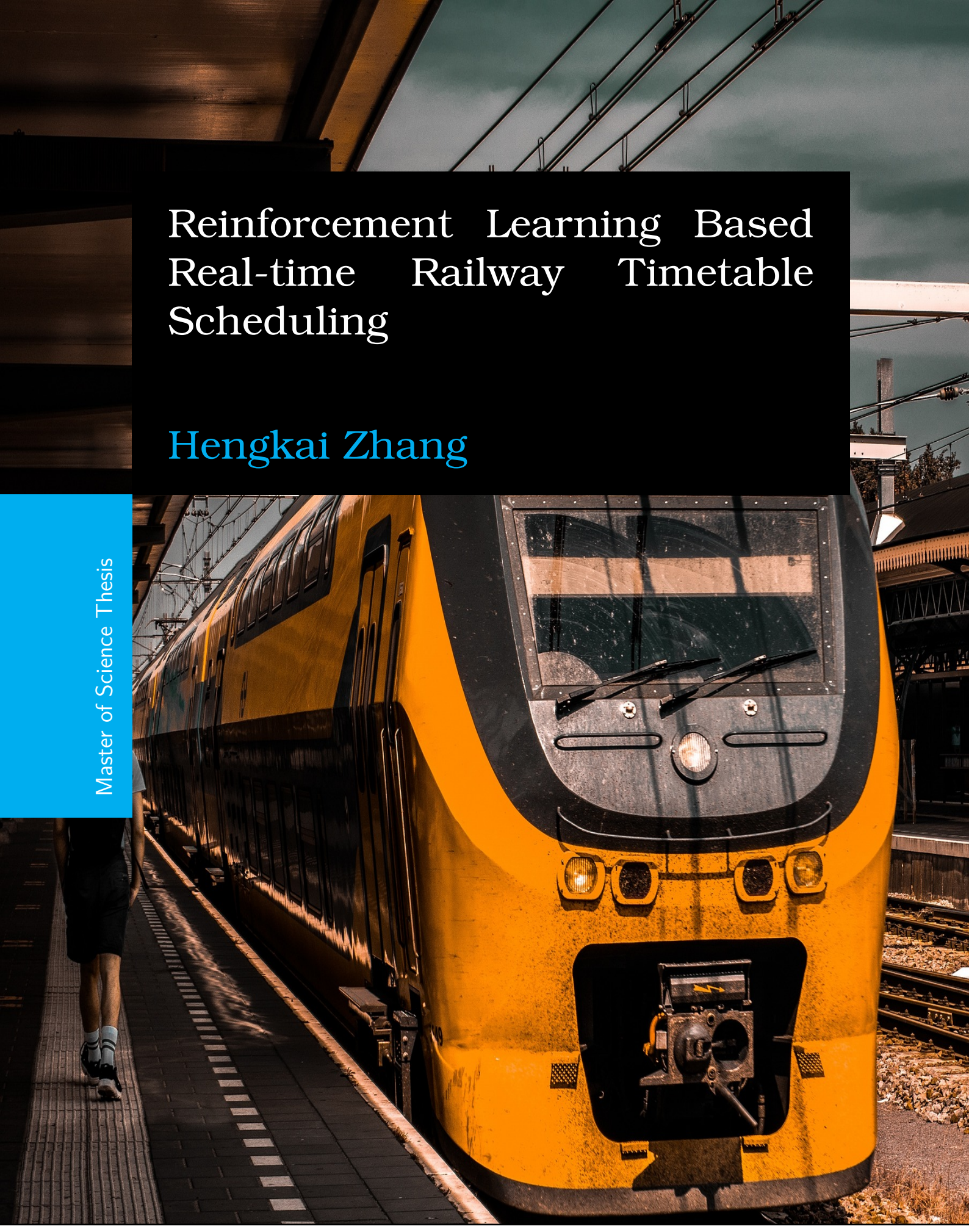

Reinforcement Learning Based Real-time Railway Timetable Scheduling

Hengkai Zhang

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Reinforcement Learning Based Real-time Railway Timetable Scheduling

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

Hengkai Zhang

June 28, 2023

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Abstract

The railway timetable rescheduling problem is a challenging problem in both industry and academia. It is required to calculate a feasible and relatively good timetable within a limited time to reduce the negative impact of disturbances or disruptions. The railway timetable rescheduling problem is typically formulated as a mixed integer linear programming (MILP) problem, which is difficult to solve due to the existence of the integer variables. To address this problem, many optimization-based studies have been conducted. The main advantage of using optimization-based methods is that they are easy to implement and more straightforward. However, the main disadvantage is that most optimization-based methods cannot reach the time requirements for large railway timetable rescheduling problems. There are also some researches using reinforcement learning techniques to solve this problem. By using reinforcement learning, the time requirement could be fulfilled.

In this thesis, an algorithm that combines both reinforcement learning and optimization approaches is proposed to solve the railway timetable rescheduling problem. In the beginning, the reinforcement learning environment is constructed from the railway timetable rescheduling problem. By selecting the independent integer variables as the action, the constraints involving the integer variables are satisfied. After that, a value-based reinforcement learning algorithm is implemented to determine the independent integer variables of the MILP problem. Then, the complete solution of the integer variables could be derived from these independent integer variables. With the solution of integer variables, the MILP problem could be transformed into a linear programming problem, which could be solved efficiently.

Several case studies are conducted in this thesis based on part of the Dutch railway network from Utrecht to 's-Hertogenbosch. The simulation results show that the proposed method makes a great improvement compared with the baseline regarding reducing the total delay of the system. Meanwhile, the reinforcement learning-based method also has an obvious advantage in terms of running time.

Table of Contents

Acknowledgements

I want to thank my supervisor Prof. Bart De Schutter for his assistance during this thesis project. The monthly meetings with him were always helpful and critical. During our discussion, his high-level research vision and patient guidance benefit me a lot.

I would like to thank my daily supervisor Xiaoyu Liu for his supervision and kind help. The discussions we had every week or even every three days were crucial to my thesis project. His many suggestions and ideas complemented this research. Meanwhile, his knowledge and experience on the research topic and scientific writing greatly helped me.

Also, I would like to thank Dingshan Sun and Caio Fabio Oliveira da Silva for helping me with this thesis project. Their extensive knowledge and practical experience in reinforcement learning bring me many inspirations.

I would want to express my gratitude to Yuxing Gao, Dong Shen, Sijun Zeng, and other friends I met here at TU Delft. Their friendship and help have always supported me.

Meanwhile, I would also like to thank my friends Yuting Xue, Bingnan Wang, Chongyang Song, Zijie Song, Xuan Wang, and Xinxin Zhang in China for helping me and trusting me for a long time.

Finally, I would like to express my deep love to my parents. Without their endless love and support, it is impossible for me to be here.

Delft University of Technology
June 28, 2023

Hengkai Zhang

Chapter 1

Introduction

1-1 Background

Railway plays an important role in the modern transportation system. Until 2022, 147 countries or regions in the world are operating railway transportation systems. In many countries, railway transport performs a lot of duties such as daily commuting, long-distance travel, and cargo transportation. In the Netherlands, the main train company Nederlandse Spoorwegen (NS) operated 623080 trips every day during 2021 according to its annual report (Nederlandse-Spoorwegen, 2021). However, it also had 4874 disruptions in 2021, which means about 13 times every day. There could be many reasons for railway disturbances and disruptions, such as extreme weather, worker strikes, and system failures. Railway disruptions normally require timetable rescheduling in real time to reduce initial delays and prevent further propagation of delays. This is the so-called railway timetable rescheduling problem (Luan et al., 2018). An efficient rescheduling algorithm should minimize the influence of disruption and improve the punctuality of trains. Therefore, developing effective rescheduling algorithms is critical to the operation of the railway system.

Nowadays, train dispatchers are responsible for timetable rescheduling. Their operations are normally made according to their experiences and skills with a certain level of computer assistance. In practice, most dispatching decisions are still made by human dispatchers, which generally results in sub-optimal decisions (Luan et al., 2020). During recent decades, with the development of computation hardware and relevant algorithms, many optimization approaches proposed in the literature are able to solve the rescheduling problem (Cacchiani et al., 2014). Nowadays, there exist many models for the railway timetable rescheduling problem. Most of them result in a mixed-integer linear programming (MILP) problem. For small-scale problems, most algorithms have excellent performance, and the optimal solution could be obtained within acceptable time and computation resources. For large-scale problems, the computation time to obtain the optimal solution is typically not acceptable. There is a trade-off between solution precision and time limitation. Meeting the solution time requirement will necessarily sacrifice some precision at the local level (Luan et al., 2020). Generally, the

computation complexity of the railway timetable rescheduling problem grows exponentially with the increase in problem size, which makes it difficult to implement in practice.

Compared with optimization-based approaches, well-trained machine learning approaches are always considered to be able to solve a problem in a limited time, regardless of the problem size. However, the main disadvantage of machine learning methods is that the optimality of solutions cannot be guaranteed and they are also less robust (Tang et al., 2022). Meanwhile, the constraint satisfaction is also a potential problem. The main reason is that the training dataset is usually a subset or part of the original problem. Same as many other machine learning applications, the test performance is always worse than the training result (Goodfellow et al., 2016). Also, the safety constraints are difficult to implement explicitly in the machine learning framework. In the last decade, the emergence of deep learning has greatly improved the applicability and performance of machine learning methods in many research areas. For the railway timetable rescheduling problem, implementations of deep learning approaches are also regarded as an efficient way to realize real-time timetable scheduling. Among various machine learning techniques, reinforcement learning is distinguished by its independence of pre-obtained training dataset and unique interaction with the environment. Following the success of AlphaGo, reinforcement learning has been applied to many different research fields, including railway timetable rescheduling problems (Šemrov et al., 2016; Zhu et al., 2020; Tang et al., 2022). The aim of this thesis is to combine reinforcement learning methods with optimization-based approaches to solve the railway traffic management problem in real time.

