Deep Reinforcement Learning Approach for Train Rescheduling Utilizing Graph Theory

Conference Paper · December 2018

DOI: 10.1109/BigData.2018.8622214

3 authors, including:

Takehiro Kashiyama
The University of Tokyo
32 PUBLICATIONS **269** CITATIONS
SEE PROFILE

Yoshihide Sekimoto
The University of Tokyo
137 PUBLICATIONS **946** CITATIONS
SEE PROFILE

Deep Reinforcement Learning Approach for Train Rescheduling Utilizing Graph Theory

Mitsuaki Obara
Faculty of Engineering
University of Tokyo
Tokyo, Japan
e-mail: obara-mitsuaki679@g.ecc.u-tokyo.ac.jp

Takehiro Kashiyama
Institute of Industrial Science
University of Tokyo
Tokyo, Japan
e-mail: ksym@iis.u-tokyo.ac.jp

Yoshihide Sekimoto
Institute of Industrial Science
University of Tokyo
Tokyo, Japan
e-mail: sekimoto@iis.u-tokyo.ac.jp

***Abstract*—Railway disturbances occur every day and train rescheduling is conducted by human experts. Approaches for automating rescheduling have been widely studied (e.g., heuristic approach and mixed integer problem-based approach). However, extant research is still inadequate for employing the approach in practical application in terms of size, run-time, and solution accuracy. In this research, we simulate train dispatching using graph theory and propose a reinforcement learning method (i.e., Deep Q-Network (DQN)) for rescheduling. We also show experimental results of this algorithm. DQN presented positive results for over 50% of test cases, and its train rescheduling decreased approximately 20% of passengers' dissatisfaction in a certain case. It can be expected that applying a DQN approach to real-world scale cases by will improve methods to handle larger-scale networks.**

Keywords—Train rescheduling, Reinforcement learning, Graph theory, Deep Q-network, PERT graph

I. INTRODUCTION

In Japan, 5,278 train disruptions were reported in 2011 [1], despite a high train service frequency, especially in major cities. Thus, there is considerable confusion when disruptions occur.

If train services are disrupted, they are brought back to regular train rescheduling by dispatchers. Tactics include altering departure sequences, turning trains back, using different tracks, or cancelling train runs. It is assumed that dispatchers' empirical knowledge supports train rescheduling. However, it is unstable in terms of individual dependency. Therefore, studies on automatic train rescheduling are socially valiant in regards to knowledge management.

Researchers worldwide have studied this problem, which is formulated as an optimization problem. Regarding operators' objectives, several variables have been proposed, such as minimization of train delays [2], minimization of passengers' travel time [3], and minimization of passengers' dissatisfaction [4]. Furthermore, researchers have proposed many approaches to affect these variables. In [5], a depth-first search was applied to solve the problem; in [6], a tabu search was applied; in [7], a genetic algorithm was applied, and an ant-colony optimization was applied in [8]. For these studies, equations were used to express optimal solutions and restrictions. A method of graph modeling for formulation of train dispatches was also proposed in [9]. This creates complicated rescheduling. In [10], graph modeling and simulated annealing was proposed.

Because all the above methods were heuristic, run time was fast. However, they tend to end at a local optimum. Further, escapes from the local minima were not guaranteed. Local optimality is difficulty in heuristic approaches.

Researchers have proposed a formulation based on the mixed integer problem (MIP). This method provides an exact solution [11]. The integer problem [12] and MIP-based formulation minimizing passengers' dissatisfaction [13] have been also proposed. In the MIP framework, researchers developed more realistic formulations [14].

Recently, other approaches (i.e., distributed models) appeared [15]. They adopted iterative procedures of timetable rescheduling, rolling-stock rescheduling, and passenger models to solve the problem interactively. However, this approach is new and requires more discussion on iterative processes.

While the MIP approach acquires an exact solution, run time and problem size depend on the performances of the solver and the computer. Because available time between the accident occurrence and the rescheduling are very short, a collateral run time is indispensable. As the scope of the handled size becomes larger, it takes more time to utilize this approach for real-scale problems.

In summary, approaches that realize the following conditions are the goal of automatic rescheduling: fast run time, capability of handling large-scale schedules, and high-quality solutions.

The development of reinforcement learning has been remarkable recently [16]. This technology may be able to deal with some kinds of task better than human, such as Alpha Go Zero [17], a heuristic solution. It is fast because only the actual delay information has to be provided to pre-learned agents when disruptions occur.

Related reinforcement learning approaches to traffic have appeared. [18] provided reinforcement learning to traffic signal controls. [19] applied deep-Q learning to traffic-signal controls. For railways, reinforcement learning was conducted for coordinated passenger inflow control [20] and train rescheduling on a single-track railway [21].

Thus, we formulated train dispatching using graph theory and solved the train rescheduling problem with a reinforcement-learning approach. We tried to balance run time,

capability of handling large-scale schedules, and high-quality solutions.

Specifically, we constructed a graph as an environment modeling double-track railroads and train delays. We adopted this modeling because it can express double-track and major rescheduling techniques in Japan. We used the Deep Q-Network (DQN), a neural network, as an agent. Train delay then becomes input to the neural network. Additionally, graph deformation instructions of the graph become the output. This translates to action in the reinforcement learning framework: the agent learns a method to minimize passengers' dissatisfaction. After training the networks, we measure its responses to unknown delays and discuss its performance and applicability.

II. METHOD

A. Overview

In this section, we explain the framework of reinforcement learning and state the overview of the research. Then, we state the contents of the rest of the sections.

Reinforcement learning is a type of machine learning from which an agent in an environment can observe the current state and determine action. The framework is expressed as the interaction between two objects: agent and environment. With thorough training, the agent can be trained to choose the most valuable action in a given environment. The framework is shown in Fig. 1. The agent observes the current environment as a state and determines the action based on this observance. After the environment receives this action and reflects it to itself, it returns an evaluation of the action to the agent as a reward. The agent then learns of its performance by the reward amount.

