applied sciences

MDPI

Article

Train Scheduling with Deep Q-Network: A Feasibility Test

Intaek Gong [1], Sukmun Oh [2] and Yunhong Min [1,*]

[1] Graduate School of Logistics, Incheon National University, Incheon 22012, Korea; itgong87@inu.ac.kr
[2] Transport System Research Team, Korea Railroad Research Institute, Gyeonggi 16105, Korea; smoh@krri.re.kr
* Correspondence: yunhong.min@inu.ac.kr; Tel.: +82-32-835-8185

Received: 1 November 2020; Accepted: 23 November 2020; Published: 25 November 2020

Abstract: We consider a train scheduling problem in which both local and express trains are to be scheduled. In this type of train scheduling problem, the key decision is determining the overtaking stations at which express trains overtake their preceding local trains. This problem has been successfully modeled via mixed integer programming (MIP) models. One of the obvious limitation of MIP-based approaches is the lack of freedom to the choices objective and constraint functions. In this paper, as an alternative, we propose an approach based on reinforcement learning. We first decompose the problem into subproblems in which a single express train and its preceding local trains are considered. We, then, formulate the subproblem as a Markov decision process (MDP). Instead of solving each instance of MDP, we train a deep neural network, called deep Q-network (DQN), which approximates Q-value function of any instances of MDP. The learned DQN can be used to make decision by choosing the action which corresponds to the maximum Q-value. The advantage of the proposed method is the ability to incorporate any complex objective and/or constraint functions. We demonstrate the performance of the proposed method by numerical experiments.

Keywords: train scheduling; reinforcement learning; deep Q-network

1. Introduction

Urban metro has played the role of a sustainable transportation mode in the sense that it provides a reliable and mass transportation service to commuters and has a relatively higher energy efficiency than other transportation modes. To achieve the desired level of customer service as well as minimizing operating costs, various aspects of operating issues need to be considered in the planning process of urban metro. As many different decisions and their levels are involved in the planning process, instead of considering these decisions simultaneously, the sequential decision making strategies are typically adopted. These strategies commonly consists of the line planning, timetabling, rolling stock circulation, and crew scheduling stages [1,2].

In the line planning stage, we want to find the operating frequencies of trains and a set of their stopping stations to satisfy travel demand in a given railway network. The decisions for the timetabling stage include the arrival and departure times of trains at each station it stops. Rolling stock circulation includes the decisions such as train composition (locomotives and carriages), vehicle circulation, maintenance policies, and so on. In the crew scheduling stage, the crew duties are allocated to train services while satisfying operational requirements. Note that decisions for a particular stage become a part of the input for the next stages.

In this paper, we focus on train timetabling for metro assuming that the decisions in the stage of line planning are given in advance. More specifically, we are interested in timetabling for Seoul metro's Line 9 in South Korea. A distinguished feature of Line 9 is that there exist two types of trains: *local trains* and *express trains*. To provide a regular service for passengers, local trains stop every station in the line. When the trains stop at all stations, however, the travel times of passengers, in particular, of long-distance commuters, increase. Therefore, another type of train, express trains, which stop only at a predefined set of stations, is introduced. When these two types of trains are simultaneously run in the line, it is important to allow express trains to pass local trains. However, Line 9 consists of a single track for each direction between two consecutive stations and hence additional infrastructures at some stations are required through which overtaking between trains is possible. In Line 9, there is a set of stations, called *overtaking stations*, at which express trains are allowed to pass local trains through another track.

In this paper, we assume that the departure sequence of trains at their common initial station is given as input. To solve the timetabling problem, we decompose the problem into several smaller problems involving smaller number of trains and by combining the results of subproblems we can obtain the timetable for all trains. The decomposition is based on the the departure sequence of trains at the initial station and each subproblem considers a single express train and its preceding local trains. Assuming that overtaking between the same type of trains is not allowed, the result of one subproblem is independent of those of other subproblems. Thus, our problem is reduced to the timetabling problem involving local trains and a single express train in which the express train is the last train departing at the initial station.

One of the promising approaches to solve the reduced timetabling problem is to construct a mixed integer programming model for the problem and solve it using commercial solvers such as Gurobi, Cplex, and so on. In this paper, instead, we consider a reinforcement learning approach in which an optimal policy is learned. Note that due to small size of the reduced problem, MIP solvers can give an optima solutions in reasonable times. However, MIP solvers rely on the mathematical model for the problem and it is not easy to incorporate complex objectives and/or constraints which cannot be represented as linear functions on decision variables into the model. Reinforcement learning approaches, on the other hand, can incorporate complex objectives and/or constraints in the form of reward functions, and therefore have more applicability for diverse situations (e.g., considering energy consumption or passenger crowdedness).

This paper is organized as follows. In Section 2, we introduce the relevant literature including the recent applications of reinforcement learning to train scheduling. A formal definition of our problem and its mathematical model are presented in Section 3. The reinforcement learning framework for our problem is introduced in Section 4, and its performance is demonstrated in Section 5. Finally, Section 6 presents the concluding remarks and possible directions for future research.

2. Literature Review

Train scheduling in metro systems has been studied for a long time. These studies can be broadly categorized into macroscopic and microscopic levels. The former considers the arrival and departure times of trains at stations ignoring some detailed information such as signaling systems and routing within stations [3]. On the contrary, at the microscopic level, more detailed information such as routing in a station is taken into account [4]. Another categorization can be made with respect to the variables representing the times (arrival/departure times of trains). In the discrete-time train scheduling, times are discretized and train schedules are defined on the time-space network where nodes represent the locations of a train at specified times [5,6]. In the continuous-time train scheduling, however, continuous variables are used to represent the arrival and departure times of trains [7,8].