1-2 Problem Description

The railway timetable rescheduling problem is commonly formulated as an MILP problem. Compared with linear programming, the MILP problem usually takes much more time to solve due to the existence of integer variables. In this thesis, an algorithm that combines both reinforcement learning and optimization approaches will be proposed to solve this problem.

Specifically, a value-based reinforcement learning model will take the parameters from the MILP problem as the state and determine the integer solutions of the MILP problem as action. With these solutions, the MILP problem will become a linear programming problem, which could be solved efficiently. After obtaining the complete solution to the MILP problem, the railway model could be updated to the next time step. The entire procedure can be described in the following figure:

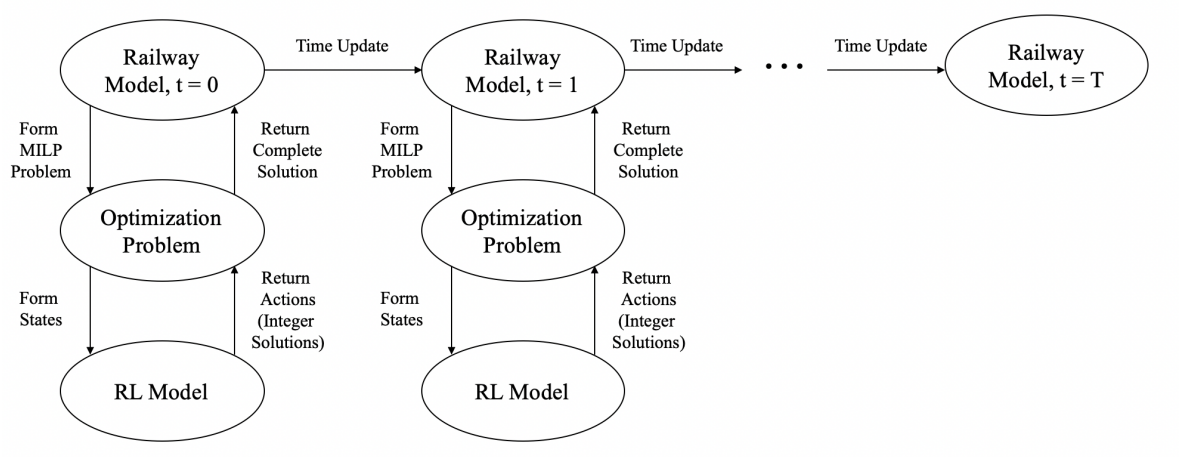

Figure 1-1: Research Problem Formulation

The main challenge of this thesis would be to train the reinforcement learning model, which can be divided into the following research questions:

- How to define a proper state representation
- How to design an efficient reward function
- How to find a suitable reinforcement learning framework

The expected contribution of this thesis can be summarized as follows:

- Develop a proper railway timetable rescheduling model and formulate the corresponding MILP problem
- Apply reinforcement learning algorithm to solve the integer variables of MILP problems that are inherently connected
- Propose a framework that combines both reinforcement learning and optimization techniques to solve the railway timetable rescheduling problem efficiently

1-3 Thesis Outline

The outline of this thesis is given as follows:

- Chapter 1 provides some background information on the main topic of this thesis. Then the research problem formulation and thesis outline are presented.
- Chapter 2 introduces the necessary background knowledge of this thesis, mainly containing two aspects: the railway timetable rescheduling problem and reinforcement learning.

- Chapter 3 thoroughly discussed the reinforcement learning-based solution to the railway timetable rescheduling problem proposed in this thesis.
- Chapter 4 provides three case studies conducted by this thesis and their experiment results.
- Chapter 5 concludes this thesis and makes an outlook for the future work.

Chapter 2

Background Knowledge

This chapter presents the necessary background knowledge for the thesis, organized into three sections. The first section provides an overview of the railway timetable rescheduling problem, including its modeling and solution methods. The second section provides a concise discussion of the fundamental concepts and principles of reinforcement learning. Lastly, the chapter concludes by summarizing the key points and emphasizing their relevance to the subsequent chapters' analyses and methodologies.

2-1 Railway Timetable Rescheduling

In this section, the basic knowledge of the railway timetable rescheduling problem is given. The first part focuses on the details of the general railway systems and some key concepts in the railway network. In the second part, several methods of modeling the railway timetable rescheduling problem are introduced. Finally, some solution methods for railway timetable rescheduling problem in current research are discussed.

2-1-1 Railway System Details

In many countries, the railway system is an important part of transportation. It has significant advantages in terms of capacity, speed, and applicability. Unlike other transport modes, a train can only run on a given track with strict limitations on the route, speed, and driver operations. A railway operation system includes several basic components: trains, stations, tracks, block sections, and signal systems. The introduction of these components is given as follows (Fang et al., 2015):

- *Train*: Different types of trains may run in the railway system simultaneously, such as long-distance passenger trains, short-distance commuter trains, light cargo and heavy cargo trains. They all have different operating properties and limits that need to be considered when making dispatching operations.

- *Station*: A station normally consists of multiple tracks and platforms that allow passengers or cargo to board and alight the train. Some stations may also have independent places to store and repair the train.
- *Track*: There are also different kinds of railway tracks in a railway system. For the direction, The track could be unidirectional or bi-directional. For the track number, it could be single, double or multiple. Different tracks may only allow certain kinds of trains to run.
- *Block Section*: A block section is defined between two block signals as a specific part of the track, in which only one train can run at the same time. A railway track could be regarded as a combination of multiple block sections.
- *Signal System*: The signal system of a railway system is used to control the traffic of the railway network and avoid any potential collision. The working mechanism of the signal system varies in different countries.

In addition to the components of the system described above, there are specific definitions of railway systems to describe the operating state and constraints of the system (Pachl, 2002):

- *Blocking Time*: The blocking time is a period in which a block section is exclusively assigned to a train. It starts from the moment that the signal system gives the authorization for the train to pass the section, and ends at the moment that the signal system becomes possible to assign this section to another train. Normally, the blocking time would be longer than the actual time that a train physically occupies the block section.
- *Safety Headway*: The safety headway time is the time interval between two consecutive trains. The minimum headway is exactly the blocking time.

In general, the railway system physically consists of trains, stations, tracks, and the signal system. Each railway track is artificially divided into block sections, and the signal system is designed to ensure that there is only one train in a block section at any given time. At the same time, the movement of trains is operated in each block section. Multiple consecutive block sections of the same train form the route of the train, and multiple routes of different trains form the operation of the railway network. In order to guarantee safety, multiple constraints are considered during railway scheduling, such as speed limits, blocking time, and safety headway.

2-1-2 Modeling of Railway Timetable Rescheduling

The railway timetable rescheduling problem refers to the task of dynamically adjusting the schedules of trains in a railway network in response to disturbances or disruptions. It involves rearranging the departure and arrival times of trains, as well as determining the optimal allocation of resources such as platforms and tracks. The goal is to minimize the impact of delays and disruptions, enhance system efficiency, and improve passenger satisfaction. Different studies may have their own problem settings and focus on different objectives. Before discussing the modeling methods of the railway timetable rescheduling problem, there are several concepts that need to be further introduced.

Disturbances and Disruptions In principle, a timetable should always be conflict free, which means that if everything goes well, there is no need for rescheduling. However, in practice, disturbances and disruptions are unavoidable. According to Cacchiani et al. (2014), disturbances refer to relatively small perturbations influencing the railway system, and disruptions are relatively large incidents that result in canceling some trips from the timetable. The key difference between these two concepts is whether there exists cancellation of trips. A disturbance is a fact that some railway operations take longer time than expected in the timetable. Common causes of disturbances include excessive crowds, slow loading, and unloading, unexpected medical or safety events, etc. A delay caused directly by the disturbance is called the primary delay. A primary delay may easily propagate from one train to another. In this case, the second train's delay is called the secondary delay. Compared with disturbances, disruptions usually cause large delays and cancellations of trips. Typical reasons for disruptions consist of extreme weather, worker strikes, and signal system failures. Similar to the delay propagation, the cancellation of certain trains may also cause more cancellations due to the lack of trains and manpower. In this thesis, only disturbances are considered for the railway timetable rescheduling problem.

Operator-centric and Passenger-centric The objective of the railway timetable rescheduling problem is to minimize the secondary delay caused by disturbances or disruptions. Depending on the different emphases, railway traffic management problems could be divided into two categories: operator-centric problems and passenger-centric problems. The operator-centric railway timetable rescheduling problems focus on minimizing the delay of trains or the number of canceled trains. While the passenger-centric railway timetable rescheduling problem focuses on minimizing the negative effects on passengers or freights. Apparently, the passenger-centric problem is more related to the satisfaction of passengers and also more complex than the operator-centric problem since it adds an additional factor. Currently, most studies consider the operator-centric problem. However, there is an increase of papers in the passenger-centric field (Cacchiani et al., 2014; Binder et al., 2017). Compared with the operator-centric problem, the main challenge of the passenger-centric railway timetable rescheduling problem is to accurately model the passenger flow, which is also called timetable-dependent passenger behavior (Zhu and Goverde, 2019). In this thesis, the passenger-centric railway timetable rescheduling is studied. However, a simplification of the passenger flow is applied to simplify the problem.

