23rd International Congress on Modelling and Simulation, Canberra, ACT, Australia, 1 to 6 December 2019
mssanz.org.au/modsim2019

Modelling railway traffic management through multi-agent systems and reinforcement learning

A. 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: Allan.MessederCaldasBretas@uon.edu.au

Abstract: Australia plays a significant role in the world's coal supply. The world's largest coal operation is located in the state of New South Wales, where more than 87% of the transport is done through railways. One of the strategies to increase throughput is to use sophisticated computational techniques for train scheduling optimisation and this study applies artificial intelligence techniques to the railway traffic management problem in the context of the Hunter Valley Coal Chain Coordinator (HVCCC). This problem has been studied mostly through centralised decision-making models, applying linear integer programming, heuristics and hybrid approaches. However, recent publications indicate a lack of practical applications (Lamorgese et al. [2018]), pointing out that low computational requirements, scalability, decentralisation and real-world commitment are key features required for deployment-ready applications.

Towards that, one option is to model system actors (trains, stations, dispatchers, operators, and more) as autonomous intelligent agents that interact, learn and act independently to reach their own objectives – thus constituting a multi-agent system (MAS). This way, the railway traffic system will be capable of making rapid, distributed decisions. Few studies have modelled railway traffic management as a MAS and they lack many of the important decisions, constraints and actors present in real-world scenarios (Lamorgese et al. [2018]).

This paper describes a discrete event simulation model of a small, closed railway, and implements a decentralised and heterogeneous MAS for train dispatching. The model was built using the Arena Simulation Software[1]. It includes several train agents and a single dispatcher agent that applies different decision methods (First-in-First-Out rule, random walk and reinforcement learning) to regulate railway traffic decisions.

The paper describes how experiments were designed, computational results, calibration of the reinforcement learning (RL) algorithm, performance tests for various levels of congestion, and tests for transfer learning between different instance configurations. The RL performance outperforms the FIFO standard dispatch rule by 10.3% for the high-congestion network configuration. In addition, transfer learning tests illustrate the generalisation capability of the RL method, where knowledge gained during the training using an instance reduces the time required for the training of additional instances. This represents an initial step towards the application of the approach in the HVCCC network traffic management problem.

Keywords: *Railway traffic management, multi-agent systems, reinforcement learning, transfer learning, discrete event simulation.*

[1] https://www.arenasimulation.com/

1 INTRODUCTION

Coal exports are the second largest source of external revenue for Australia. The country is the world's largest coal exporter – with 36.6% of the volume (IEA [2018]). In the state of New South Wales, more than 87% of the coal is transported through railways. The Hunter Valley Coal Chain (HVCC) is the largest coal export operation in the world. The rail network has haulage distances of up to 380 kilometres and reaches 31 loading points. The port is composed of three terminals, able to load more than 1400 vessels per year (TNSW [2018]). The planning and scheduling activities are conducted by the Hunter Valley Coal Chain Coordinator (HVCCC). The start point for the planning process is the vessels line-up sequence. Trains leave the terminals empty, go to one of the load points, and bring the required type of coal for each vessel before it berths, so loading operations can start without delay. Besides the operational and safety constraints, dispatchers must also deal with capacity and time constraints, while satisfying any contractual constraints imposed by the stakeholders.

2 BACKGROUND

2.1 Railway Traffic Management

The general railway traffic management problem is very complex and a core activity is the definition of an optimal and robust timetable. A timetable is a conflict-free schedule of trains, with specific arrival and departure times at relevant checkpoints (e.g. stations, terminals and track sections). Among the measures that are normally optimised, there is total delay, passenger satisfaction, transported weight (or throughput) and several others. Moreover, the timetable should be robust enough to handle unexpected events with minimal perturbation to the rest of the schedule. This paper addresses the problem of assigning time slots to trains at stations and main checkpoints, delivering a timetable that assures capacity feasibility.