Due to the intractability of the problem, both the exact algorithms and heuristics have been proposed to solve the train scheduling problem. Many of these approaches rely on mixed integer linear/nonlinear

programming formulations. Some of them are solved via efficient enumeration schemes such as branch-and-bound and branch-and-cut algorithms [8,9]. Various metaheuristics have been also proposed. For example, Dundar and Sahin [10] and Niu and Zhou [11] proposed a genetic algorithm to solve the train scheduling problem. In particular, Dundar and Sahin [10] combine an artificial neural network (ANN) into the genetic algorithm so that ANN is trained to mimic the decision behavior of train dispatchers.

As a relatively new approach, approximate dynamic programming (ADP), or reinforcement learning (RL), has also been used to solve train scheduling problems. This approach relies on formulating the problem as Markov decision processes (MDPs). To obtain an MDP formulation, it is necessary to recast the problem as a multi-stage decision process. There are two types of train scheduling problems to which ADP or RL approach has been applied: timetabling and rescheduling problem. Essentially, these two types of problems want to decide the optimal values of the same decision variables such as arrival/departure times of trains at stations. However, two problems lie in the different stages in the planning process. Timetabling is solved in the basic planning process introduced in Section 1. On the other hand, rescheduling problem arises when the planned schedule resulted from solving timetabling problem is infeasible due to unplanned train delays or infrastructure failures. Thus, both problems can have different objectives and requirement for solution approaches.

Khadilkar [12] and Liu, Li, Yang, and Yin [13] consider the timetabling problems at a macroscopic level. Khadilkar [12] considers the timetabling problem for bidirectional railway lines and therefore track allocations need to be also considered as well as arrival/departure times for trains at stations. The goal is to minimize the total priority-weighted delay. Liu et al. [13] consider the timetabling problem for energy-efficient management of subway trains. It considers train headway for safety, train passenger loads, and the energy consumption along the subway line simultaneously.

In the rescheduling problem, both macroscopic [14,15] and microscopic [16,17] approaches exist. Jiang, Gu, Fan, Liu, and Zhu [14] and Yin, Tang, Yang, Gao, and Ran [15] want to minimize the inconvenience of passengers due to train delays, measured by waiting times of passengers at stations. In the case of Yin et al. [15], minimizing the energy consumption is additionally considered as well as the waiting times of passengers. On the other hand, Ghasempour and Heydecker [16] and Semrov, Marsetic, Zura, Todorovski, and Srdic [17] try to minimize the train delays which are determined by the difference between the planned arrival/departure times of trains and the rescheduled arrival/departure times.

In the scheduling problem considered in this paper, we are only interested in the arrival and departure times of trains, and therefore we consider a train scheduling problem of a macroscopic level. The objective is to find a schedule which is as close as possible to some reference schedule. This objective is similar to the one used in Ghasempour and Heydecker [16] and Semrov et al. [17], but, as explained later, the reference schedule is not a feasible train schedule. Our approach is based on MDP formulation which will be solved via deep Q-learning technique (DQN [18]) similar to Jiang et al. [14]. As explained in Section 4.1, however, our MDP formulation is different to those proposed in the previous literature. More precisely, our MDP formulation has different sets of states and actions. In particular, in our MDP formulation the schedule of only one train is directly handled through the choice of actions and the schedules of remaining trains are indirectly changed through the transition function. This contrasts with the previous works in which the schedules of all trains are sequentially handled by the choice of actions. Another distinction of our formulation is the use of linear programming (LP) model for the transition function. In our formulation, the choice of action corresponds to the determination of integer variables in the mixed integer programming (MIP) formulation of the problem. Thus, the resulting change of schedules can be obtained by solving LP model.

3. Problem Description

We consider a train scheduling problem on single-track unidirectional railway. Let $\mathcal{S} = \{1, \ldots, S\}$ be the set of stations, and all trains are assumed to begin their trips from station 1 and end at station S. There are two types of trains—local trains and express trains—denoted by $\mathcal{T}_l$ and $\mathcal{T}_e$, respectively. The difference in type is due to the stopping patterns of trains. Local trains stop at every station, while express trains stop only at the predetermined set of stations, denoted by $\bar{\mathcal{S}} \subseteq \mathcal{S}$. We assume that all express trains have the same set of stopping stations. If there are only local trains, no overtaking is occurred. If not, an express trains can overtake its preceding normal trains at a predefined set of stations, denoted by $\mathcal{S}_o$, in which a sidetrack is prepared.

We consider the following operational environment.

(E1) There exists a safety headway, say h. Therefore, a train arriving (departing) at a station can arrive at (depart from) the station h time units after its immediate predecessor.
(E2) For two consecutive stations s and s', where $s' = s + 1$, there are minimum transit times between s and s' for local and express trains, denoted by $l^l_{ss'}$ and $l^e_{ss'}$, respectively.
(E3) If a train stops at station, it should stay at least d time units.
(E4) Overtaking only occurs between local and express trains. Any overtaking between the same type of trains is not allowed.