Rescheduling Decisions When a disturbance or disruption happens, the timetable has to be rescheduled. Compared with timetable planning, rescheduling must be done in a short time period. Normally, the decision is expected within minutes. When a disturbance occurs, only the timetable itself needs to be adjusted. However, when a disruption happens, the railway traffic manager has to consider more issues, such as the type and number of different rolling stocks, the deployment of crews and their working qualifications, and also the possible maintenance. The common decisions for rescheduling include:

- *Retiming*: Retiming refers to the change of departure and arrival times of the train in a block section. Since the departure time from a station cannot be earlier than the

scheduled time, retiming usually causes the delay of trains. The retiming decision may not only consider the situation of the current block section but also the availability of the following block sections on the route.

- *Rerouting*: Rerouting means changing the assigned route of the train. In general, rerouting can be divided into two categories: local and global. Local rerouting is performed by using parallel block sections during the trip. It will have the same destination and normally no delay. Global rerouting usually considers a totally different route in the railway network. It may also add some extra stops or stations. Global rerouting usually results in certain delays.
- *Reordering*: Reordering changes the passing order of trains on their common block sections. The main reason for reordering is related to the delay of the front train. By reordering, the successive train may be able to run on time. In this way, the secondary delay could be effectively reduced. In fact, reordering could be regarded as a special kind of retiming.
- *Cancellation*: Normally cancellation is only used when a disruption happens. A certain number of trips will not depart from the station.

The following figure gives an illustration of these rescheduling approaches:

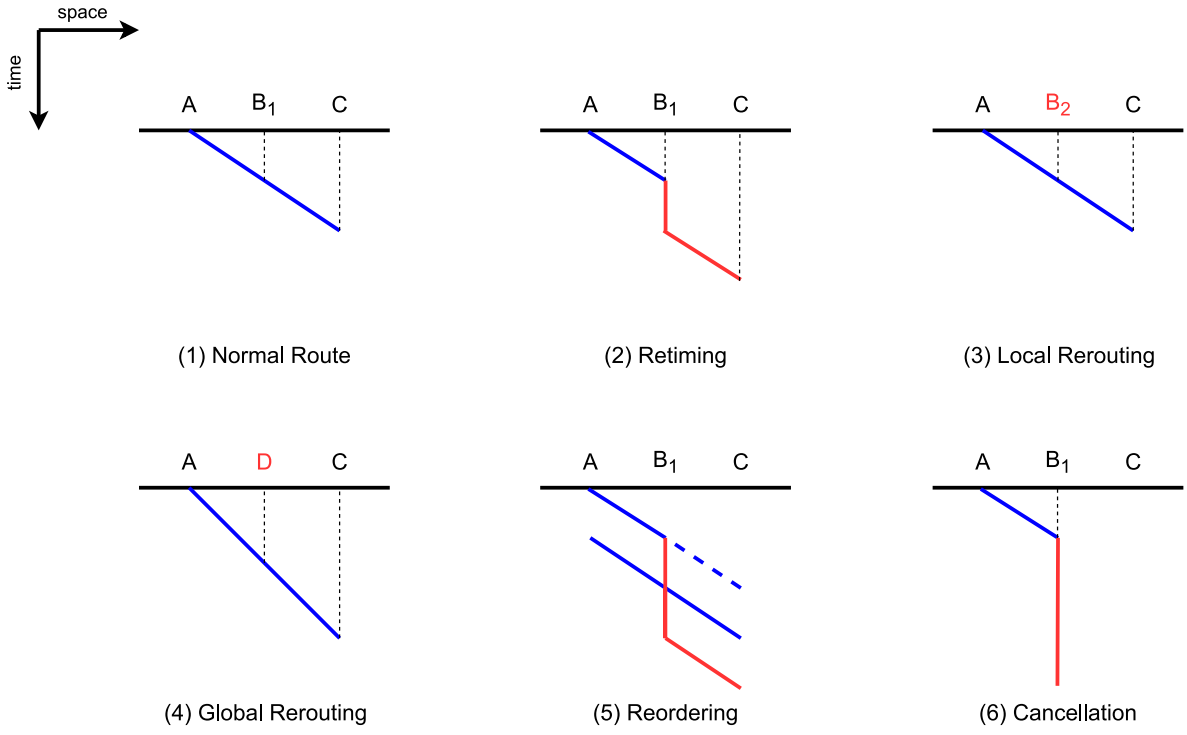

Figure 2-1: Common Rescheduling Decisions

In this thesis, only retiming and reordering are considered for the rescheduling.

In order to solve the railway timetable rescheduling problem in an effective way, an accurate

mathematical model is required. Currently, there are several different problem formulations for the railway timetable rescheduling problem, such as mixed logical dynamical (MLD) systems, alternative graph (AG), fuzzy petri net (FPN) and expert system (ES), discrete event model and simulation model, etc (Fang et al., 2015). Among all these models, MLD, and AG are the most used formulations to describe the rescheduling problem in the railway network. With the same problem setting, different modeling methods do not change the nature or the complexity of the railway management problem. These modeling approaches are mainly used to formulate the problem in an easily understandable form. In fact, most railway timetable rescheduling problems will be formulated as an integer programming (IP) problem or a mixed-integer linear programming (MILP) problem (D'Ariano et al., 2007; Corman et al., 2011; Luan et al., 2018). In this thesis, the railway traffic management problem will be formalized as a mixed logical dynamical system and solved by mixed-integer linear programming (MILP). Therefore, a brief introduction to MILP is given as follows.

The MILP problem is a kind of optimization problem that involves both integer-valued and real-valued parameters (Schrijver, 1998). It could be written as:

$$\begin{aligned} \min_{x} \quad & \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \\ \text{s.t.} \quad & A\boldsymbol{x} \leq \boldsymbol{b} \\ \text{where} \quad & \boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_{\mathrm{r}} \\ \boldsymbol{x}_{\mathrm{i}} \end{bmatrix} \\ & \boldsymbol{x}_{\mathrm{r}} \in \mathbb{R}^{n_{\mathrm{r}}}, \boldsymbol{x}_{\mathrm{i}} \in \mathbb{R}^{n_{\mathrm{i}}} \end{aligned} \tag{2-1}$$

A basic solution approach for the MILP problem is the branch-and-bound algorithm. In each node, the MILP is relaxed to a linear programming problem by relaxing the integer values to be continuous. One may also use heuristic search techniques, such as random search, genetic algorithms, simulated annealing, etc, to solve the MILP problem. Nowadays, there also exist commercial solvers to solve the MILP problem. Most commercial solvers are able to solve small-scale problems efficiently. However, they typically suffer from computational complexity issues when encountering large-scale problems. The fundamental reason for that is that the MILP problem is regarded as an NP-hard problem.

The key to transforming the railway timetable rescheduling problem into MILP is to properly define binary and continuous variables. Depending on different focuses, problem settings, and application backgrounds, the definition of variables varies among existing studies. A comprehensive survey given by Fang et al. (2015) offers a very detailed summary to compare these choices of decision variables and also the number of constraints.

For the binary decision variables, one of the commonly-used definitions is to indicate whether or not a block section or route is assigned or occupied to a train (Törnquist and Persson, 2007; Min et al., 2011; Pellegrini et al., 2014). This kind of binary variable is used to indicate the status of the tracks and stations in the railway network. Another type of binary variable represents the order of trains in certain situations. In van den Boom et al. (2011), the binary variables indicate the order of two trains. In Acuna-Agost et al. (2011), the binary variable compares whether a given train passes before another train or not. There are also other definitions of the binary variable, but the common intention of all these studies is to accurately

and effectively represent the status of the railway network under certain problem settings. The definitions of binary variables in this thesis will be discussed in Chapter 3.

For the continuous decision variables, the most common choice is the operation time of trains, such as the departure and arrival times of a train with respect to a block section or a station (Min et al., 2011; Mu and Dessouky, 2011). Many studies also include the delay of a train or an event as the continuous variable (Acuna-Agost et al., 2011; Pellegrini et al., 2014). Another choice is related to the time differences, such as the difference between the original arrival time and actual arrival time (Min et al., 2011) and the time difference between the end of an event for a train (Törnquist and Persson, 2005). In general, most continuous variables focus on the time property of the railway network.

As mentioned above, the objective of the railway timetable rescheduling problem depends on the type of the problem. In an MILP formulation, the objective function could be easily defined as the sum of delays of all trains or all passengers' delays. The block section rules, safety headways, and station capacities are often considered as different constraints by most studies. The passenger-centric problem may also include some constraints related to passenger satisfaction. In practice, most studies will simplify the railway timetable rescheduling problem according to their own needs.

2-1-3 Solution Methods for Railway Timetable Rescheduling

As discussed in the previous part, the key to solving the railway timetable rescheduling problem is to solve the corresponding MILP problem. There are two main threads of solving the MILP problem: one is the branch and bound algorithm, which is widely used by commercial solvers; the other is using the heuristic, which is actually a sub-optimal method and has various algorithms. Both methods have their own advantages and disadvantages. In the remainder of this part, these two solving methods will be described in detail.