The term railway traffic management problem (RTMP) (Corman and Meng [2015]) will be adopted here to describe the problem under consideration. It has a broad scope and groups the main features of interest. Since most variations of the RTMP are NP-hard (Lamorgese et al. [2018]), commercial solvers cannot deal with real-world instances. In that case, metaheuristics, decomposition methods, simulation-guided optimisation and hybrid approaches are frequently applied (Mendes et al. [2017]; Ho and Yeung [2001]; Corman and Meng [2015]). Although some studies achieved significant results, they can't be used directly in a decision support system (Lamorgese et al. [2018]). Most papers propose centralised strategies, dedicated to finding a global optimum for the whole system, and address simplified versions of the problem without many of the constraints faced by train drivers, dispatchers, controllers and operators (Corman and D'Ariano [2008]).

2.2 Agent Technology

Humans act by following a sequence of processes: perceiving the environment, understanding the information; predicting what can happen next; and finally, taking an action. The goal in Artificial Intelligence (AI) is to reproduce this process, not just by imitating human behaviour, but trying to achieve the ideal performance, being rational (Corman and D'Ariano [2008]). A multi-agent system (MAS) is composed of intelligent autonomous agents that coexist and interact, sharing environment and information; and the action of one agent could affect others. MAS has been considered a suitable option for modelling complex systems with decentralised decisions, such as those present in traffic and transportation management (Bazzan and Klügl [2014]). Recent publications indicate that low computational power requirements, scalability, decentralisation and real-world commitment are key features to address the RTMP successfully (Lamorgese et al. [2018]; Corman and Meng [2015]). Those features are compatible with MAS approaches, where the related actors (trains, stations, dispatchers, operators, etc.) are modelled as autonomous intelligent agents (Corman and D'Ariano [2008]; Narayanaswami and Rangaraj [2015]).

2.3 Reinforcement Learning

A learning agent can use a wide variety of algorithms. Reinforcement Learning (RL) methods promote learning by experimentation and reward evaluation, i.e. positive and negative rewards are used to guide future actions. When applying single-agent reinforcement learning (SARL), the environment is usually modelled as a Markov Decision Process (MDP).

Definition 1. A finite Markov decision process is a tuple (S, A, f, R), where S is a finite set of environment states, A is a finite set of agent actions, $f : S \times A \times S \to [0, 1]$ is the state transition probability function, and $R : S \times A \times S \to \mathbb{R}$ is the reward function (Russell [2010]).

In a MDP, the environment is composed of finite discrete time steps and ruled by a stochastic state transition function f. At each time step k, the RL agent observes the state $s_k \in S$ and chooses the action $a_k \in A$ based on a policy π. The environment then changes to a state $s_{k+1} \in S$, determined by $f(s_k, a_k, s_{k+1})$. A reward function R is used to calculate the reward r_{k+1} for the choice of action a_k in state s_k: $r_{k+1} = R(s_k, a_k, s_{k+1})$. In deterministic models, the transition function $\bar{f} : S \times A \to S$ results always in the same state s_{k+1} and the same reward r_{k+1} from a given pair (s_k, a_k). The objective of the RL agent is the maximisation of its performance in all k time steps of a given episode (Russell [2010]). Figure 1 illustrates the relationship between the SRLA and the environment.

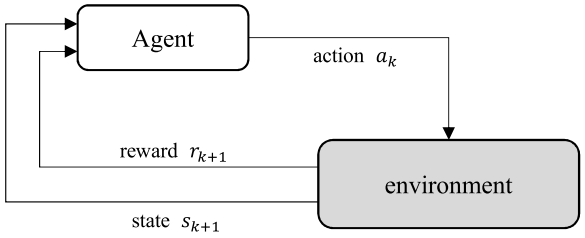

Figure 1: Illustration of the dynamics of a single reinforcement learning agent.