We assume that the departing sequence π of trains at station 1 is given. Thus, $\pi(g) \in \mathcal{T} := \mathcal{T}_l \cup \mathcal{T}_e$ is the train which departs the initial station gth for $g = 1, 2, \ldots, |\mathcal{T}|$. Conversely, $\pi^{-1}(i)$ is the departing order of train $i \in \mathcal{T}$ at the initial station. We also assume that there are departure times $\bar{D}_{i1}$ for $i = 1, \ldots, |\mathcal{T}|$ after which train i can depart from station 1. Let us consider the departure times $\bar{D}_{is}$ and arrival times $\bar{A}_{is}$ for $i = 1, \ldots, |\mathcal{T}|$ and $s = 2, \ldots, \mathcal{S}$ obtained by scheduling all trains separately. This schedule is possibly infeasible as it can violate (E1) and (E4). However, this schedule is the most desirable one and is easily obtainable, and therefore we regard it as our target schedule, i.e., we want to find a schedule which is as close as possible to the target schedule.

MIP Formulation

For a more precise description of our scheduling problem and later use for its LP relaxation, we introduce a mixed integer programming (MIP) model. The model uses the following decision variables.

- $A_{i,s}$: the arrival time of train i at station s for all $i \in \mathcal{T}$ and $s \in \mathcal{S}$
- $D_{i,s}$: the departure time of train i at station s for all $i \in \mathcal{T}$ and $s \in \mathcal{S}$
- $Y_{i,j,s}$: the binary variable of which value is 1 if train i arrives at station s earlier than train j, and 0 otherwise for all $i, j \in \mathcal{T}$ and $s \in \mathcal{S}$
- $Z_{i,j,s}$: the binary variable of which value is 1 if train i departs from station s earlier than train j, and 0 otherwise for all $i, j \in \mathcal{T}$ and $s \in \mathcal{S}$

To define objective function and constraints, the planned arrival and departure times (or the target arrival and departure times) are needed, and we use $\bar{A}_{i,s}$ and $\bar{D}_{i,s}$ to denote these parameters, respectively.

The objective function is

$$\min \quad \sum_{i \in \mathcal{T}} \sum_{s \in \mathcal{S}} |\bar{A}_{i,s} - A_{i,s}| + |\bar{D}_{i,s} - D_{i,s}| \tag{1}$$

which represents the total difference with the target schedule.

Regarding (E1)–(E4) and other operational considerations, we consider the following constraints.

- The departing order at the initial station given by π should be observed:

$$D_{i,1} \geq D_{j,1}, \quad \text{for all } i \neq j, i, j \in \mathcal{T} \text{ such that } \pi^{-1}(i) > \pi^{-1}(j) \tag{2}$$

- The departure times of a train at stations should be later than the planned departure times.

$$D_{i,s} \geq \bar{D}_{i,s}, \quad \text{for all } i \in \mathcal{T} \text{ and } s \in \mathcal{S} \tag{3}$$

- The trains which stops at stations should dwell at least d time units.

$$D_{i,s} - A_{i,s} \geq d, \quad \text{for all } i \in \mathcal{T}_l \text{ and } s \in \mathcal{S} \tag{4}$$

$$D_{i,s} - A_{i,s} \geq d, \quad \text{for all } i \in \mathcal{T}_e \text{ and } s \in \bar{\mathcal{S}} \tag{5}$$

- Each train cannot move faster than its maximum speed.

$$A_{i,s+1} - D_{i,s} \geq l^l_{ss+1}, \quad \text{for all } i \in \mathcal{T}_l \text{ and } s \in \mathcal{S} \setminus \{S\} \tag{6}$$

$$A_{i,s+1} - D_{i,s} \geq l^e_{ss+1}, \quad \text{for all } i \in \mathcal{T}_e \text{ and } s \in \mathcal{S} \setminus \{S\} \tag{7}$$

- The arrival headway between two trains at the same station should be observed.

$$A_{j,s} - A_{i,s} \geq hY_{i,j,s} - M \times (1 - Y_{i,j,s}), \quad \text{for all } i \neq j \in \mathcal{T} \text{ and } s \in \mathcal{S} \tag{8}$$

- The departure headway between two trains at the same station should be observed.

$$D_{j,s} - D_{i,s} \geq hZ_{i,j,s} - M \times (1 - Z_{i,j,s}), \quad \text{for all } i \neq j \in \mathcal{T} \text{ and } s \in \mathcal{S} \tag{9}$$

- The arrival sequences of two trains of the same type at all stations should be same as those in π.

$$A_{i,s} \geq A_{j,s}, \quad \text{for all } i \neq j \in \mathcal{T}_l \text{ and } s \in \mathcal{S} \text{ such that } \pi^{-1}(j) < \pi^{-1}(i) \tag{10}$$

$$A_{i,s} \geq A_{j,s}, \quad \text{for all } i \neq j \in \mathcal{T}_e \text{ and } s \in \mathcal{S} \text{ such that } \pi^{-1}(j) < \pi^{-1}(i) \tag{11}$$

- The departure sequences of two trains of the same type at all stations should be same as those in π.

$$D_{i,s} \geq D_{j,s}, \quad \text{for all } i \neq j \in \mathcal{T}_l \text{ and } s \in \mathcal{S} \text{ such that } \pi^{-1}(j) < \pi^{-1}(i) \tag{12}$$

$$D_{i,s} \geq D_{j,s}, \quad \text{for all } i \neq j \in \mathcal{T}_e \text{ and } s \in \mathcal{S} \text{ such that } \pi^{-1}(j) < \pi^{-1}(i) \tag{13}$$

- For any trains $i \in \mathcal{T}$ and $j \in \mathcal{T}$, train i arrives at station s either before train j or after train j.

$$Y_{i,j,s} + Y_{j,i,s} = 1 \tag{14}$$

- For any trains $i \in \mathcal{T}$ and $j \in \mathcal{T}$, train i departs station s either before train j or after train j.

$$Z_{i,j,s} + Z_{j,i,s} = 1 \tag{15}$$

4. The Proposed Method

The basic idea of the proposed method is to decompose a given instance of scheduling problem and then solve each subproblem separately. The solution of the problem is obtained simply by merging the solutions of subproblems.