Basic Methods for Railway Timetable Rescheduling

For the railway timetable rescheduling problem, the branch and bound algorithm is widely used to solve the AG-based MILP problem. D'Ariano et al. (2007) used a truncated branch and bound algorithm, which includes two implication rules: First-Come-First-Served (FCFS) and First-Leave-First-Served (FLFS) to speed up the computation. Corman et al. (2010) combined the branch and bound algorithm with a heuristic tabu search to minimize the maximum consecutive delay on the Utrecht Den Bosch railway. Specifically, the branch and bound algorithm is used for reordering and tabu search is used for rerouting. Another paper by Corman et al. (2011) considered different priorities of running traffic within the branch and bound framework. In Kecman et al. (2013), the impact of different detail levels is investigated with respect to the solution quality and computational efficiency.

As mentioned in the previous part, there exist several commercial solvers that are able to solve small-scale MILP problems efficiently. However, these solvers usually become very slow when dealing with large-scale problems. For the railway traffic management problem, the most commonly used solver is CPLEX developed by IBM (Fang et al., 2015). The CPLEX

solver uses the branch and cut algorithm to optimize the mixed integer programming problem (Cplex, 2021), which is considered as a general and robust method. In practice, some studies also implement the branch and cut algorithms by themselves to solve the railway rescheduling problem (Lamorgese et al., 2016). Addressing the railway timetable rescheduling problem with commercial solvers has several advantages. First, it is easy to implement and tune. Nowadays, most commercial solvers could be implemented with different programming languages. Second, using commercial solvers makes researchers focus more on how to derive a representative MILP problem instead of how to solve the MILP problem quickly. The latter question should be considered by optimization research. Third, commercial solvers provide a benchmark that allows the results of studies to be compared with each other.

Unlike self-implementation of branch and bound algorithms, which are mostly used for solving AG-based MILP problems, commercial solvers are commonly implemented for MILP problems based on different models (Luan et al., 2018). Törnquist and Persson (2007) directly formulated the railway timetable rescheduling problem as a MILP problem, whose objective is to minimize the consequences of a single disturbance. The CPLEX solver was used in this paper with different rescheduling strategies, such as track change or order modification. Dollevoet et al. (2012) proposed an event-activity-based model to describe the railway timetable rescheduling problem by rerouting passengers. It combined the CPLEX solver with a modified Dijkstra's algorithm to solve the MILP problem. In Pellegrini et al. (2014), the railway traffic management problem is directly formulated as an MILP problem, where the infrastructure is represented with fine granularity. The paper studied both simple junctions and complex networks. Similarly, the CPLEX solver is used to solve the MILP problem. In Xu et al. (2017), a MILP formulation based on the alternative graph is proposed to reschedule the timetable after a disruption in the high-speed railway system with a quasi-moving signaling system.

In summary, the basic approach to solving the MILP-based railway traffic management problem is using the branch and bound algorithm. Some studies implement the algorithm by themselves, while others use commercial solvers that are also based on the branch and bound algorithm. The main advantage of this method is that it is easy to implement and could also be used as a benchmark. The main disadvantage is that when the problem size increases, the computation time becomes unacceptable.

Heuristics for Railway Timetable Rescheduling

A heuristic algorithm or simply heuristic is defined as a suboptimal technique that is able to find relatively good solutions with acceptable computation time (Gendreau et al., 2010). There is no guarantee of optimality or even feasibility when using heuristic methods. In many cases, they are also not able to indicate how close to optimality a particular feasible solution is. Most heuristics first generate candidate solutions and then use a simulation to evaluate the criterion function and the feasibility. Some commonly used heuristic algorithms include random search (Mladenović and Hansen, 1997), tabu search (Glover and Laguna, 1997), genetic algorithm (Davis, 1991), greedy algorithm, and ant colony (Dorigo et al., 2006).

Heuristic algorithms are broadly applied to solve the MILP-based railway timetable reschedul-

ing problem due to the feature that this kind of algorithm could obtain relatively good results in a short time. In fact, some simple rule-based heuristics are also widely used (Luan et al., 2018), such as First-Come-First-Served (FCFS), First-Leave-First-Served (FLFS), First-Scheduled-First-Served (FSFS), First-Rescheduled-First-Served (FRFS). In D'Ariano et al. (2007), researchers solved the AG-based MILP problem by combining the branch and bound algorithm with FCFS and FLFS.

In addition to the rule-based heuristic algorithms mentioned above, other types of heuristics have also been applied to solve the railway timetable rescheduling problem. Some studies only use heuristics to solve the problem. For instance, in Mu and Dessouky (2011), a freight train scheduling problem was formulated as an MILP problem, which was solved by the greedy algorithm and neighborhood search algorithm. In Krasemann (2012), a greedy heuristic was developed to further improve the computation efficiency, using the same problem formulation as in Törnquist and Persson (2007). Meanwhile, some studies also designed heuristics for the railway timetable rescheduling problem. In Pellegrini et al. (2015), a special heuristic method called RECIFE-MILP was developed based on the MILP formulation given in Pellegrini et al. (2014).

Some studies focus on combining numerical approaches with heuristics. In the beginning, heuristic algorithms are implemented to obtain a relatively good result in a short time. After that, the optimization method will find the final optimum. In Corman et al. (2010), the tabu search and the branch and bound algorithm are alternatively used to solve the railway rescheduling problem. The final result saved more than 80% computation time and also have 15% better results. In Corman et al. (2011), the priority rule-based heuristic was added to the branch and bound algorithm, which considered multi classes of trains. Corman et al. (2012) tried to achieve two objectives simultaneously: reducing the delay and maintaining as many connections as possible by combining the branch and bound with Pareto frontier technique. In Šama et al. (2016), the ant colony was applied to select the best train routes from all candidates.

Compared with the two approaches mentioned previously, heuristics are more complicated to implement. However, they can always obtain a near-optimal result within a limited time, when dealing with large-scale problems. This important feature makes them widely studied. In fact, since the MILP problem is typically regarded as an NP-hard problem, heuristic approaches are the most commonly used algorithms to solve the MILP-based railway traffic management problem (Fang et al., 2015). Also, it is noticed that most studies combine the heuristic methods with numerical approaches, which results in good performance (D'Ariano et al., 2007; Corman et al., 2010; Šama et al., 2016).

2-2 Reinforcement Learning

In this section, the general introduction to the definition of reinforcement learning and some key concepts will be first given. Then a more detailed description of value-based methods will be provided. After that, the last part will summarize the current applications of reinforcement learning in railway timetable rescheduling problems.

2-2-1 Basic Knowledge of Reinforcement Learning

The early history of reinforcement learning has two main threads. One focused on learning by trial and error while the other concerned the optimal control problem and dynamical programming (Sutton and Barto, 2018). In the 1990s, the first wave of reinforcement learning raised and combined two main threads above into modern reinforcement learning techniques. A large number of classical algorithms were developed in this period, including Q-Learning by Watkins and Dayan (1992), REINFORCE by Williams (1992), and SARSA by Rummery and Niranjan (1994).

Over the past decade, reinforcement learning has once again gained tremendous attention from both academia and industry. With the growth of hardware and computing power, deep reinforcement learning has become the hot spot of this second wave. During this period, some remarkable algorithms have been proposed, like Deep Q-Network (DQN) by Mnih et al. (2015), DDPG by Lillicrap et al. (2015), and A3C by Mnih et al. (2016).

In this part, the definition of reinforcement learning and some key concepts will be given first, which will be followed by an introduction to the value function and Bellman equation. Finally, a brief classification of existing reinforcement learning algorithms will be provided.

Definition of Reinforcement Learning

According to the widely accepted description in Sutton and Barto (2018), reinforcement learning refers to a problem that an agent learns how to map states to actions, so as to maximize a numerical reward signal. The reinforcement learning problem is always described by a Markov Decision Process (MDP), which means that the next state will only be determined by the current state and current action. Specifically, in each time step, the agent first measures the states of the environment and then takes the corresponding action according to its current policy. Depending on different settings, the reward signal may be given at each time step or at the end of the game. The learning target of the agent is to learn an appropriate strategy so that it can maximize the final reward. Here, some formal definitions of these key elements are given according to Sutton and Barto (2018) as follows:

- *State* is the set of information regarding the environment that the agent can get at every time step
- *Action* is the behavior that the agent takes according to its policy when it senses current states
- *Policy* is a mapping from perceived states of the environment to actions to be taken when in those states
- *Reward* is a signal that implicitly specifies the goal of the task

Value Function and Bellman Equation

The value function is an important concept in almost all reinforcement learning algorithms. There are two kinds of value functions: the value function of states and the value function of

state-action pairs. Both of them are trying to estimate the expected final rewards from the current situation. Specifically, the value function of states measures how good the given state is for the agent, while the value function of state-action pairs measures how good it is to take the given action in this given state. Normally, value functions are defined with respect to a particular policy. For a reinforcement problem defined by an MDP, the value of state λ under policy π could be defined as:

$$v_\pi(\lambda) = \mathbb{E}_\pi[G_t \mid \Lambda_t = \lambda] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \tau^k \Psi_{t+k+1} \mid \Lambda_t = \lambda\right] \tag{2-2}$$

where $\mathbb{E}_\pi[\cdot]$ denotes the expectation of a random variable given policy π, G_t is the cumulative future reward at t, Ψ_t and Λ_t are the reward and state at t, respectively. The function $v_\pi(\cdot)$ is called the state value function of policy π.