Q-learning Q-learning (Watkins [1989]) is a model-free RL algorithm that learns a utility function for the state-action pairs, and decides the next action based on its estimated value. In Q-learning, the action a_k should be taken considering only the state s_k. An action-value function (Q-function) $Q^\pi : S \times A \to \mathbb{R}$ can be used to calculate the expected return and determine how good are the episodes in which (s_k, a_k) are involved: $Q^\pi(s, a) = E\{\sum_{j=0}^{\infty} \gamma^j \cdot r_{k+j+1} | s_k = s, a_k = a, \pi\}$. The discount factor γ is used to balance the contribution of future rewards to the Q-function due to uncertainty. Past decisions (subsequent states (s_k), actions (a_k) and rewards (r_{k+1})) are used to update Q in an iterative approximation procedure: $Q_{k+1}(s_k, a_k) = Q_k(s_k, a_k) + \alpha_k \cdot [r_{k+1} + \gamma \cdot \max_{a' \in A} Q_k(s_{k+1}, a') - Q_k(s_k, a_k)]$

The learning rate $\alpha_k \in [0, 1]$ dictates the level of adjustment of the current $Q_k(s_k, a_k)$ and usually decreases with time. The sequence Q_k converges to optimal (Q^*) by combining exploration and exploitation episodes, as in the ϵ-greedy policy (Russell [2010]). This policy chooses the state-action pair with the highest Q-value with a (1-ϵ) probability. Otherwise, the action is determined randomly. Since ϵ starts with value 1 and decreases with the number of episodes, the training starts with a high level of exploration (and no exploitation) and finishes with no exploration and high level of exploitation.

3 Related Work

The use of autonomous intelligent agents to represent elements in traffic and transportation problems has gained momentum in the last two decades (Bazzan and Klügl [2014]; Chen and Cheng [2010]; Davidsson et al. [2005]). However, there is still a lack of deployed real-world applications. The study by Corman and D'Ariano [2008] analysed the few applications of agent technology to railway traffic management problems. Some of these studies apply methods such as auction, negotiation protocols and learning to solve the RTMP in cooperative and non-cooperative environments and are discussed next.

Agent Technology Related Works Törnquist and Davidsson [2002] propose a conceptual model with dispatcher and transport operator agents with the objective to maximise punctuality after a disturbance. In D'Ariano and Hemelrijk [2006], the authors suggest a novel MAS architecture to deal with train dispatching. In this application, geographically assigned dispatcher agents proactively try to find conflicts in a future time horizon and generate a set of feasible solutions to be used by the train agents for conflict resolution. Proença and Oliveira [2004] present a conflict detection approach that minimises train conflicts. Supervisor, train, station and learning agents are present in this MAS. Results show improvement in performance through the use of rules learned during past conflict scenarios to predict and prevent new conflicts. Narayanaswami and Rangaraj [2015] introduce a similar architecture with a new auction-based procedure. They solve the dynamic and real-time reschedule problem in a linear, single-track railway, with freight and passenger trains sharing the same resources. Finally, the recent work of Savelsbergh and Talebian [2018] presents an alternative approach that uses non-cooperative game theory for rail track allocation to rail operators. The players (operators) propose train configurations that result in different access charge discounts in a cost allocation game.

Agent Technology with Reinforcement Learning There are few examples of applications of Reinforcement Learning (RL) in railway traffic management problems. Khadilkar [2019] applied a table-based Q-learning algorithm, combined with a discrete event simulation model, to generate a feasible timetable for passenger trains, allowing scheduling and rescheduling applications. In Šemrov et al. [2016], an agent-based

model using RL is applied to the conflict detection and resolution problem. The method implements a dispatcher agent using Q-learning and a simulation model for timetable rescheduling for passenger and freight trains. The objective was to minimise total delay, and tests conducted in instances with 23 block sections, 14 stations and 26 trains revealed deadlock-free solutions.

4 A MULTI-AGENT SYSTEM FOR A SMALL CLOSED RAILWAY NETWORK

This work addresses the real-time railway traffic management of trains travelling in a small closed network, with a tree topology (see Figure 2). Trains are scheduled from their initial location to their final destination, and the objective function is to minimise the total dwell time. In our model, trains travel at a maximum speed when moving along a section, and only one train can be in a track section at any given time. Passing loops allow overtaking, and balloon loops provide access to load/unload points. Trains are only allowed to stop at passing loops. In that case, a train can only leave a passing loop if the resources required to get to the next passing loop are available. Every time a train stops, it must wait a minimum penalty time for recharging the breaks. Also, even if a train does not stop at a passing loop, it must decelerate slightly while crossing it, due to safety constraints. Finally, the unload point and the load points can only receive one train at a time.