In this research, we use a DQN (a neural network) as the agent. The environment is a graph that expresses train dispatching activities, the state is a delay situation described in the graph, and an action is an instruction in which delays should be minimized. We provide further details in later sections

In Section B, we specifically state problem settings and discuss objective criteria. The reward is the difference of this objective evaluation before and after the action is applied. In Section C, we explain the structure of the graph. The delay description is provided in Section D. In Section E, we explain the deformation of the graph and its meaning. In Section F, the structure of neural networks and actions based on deformations are explained.

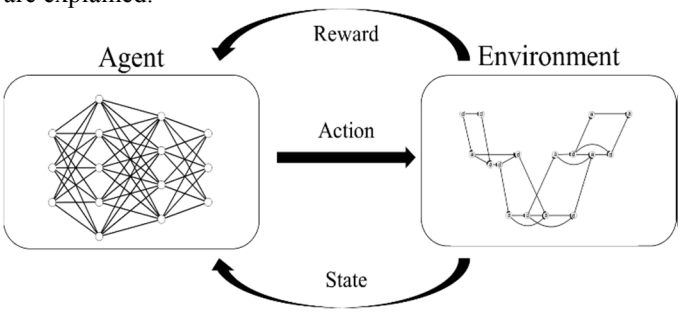


Fig. 1. Overview of reinforcement learning for train reschedule

B. Problem setting and evaluation of deformation

In this section, we explain problem settings and objective criteria. We propose using "passenger dissatisfaction" as a variable. Table 1 lists the objective criteria. Train delays and some inconveniences are assumed as dissatisfaction criteria from passengers' viewpoints. These criteria were also adopted in [10]. The background of this idea stems from the diversity of train rescheduling. A single criterion of train rescheduling is insufficient. For example, if we only aim to minimize total delay of trains, the simplest solution is to cancel all trains until the delay converges. However, in reality, this is a terrible rescheduling method from a passenger's viewpoint.

In Tokyo, trains usually arrive in intervals of several minutes. In many stations, passengers are continuously arriving. Thus, if stopping trains were easy, stations would overflow with passengers and become very confusing quickly. In this case, even if frequency decreases and the delay recovery is prolonged, it would be more appropriate to make the train interval constant. Therefore, it is more appropriate to score dissatisfactions using several criteria.

In this research, we tackle the problem of scoring passengers' dissatisfaction using several criteria and work to minimize them as we manipulate the train schedule.

We can calculate the delay by graph modeling. Passengers' discontent types are defined in Table 1. Then, we calculate the dissatisfaction score. With respect to these scores, we refer to the entirety of the graph and calculate the score as the sum of the products of delay seconds and default magnifications of each item, and products of the number stations where passengers are not dropped and prescribed deduction points. We can create gradients among dissatisfaction types by adjusting magnifications.

TABLE I. PASSENGERS' DISSATISFACTION TYPE

Dissatisfaction type	Explanation
Delay itself	Departure/arrival at a station is out of time in the timetable.
Stoppage	The train stops at a station longer than scheduled.
	The train does not drop passengers off at the scheduled stations.
Driving time	Driving time between stations is longer than scheduled. It is difficult to get to the station on time.
Frequency	Interval between trains at certain stations in the same direction is longer than scheduled.
Connection	Intervals between trains in the same direction and period at certain stations is longer than scheduled

C. Modeling

For a neural network to manipulate a timetable, the train schedule must be expressed in a graph form. This creates the environment for reinforcement learning. The graph we used was a program evaluation and review technique (PERT) graph. Because PERT is a directed acyclic graph, it finds critical paths at the rate O(|V|+|E|) by using breadth-first search. Critical path refers to the longest path from one node to another. The length of the critical path provides the shortest time to finish a task. This method to calculate the required time is called the critical-path method (CPM). Thus, we apply the CPM to the train

schedule. Train management imposes different kinds of constraints. For example, it requires time to securely run between stations and specifies an interval for trains to guarantee platform safety. A graph realizes these constraints with its edges; time-table states of each train are the nodes.

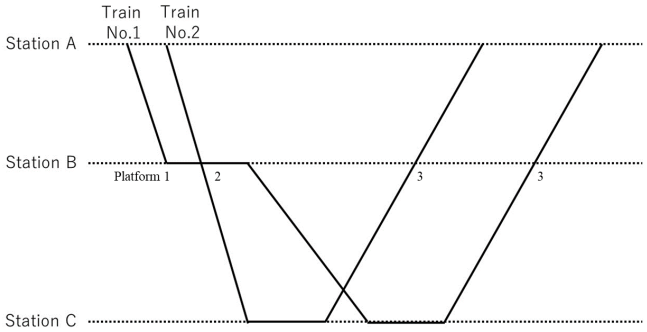


Fig. 2. Two trains' schedule expression

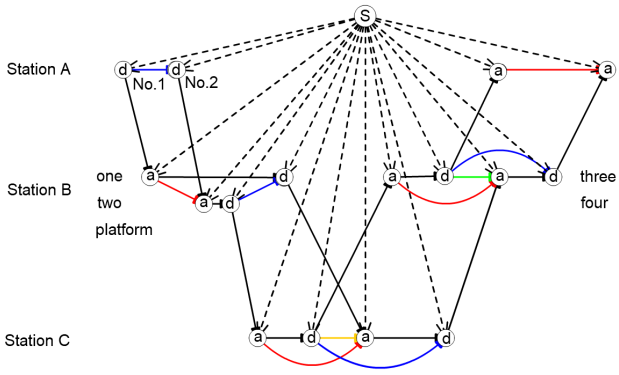


Fig. 3. PERT graph expression of the train schedule

TABLE II. EDGE TYPE OF RESTRICTIONS FOR SAFE TRAIN DISPATCHING

Edge Type	Weight and meaning	Edges in Fig. 3
Running	minimum time required for trains to run from one station to the next	Black (d to a)
Stoppage	minimum time required for the passengers to get on and off the train safely	Black (a to d)
Turning	minimum time required for trains to shuttle continuously	Black at last stations (a to d)
Departure order	minimum time required time for the departure of trains bound for the same station continuously	Blue
Arrival order	minimum time required for the arrival at the same station continuously	Red
Same platform	minimum time required for same-bound trains to use the same platform continuously	Green
Cross hindrance	minimum time required time for the departure or arrival at the last station continuously	Orange
Schedule	edges from starting node to each node – weight is schedule time of each event	Dotted black

