for feasible solutions, otherwise $D_{RW} = -1$. In order to achieve a positive and representative normalisation for the dwell time, we use the dwell time obtained by a improved FCFS heuristic available in the eWoCC simulation model ($FCFS^{DT}_{1TS}$) as a reference. Hence, a reward greater than 1 indicates that the MADRL has a smaller dwell (RL^{DT}) than the $FCFS_{1TS}$ heuristic.

3.2 MADRL implementation

The MAS works on a decentralised execution with centralised learning framework, where each agent makes independent decisions based on local information, constituting a dec-POMDP. The agents learn a common single policy through PPO and the Pathmind[2] platform. Pathmind leverages the RLlib library[3] with a ded-

[1] https://www.anylogic.com
[2] https://pathmind.com/
[3] https://docs.ray.io/

Table 1. Description of the state layers that compose each observation of the train agents

State Layer	Size	Description	Examples
Occupation	$b + f$	A vector S of integer variables that uses Equation 1 to calculate the level of occupation in each resource r. The resources are given by the b sections behind the train, the section currently occupied by the train and the f sections in front of the train.	$S_r = 0$; $S_r = 3$; $S = \{1, 1, 2, 3, 2, 3\}$: example with $b = 2$, $f = 3$ and high occupation in the sections ahead.
Movement	1	Single binary variable tm that indicates if the train is currently in movement	$tm = 0$: train waiting; $tm = 1$: train currently in movement.
Load	1	Single binary variable tl that indicates if the train is currently loaded	$tl = 0$, empty train; $tl = 1$: loaded train.
Y-junction occupation	$b + f/2$	Integer vector J of size $b+f/2$ that allows values from 0 to 2. J_r measures the occupation of the junction in resource r for the b sections behind the decider train (train currently checking the state) and the $f/2$ sections in front of it. Values greater than 0 indicate a train in the first section of a Y-junctions's alternative branch coming towards the decider train.	$J_r = 0$: No trains coming; $J_r = 1$: All trains coming will reach the Y-junction later the decider train; $J_r = 2$, At least one train will reach the Y-junction before the decider train.
Alternative path density	$b + f/2$	Integer vector D of size $b + f/2$, where D_r indicates the occupation in the following $f/2$ sections of the alternative path of the Y-junction in the resource r, *after the first section*. D_r assumes a density value between 0 and 2, given by the rounded average of the occupation (calculated by Equation 1) over the sections.	$D_r = 0$: Low density of trains coming from an alternative path of a Y-junction $D_r = 1$: Medium density of trains coming from an alternative path of a Y-junction; $D_r = 2$: High density of trains coming from an alternative path of a Y-junction
Track length detection	4	It is a binary vector tld of length 4 that tells if the size of the current and the next three sections are greater than the length of the train. We chose this number because it represents the maximum distance between a passing loop and an LP in the HVCC.	$tld_r = 0$: the train is longer than section r $tld = \{1, 0, 1, 1\}$: the train does not fit in the next section
Conflict detection	3	Binary vector that indicates, for the next three sections, if there are trains coming towards the decider train.	$cfd = \{0, 0, 0\}$: no train coming in the opposite direction $cfd = \{0, 1, 0\}$: there is an incoming train in the second section ahead.
Load point agent	1	Calculated by the LP agent, it informs through a binary variable if the LP is ready to receive the train.	$lp = 0$: there is enough space to receive the train in the arrival track *or* the arrival track fits the train, but is still partially occupied by a train that is currently loading. $lp = 1$: no track to receive the train in the LP
Dump station agent	1	It is a binary variable ds that indicates the occupation of the dump station (or unloading point) and the number of trains on the way to the dump station.	$ds = 0$: there is at most one train in / approaching the dump station; $ds = 1$: there is more than one train currently in / approaching the dump station.

icated cloud-based testing environment and provides a friendly integration with Anylogic. Pathmind applies population-based training (PBT) to conduct the hyperparameter tuning, which is a method developed by Google's Deepmind to find the best hyperparameter configuration of a neural network by running several trials in parallel (Jaderberg et al. [2017]).

4 EXPERIMENTS

This section discusses the experiments using ten instances based on the HVCC railway network. These instances have between 16 and 80 trains, and different numbers of sections, LPs and terminals. The second column of Table 2 shows the details of each instance. The first goal of the test phase refers to the learning and convergence of the MADRL approach. This is analysed by observing the reward function and the number of steps taken over the training episode. The first metric should increase from small (or even negative, i .e. infeasible) values to positive values. From iteration 250 onward, Pathmind checks if the reward mean variation over the last 50 iterations does not exceed 1%. If this convergence criterion is satisfied, training is stopped. For the number of steps, values would likely start at the highest levels and decrease as the method finds feasible solutions. Figure 2 shows the analysis of both metrics for SLP-01 and MLP-06.