Agent Architecture The MAS in this study is heterogeneous, with a dispatcher agent and several train agents. The dispatcher has a global objective and deliberates permission concessions for trains travelling in the network. Meanwhile, train agents display a competitive behaviour. Each train aims to minimise its dwell time by travelling without interruptions. Trains only have autonomy for the allocation of tracks in load/unload points and performing loading/unloading activities. The allocation of the remaining tracks by train agents only occurs if (1) the track is available and (2) if the dispatcher agent allows it. If the dispatcher does not concede permission to proceed, the train will stop at the passing loop. The dispatcher is activated every time a train releases a resource/track, and the system waits until it communicates its action. The communication happens through a Blackboard system, which is shared and updated by the agents. When the dispatcher is activated, it checks the state of the system in the blackboard, applies one of the available decision methods and communicates the action to all train agents. The action can be interpreted as a group of permissions for each train to go ahead or stop at the strategic checkpoints of the network. Both the state and the action are related to the entire system and not to a single train.

State and Action Policy The train agent environment concerns (1) the section track that the train is currently placed, (2) the allocation status of the following tracks in its route and (3) the dispatcher permissions written in the Blackboard. After the agent observes the environment, it tries to allocate the tracks (as discussed in the previous paragraph) and then moves in case of success. The action taken by the dispatcher agent depends on the state vector and the decision method being used. The state vector indicates where the trains are currently located in the system, their travel direction and final destination. The action vector allows the movement of trains and is composed of binary variables that map to the locations where trains need permission to move forward (i.e. the passing loops). For example, a value 1 in the first element of the vector indicates that permission to go ahead is given to the train in passing loop one, travelling towards the loading points. An action vector that only contains zeros indicates that no permissions to move forward were given. The number of possible actions is 2^p (where p is the number of locations where trains can stop). The decision method policy determines the action selection. Three decision methods are available: First-in-First-Out (FIFO), Random Walk (RW) and Reinforcement Learning (RL) by Q-learning procedure. If the FIFO method is chosen for the episode, all actions are guided by it, and the first train that requests a resource will have it. For the RW policy,

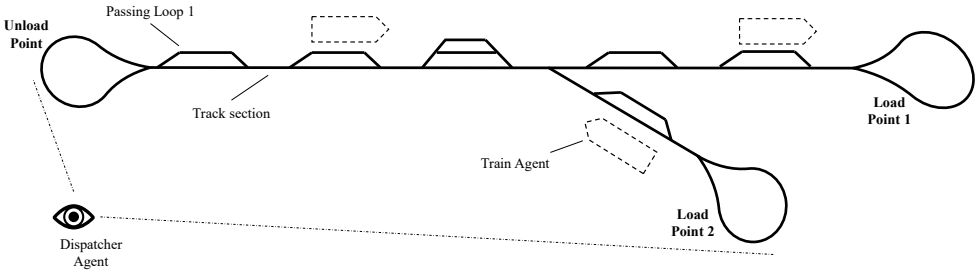

Figure 2: Railway network implemented in this study. Train agents have decentralised local actions and the dispatcher agent has a global vision of the system.

every train waiting for permission to go ahead in one of passing loops will have it granted with a probability of 50%; and finally, the RL method uses the ϵ-greedy policy to determine its next action.

Utility Function The total dwell time (DT) is the utility function used to guide the RL. It is calculated as the summation of the time τ_p^t spent by each train t in each passing loop p. Thus, given the set of trains T and the set of passing loops PL, DT is given by: $DT = \sum_{t \in T} \sum_{p \in PL} \tau_p^t$. When RL is applied, DT is used to evaluate the results of the iteration and calculate the reward obtained by the state-action pairs used. If the observed dwell time does not exceed the historical minimum dwell time DT' in more than ρ units, a +1 reward is added to all (S, a) pairs used in the iteration. Negative rewards are not used in our implementation.