Fig. 3 describes the train schedule from Fig. 2 as a PERT graph. Arrival or departure of a train corresponds to a node. Each node contains information about a location (i.e., platform and schedule time). Additionally, we establish a starting node — a virtual node and starting point of CPM. The edge types of restrictions for safe train dispatching are described in Table 2. We sometimes omit schedule edges for figure simplification.

D. Delay simulation and calculating

In this section, we delineate how to express an actual delay on the graph. This corresponds to states of reinforcement learning.

1) Delay simulation

The weight of each edge indicates the minimal time required for events on a node to occur successively. It implies that, by increasing this weight on a specific edge, we can simulate a train delay. We classify delays into two types; this classification includes most delays, except for cases where more than one delay on the same line occur simultaneously (e.g., damage to the area by earthquake or typhoon).

a) Delay at the station: If there is an issue at the station, train arrival or departure will be delayed. This can be expressed by adding adequate weight (i.e., delay time) to the schedule edge of the corresponding node.

b) Disruption in railway track: It is caused by issues on the rail line. This is expressed by specifying the time and location of the issue, and the time when that is solved. We put weight on the schedule edges of train nodes that would otherwise avoid issues and set the end of schedule edge as the arrival time after the issue is resolved. This describes an accident directly affecting multiple trains according to the location and time.

2) Delay expression in a PERT graph,

Actual timetables are drawn up with enough time space; hence, critical path to each node is its scheduled edge, and the weight corresponds to schedule time on the node.

However, when a delay occurs and edge-weights increase, some critical paths may change, and they may no longer be equal to the schedule time. Then, the total length of critical paths represents the minimal time required to execute the event. We call this the "practical time". For each node, the difference between the critical path and scheduled time on a timetable provides the total delayed time on the node. The feasible train dispatching is obtained along the practical time and platform information on each node. However, the crew schedule is not considered here. Thus it should be made separately along the proposed rescheduling plan. It is beyond this paper.

E. Deformation of the PERT Graph

In this section, we explain the transformation of graphs, by which the actions in reinforcement learning in this paper are defined.

When a train is delayed, some transformations in the graph can alleviate the impact of the delay. Here, the variables in the graph are "platform number", "arrival (or departure) order", and "turning edge". Other edges (e.g., cross-hindrance edge, platform edge, schedule edge, running–stoppage edge, departure (arrival) edge) are automatically determined by these variables, or they can be considered constant with certain manipulations. Therefore, we explain how to transform the graph by focusing on these three variables. The initial state is illustrated in Fig. 4 a).

1) Order change

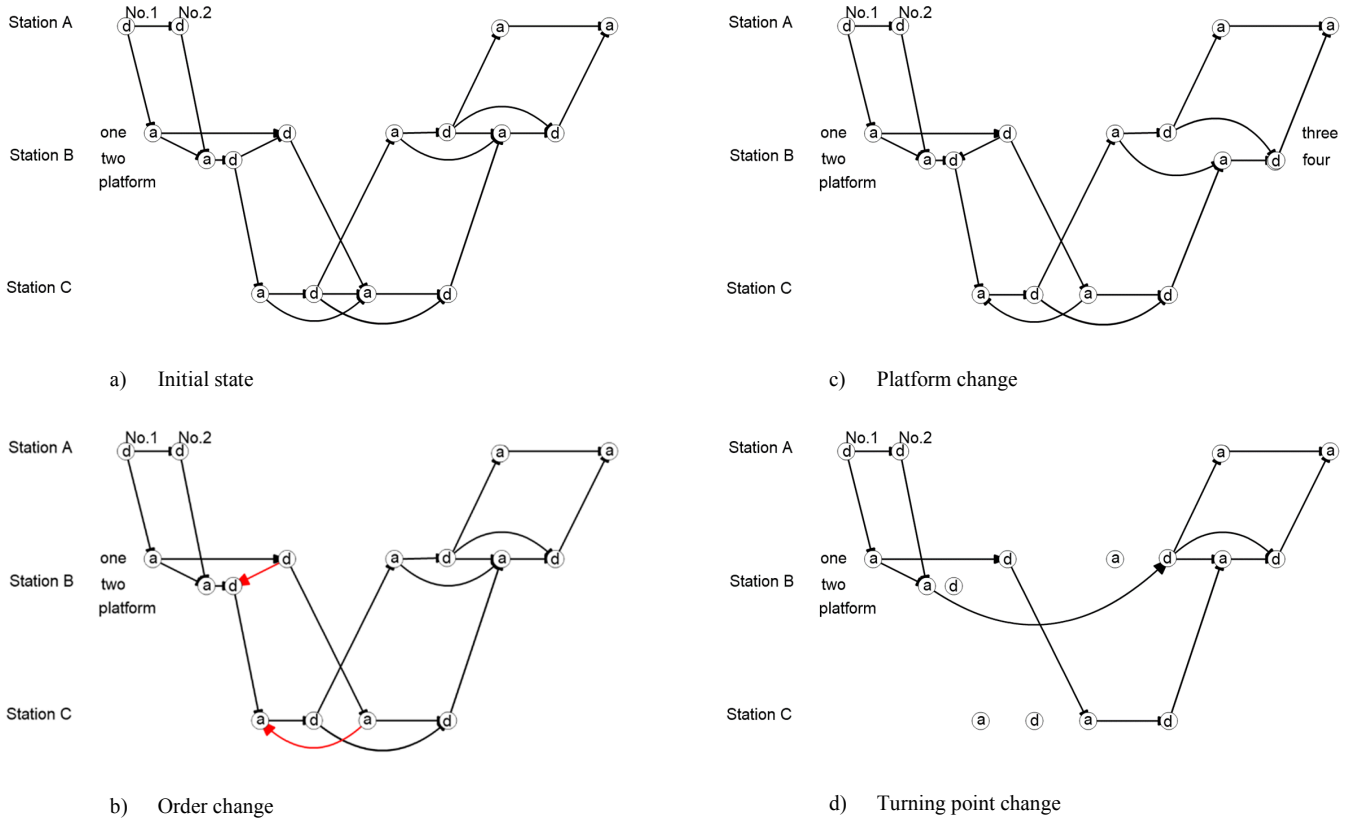
The order edge specifies the order of trains arriving at a certain station. Thus, transpositions of order edges correspond to the change in orders of train departure (or arrival). In many cases in Japan, each station has multiple rail tracks, but only one for each direction. Therefore, by changing the ordering of arrival/departure of many trains, including local and express trains, the schedule is rearranged. Manipulating this ordering can suppress the contagion effect of train delays. This is illustrated in Fig.4 b).