The experiment's configuration allows the learning process to happen until it matches the convergence criteria or at least one of the following thresholds is reached: 200,000 episodes, 12 hours of training time, or 500 iterations. While MLP03 converged since iteration 100, SLP01 takes more steps to find the best value for the

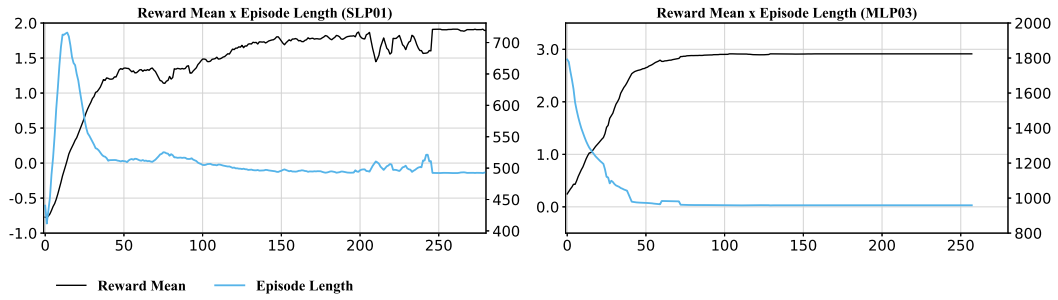

Figure 2. Analysis of the behaviour of the reward mean and the episode length mean (number of steps taken by the DRL) with the number of iterations for instances SLP01 and MLP03.

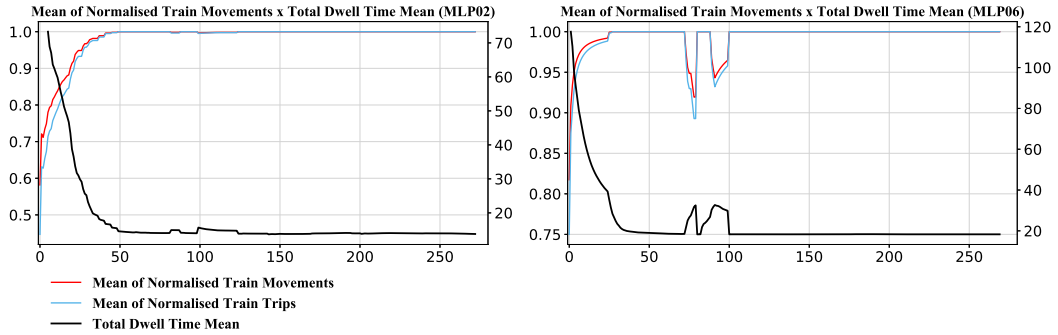

Figure 3. Analysis of the behaviour of the three metrics that compose the reward: normalised number of train movements, normalised number of train trips and normalised total dwell time for instances MLP02 and MLP06.

reward mean. The episode length curve for SLP01 suggests that it initially finds a few deadlock solutions; followed by infeasible solutions with the maximum allowed episode length, and finally, feasible solutions with fewer DRL decision steps.

We also decomposed the reward function to evaluate the behaviour of the number of train movements, the number of train trips completed and total dwell time over the training episodes. As the number of train movements and train trips reach their maximum values, feasible solutions are found, and then the learning focuses on reducing the total dwell time. Figure 2 shows a graphical analysis of these metrics for MLP02 and MLP06. While the curves for MLP02 showed a smooth behaviour, the MADRL found a few low quality solutions for MLP06, and even infeasible ones, before it converged. That can be explained by the complexity of the instance, plus the multi-agent learning with a shared policy.

Once the learning curve of the MADRL method was satisfactory, the performance evaluation was extended to all instances. In these experiments, we compare the MADRL against the $FCFS_{1TS}$ and the Genetic Algorithm (GA) presented in our previous paper (Bretas et al. [2021]). In the previous approach, the MAS uses a GA to learn a common policy for all agents. Table 2 shows the results.

Table 2. Compilation of the results for the performance experiments between $FCFS_{SZ}$, $FCFS_{1TS}$ and the GA-based MAS. Each instance was tested with 20 different seeds.

Instance	Number of Trains | Sections | LPs | Terminals	Dwell time $FCFS_{1TS}$ (hours)	Dwell time Average GA (hours)	Dwell time MADRL (hours)	GAP eFCFS / Avg GA (%)	GAP eFCFS / MADRL (%)	GAP Avg GA / MADRL (%)
SLP-01	16 | 30 | 1 | 1	9.26	5.27	4.84	43.09	47.73	8.16
SLP-02	20 | 30 | 1 | 1	13.47	8.92	8.63	33.78	35.93	3.25
MLP-01	30 | 32 | 3 | 1	25.71	9.10	7.20	64.61	72.00	20.88
MLP-02	40 | 35 | 4 | 1	27.77	25.79	13.56	7.13	51.17	47.42
MLP-03	40 | 35 | 4 | 1	29.65	14.07	10.18	52.55	65.67	27.65
MLP-04	60 | 49 | 6 | 2	32.19	22.71	15.01	29.45	53.37	33.91
MLP-05	60 | 57 | 6 | 2	35.59	19.61	15.30	44.90	57.01	21.98
MLP-06	60 | 60 | 6 | 2	22.86	19.28	18.29	15.66	19.99	5.13
MLP-07	60 | 73 | 6 | 2	26.28	27.95	21.73	-6.35	17.31	22.25
MLP-08	80 | 66 | 8 | 2	49.54	39.26	31.12	20.75	37.18	20.73