At the end of each episode, the dispatcher agent updates the value DT', the state-action pairs (S, a), their Q-value $q(S, a)$, the number of times they were observed $\beta_{S,a}$ and their accumulated reward $\delta_{S,a}$. The last two parameters are used to calculate the proportion of success $\sigma_{S,a} = \beta_{S,a}/\delta_{S,a}$. The Q-value $q(S, a)$ of a state-action pair is a value between 0 and 1, calculated as the proportion of success $\sigma_{S,a}$ of the pair divided by the average $\sigma_{S,a}$ of its N neighbour state-action pairs. A parameter ω is used to balance the two factors – in this study, $\omega = 0.5$. Equation 1 shows how the Q-value $q(S, a)$ is calculated.

$$q(S, a) = \omega \cdot \alpha(S, a) + (1 - \omega) \sum_{n=1}^{N} \frac{\alpha(S_n, a_n)}{N} \tag{1}$$

5 EXPERIMENTS

The MAS was modelled using a discrete event simulation engine implemented in Arena[2]. There are three test phases: calibration, performance and transfer learning. The calibration experiments used 5 instances of 7 trains each to determine the best value for ρ (tested between 0.01 and 0.30) and the best lengths of the exploration and exploitation phases (tested between 50 and 400). The value of 7 trains was chosen because it represents an average level of congestion for the network. The best combination of parameters were $\rho = 0.03$, 300 exploration episodes and 200 exploitation episodes.

5.1 Performance Evaluation

The experiments compare RL against RW and FIFO for the minimisation of total dwell time. RW uses 1000 episodes with 100% of random actions by the dispatcher agent. FIFO uses only one episode (since it is deterministic) and RL uses the parameter configuration described in the previous section. Initially, five instances with 7 trains were tested with all methods, with 30 different seeds for the random number generator. On average, RL achieved results 7% better than FIFO and 31% better than RW. Given the very poor result of RW (as expected), we excluded that method from the remaining tests.

Then, we proceeded to test the approach for different levels of network congestion, using between 5 and 10 trains. Table 1 shows the results for total dwell time. RL performs better than FIFO in 88.2% of the tests. On average, RL produces better results than FIFO for all levels of congestion.

5.2 Transfer Learning

In this section, we assess the suitability of using transfer learning to reduce training effort. The first experiment used instance 7TC1 as the target task and instance 7TC2 as source task, both with 7 trains. Table 2 shows the

[2]https://www.arenasimulation.com/

Table 1: Total dwell time (in hours) for RL and FIFO, for instances representing different levels of congestion – with 5 to 10 trains.

	Reinforcement Learning					FIFO
Instance size	1^{st} quartile	Median	3^{rd} quartile	Average	Standard deviation	
5 trains	4.48	4.7	5.05	4.79	0.37	5.54
6 trains	8.92	9.11	9.56	9.18	0.38	9.44
7 trains	13.08	13.29	13.67	13.3	0.48	14.22
8 trains	14.32	14.58	15.43	14.79	0.74	16.41
9 trains	17.78	18.05	18.36	18.02	0.36	20.16
10 trains	27.23	28.34	30.9	28.98	2.17	31.59

Table 2: Results comparing transfer learning strategies (or training schemes) for the total dwell time. For the source task, the experiments can have 300 (only exploration phase) or 500 episodes (exploration and exploitation). For the target task, the experiments include both phases (500 episodes) or no additional training.

Training scheme	Number of episodes for source task (7TC2) (exploration / exploitation)	Number of episodes for target task (7TC1) (exploration / exploitation)	1^{st} quartile	Median	3^{rd} quartile	Average	Standard deviation
1	Standard (300/200)	Standard (300/200)	13.53	13.12	12.83	13.10	0.57
2	Exploration only (300/0)	Standard (300/200)	13.66	13.36	12.95	13.30	0.48
3	Standard (300/200)	No training (0/0)	15.38	14.58	14.10	14.68	0.86
4	Exploration only (300/0)	No training (0/0)	14.41	14.00	13.89	14.69	1.46