2) Platform change

In general, platforms are allocated according to the types of trains and when a delay occurs consecutive trains get stuck because the preceding trains have stopped at the platform. Rearrangement of platform allocation can allow dispatchers to make better use of platforms. This is illustrated in Fig.4 c).

3) Turning point change

When the delay is long, some trains must turn before the terminal. This is effective for recovering from delays, but it can prevent passengers from reaching their destinations. Hence, turning-point changes must be made only after careful deliberation. Thus, the decision to take this action should be based on a global judgement, not a local one. This is illustrated in Fig. 4 d).



a) Initial state

b) Order change

c) Platform change

d) Turning point change

Fig. 4. Deformation of PERT graph

F. *Deformation process by neural network*

In this section, we introduce the structure of a neural network — an agent in the reinforcement learning framework. We also explain actions indicated by the agent.

1) The structure of neural network

We modeled train timetables utilizing graph theory and proposed rescheduling methods as a form of graph deformation. We also quantified delays using this graph. Thus, we scored

passengers' dissatisfaction according to the criteria considered in Section B. Next, we introduced the process of reinforcement learning. The input and output to the neural network are explained here.

Input is the delay value in each node occurring in the current graph. Specifically, a neural network has an input layer. The number of artificial neurons is equal to the nodes in the graph. Each artificial neuron corresponds to a node, and the input value of each neuron is the delay in the corresponding node's delay (practical time - schedule time). Thus, neural network receives the information on how delays occurs from the graph. If we input all information of the current graph (i.e., the structure of graph, including the edges' information and nodes' platform information), space complexity of input layer becomes $O(|V|+|E|) \leq O(|V^2|)$ in this case. This is too large and sparse for a network to learn easily. Thus, we adopted the information showing the strictest current state.

The number of artificial neurons in the output layer is equal to the nodes in the graph. The output is node's number required to minimize its delay. A deterministic algorithm chooses an actual deformation to minimize the node's delay. It is designed to simplify the task from which neural networks learn while keeping their original purpose (i.e., improvement of solution quality). We explain this algorithm in detail in the next section.

We confined the scope of neural networks to make the training easier. We explain this in detail later in the experiments section.

2) Deterministic algorithm to minimize delay in the chosen node

The node selected by the neural network indicates a delay point to be minimized.

First, the algorithm finds a critical path between the starting point and the selected node and tries all possible deformations about all edges and nodes on the path. It compares them and decides the deformation that minimizes the node's delay the most. When a certain node is chosen, the algorithm tries the deformation changing the platform number. It can cause deformation of platform edges. When an order edge on the graph is chosen, the edge is transposed. In that time, platform change is executed simultaneously if needed. In some situations, dispatches can become unusual without this platform change. When turning edges are chosen, the algorithm tries the deformation of turning point changes.

In every deformation, the algorithm compares every deformed graph and calculates the delay in the selected node, regarding the one minimizing the delay the most as the deformation to be selected. Although turning point change is the best deformation for minimizing delay, it can be adopted only when the total score of the graph is improved. This is because the deformation affects many passengers and should be based on global judgement, not a local one, as mentioned before.

After the deformation, the deformed graph becomes the next input and selects the next deformation to the graph. This cycle provides train rescheduling via a neural network. For example, we show a concrete process of this deterministic algorithm. The initial state after delay input is described in Fig. 5. The red line is the initial critical path. The algorithm tries all possible deformations below:

Case 1: Platform change of No. 1 and 2 at Station B, outbound in Fig. 6 a).

Case 2: Change of departure sequence at Station A, outbound in Fig. 6 b).

Case 3: Turning point change from Station C to Station B in Fig. 6 c).

Case 4: Platform change of No. 1 at Station B, inbound in Fig.6 d).

Table 3 summarizes the result of each deformation. Case 3 isn't adopted because the score falls after the deformation is applied. Case 2 is the most effective deformation for valid results. Thus, the algorithm adopts this as the best deformation and executes it. This cycle creates train rescheduling for this research.