Similarly, the value of taking action ω in state λ under policy π could be written as:

$$q_\pi(\lambda, \omega) = \mathbb{E}_\pi[G_t \mid \Lambda_t = \lambda, \Omega_t = \omega] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \tau^k \Psi_{t+k+1} \mid \Lambda_t = \lambda, \Omega_t = \omega\right] \tag{2-3}$$

where Ω_t is the action at t, and τ is the discount factor. The function $q_\pi(\cdot, \cdot)$ is called the action value function of policy π.

From these two definitions, there exists a substitution for both value functions. For the state value function:

$$v_\pi(\lambda) = \sum_{\omega} \pi(\omega|\lambda) q_\pi(\lambda, \omega) \tag{2-4}$$

where $\pi(\omega|\lambda)$ represents the probability of taking action ω in state λ under policy π, and clearly $\sum_\omega \pi(\omega|\lambda) = 1$. This equation represents that the value of state λ is the expectation of all feasible actions' value in this state and under current policy.

Similarly, the action value function could be written as:

$$q_\pi(\lambda, \omega) = \sum_{\lambda', \psi} p(\lambda', \psi|\lambda, \omega) \left[\psi + \tau v_\pi(\lambda')\right] \tag{2-5}$$

where $p(\lambda', \psi|\lambda, \omega)$ represents the probability of transitioning to λ' with reward ψ, from state λ and taking action ω, also $\sum_{\lambda', \psi} p(\lambda', \psi|\lambda, \omega) = 1$. Here the reward ψ of current time step depends on the next state λ'. This equation represents that the value of taking action ω in state λ is the expectation of all possible next states' value and transition rewards under current policy.

Further more, the recursive relationship of state value function is given by substituting equation (2-5) into (2-4):

$$\begin{aligned} v_\pi(\lambda) &= \sum_{\omega} \pi(\omega|\lambda) q_\pi(\lambda, \omega) \\ &= \sum_{\omega} \pi(\omega|\lambda) \sum_{\lambda', \psi} p(\lambda', \psi|\lambda, \omega) \left[\psi + \tau v_\pi(\lambda')\right] \end{aligned} \tag{2-6}$$

Equation (2-6) is called *Bellman equation for* v_π, which shows that the value of a state is equal to the discounted expectation of next state's value, plus the expected reward along the transition. The action value function could also be calculated recursively by substituting (2-4) into (2-5):

$$\begin{aligned} q_\pi(\lambda, \omega) &= \sum_{\lambda', \psi} p(\lambda', \psi | \lambda, \omega) \left[\psi + \tau v_\pi(\lambda')\right] \\ &= \sum_{\lambda', \psi} p(\lambda', \psi | \lambda, \omega) \left[\psi + \tau \sum_{\omega'} \pi(\omega' | \lambda') q_\pi(\lambda', \omega')\right] \end{aligned} \tag{2-7}$$

Equation (2-7) is called *Bellman equation for* q_π. These two Bellman equations build the recursive relationship of state values and state-action values. Almost all reinforcement learning algorithms use this recursive property to estimate the values and determine the action.

2-2-2 Value-based Methods

One of the most important branching points of a reinforcement learning algorithm lies in its ability to access information about the model. Here, the word "model" refers to the dynamic model of the environment. Normally, model-based algorithms are expected to have better performance than model-free algorithms, since they have extra information about the environment. Algorithms that do not use a model of the environment are called model-free. Although some sample efficiency is lost, this kind of algorithm is usually easier to implement and tune, which also makes it a more popular choice compared with model-based algorithms. For the model-free algorithms, there are two different things to learn: one is the action value function (Q-function), and the other is the policy itself. Here, a non-exhaustive but useful taxonomy is presented to highlight some fundamental differences among reinforcement learning algorithms (Achiam, 2018).

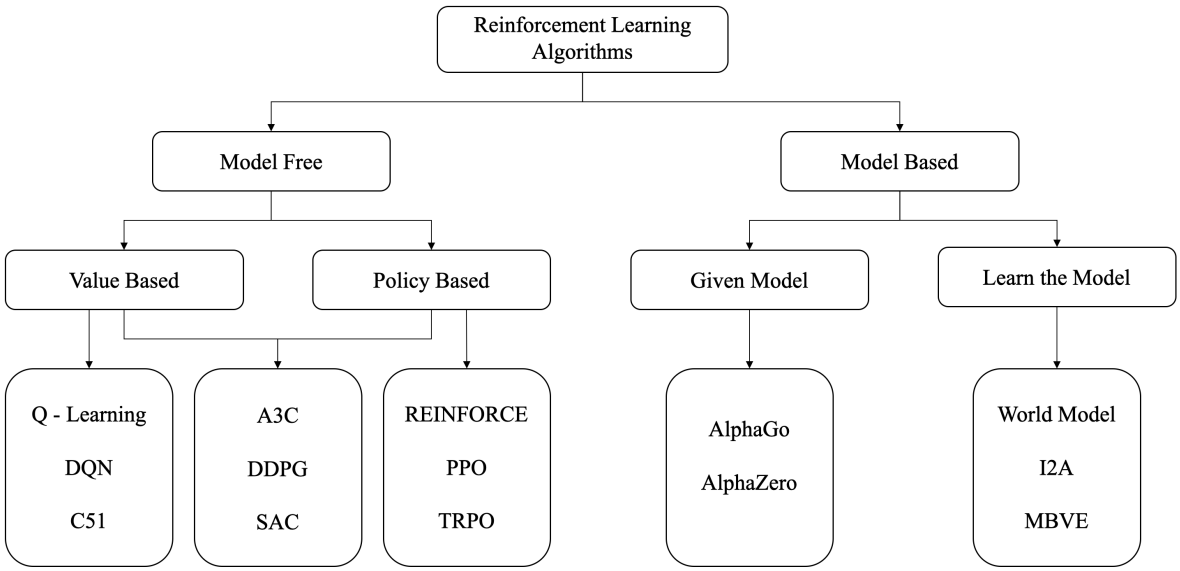

Figure 2-2: A Taxonomy of Reinforcement Learning Algorithms

For value-based algorithms, as mentioned in the last part, the action value function measures how good to take given action in the current state. Therefore, after learning the action value function, the agent would be able to determine the value of all actions and then simply choose the action with the highest value to execute. The key idea to this kind of algorithm is fitting the action value function given in (2-3). Value-based algorithms gain the advantage of sampling efficiency since during the update, they can use all the data collected before. However, compared with policy-based methods, there are still two main limitations: first is that most value-based methods can only deal with discrete and relatively small action space since they need to calculate all possible actions' values at each time step. Large or continuous would bring high calculation requirements for value-based algorithms. Another disadvantage is that value-based methods are always considered less "direct" on the optimization target, which may cause unstable performance. The second section of this chapter will provide a more detailed description of value-based methods. In this thesis project, the action space will be discrete. Therefore, the thesis is going to mainly consider value-based reinforcement algorithms. Here, this part will give a brief introduction to the Q-Learning and Deep Q-Network (DQN) algorithms, which are two basic algorithms of all value-based methods in principle.

Q-Learning

Proposed by Watkins and Dayan (1992), Q-Learning is regarded as one of the most important breakthrough of reinforcement learning and also the starting point of value-based methods. The basic one-step Q-Learning is defined as follows:

$$Q(\Lambda_t, \Omega_t) \leftarrow Q(\Lambda_t, \Omega_t) + \zeta \left[\Psi_{t+1} + \tau \max_{\omega} Q(\Lambda_{t+1}, \omega) - Q(\Lambda_t, \Omega_t) \right] \tag{2-8}$$

where ζ is the learning rate. In this case, the learned action value function Q directly approximates the optimal action value function despite the policy. It can be proved that Q will converge to the optimal action value function with probability 100% under the assumption that the value of all state-action pairs continues to be updated (Sutton and Barto, 2018). The *Q-Learning* algorithm is shown as follows:

Algorithm 1 Q-Learning

Initialize $Q(\lambda, \omega)$
Repeat:
  Initialize Λ
  Repeat:
    Choose Ω from Λ using policy derived from Q (e.g. ϵ-greedy)
    Take action Ω, observe Ψ, Λ'
    $Q(\Lambda_t, \Omega_t) \leftarrow Q(\Lambda_t, \Omega_t) + \zeta \left[\Psi_{t+1} + \tau \max_{\omega} Q(\Lambda_{t+1}, \omega) - Q(\Lambda_t, \Omega_t) \right]$
    $\Lambda \leftarrow \Lambda'$
  until Λ is terminal
until End of the episode

The Q-Learning algorithm is widely used for reinforcement problems due to its small requirement of computation power. It has great performance on small-scale problems. However, for

problems with large state or action space, it may take quite long time to converge since some state-action pairs may be difficult to explore and update.

Deep Q-Network (DQN)

In order to solve the updating problem for Q-Learning, the DQN algorithm was proposed in 2015 (Mnih et al., 2015). The key idea of DQN is to use a deep convolutional neural network to approximate the following optimal action-value function:

$$Q^*(\lambda, \omega) = \max_{\pi} \mathbb{E}\left[\Psi_t + \tau\Psi_{t+1} + \tau^2\Psi_{t+2} + \ldots | \Lambda_t = \lambda, \Omega_t = \omega, \pi\right] \tag{2-9}$$

In the original paper (Mnih et al., 2015), the neural network model consists of a data preprocessing layer ϕ, three convolutional layers, followed by two fully connected layers and a single output layer for all actions. Each hidden layer is followed by a rectifier non-linearity layer. The main reason for designing such a neural network model is that the input to Atari 2600 is image information. The structure of the neural network could be redesigned according to the need of different problems. In this thesis, since the input does not contain image information, the neural network will not use the convolutional layer. Besides the special neural network structure, there are two main ideas that benefit the convergence of the reinforcement learning model: experience replay and the target network.