In a typical daily schedule of metro, a number of local trains are arranged between two consecutive express trains. For example, in Seoul metro, 2–3 local trains are serviced between two express trains at peak times, while there are 4–5 local trains at non-peak times. The particular pattern of trains is determined by the sequence of local and express trains, for example, $(L, L, E, L, L, L, E, L, L, E)$ is a service pattern of 7 local trains and 3 express trains where L and E denote the normal and rapid trains, respectively.

The decomposition is based on the service pattern of the trains given as π. Each subproblem consists of a single express train and its preceding local trains. Thus, for example, if a service pattern is $(L, L, E, L, L, L, E, L, L, E)$, we obtain three subproblems, (L, L, E), (L, L, L, E), and (L, L, E). Note that according to the service pattern π, we can order the subproblems. Let us consider the ith subproblem. By the operational environment (E4), the last local train in the ith subproblem cannot overtake the first local train in the $(i+1)$th subproblem. Moreover, the express train in the ith subproblem usually moves faster than any local trains; the first local train in the $(i+1)$th subproblem also cannot overtake the express train in the ith subproblem. Thus, if we assume that the express train in the ith subproblem cannot overtake any local trains in the $(i-1)$th subproblem, subproblems can be assumed to be independent to each other. Therefore, in the remaining part of this section we focus on the method to solve the subproblem.

4.1. MDP Formulation

The subproblem consisting of an express train and its preceding local trains is essentially same as the original problem with smaller number of trains. Thus, it has the same MIP formulation as the original one, and its MIP formulation involves smaller number of variables and constraints than the original problem. In particular, integer variables only appear between a single express train and local trains at the overtaking stations. A key observation is that if we specify the overtaking stations at which the express train overtakes its immediate predecessor, we can determine the values of all integer variables in the MIP formulation of subproblem. In addition, an optimal solution with the corresponding set, say O, of chosen overtaking stations can be represented as the increasing sequence of sets of overtaking stations which ends with O. For example, an optimal solution with $O = \{4, 11, 19\}$ can be represented as $(\emptyset, \{4\}, \{4, 11\}, \{4, 11, 19\})$ and each schedule corresponding the subset of O in the sequence can be obtained by solving LP problem by fixing all integer variables.

Based on these observations, we reformulate the subproblem as a sequential decision-making problem such that at each stage we determine the next overtaking station at which the express train overtakes its immediate predecessor. This sequential decision making problem can be modeled as a Markov decision process (MDP) in which the state space, the action space, the reward, and the transition function are defined as follows (also represented in Figure 1).

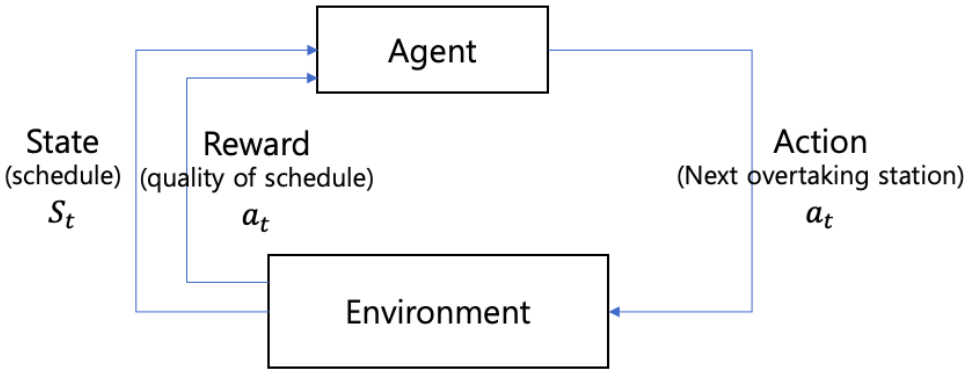

Figure 1. The proposed Markov decision process (MDP) framework.

- *State space*: A state represents the schedule of trains, that is, the arrival and departure times of trains at every station. Thus, a state can be represented by a vector $s \in \mathbb{R}^{2\times|\mathcal{S}|\times|\mathcal{T}|}$.
- *Action space*: At each time a decision is made, that is, at each state, the number of possible actions is equal to the number of overtaking stations plus 1. For each action corresponding to an overtaking station, a new state is determined by the transition function introduced later. These actions are called *overtaking actions*. A single additional action is reserved for the decision for stopping the further exploration of states or schedules, and it is called the *finishing action*.
- *Transition function*: For each overtaking action, a new overtaking station is added to the current set of overtaking stations used by the current schedule. There are two cases: (1) a new overtaking station is either already chosen previously or just passed in the current schedule, and (2) a new feasible schedule is generated by using the resulting set of overtaking stations. For case (1), the current schedule is not changed, and therefore actions corresponding to this case are called the *infeasible actions*. The other actions are called the *feasible actions*. For each feasible action, a new set of overtaking stations is well defined and values for integer decision variables for the MIP formulation of the subproblem can be fixed. Thus, we can obtain linear programming (LP) model for the subproblem which is efficiently solvable. An optimal solution of LP can provide a new schedule of trains, i.e., a new state s'. When the finishing action is chosen, an agent stops exploring the states with declaring the current state or schedule is the final one.
- *Reward*: For any state s, feasible actions generate the rewards of zero but for an infeasible action small negative reward is given to the agent. For the finishing action, the difference of qualities between the initial schedule and the current schedule is given as the reward. Thus, if the final state or schedule is better than the initial schedule a positive reward is awarded and the magnitude of the reward represents the level of improvement.