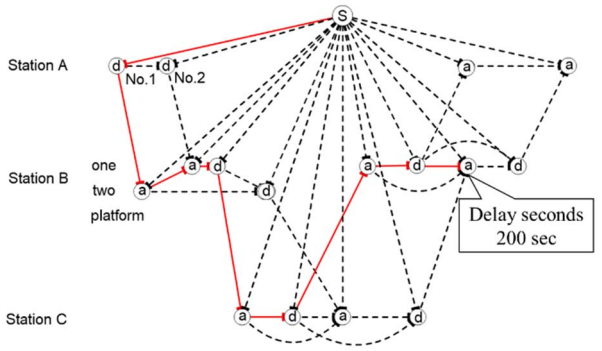
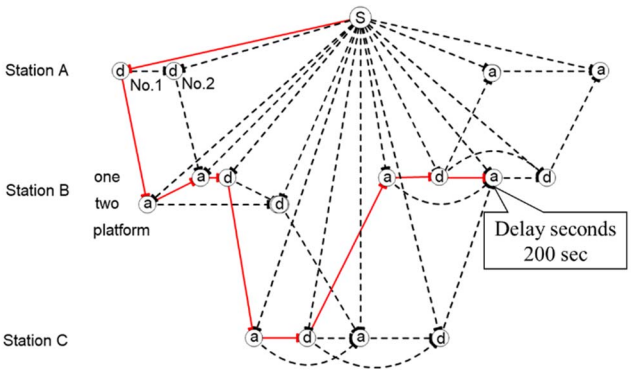


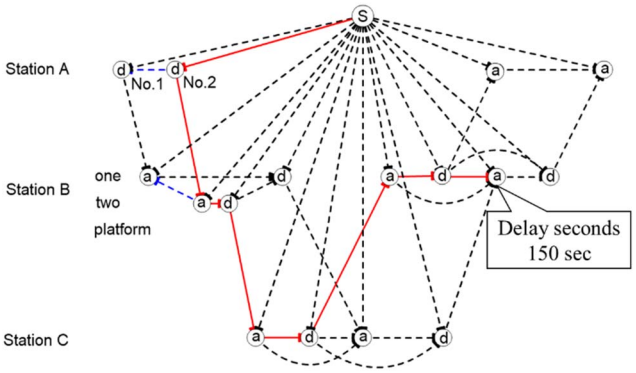
Fig. 5. Deterministic algorithm: Initial state

TABLE III. DEFORMATIONS RESULT BY DETERMINISTIC ALGORITHM

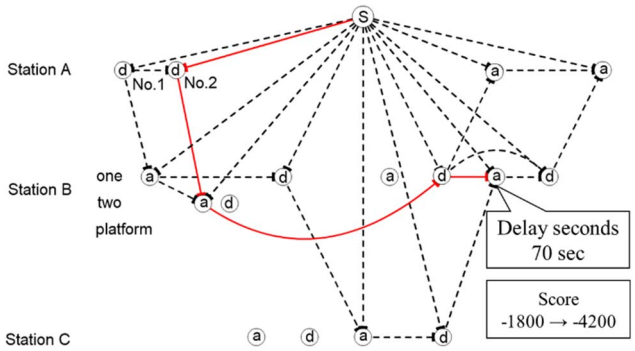
State	Delay seconds (s)	Supplement
Initial state	200	-
Case 1	200	-
Case 2	150	-
Case 3	70	Invalid (the score doesn't improve)
Case 4	180	-



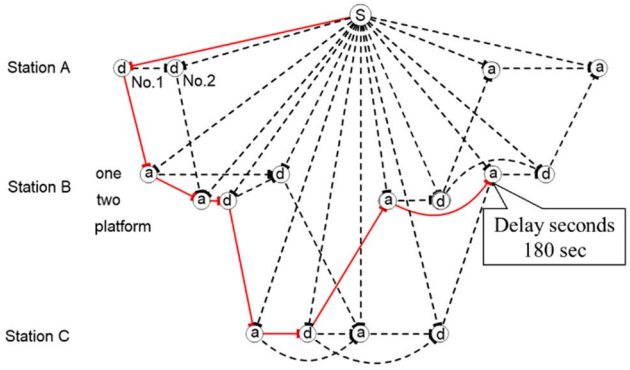
a) Case 1: Platform change at station B, outbound



b) Case 2: Departure sequence change at station A, outbound



c) Case 3: Turning point change from station C to B



d) Platform change of No. 1 at station B, inbound

Fig. 6. Possible change of the graph in Fig.5

III. EXPERIMENT

A. Settings

We generated a delay in the virtual train schedule, and allowed the neural network to learn and adapt to the situation. Then, we evaluated how effectively it could adapt to unknown cases.

The virtual train schedule used in this experiment included some "overtake", where a consecutive express train passes a local one; this is often seen in Japan. We restricted the number of stations to eight. This is a realistic number of stations where the order of trains can be changed for each line. Station A is the city center, so trains from Stations A to H are outbound, whereas ones from H to A are inbound. At Station F, it is impossible to change the order of outbound trains because Station F has only one platform. This reflects the fact that there exists a line where some stations have different numbers of available platforms for inbound and outbound trains. The scoring criteria are listed in Section 2. B but we omitted "connection" for simplification. Table 4 lists the magnification weights and thresholds above which dissatisfaction occurs.