results for 4 different training schemes. The performance is measured by the total dwell time, and the numbers show the median values obtained after 30 runs with different seeds. The best performances come from training schemes 1 and 2. Training scheme 1 is equivalent to running RL normally on 7TC2 (as the source task) and using the resulting Q-table in the beginning of the RL on 7TC1 (as the target task). Training scheme 2 runs only the exploration phase of the RL in the source task, and then exploration plus exploitation in the target task. Training scheme 3 runs exploration plus exploitation in the source task, and then evaluates in the target task (i.e. train the RL strategy on one instance, but using the results in a different one). Finally, training scheme 4 runs only exploration in the source task, and then evaluates in the target task. For reference, the dwell times for random walk (RW) is 18.76, for FIFO is 14.22 and for RL without transfer learning is 13.29.

From the results, we can see that without further training in the target task (training schemes 3 and 4), the total dwell time degrades, i.e. generality of the RL is limited, but still remains similar to FIFO. More tests will be required to explore the full potential of this approach, but it might be possible to reduce the number of training episodes, and/or use less instances to train a robust RL approach for the problem.

In order to reduce the training effort, we tested a strategy that includes a small exploration phase in the target task. Thus, the last experiments used a training scheme with 100 exploration episodes and 100 exploitation episodes. Four different instances were used as source task, and all of them presented similar behaviour. For illustrative purposes, Figure 3 shows the learning curve for the instance 7TC1 with and without transfer from instance 7TC3. *RL 7TC1 (SELF)* is the result for instance 7TC1 with 100 exploration and 100 exploitation episodes. *RL 7TC1 (TR-C3)* is similar, but with the Q-table learned from 7TC3. Finally, *RL 7TC1 (TR-EXC3)* starts with the Q-table learned from 7TC3 with the 300-episode exploration phase only. The tests confirmed that the most significant asymptotic improvement is achieved when transfer learning comes from the complete RL run in the source task, and is complemented by 100 exploration and 100 exploitation episodes in the target instance – RL 7TC1 (TR-C3). Performance is relative to training scheme 1 (i.e. 500 episodes used for the training of the target instance, instead of only 200).

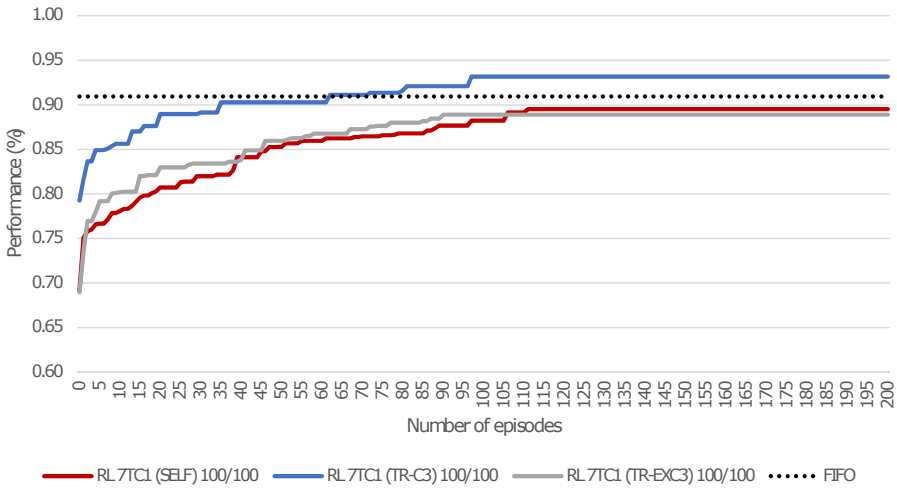

Figure 3: Learning curves for instance 7TC1 using 100 exploration and 100 exploitation episodes. The scenario with transfer learning from a complete run of 7TC3 shows the best learning performance.

6 Conclusion

The railway traffic management approach presented in this work uses a heterogeneous MAS with autonomous agents that represent the main parts of the daily traffic control process at HVCCC. Reinforcement Learning produced better results than the FIFO strategy for a range of congestion scenarios. These results encourage the application of RL in larger, real-world instances to measure its potential benefits. The RL was better than FIFO in all instances, with 5 to 10 trains.