Specifically, experience replay builds a dataset $E_t = \{e_1, \ldots, e_t\}$ to store the agent's experience $e_t = (\Lambda_t, \Omega_t, \Psi_t, \Lambda_{t+1})$ at each time step. By using the experience replay, there are several advantages for convergence compared with normal Q-Learning methods. First, data efficiency is improved, since each data point can be used for many weight updates. Second, randomizing samples breaks the correlation between consecutive samples. Third, experience replay will make the behavior distribution more averaged and more smoothing. Notably, this uniform sampling gives equal importance to every sample, one may also design other sampling strategies to emphasize some particular data points.

Another modification is a separate target network $\hat{Q}$ that is used to generate targets y_j in the Q-learning update. Every C updates, the target network $\hat{Q}$ will be replaced by the current network Q. In this way, the stability of the algorithm is further improved since the correlation between action values and target values is reduced. The detailed algorithm is given as follows:

Algorithm 2 Deep Q-Learning with Experience Replay

Initialize replay memory E with capacity N
Initialize action value function Q with random weights η
Initialize target action value function $\hat{Q}$ with weights $\eta^- = \eta$
for episode$= 1, M$ **do**
  Initialize state sequence $\Lambda_1 = \langle x_1 \rangle$ and preprocessed sequence $\phi_1 = \phi(\Lambda_1)$
  for $t = 1, T$ **do**
    With probability ϵ select and action Ω_t
    otherwise select $\Omega_t = \arg\max_\omega Q(\phi(\Lambda_t), \omega; \theta)$
    Execute action Ω_t and observe reward Ψ_t and image x_{t+1}
    Set $\Lambda_{t+1} = \Lambda_t, \Omega_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(\Lambda_{t+1})$
    Store transition $(\phi_t, \Omega_t, \Psi_t, \phi_{t+1})$ in E
    Sample random minibatch of transitions $(\phi_t, A_t, R_t, \phi_{t+1})$ from E
    Set $y_t = \begin{cases} \psi_j, & \text{if episode terminates at step } j+1 \\ \psi_j + \tau \max_{\omega'} \hat{Q}(\phi_{j+1}, \omega'; \eta^-), & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $(y_t - Q(\phi_j, \omega_j; \eta))^2$ with respect to the network parameters η
    Every C steps reset $\hat{Q} = Q$
  end for
end for

Comparing the algorithm of DQN with the standard Q-Learning, there are three main improvements. First, DQN uses a convolutional neural network to represent the action value function (Q function). Second, an experience replay pool is introduced to improve data efficiency. Third, a separate target network is used to generate targets during training, which breaks the correlation between targets and action values. By using these techniques, the convergence and performance are significantly improved. The DQN method also indicates that the same algorithm, network architecture, and hyperparameters can successfully learn the control policy of many different tasks through extensive experiments in Atari 2600 games.

2-2-3 Application of Reinforcement Learning in Railway Timetable Rescheduling

This part will introduce several studies that use reinforcement learning to solve the railway timetable rescheduling problem. As discussed in the first section, solving the railway timetable rescheduling problem by optimization methods has been extensively developed. However, only a few studies have addressed this problem using reinforcement learning techniques. Most literature considers a traffic controller as the agent that directly takes actions to control the railway traffic system, which is regarded as the environment. The choice of states and actions could be quite different according to the different focus and setting of the research. The following table lists the state, action, and algorithm of current research.

From Table 2-1, it is noticed that these studies' selections of state and action are rather diverse. These variations are related to their different focuses and problem settings. For instance, in Ghasempour and Heydecker (2019), only one junction is considered, so trains waiting for passing through becomes the state of the environment and the sequence of trains

Table 2-1: Comparison of State, Action and Algorithm of RL-based Railway Timetable Rescheduling

Paper	State	Action	Algorithm
Šemrov et al. (2016)	availability of all tracks, locations of all trains and current system time	signals along the track	Q-Learning
Ning et al. (2019)	actual departure and arrival times of all trains	the departure sequence of trains	DQN
Ghasempour and Heydecker (2019)	trains waiting for passing through the junction	sequence of trains passing through a junction	Approximate dynamic programming (ADP)
Khadilkar (2019)	availability of a few resources close to the train	train movements	Q-Learning
Zhu et al. (2020)	current delay of the event, resources congestion level and the resource to be occupied	implement the event or wait	Q-Learning

is the action. While in Khadilkar (2019), the agent only operates one particular train at one time step, using the availability of close resources as the state. From the experiment result in Zhu et al. (2020), it is shown that larger state space leads to better solution quality but also slower convergence. Meanwhile, including some crucial information may bring significant improvement to the final performance. The experiment result shows that broader representativity of state choice is even more important than a larger state space (Zhu et al., 2020).

Most existing research chooses to use value-based, model-free reinforcement learning algorithms, like Q-Learning and DQN. Only Ghasempour and Heydecker (2019) implements the approximate dynamic programming (ADP), which could be regarded as a model-based reinforcement learning method. The main reason for using value-based reinforcement learning algorithms, especially Q-Learning, is that they are easy to implement and tune. Also, the state space and action space of railway timetable rescheduling problems considered in these studies are both discrete and relatively small.

The major limitation of all research mentioned above is that only microscopic problem is considered in their case studies. Specifically, the existing studies only concentrate on the individual railway line instead of a railway network. In Šemrov et al. (2016) and Ning et al. (2019), only a single-track railway line is considered. While Zhu et al. (2020) focuses on a double-track railway line and Khadilkar (2019) studies both single and double-track railway lines. While paper Ghasempour and Heydecker (2019) concentrates on one isolated junction. An improvement is made in Ghasempour et al. (2019) to operate multiple independent junctions simultaneously, that have no traffic coordination. Understandably, as using reinforcement learning to solve the railway timetable rescheduling problem is still in its early stages, most of the available results have been generated from smaller scale experiments. However, as the size of the problem grows, it will become more difficult to implement a value-based algorithm due to the large computation requirement. Future work may consider solving networked macroscopic problems.

In addition to this common limitation, these studies also have some individual problems. In Ning et al. (2019), the state representation is closely related to the training railway structure, which means that the agent cannot be used in other railway networks. Research in Khadilkar (2019) makes an assumption that a segment can only be occupied by one train at a time, which is not applicable in real-world scenarios. In Zhu et al. (2020), one open track is allowed to be occupied simultaneously by several trains as long as the safety requirements are fulfilled.

Overall, until now there have been several studies using reinforcement learning methods to solve railway timetable rescheduling problems. Most of these studies have implemented value-based algorithms on small-scale problems, but they still have many limitations and drawbacks. Compared with the widely studied optimization methods, reinforcement learning-based methods still have many directions to be explored and extended. Examples of possible future work include applying reinforcement learning methods on larger-scale railway network problems, the selection and effective representation of state and action, and the possible application of policy-based reinforcement algorithms.

2-3 Conclusions

The railway timetable rescheduling problem is a real-time rescheduling problem, which is usually caused by a disturbance or a disruption. To accurately model the railway timetable rescheduling problem, a railway way model must be given first. The railway system physically consists of several kinds of components. The proper operation of it depends on many constraints in both space and time. The movement of a train is defined on a specific block section at a certain time instant. Some consecutive block sections form the route of a train. Many routes together establish the railway network.

In fact, the railway timetable rescheduling problem could be modeled with different approaches. Most of these modeling techniques typically result in an MILP problem. To solve the MILP problem, one may choose the branch and bound algorithm or heuristic methods. There also exist several commercial solvers that are able to optimize the MILP problem. Depending on different focuses, the objective of the problem is also different. There also exist different rescheduling techniques applied in time or space.

For the MILP-based railway traffic management problem, the basic solution is using the branch and bound algorithm (B&B). Some studies implement the algorithm by themselves, while others use commercial solvers. The common advantage of these two approaches is that they are relatively easy to implement and could be used as a benchmark. By using B&B, researchers could focus more on the problem formulation instead of solving the problem. At the same time, the main disadvantage of these two methods is that they both suffer from the issues of computation time when dealing with large-scale problems. Heuristic is a kind of suboptimal algorithm that could find relatively good solutions within a limited time. This property makes it suitable for solving large-scale railway rescheduling problems. However, the disadvantage is also obvious. In most cases, heuristic methods cannot reach the global optimum of the problem. Currently, both directions have been widely studied, more researchers tend to combine two kinds of algorithms to obtain better results. In summary, most of the existing studies are unable to balance the requirements in terms of optimality and computation time. For this reason, a method that combines reinforcement learning with optimization techniques will be proposed in this thesis project.

Reinforcement learning is a special kind of machine learning. It refers to a problem that an agent learns how to map states to actions, so as to maximize a numerical reward signal. By introducing the concepts of the state value function and action value function, the importance of different states and actions is represented. Meanwhile, depending on different learning methodologies, the model-free reinforcement learning algorithms could be divided into value-based and policy-based algorithms. In this thesis, only value-based reinforcement learning methods are considered.

Compared with the traditional optimization-based methods, addressing the railway timetable rescheduling problem with reinforcement learning is still in a very early stage. Most studies are still using relatively simple value-based algorithms, such as Q-Learning and DQN to solve small scale problems with certain limitations and constraints. Future research could be conducted from both algorithm and problem scale.