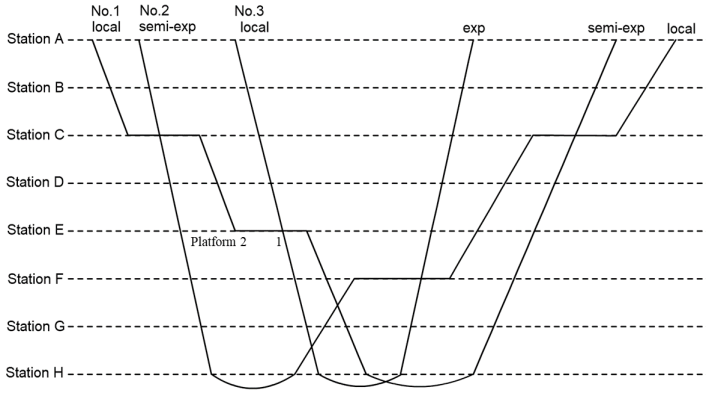


Fig. 7. A timetable used in the experiment

TABLE IV. MAGNIFICATION WEIGHT OF PASSENGERS' DISSATISFACTION

Dissatisfaction		Magnification weight	Threshold(s)
Delay itself		0.03	120
Stoppage	Stoppage delay	0.01	180
	Not dropping off	400 (penalty points/node)	-
Driving time		0.01	180
Frequency		0.05	210

For "not dropping off", 400 points are subtracted for each node where a train could not arrive after rescheduling. For the other criteria, we subtracted the product of magnification and delayed seconds for corresponding nodes. With these criteria, we conducted the learning procedure.

First, to make tasks easier and improve training speeds, we provided 150 typical patterns of delays for learning. We took two examples from every station and every interval between adjacent stations. Moreover, we classified the delay time into small (600 s), medium (1,200 s), and large (1,800 s); then we constituted a neural network to learn to each category. We used the package named "Keras-RL" for implementation of neural networks, training and testing.

The neural network comprises four layers: an input layer of 77 neurons, 100 hidden first neurons, 80 hidden second neurons, and 77 output neurons.

Regarding hyper parameters, we used epsilon greedy policy (ε=0.1), and the learning method was Adam, with a learning rate = 10^{-2}. The maximal number of steps for each episode was 20. Additionally, we provided difference scores before and after transformation as the reward every time. For each episode, if no score improvement till the $(n-5)^{th}$ step, we break it off. We did not want to omit patterns that show inefficient results halfway but out good in the end. However, because it is inefficient to conduct 20 steps per episode, we decided to give just five step reprieves.

We scored 150 delay patterns with four types of delay scales (300, 900, 1,500, and 2,100 s) as test cases to deal with unknown cases. As Fig. 8 shows, we input a delay in three neural networks, and the network adopted the best rescheduling plan.

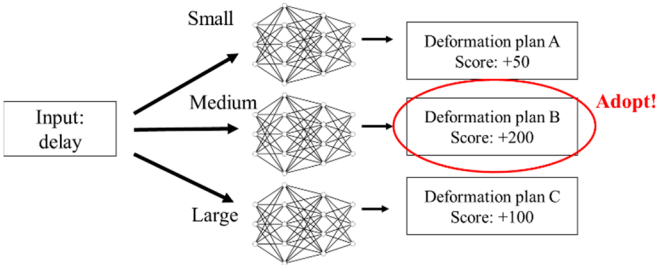


Fig. 8. Usage of three neural networks in test capability

B. Result

1) Overall results

Table 5 classifies the results of 600 delay patterns. Here, we count a delay between stations in the row of the preceding stations (i.e., a delay in outbound trains between A and B is included in the row of station A).

In 57% of the cases (342 out of 600), we obtained positive rescheduling, and for 249 patterns, the neural network did not carry out rescheduling. Neural networks were not able to rearrange nine patterns. For delays in outbound trains, the rate of positive rescheduling was 79.7 % (220 out of 276), whereas it was 37.6% (122 out of 324) for inbound trains. We next discuss one case to demonstrate how the neural network performs rescheduling.

2) An example

Owing to issues in outbound railroads between Stations D and E, outbound train No. 1 must wait 900 s for its departure. This delay affects train No. 3. The dotted line in Fig. 9 denotes the schedule without delay. If there is no rescheduling, trains 1 and 3 must wait until the issue is resolved and then run according to the schedule. In this case, the neural network minimizes the delay in the inbound train 2's arrival at Station A (the blue point in Fig. 9). Thus, the neural network reschedules as shown in Fig. 9.

First, it changes the order of arrival and departure of inbound trains 2 and 3 at Stations E and F. Thus, train 3 exceeds train 2 at Station E, not at Station F. Next, it changes the order of outbound trains 1 and 3 at Station D and E, resulting in train 3 exceeding train 1 at Station D, not at Station E. Finally, outbound train 1's arrival at Platform 2 changes to Platform 1 at Station D. Therefore, the score is improved from -1,290.73 (initial score) to -1,235.82 (1^{st} change), -1,140.42 (2^{nd} change), - 1,039.32 (3^{rd} change, final score), per change. Comparison between the initial score and the rescheduled one provides an overall improvement of +251.41. The improvement rate is 19.4%.

TABLE V. RESCHEDULING RESULT OF 600 TEST CASES

Outbound				Inbound			
Disrupted Station/Railway	Positive change	No change	Negative change	Disrupted Station/Railway	Positive change	No change	Negative change
A/A-B	18	6	0	A	0	12	0
B/B-C	40	8	0	B/B-A	18	30	0
C/C-D	40	8	0	C/C-B	0	48	0
D/D-E	40	8	0	D/D-C	31	8	9
E/E-G	42	6	0	E/E-D	16	32	0
				F/F-E	6	42	0
G/G-H	32	16	0	G/G-F	37	11	0
H	8	4	0	H/H-G	14	10	0
Subtotal	**220 (79.7%)**	**56 (20.3%)**	**0 (0.0%)**	**Subtotal**	**122 (37.6%)**	**193 (59.6%)**	**9 (2.8%)**
Overall							
Positive change	**342 (57%)**	**No change**	**249 (41.5%)**	**Negative change**		**9 (1.5%)**	