Figure 2 illustrates the process of the agent in the proposed MDP framework. Figure 2 describes four steps of the agent's decision-making. Step 1 is the initial schedule of four trains in which the first three trains (blue, yellow, and green lines) are local trains and the last train is an express train (red line). According to the decision of overtaking station the agent makes, the express train overtakes its preceding local train as shown in Step 2. Following the decisions shown in Step 3 and Step 4, the agent returns the schedule in Step 4 as a result of its finishing action.

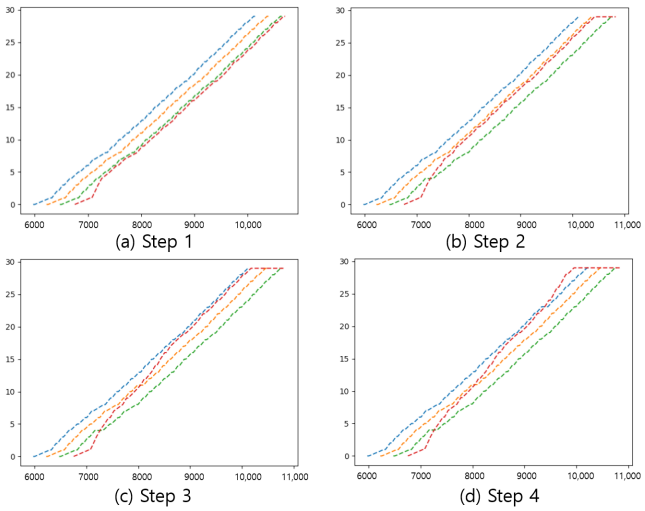

Figure 2. An example of sequential decision-making. (**a**) Initial step, (**b**) Step 2, (**c**) Step 3, (**d**) Final step.

4.2. Deep Q-Learning

Each instance of subproblem is defined by specifying the number of local trains preceding the express train and the target schedule of trains. In this paper, instead of trying to solve each instance whenever it is necessary, we try to learn an agent or a policy which can make near-optimal decisions for all instances. There has been several methods to learn such an agent (see, e.g., in [19]). Among them, we use deep neural networks, i.e., deep Q-network (*DQN*) [18], which we estimate *Q-values* defined as

$$Q^*(s,a) = \max_{\pi} \mathbb{E}\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots | s_t = s, a_t = a, \pi\right]$$

which is the maximum sum of rewards r_t discounted by γ at each time step t after making an observation s and action a is chosen a policy $\pi = P(a|s)$ is used. The idea is to represent Q-value function as a recursive equations, known as Bellman equation, and iteratively update Q-values with these equations, i.e.,

$$Q_{i+1}(s,a) = \mathbb{E}\left[r + \gamma \max_{a'} Q_i(s',a')\right].$$

$Q(s,a)$ is parameterized by θ in the form of the deep neural network. The loss function which guides the update of θ is

$$L(\theta) = \mathbb{E}\left[\left(r + \gamma \max_{a'} Q(s',a';\theta) - Q(s,a;\theta)\right)^2\right]. \tag{16}$$

To train the neural network, random scenarios, i.e., instances of the subproblem are generated. For each scenario, the agent choose an action following the ϵ-greedy policy in which most of the time the agent choose the action of the large Q-value but there exists a small chance, ϵ, of choosing a random action. Then, the obtained experiences, $(s_t^i, a_t^i, r_t^i, s_{t+1}^i)$ for scenario i and time t, are used for defining the loss function $L(\theta)$, which is then minimized with respect to θ. The learned Q-network can be used to make a decision by choosing the action which attains the largest Q-value.

We also adopt two strategies proposed in [18] to prevent a possible divergence of learning. The first strategy is using the experience replay memory which stores experiences (s,a,r,s') during the training and whenever updates of θ are performed several experiences are randomly chosen in the memory to form the loss function. In the second strategy, we maintain two neural networks, called the target network and the behavior network. The target network, parameterized by θ^-, is used to compute the target value in (16) and the behavior network is used to explore the scenarios. Thus, at each iteration i, the loss function is defined as

$$L_i(\theta) = \mathbb{E}\left[\left(r + \gamma \max_{a'} Q(s',a';\theta_i^-) - Q(s,a;\theta_i)\right)^2\right]$$

where the expectation is taken over the random experience in the memory. The parameters θ^- of the target network are updated to the parameter θ of the behavior network according to the predefined schedule.

5. Experiments

5.1. Instances

In our experiments, we generate instances for the subproblem consisting of a single express train and multiple local trains preceding the express train. These trains are assumed to run Seoul metro's Line 9 in South Korea. There are 30 stations from station 1 to station 30, and all express trains stop at 12 of them: stations 1, 6, 9, 12, 14, 16, 19, 22, 24, 26, 28, and 29. Overtaking between local trains and express train can occur at stations 4, 6, 11, 15, 19, 23, and 27. Therefore, the number of actions is equal to 8, i.e., the number of

overtaking stations plus 1. The dwell time of each train at its stopping station is equal to 30 s and safety headway is given as 60 s. The minimal transit times of each type of trains between two consecutive stations are given in Table 1.

Table 1. Minimum transit times (seconds) of local and express trains.