Chapter 3

Reinforcement Learning-based Railway Timetable Rescheduling

In this chapter, the reinforcement learning-based solution method will be proposed to solve the railway timetable rescheduling problem. As discussed in Chapter 2, the railway timetable rescheduling problem is usually formulated as a mixed-integer linear programming (MILP) problem by existing research. In the first section of this chapter, the formulation of the MILP problem will be given. After that, the second section will first introduce how to formulate the railway timetable rescheduling and MILP problem together as the environment of the reinforcement learning model. Then the reinforcement learning algorithm used in this thesis will be presented. In the third section, the complete reinforcement learning-based solution approaches to the railway timetable rescheduling problem will be presented. In the end, a short conclusion will summarize this chapter.

3-1 Formulation of MILP Problem

In this section, the railway timetable rescheduling problem will be formulated as an MILP problem. As discussed in Chapter 2, the MILP problem is a special kind of optimization problem, some of whose variables are constrained to be integer or binary. Depending on different problem settings and objectives, the choice of both integer and continuous variables of the railway timetable rescheduling problem is various. The selection of these variables in this thesis will be presented first. After that, the introduction of the objective function and constraints will be given.

3-1-1 Variables

According to different focuses and problem setting of the railway timetable rescheduling problem, the definition of variables will naturally be various (Fang et al., 2015). In this thesis, the main objective of the railway timetable rescheduling problem is to minimize the total

delay for all passengers. The rescheduling decisions are limited to reordering and retiming. Therefore, the continuous variables are related to the operational times of the train, and the integer variables are given as the order between different trains. Detailed definitions are given as follows:

Continuous Variables

In a railway network, there are several different times regarding the operation of trains. In this thesis, they are defined as continuous variables as follows:

- $a_{i,s}$: the arrival time of train i at station s
- $d_{i,s}$: the departure time of train i from station s
- $\gamma_{i,s}$: the dwelling time of train i in station s
- $r_{i,u,s}$: the running time of train i between station u and s

These time-related continuous variables have different properties. The arrival time and departure time are points of time while the running time and dwelling time are periods of time. The constraints between them will be further introduced in the third part of this section. In this thesis, the unit of time is minutes.

Integer Variables

As mentioned above, another rescheduling measure used in this thesis is reordering. The order between different trains can be represented as the integer variable. First, the departure order is defined as follows:

$$\delta_{i,j,s} = \begin{cases} 1, & \text{if } d_{i,s} - d_{j,s} \geq 0 \\ 0, & \text{otherwise} , \end{cases} \tag{3-1}$$

where $\delta_{i,j,s}$ represents the departure order between train i and train j at station s. If $\delta_{i,j,s} = 1$, it means that train i departs later than j. Otherwise if $\delta_{i,j,s} = 0$, train j departs later than train i. Then, the arrival order could also be defined as:

$$\alpha_{i,j,s} = \begin{cases} 1, & \text{if } a_{i,s} - a_{j,s} \geq 0 \\ 0, & \text{otherwise} , \end{cases} \tag{3-2}$$

where $\alpha_{i,j,s}$ represents the arrival order between train i and train j at station s. If $\alpha_{i,j,s} = 1$, it means that train i arrives later than j. Otherwise if $\alpha_{i,j,s} = 0$, train j arrives later than train i.

3-1-2 Objective Function

In this thesis, the objective is to minimize delays for all passengers at all stations. The objective function of the railway timetable rescheduling problem is given as follows:

$$\min \quad \sum_{s \in \mathcal{S}} \sum_{i \in \mathcal{T}} p_{i,s}(a_{i,s} - A_{i,s}) \ , \tag{3-3}$$

where $\mathcal{S}$ and $\mathcal{T}$ are sets of all stations and all trains respectively. Since the passenger delay is considered, it is necessary to multiply the passenger number $p_{i,s}$ with the delay of the corresponding train and station. Here, $p_{i,s}$ represents the number of passengers on train i with the destination of station s. The $A_{i,s}$ represents the ideal arrival time of train i at station s from the pre-defined timetable.

3-1-3 Constraints

In order to formulate the railway timetable rescheduling problem into the MILP problem, there are several kinds of constraints that need to be considered. Some typical constraints are related to the operation of the railway system, the safety requirements between trains, the availability of platforms, tracks, and trains, the limitation of train speed, etc. In this thesis, we mainly consider constraints of train operation, safety headways, and the availability of platforms in all stations.

Train Operation Constraints

For a train i in the railway system, the train operation should satisfy the following two basic constraints:

$$d_{i,s} = a_{i,s} + \gamma_{i,s} \ , \tag{3-4}$$

$$a_{i,s} = d_{i,s_i^-} + r_{i,s_i^-,s} \ , \tag{3-5}$$

Here, equation (3-4) represents that the departure time of the train i at the station s is the sum of its arrival time and dwelling time at this station. This constraint actually describes the operation of train i within the station s.

In equation (3-5), s_i^- represents the preceding station of station s on the route of train i. This constraint described the operation of train i between two stations s_i^- and s. Specifically, it means that the arrival time of train i at the station s is the sum of its departure time in the last station and the running time between them. Both (3-4) and (3-5) should be satisfied by all trains at all stations. These two constraints form the basic operation of the railway system.

Besides the constraints above, there are also some lower bounds regarding the continuous decision variables. For the departure time $d_{i,s}$ and the arrival time $a_{i,s}$, the train cannot be earlier than the original time from the timetable. The running time $r_{i,u,s}$ and dwelling time $\gamma_{i,s}$ could be shorter than the original time indicated on the timetable, but they also need to have a minimum value to guarantee the operation safety. These lower bounds are given by inequalities as follows:

$$d_{i,s} \geq D_{i,s} \tag{3-6}$$

$$a_{i,s} \geq A_{i,s} \tag{3-7}$$

$$r_{i,u,s} \geq R_{i,u,s}^{\min} \tag{3-8}$$

$$\gamma_{i,s} \geq \Gamma_{i,s}^{\min} \quad , \tag{3-9}$$

where $D_{i,s}$ and $A_{i,s}$ represent the ideal departure time and arrival time of train i at station s from the timetable, respectively. $R^{\min}_{i,u,s}$ is the minimum running time of train i between station u and s. By applying constraint (3-8), an upper bound for the train speed is added. Also, $\Gamma^{\min}_{i,s}$ represents the minimum dwelling time of train i at station s. Constrain (3-9) requires a minimal operation time within the station.

Safety Headway Constraints

In the railway timetable rescheduling problem, safety headway constraints require that there must be a certain distance or time slot between two trains on the same track to guarantee the safe operation of the railway system. However, it is not necessary to consider the constraint of safety distance between any two trains. Suppose that two trains are not operating on the same platform or on the same track, then there is no need to consider the safety headway between them. Without losing generality, some assumptions are made in this thesis:

- One platform can only be occupied by one train at a time.
- The connection between platforms and tracks is fixed.
- All tracks and platforms are unidirectional.

Then, in order to investigate the headway constraint, some binary parameters need to be defined. First, the parameter indicating the departure situation is defined as follows:

$$\beta_{i,j,s} = \begin{cases} 1, & \text{if train } i \text{ and train } j \text{ depart from the same platform or to the same track at station } s \\ 0, & \text{otherwise ,} \end{cases} \tag{3-10}$$

From the definition above, when $\beta_{i,j,s}$ is 0, there is no need to consider the departure order of train i and train j at station s, since they will not run on the same track nor from the same platform. In other words, the departure safety headway constraint between train i and j at station s will only need to be considered when $\beta_{i,j,s} = 1$. An illustration figure of $\beta_{i,j,s}$ is given as follows:

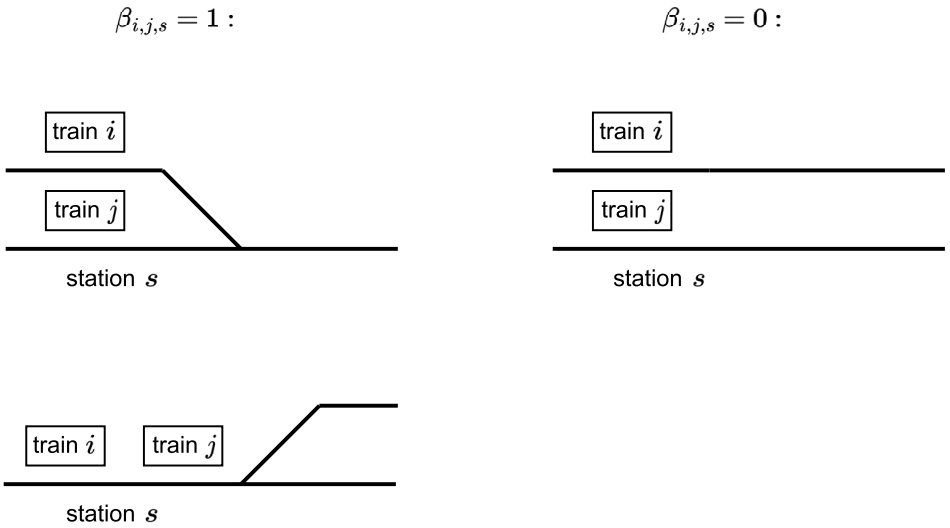

Figure 3-1: Illustration Figure of the Departure Situation Parameter $\beta_{i,j,s}$

Similarly, the parameter indicating the arrival situation is defined as follows:

$$\theta_{i,j,s} = \begin{cases} 1, & \text{if train } i \text{ and train } j \text{ arrive from the same track or to the same platform at station } s \\ 0, & \text{otherwise ,} \end{cases} \tag{3-11}$$