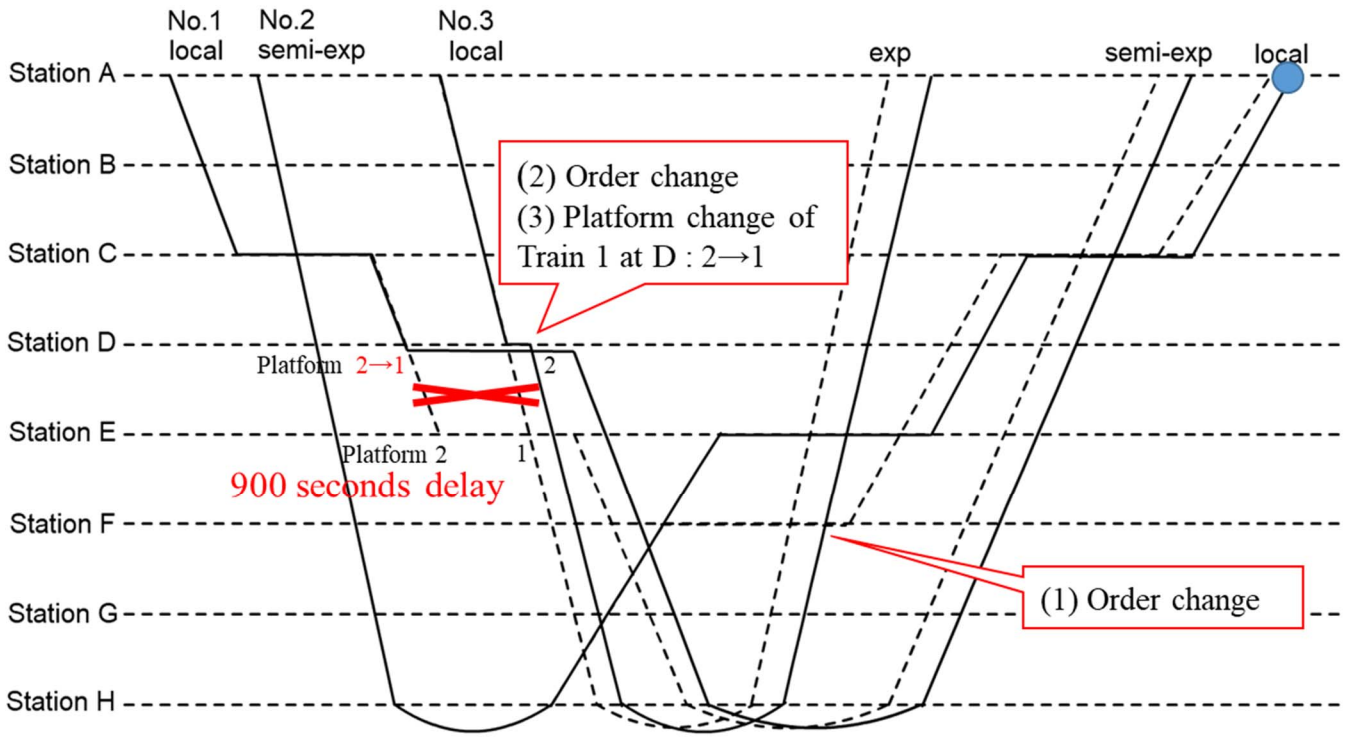


Fig. 9. Train rescheduling by the neural network in the B.2)'s example (outbound train No. 1 must wait 900 s for station D's departure.)

IV. Discussion

A. Consideration of the experiment

Neural networks can provide positive rescheduling in several cases. Although this experiment makes many simplifications, such as input and neural-network tasking, it is certain that neural networks provide effective solutions within this simplified range. It is especially effective against delays in outbound railways, giving an improvement of around 80%. However, for delays in inbound railways, we have many cases in which neural networks do not provide a solution resulting in score improvement. One of the reasons is that the number of rescheduling candidates increases the earlier a delay occurs.

If there is no score improvement, either the neural network has judged that no rescheduling is the optimal solution or learning is not sufficient to find an efficient solution.

For the former case, because actual timetables often have enough margins of time, if trains run fast within the stipulated safety speed, they can finally catch up; this serves as rescheduling. However, we cannot confirm this from the results we obtained. It is, however, possible that neural networks could try different patterns and conclude that no rescheduling is the optimal solution when there is little room for delay contagion or adjustment, and there are many cases with no rescheduling options for inbound delays.

Furthermore, in the example mentioned in Section III. B. 2, we obtained approximately 20% improvement in dissatisfaction. We conclude, therefore, that this is an effective rescheduling method. Moreover, the graph expression and delay simulation we proposed allow us to find weak points of a particular train schedule, implying that this technique can be used for making a robust time table.

B. Future work

1) Handling a large-scale network

The network we developed in this research is relatively small-scale and cannot deal with railways in the real world yet. Even if we limit the input to delays, space complexity is O(|V|). Thus, it would still be difficult to deal with a large train schedule. The network is likely to improve if we develop a higher-performance reinforcement learning technique; it also depends on the capacity of computers.

2) Relaxing simplified restrictions for more flexible deformations

When we developed the proposed model, we made some simplifications. Thus, we had to allow for some information omissions. Namely, we limited inputs to train delays on each node and confined the delay patterns and periods we made to learn the typical ones. We also admit that some parts of determining changes were executed by the greedy algorithm and not optimal in a strict sense. We made these restrictions to simplify the tasks for our neural network and to allow easy learning. If we relaxed such conditions, we could get better solutions while dealing with larger networks.

3) Defining dissatisfaction and magnification reflecting actual passengers.

We gave priority to the frequency when we defined the dissatisfaction and magnification for this research. However, if we define criteria by considering actual settings, we could improve the result further. For example, we could set magnification by tabulating actual passengers' demands. We can also consider a method by which we could simultaneously simulate the flow of passengers and evaluate how much passengers' dissatisfaction could be relaxed.

4) Developing criteria for evaluating the goodness of the deformation plan.