(From, To)	Local Train	Express Train	(From, To)	Local Train	Express Train
1–2	291	291	16–17	112	83
2–3	91	60	17–18	100	76
3–4	85	57	18–19	102	78
4–5	102	68	19–20	137	105
5–6	132	88	20–21	100	66
6–7	127	84	21–22	84	56
7–8	93	66	22–23	88	58
8–9	203	173	23–24	111	82
9–10	99	67	24–25	104	77
10–11	94	67	25–26	89	66
11–12	118	84	26–27	87	65
12–13	103	74	27–28	100	70
13–14	121	104	28–29	98	69
14–15	89	75	29–30	124	124
15–16	91	67	-	-	-

For a given departure time of a train at the initial station (station 1), the target schedule of the trains consists of the optimal schedules of each train ignoring other trains, that is, each train moves the line according to its minimum transit times between stations and staying for 30 s if it stops at stations. The aggregation of such schedules for each train possibly results in an infeasible schedule, but it is enough to take a role of the target schedule. Other choices of target schedules are also possible. On the other hand, for a given instance, the initial schedule is obtained by solving MIP of which integer variables are fixed by assuming that the orders of two trains are same with one at their initial station.

We assume that the subproblem consists of at most 4 trains, and therefore the number of local trains ranges from 1 to 3. For a given number of local trains and a single express train, the intervals of departure times at the initial station are randomly chosen between 120 s and 600 s.

5.2. Q-Learning

To train the agent, represented by a single neural network explained in the next subsection, we randomly generate an instance and this instance is solved by the agent. Random instances of a particular number of trains can vary according to the departure interval at their initial station. This process is called an *episode*. The experiences of the agent, consisting of the state, action, and the next state, will be used to train the neural network. The size of the experience memory is set to 200,000. To accumulate enough experiences, the update of the behavior network is postponed until 4000 episodes were explored. When the train begins, we randomly choose 8 experiences, and the weights of the neural network are updated whenever the agent takes an action. In addition, the parameter ϵ, initially set to 1, is diminished according to the factor of 0.9999 after each update of weights of the neural network. When ϵ is less than or equal to 0.01, the entire process of train ends. The target network is updated to the behavior network whenever an episode ends.

As explained earlier, for a given state s and the action a taken by the agent, the new state s' is obtained by solving the corresponding LP. LP is solved by PuLP. The optimal objective value is the absolute difference between the target schedule and the current schedule (the optimal solution of LP).

Thus, the performance measure used in our experiments is the objective value of LP defined by the final state, i.e., the final schedule.

5.3. Deep Q-Network

The neural network representing the agent consists of 4 layers with 512, 256, 128, and 64 units, respectively. These 4 layers are fully connected. All layers use the rectified linear activation. The output layer consists of the units of which number is same as the number of actions, i.e., 8. The linear activation is used for the output layer. The neural network is randomly initialized with respect to the normal distribution with mean of zero and standard deviation of 1.

The input of the network is the state and therefore it is $(2 \times 30 \times n)$-dimensional vector where n is the number of trains. As our instances can have training up to 4 and therefore we can fix n to 4. Whenever an instance has trains less than 4, the remaining entries of the input are filled with zeros.

As stated earlier, 8 experiences randomly chosen from the experience memory constitute a mini-batch of size 8. We use Adam [20] with a learning rate of 0.00001 as the training algorithm for the neural network and 16 epochs are repeated with the same mini-batch.

The neural network and its training algorithm are implemented with Keras library with TensorFlow backend. All experiments are performed with a single NVDIA Quadro P500.

5.4. Results

We train the neural network with 19,264 episodes and it takes 29,878.40 s to finish training. We first evaluate the learned neural network with 50 random instances consisting of only two trains: one local train and one express train. Table 2 demonstrates the result. The columns of the table indicate the instance number (#), the optimal objective value (MIP), the objective value from the learned agent (DQN), and optimality gap (Gap), respectively. The optimal gap is computed by $(\text{DQN} - \text{MIP})/\text{MIP} \times 100$.

Table 2. Comparison on 50 2-train instances.

#	MIP	DQN	Gap (%)	#	MIP	DQN	Gap (%)
1	5976	5976	0.00	26	7755	7755	0.00
2	7406	7406	0.00	27	8510	8510	0.00
3	5822	5822	0.00	28	8162	8162	0.00
4	9658	9658	0.00	29	6502	6502	0.00
5	5742	5742	0.00	30	7512	7512	0.00
6	5318	5318	0.00	31	6273	6273	0.00
7	5266	5266	0.00	32	6528	6528	0.00
8	10,666	10,666	0.00	33	7446	7446	0.00
9	10,610	10,610	0.00	34	7003	7003	0.00
10	7905	7905	0.00	35	5902	5902	0.00
11	9112	9112	0.00	36	8415	8415	0.00
12	4662	4662	0.00	37	8178	8178	0.00
13	8466	8466	0.00	38	9112	9112	0.00
14	8036	8036	0.00	39	8204	8204	0.00
15	6466	6466	0.00	40	4622	4622	0.00
16	5822	5822	0.00	41	6681	6681	0.00
17	8084	8084	0.00	42	6439	6439	0.00
18	6422	6422	0.00	43	6579	6579	0.00
19	9170	9170	0.00	44	9490	9490	0.00
20	5640	5640	0.00	45	6783	6783	0.00
21	7191	7191	0.00	46	8454	8454	0.00
22	9274	9274	0.00	47	6520	6520	0.00
23	8160	8160	0.00	48	6792	6792	0.00
24	5994	5994	0.00	49	8211	8211	0.00
25	9714	9714	0.00	50	8211	8211	0.00

As observed in Table 2, the learned policy successfully finds optimal schedules for all 50 instances.