The experiments show that the MADRL method outperformed both the $FCFS_{1TS}$ and the GA in all instances. Even for the only instance where the GA fails to outperform the $FCFS_{1TS}$ (MLP-07), the MADRL achieves a total dwell time of 21.73 hours, which is 17.31% less than the heuristic. In terms of CPU time, the MADRL experiments ran for an average of 3.12 hours, and the GA experiments ran for 5.54 hours, on average. However, since they run on different platforms, there is no direct way to compare their computational times.

5 Conclusion

This paper describes the implementation of a MADRL method for the RTM problem reproducing the challenges faced by the Hunter Valley Coal Chain Coordinator (HVCCC). The MADRL outperformed the heuristic in use by HVCCC's simulation model and a high-performance GA in all instances, reaching improvements of up to 72.00% and 47.42%, respectively. The framework with the MADRL and the simulation model allows the application to a variety of situations. The approach is efficient, and at the same time reliable, since the simulation model limits eventual non-consistent solutions that represent a risk in black-box methods using ANN. The centralised learning scheme allows all agents to share a single policy, even when they are acting at the same time and in physically distant locations. It enables the exploration of a common knowledge base across local problems and accelerates the learning process. Hence, the next steps will consider using the method with larger instances and evaluating the use of transfer learning across different instances.

Acknowledgments

This work was supported by a joint HVCCC / University of Newcastle 50/50 business and industry PhD scholarship. Furthermore, the authors praise and thank the Pathmind team for their support.

References

Arulkumaran, K., Deisenroth, M.P., Brundage, M., Bharath, A.A., 2017. A brief survey of deep reinforcement learning. arXiv preprint arXiv:1708.05866 .

Bazzan, A.L.C., Klügl, F., 2014. A review on agent-based technology for traffic and transportation. The Knowledge Engineering Review 29, 375–403.

Bretas, A.M.C., Mendes, A., Jackson, M., Clement, R., Sanhueza, C., Chalup, S., 2021. A decentralised multi-agent system for rail freight traffic management. Annals of Operations Research .

Claus, C., Boutilier, C., 1998. The dynamics of reinforcement learning in cooperative multiagent systems, in: Proceedings of AAAI'98, American Association for Artificial Intelligence, 295800. pp. 746–752.

Corman, F., Meng, L., 2015. A review of online dynamic models and algorithms for railway traffic management. IEEE Transactions on Intelligent Transportation Systems 16, 1274–1284.

Jaderberg, M., Dalibard, V., Osindero, S., et al., 2017. Population based training of neural networks. arXiv preprint arXiv:1711.09846 .

Khadilkar, H., 2019. A scalable reinforcement learning algorithm for scheduling railway lines. IEEE Transactions on Intelligent Transportation Systems 20, 727–736.

Lamorgese, L., Mannino, C., Pacciarelli, D., Krasemann, J.T., 2018. Train dispatching, in: Handbook of Optimization in the Railway Industry, Springer International Publishing (Ed). pp. 265–283.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., et al., 2015. Human-level control through deep reinforcement learning. Nature 518, 529.

Ning, L., Li, Y., Zhou, M., Song, H., Dong, H., 2019. A deep reinforcement learning approach to high-speed train timetable rescheduling under disturbances, in: 2019 IEEE Intell. Transportation Systems Conf, IEEE.

Obara, M., Kashiyama, T., Sekimoto, Y., 2019. Deep reinforcement learning approach for train rescheduling utilizing graph theory, in: 2018 IEEE International Conference on Big Data, IEEE. pp. 4525–4533.

Panait, L., Luke, S., 2005. Cooperative multi-agent learning: the state of the art. Autonomous Agents and Multi-Agent Systems 11, 387–434.

Parvez Farazi, N., Zou, B., Ahamed, T., Barua, L., 2021. Deep reinforcement learning in transportation research: A review. Transportation Research Interdisciplinary Perspectives 11, 100425.

Russell, S.J., 2010. Artificial intelligence: a modern approach. N.J, United States.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O., 2017. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 .

Watkins, C.J.C.H., 1989. Learning from delayed rewards. Thesis.

Ying, C.S., Chow, A.H.F., Wang, Y.H., Chin, K.S., 2021. Adaptive metro service schedule and train composition with a proximal policy optimization approach based on deep reinforcement learning. IEEE Transactions on Intelligent Transportation Systems , 1–12.