The transfer learning experiments show a worsening in total dwell time when the patterns observed in the chosen state representation is transferred directly to a different instance, without further training. However, using transfer learning with limited additional learning for the target instance shows promising results, with a noticeable improvement in total dwell time, compared to running reinforcement learning from scratch.

Although the MAS approach achieved good results, the centralised dispatcher decision makes the size of the state and action spaces increase proportionally with the size of the network, which makes it challenging to model large size networks. It also limits transfer learning to instances with the same number of trains, since the vectors are specific to the number of trains in the system. Therefore, the next step of the research will focus on a more autonomous train agent, aiming at a totally decentralised approach.

References

Bazzan, A. L. C. and F. Klügl (2014). A review on agent-based technology for traffic and transportation. *The Knowledge Engineering Review 29*(3), 375–403.

Chen, B. and H. H. Cheng (2010). A review of the applications of agent technology in traffic and transportation systems. *IEEE Transactions on Intelligent Transportation Systems 11*(2), 485–497.

Corman, F. and A. D'Ariano (2008). Existing and future approaches on railway traffic control from an agent-based perspective. pp. 1–15. Proceedings of 10th TRAIL Congress, Rotterdam, The Netherlands.

Corman, F. and L. Meng (2015). A review of online dynamic models and algorithms for railway traffic management. *IEEE Transactions on Intelligent Transportation Systems 16*(3), 1274–1284.

D'Ariano, A. and R. Hemelrijk (2006). Designing a multi-agent system for cooperative train dispatching. *IFAC Proceedings Volumes 39*(3), 369–374.

Davidsson, P., L. Henesey, L. Ramstedt, J. Törnquist, and F. Wernstedt (2005). An analysis of agent-based approaches to transport logistics. *Transportation Research Part C: Emerging Technologies 13*(4), 255–271.

Ho, T. K. and T. H. Yeung (2001). Railway junction traffic control by heuristic methods. *IEE Proceedings - Electric Power Applications 148*(1), 77–84.

IEA (2018). *World energy outlook 2018*. International Energy Agency, Australia.

Khadilkar, H. (2019). A scalable reinforcement learning algorithm for scheduling railway lines. *IEEE Transactions on Intelligent Transportation Systems 20*(2), 727–736.

Lamorgese, L., C. Mannino, D. Pacciarelli, and J. T. Krasemann (2018). *Train dispatching*, Book section 12, pp. 265–283. Cham: Springer International Publishing.

Mendes, A., M. Jackson, M. Paula, and O. Rojas (2017). Iterative train scheduling in networks with tree topologies: a case study for the hunter valley coal chain. pp. 1337–1343. 22nd International Congress on Modelling and Simulation (MODSIM2017), Hobart, Australia.

Narayanaswami, S. and N. Rangaraj (2015). A mas architecture for dynamic, realtime rescheduling and learning applied to railway transportation. *Expert Systems with Applications 42*(5), 2638–2656.

Proença, H. and E. Oliveira (2004). Marcs: Multi-agent railway control system. In C. Lemaître, C. A. Reyes, and J. A. González (Eds.), *Advances in Artificial Intelligence – IBERAMIA 2004*, pp. 12–21. Springer.

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

Savelsbergh, M. and M. Talebian (2018). Cost allocation under competition: a new rail access charging policy. *EURO Journal on Transportation and Logistics*.

TNSW (2018). *NSW freight and ports plan 2018-2023*. Transport for NSW, Australia.

Törnquist, J. and P. Davidsson (2002). A multi-agent system approach to train delay handling. In *European Conference on Artificial Intelligence, Amsterdam, The Netherlands*.

Watkins, C. J. C. H. (1989). *Learning from delayed rewards*. Phd thesis, King's College, Oxford, UK.

Šemrov, D., R. Marsetič, M. Žura, L. Todorovski, and A. Srdic (2016). Reinforcement learning approach for train rescheduling on a single-track railway. *Transportation Research: Part B 86*, 250–267.