Table 3 demonstrates the result tested on 50 random instances consisting of 3 trains, that is, 2 local trains. The structure of Table 3 is same as that of Table 2.

Table 3. Comparison on 50 3-train instances.

#	MIP	DQN	Gap (%)	#	MIP	DQN	Gap (%)
1	12,592	15,352	21.92	26	14,336	14,336	0.00
2	12,626	12,626	0.00	27	17,746	17,746	0.00
3	15,562	15,562	0.00	28	14,048	14,048	0.00
4	6444	6444	0.00	29	12,342	15,640	26.72
5	15,220	15,220	0.00	30	14,624	14,624	0.00
6	6600	6600	0.00	31	9713	9713	0.00
7	12,800	12,800	0.00	32	15,262	15,262	0.00
8	13,464	13,464	0.00	33	12,650	12,650	0.00
9	13,289	13,289	0.00	34	12,666	12,666	0.00
10	5944	5944	0.00	35	13,649	13,649	0.00
11	9640	10,396	7.84	36	8912	9668	8.48
12	6444	6444	0.00	37	12,691	13,178	3.84
13	14,882	14,882	0.00	38	12,249	12,249	0.00
14	12,134	12,629	4.08	39	13,058	13,058	0.00
15	6144	6144	0.00	40	9288	9288	0.00
16	8628	9276	7.51	41	8599	9220	7.22
17	12,452	12,452	0.00	42	11,708	11,708	0.00
18	11,354	11,354	0.00	43	11,174	11,930	6.77
19	12,644	12,644	0.00	44	13,092	15,490	18.32
20	8548	8548	0.00	45	12,597	12,597	0.00
21	15,031	15,031	0.00	46	11,708	11,708	0.00
22	12,566	12,636	0.56	47	13,289	13,289	0.00
23	13,092	15,490	18.32	48	9136	9892	8.27
24	17,594	17,594	0.00	49	9756	9756	0.00
25	7328	7328	0.00	50	8253	8550	3.60

Among 50 instances, the learned policy succeeds at finding the optimal solutions for 36 instances. The gaps for the instances for which the learned policy fails to give optimal solutions are relatively low (within 10%), except for instances #1, #23, #29, and #44. However, the schedules obtained by the learned policy turn out to be largely improved from their initial schedules of which objective values are $28,112, 28,412, 28,712$, and $28,412$, respectively. Furthermore, the sequences of overtaking stations determined by two schedules are quite similar, for example, $(6, 15)$ for optimal schedule and $(4, 15)$ for the schedule by the learned policy.

Finally, we tested on random instances consisting of 4 trains, that is, three local trains. The structure of Table 4 is again same as the previous two.

Table 4. Comparison on 50 4-train instances.

#	MIP	DQN	Gap (%)	#	MIP	DQN	Gap (%)
1	11,186	11,942	6.76	26	11,278	12,034	6.70
2	9248	9248	0.00	27	19,554	29,198	49.32
3	18,492	18,492	0.00	28	20,148	21,424	6.33
4	11,460	16,561	44.51	29	10,542	11,298	7.17
5	10,529	10,529	0.00	30	10,276	10,330	0.53
6	24,530	35,364	44.17	31	22,158	33,090	49.34
7	19,665	22,278	13.29	32	12,442	12,442	0.00
8	16,589	16,589	0.00	33	25,260	36,034	42.65
9	19,803	22,034	11.26	34	12,194	12,194	0.00
10	10,529	10,529	0.00	35	20,079	21,546	7.31
11	7796	7796	0.00	36	14,656	17,552	19.76
12	17,125	17,125	0.00	37	10,162	10,162	0.00
13	20,186	21,058	4.31	38	24,092	34,962	45.12
14	7536	7536	0.00	39	10,694	10,694	0.00
15	12,736	14,592	14.57	40	18,853	23,766	26.06
16	13,556	18,267	34.75	41	16,059	17,919	11.58
17	13,656	18,202	33.29	42	12,644	13,062	3.31
18	16,907	16,907	0.00	43	7172	7172	0.00
19	18,618	18,618	0.00	44	14,656	17,552	19.76
20	12,210	16,311	33.59	45	9080	9080	0.00
21	8172	8388	2.64	46	11,610	16,511	42.21
22	11,954	11,954	0.00	47	25,698	36,436	41.79
23	5694	5694	0.00	48	19,805	29,582	49.37
24	19,018	19,016	0.46	49	7983	8010	0.34
25	8367	8772	4.84	50	22,632	33,622	48.56

Table 4 shows that the performance of the learned policy is quite poor compared with the previous two results. The average gap over 50 instances is 14.51% (0.00% and 2.87% for the previous two results, respectively). Four-train instances are more difficult than the 2- and 3-train instances. In particular, during the training phase of the neural network, the agent has little experiences on the cases of 2 and more overtaking stations. Indeed, by investigating the experience memory which stores the experience the agent used during training, we found that the proportions of the experiences considering 0, 1, 2, and 3 overtaking stations are about 10%, 45% 25%, and 20%, respectively. Therefore, there exists a severe imbalance of experiences if a simple ϵ-greedy exploration strategy is adopted. We conjectured that this phenomena can be avoided when we design more sophisticated strategies for the exploration during training.

6. Conclusions

In this paper, we consider a train scheduling problem in which both local and express trains are scheduled. For a given target schedule on a unidirectional single-track railway, we decompose the problem into separate subproblems in which a single express train and its preceding local trains are to be scheduled. We formulate each subproblem as a Markov decision problem and solve it with deep reinforcement learning technique.