Although we obtained score improvements with our model, we were unable to analyze how efficient our deformation plan really is. We need, for example, to compare our model with other algorithms, including simulated annealing, to evaluate its performance. Additionally, if we could find a global optimized solution, we could compare it to our solution and provide a discussion about the local optimum.

5) Sophisticated modeling

The graph modeling we used is advantageous as we can consider train turning and connections. However, it does not deal with several rescheduling methods (e.g., changes of train types). It refers to the procedure of replacing an express train with a local one. Theoretically, it is possible to use an express train in place of a local train. However, this practice is tabooed in the Japanese railroad business. In practice, we may achieve this by changing running edges and stoppage edges. However, we must leave this discussion to future research.

V. Conclusion

We introduced a train rescheduling method based on DQN for trains running behind schedule. Using graph modeling, we obtained effective results for small-scale train delays. However, it continues to be difficult to deal with large railway delays, the model we developed could manage larger timetables, ensuring flexible train rescheduling as learning techniques of neural networks improves. Furthermore, by accommodating more sophisticated input, output, and modeling into neural networks, it will become possible to better reschedule train tables autonomously. The issue of automation of train rescheduling still has many important issues to be explored in future research. Our modeling, using graph theory and solution methods with reinforcement learning, offers an effective approach.

VI. ACKNOWLEDGMENT

We would like to especially thank Tokyu Corporation for providing information.

VII. References

[1] http://www.mlit.go.jp/tetudo/tetudo_fr8_000027.html

[2] P. Brucker, S. Heitmann, and S. Knust, "Scheduling railway traffic at a construction site," Contain. Terms. Autom. Transp. Sys., pp. 345–356, May 2001.

[3] K. Ghoseiri, F. Sszidarovsky, and M. J. Asgharpour, "A multi-objective train scheduling model and solution," Transp. Res. Pt. B, vol. 38, pp. 927–952, February 2004.

[4] S. Wegele and E. Schnieder, "Dispatching of train operations using genetic algorithms," Proc. 9th Intl. Conf. Computer-Aided Sched. Public Transp., pp. 1–9, January 2004.

[5] J. T. Krasemann, "Design of an effective algorithm for fast response to the re-scheduling of railway traffic during disturbances," Transp. Res. Pt. C, vol. 20, pp. 62–78, February 2012.

[6] F. Corman, A. D'Ariano, D. Pacciarelli, and M. Pranzo, "A tabu search algorithm for rerouting trains during rail operations," Transp. Res. Pt. B, vol. 44, pp. 175–192, January 2010.

[7] S. Dündar and İ. Şahin, "Train re-shceduling with genetic algorithms and artificial neural networks for single-track railways," Transp. Res. Pt. C, vol. 27, pp. 1–15, February 2013.

[8] B. Fan, C. Roberts, and P. Weston, "A comparison of algoritms for minimising delay costs in disturbed railway traffic scenarios," J. Rail Transp. Mgmt., vol. 2, pp. 23–33, November 2012.

[9] Y. Nagasaki, M. Eguchi, and T. Koseki, "Application of graph theory to scheduling of train operation," Tech. Metg. Transp. Electr. Railway, IEE Japan, pp. 17–22, June 2003.

[10] N. Tomii., Y. Tashiro, N. Tanabe, C. Hirai, and K. Muraki, "Train Rescheduling Algorithm Which Minimizes Passengers' Dissatisfaction," M. Ali and F. Esposito (eds), Innov. Appl. Artif. Intel., IEA/AIE 2005, Lecture Notes in Computer Science, vol. 3,533, pp. 829–838, June 2005.

[11] N. Lengm and U. Weidmann, "Discussions of the reschedule process of passengers, train operators and infrastructure managers in railway disruptions," 20th EURO Wkg. Gp. Transp. Mtg., September 2017.

[12] S. P. Fekete, A. Kröller, M. Lorek, and M. Pfetsch, "Disruption Management with Rescheduling of Trips and Vehicle Circulations," presented at 5th ASME/ASCE/IEEE Jt. Rail Coinf., January 2011.

[13] K. Sato, K. Tamura, and N. Tomii, "A MIP-based timetable resheduling formulation and algorithm minimizing further inconvenience to passengers," J. Rail Transp. Plan. Mgmt., November 2013.

[14] K. Imada and N. Tomii, "Recheduling Algorithm Based on MILP Formulation Considering Partial Cancellation and Turning Back," IEEJ Trans. Indust. Appl., vol. 137(6), pp. 484–491, February 2017.

[15] L. P. Veelenturf, L. G. Kroon, and G. Maróti, "Passenger oriented railway disruption management by adapting timetables and rolling stock schedules," Transp. Res. Pt. C, vol. 80, pp. 133–147, May 2017.

[16] R. S. Sutton and A. G. Barto, "Reinforcement learining: An introduction," MIT Press Cambridge, vol.1(1), March 1988.

[17] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez et al., "Mastering the game of Go without human knowledge," Nature, vol. 550, pp. 354, October 2017.

[18] B. Abdulhai, R. Pringle, and G. J. Kerakoulas, "Reinforcement Learning for True Adaptive Traffic Signal Control,"] J. Transp. Engin., vol. 129(3), pp. 278–285.

[19] X. Liang, X. Du, G. Wang, and Z. Han, "Deep Reinforcement Learning for Traffic Light Control in Vehicular Networks," in press.

[20] Z. Jiang, W. Fan, W. Liu, B. Zhu, and J. Gu, "Reinforcement learining approach for coordinated passenger inflow control of urban rail transit in peak hours," Transp. Res. Pt. C, vol. 88, pp. 1–16, March 2018.

[21] D. Šemrova, R. Marsetiča, M. Žuraa, L. Todorovskib, and A. Srdica, "Reinforcement learning approach for train reshceduling on a single-track railway," Transp. Res. Pt. B: Methodological, Vol. 86, pp. 250–267, April 2016.