We demonstrated the performance of the proposed method over 150 instances which are further divided into three groups with respect to the number of local trains. For the instances with a small number (2 and 3) of local trains, the proposed method is compatible with the exact algorithm (e.g., branch-and-bound or branch-and-cut methods). However, for larger instances, it still requires an improvement to obtain better schedules.

The advantage of the proposed method is that the learned policy can be used for any instances regardless of the number of local trains. Another advantage is its possible extension to use more complex objective function (e.g., energy consumption, passenger crowdedness, etc.) with alternative forms of rewards. However, this possibility is still conceptual and therefore empirical validations are needed as one of the future studies.

The obvious next step for future research is to devise an effective exploration strategy by which the agent can approximate the Q-value for diverse states. Furthermore, we do not have enough experiments on possible values of hyperparameters such as the architecture of the neural network and the parameters related to its training (e.g., learning rate). Therefore, more careful investigations, for example, Bayesian optimization, are required.

Author Contributions: Conceptualization, S.O. and Y.M.; methodology, Y.M.; software, I.G. and Y.M.; validation, I.G., S.O., and Y.M.; formal analysis, Y.M.; investigation, I.G.; resources, S.O.; data curation, I.G.; writing—Original draft preparation, I.G.; writing—Review and editing, Y.M.; visualization, I.G.; supervision, Y.M.; project administration, S.O.; funding acquisition, S.O. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Korea Railroad Research Institute grant number PK2002B2.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Corman, F.; D'Ariano, A.; Marra, A.D.; PAcciarelli, D.; Sama, M. Integrating train scheduling and delay management in real-time railway traffic control. *Transp. Res. Part Logist. Transp. Rev.* **2017**, *105*, 213–239. [CrossRef]
2. Lusby, R.M.; Larsen, J.; Ehrgott, M.; Ryan, D. Railway track allocation: Models and methods. *OR Spectr.* **2011**, *33*, 843–883. [CrossRef]
3. Brannlund, U.; Lindberg, P.O.; Nou A.; Nilsson, J.E. Railway timetabling using Lagrangian relaxation. *Transp. Sci.* **1998**, *32*, 358–369. [CrossRef]
4. Caimi, G.; Fuchsberger, M.; Laumanns, M.; Schupbach, K. Periodic railway timetabling with event flexibility. *Networks* **2011**, *57*, 3–18. [CrossRef]
5. Cacchiani, V.; Furini, F.; Kidd, M.P. Approaches to a real-world train timetabling problem in a railway node. *Omega* **2016**, *58*, 97–110. [CrossRef]
6. Jiang, F.; Cacchiani, V.; Toth, P. Train timetabling by skip-stop planning in highly congested lines. *Transp. Res. Part B Methodol.* **2017**, *42*, 553–570. [CrossRef]
7. Fischetti, M.; Salvagnin, D.; Zanette, Z. Fast approaches to improve the robustness of a railway timetable. *Transp. Sci.* **2009**, *43*, 321–335. [CrossRef]
8. Zhou, X.; Zhong, M. Single-track train timetabling with guaranteed optimality: Branch-and-bound algorithms with enhanced lower bounds. *Transp. Res. Part Methodol.* **2007**, *41*, 320–341. [CrossRef]
9. Wang, Y.; De Schutter, B.; van den Boom, T.J.J.; Ning, B.; Tang, T. Efficient bilevel approach for urban rail transit operation with stop-skipping. *IEEE Trans. Intell. Transp. Syst.* **2014**, *15*, 2658–2670. [CrossRef]
10. Dundar, S.; Sahin, I. Train re-scheduling with genetic algorithms and artificial neural networks for single-track railways. *Transp. Res. Part C Emerg. Technol.* **2013**, *27*, 1–15. [CrossRef]
11. Niu, H.; Zhou, X. Optimizing urban rail timetable under time-dependent demand and oversaturated conditions. *Transp. Res. Part Emerg. Technol.* **2013**, *36*, 212–230. [CrossRef]
12. Khadilkar, H. A scalable reinforcement learning algorithm for scheduling railway lines. *IEEE Trans. Intell. Transp. Syst.* **2019**, *20*, 727–736. [CrossRef]
13. Liu, R.; Li, S.; Yang, L.; Yin, J. Energy-efficient subway train scheduling design with time-dependent demand based on an approximate dynamic programming approach. *IEEE Trans. Syst. Man Cybern. Syst.* **2018**, *50*, 2475–2490. [CrossRef]

14. Jiang, Z.; Gu, J.; Fan, W.; Liu, W.; Zhu, B. Q-learning approach to coordinated optimization of passenger inflow control with train skip-stopping on a urban rail transit line. *Comput. Ind. Eng.* **2019**, *127*, 1131–1142. [CrossRef]
15. Yin, J.; Tang, T.; Yang, L.; Gao, Z.; Ran, B. Energy-efficient metro train rescheduling with uncertain time-variant passenger. *Transp. Res. Part Methodol.* **2016**, *91*, 178–210. [CrossRef]
16. Ghasempour, T.; Heydecker, B. Adaptive railway traffic control using approximate dynamic programming. *Transp. Res. Part C Emerg. Technol.* **2019**, *38*, 201–221. [CrossRef]
17. Semrov, D.; Marsetic, R.; Zura, M.; Todorovski, L.; Srdic, A. Reinforcement learning approach for train rescheduling on a single-track railway. *Transp. Res. Part B Methodol.* **2016**, *86*, 250–267. [CrossRef]
18. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G. et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]
19. Sutton, R.; Barto, A. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 1998.
20. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.