Similar to $\beta_{i,j,s}$, the definition of $\theta_{i,j,s}$ also indicates that the arrival safety headway constraint only needs to be considered when $\theta_{i,j,s} = 1$. An illustration figure of $\theta_{i,j,s}$ is given as follows:

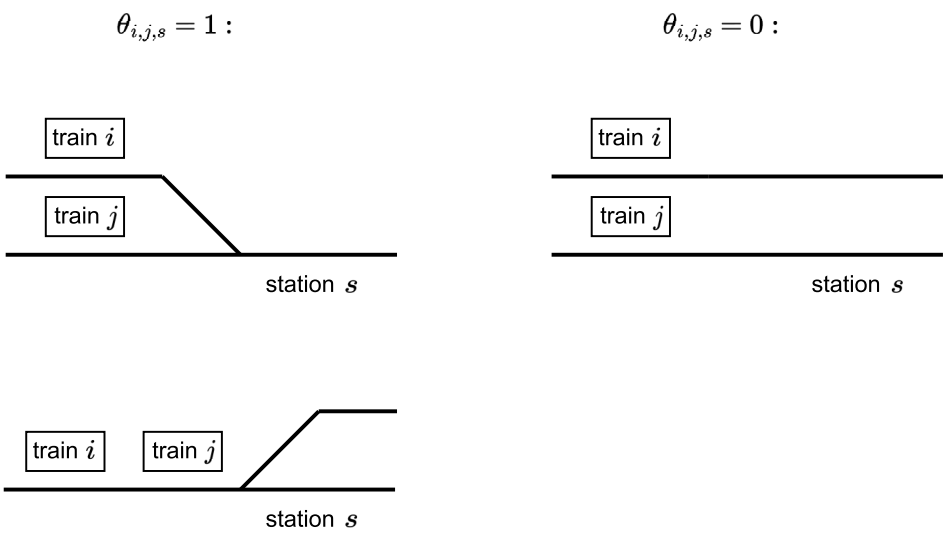

Figure 3-2: Illustration Figure of the Arrival Situation Parameter $\theta_{i,j,s}$

Given definitions of the departure order in (3-1) and the departure situation parameter in (3-10), the departure safety headway constraint between any trains i and j at station s could be written as follows:

$$d_{i,s} - d_{j,s} \geq h - M(2 - \beta_{i,j,s} - \delta_{i,j,s}) \ , \tag{3-12}$$

where h is a time constant, representing the required minimum safety headway in the railway system, and M is a large positive constant.

It is noticed that when $\beta_{i,j,s} = 0$, constraint (3-12) will hold automatically due to the large positive constant M, i.e. when train i and train j do not depart from the same platform or to the same track at station s, there is no need to consider the departure safety headway between train i and train j.

Only when $\beta_{i,j,s} = 1$, does the departure headway between train i and train j need to be considered. Then, if $\delta_{i,j,s} = 0$, constraint (3-12) will also hold automatically. According to the definition of the departure order in (3-1), $\delta_{i,j,s} = 0$ represents that train i departs earlier than train j. In this situation, constraint (3-12) still does not guarantee the safety headway.

When $\beta_{i,j,s} = 1$ and $\delta_{i,j,s} = 1$, train i departs later than train j, and these two trains depart from the same platform or to the same track. Then, the constraint (3-12) become:

$$d_{i,s} - d_{j,s} \geq h \tag{3-13}$$

In such case, this constraint requires that the departure time of train i should be h minutes later than the departure time of train j at station s. From the analysis above, it is noticed that constraint (3-12) could only demand the safety headway when the train i departs later than train j. In other words, both departure order $\delta_{i,j,s}$ and $\delta_{j,i,s}$ need to be determined in the MILP problem. Naturally, it requires the following constraint:

$$\delta_{i,j,s} + \delta_{j,i,s} = 1 \tag{3-14}$$

Constraint (3-14) requires that there must be a departure order between train i and train j from station s. Similar to constraints (3-12) and (3-14), the arrival headway constraints are defined as follows:

$$a_{i,s} - a_{j,s} \geq h - M(2 - \theta_{i,j,s} - \alpha_{i,j,s}) \tag{3-15}$$

$$\alpha_{i,j,s} + \alpha_{j,i,s} = 1 \tag{3-16}$$

Similar to the analysis of departure safety headway above, only when $\theta_{i,j,s} = 1$ and $\alpha_{i,j,s} = 1$, will constraint (3-15) guarantee the arrival safety headway.

Platform Flexibility and Station Connection Constraints

As mentioned before, in this thesis, the rescheduling decisions considered are reordering and retiming. On the same track, it is impossible for the train behind to overtake the train in front. On different tracks, reordering is achieved automatically by retiming without influencing other trains' operations. Therefore, only the reordering happened within the station requires determining the binary variables. However, not all stations can achieve reordering. For example, suppose a station s only has one platform, then it is obvious that the departure order of trains from this station must be the same as their arrival order. Clearly, the flexibility of doing reordering is very related to the number of platforms in the station. Therefore, the number of platforms in station s is defined as a positive integer parameter ξ_s. After that, the constraint of platform flexibility could be written as:

$$\sum_{j \in \mathcal{T}(s) \setminus \{i\}} (\alpha_{i,j,s} - \delta_{i,j,s}) \leq \xi_s - 1 \ , \tag{3-17}$$

where $\mathcal{T}(s)$ is the set of all trains that pass the station s. From the definitions of departure order $\delta_{i,j,s}$ in (3-1) and arrival order $\alpha_{i,j,s}$ in (3-2), $\alpha_{i,j,s} = 1$ means that train j arrives earlier than train i at station s. Therefore, the number of trains that arrive earlier than train i at station s could be calculated as:

$$n_\alpha = \sum_{j \in \mathcal{T}(s) \setminus \{i\}} \alpha_{i,j,s} \tag{3-18}$$

Similarly, the number of trains that depart earlier than train i at station s could be calculated as:

$$n_\delta \sum_{j \in \mathcal{T}(s) \setminus \{i\}} \delta_{i,j,s} \tag{3-19}$$

Therefore, the left part of the constraint (3-17) represents the number of trains that train i overtakes at station s. Since there are only ξ_s platforms in station s, and train i also need to

take 1 platform to operate, the maximal number of trains that could be overtaken by train i at station s is $\xi_s - 1$.

In fact, constraint (3-17) reflects the relationship between arrival orders and departure orders within a station. Naturally, it is also necessary to consider the relationship between arrival orders and departure orders between two connected stations. Using the definitions of departure and arrival parameters in (3-10) and (3-11), the constraint of train orders between two connected stations is written as:

$$\beta_{i,j,s_i^-}\theta_{i,j,s}\alpha_{i,j,s} = \beta_{i,j,s_i^-}\theta_{i,j,s}\delta_{i,j,s_i^-} \tag{3-20}$$

From the constraint above, it can be easily known that if any of $\beta_{i,j,s_i^-} = 0$ or $\theta_{i,j,s} = 0$, the constraint will always hold. This property indicates that if train i and train j depart to different tracks at station s_i^- or arrive from different tracks to station s, there are no restrictions between their departure order and arrival order. Conversely, if and only if train i and train j use the same track to travel from station s_i^- to station s, their arrival order $\alpha_{i,j,s}$ must be as same as their departure order δ_{i,j,s_i^-}.

Now, all constraints considered in this thesis are presented. Combining them with the objective function in (3-3), the railway timetable rescheduling problem could be given as:

$$\begin{aligned}
\min \quad & \sum_{s\in\mathcal{S}}\sum_{i\in\mathcal{T}} p_{i,s}(a_{i,s} - A_{i,s}) \\
\text{s.t.} \quad & d_{i,s} = a_{i,s} + \gamma_{i,s} \\
& a_{i,s} = d_{i,s_i^-} + r_{i,s_i^-,s} \\
& d_{i,s} \geq D_{i,s} \\
& a_{i,s} \geq A_{i,s} \\
& r_{i,u,s} \geq R^m_{i,u,s} \\
& \gamma_{i,s} \geq \Gamma^m_{i,s} \\
& d_{i,s} - d_{j,s} \geq h - M(2 - \beta_{i,j,s} - \delta_{i,j,s}) \\
& \delta_{i,j,s} + \delta_{j,i,s} = 1 \\
& a_{i,s} - a_{j,s} \geq h - M(2 - \theta_{i,j,s} - \alpha_{i,j,s}) \\
& \alpha_{i,j,s} + \alpha_{j,i,s} = 1 \\
& \sum_{j\in\mathcal{T}(s)\setminus\{i\}} (\alpha_{i,j,s} - \delta_{i,j,s}) \leq \xi_s - 1 \\
& \beta_{i,j,s_i^-}\theta_{i,j,s}\alpha_{i,j,s} = \beta_{i,j,s_i^-}\theta_{i,j,s}\delta_{i,j,s_i^-}
\end{aligned} \tag{3-21}$$

which is an MILP problem. Suppose that all trains will pass all stations, then the largest possible numbers of variables and constraints are summarized in the following table:

Table 3-1: Largest Possible Numbers of Variables and Constraints

Variables or constraints	Number
Continuous Variables	$4\|\mathcal{S}\|(\|\mathcal{T}\|-1)$
Binary Variables	$2\|\mathcal{T}\|(\|\mathcal{T}\|-1)(\|\mathcal{S}\|-1)$
Train Operation Constraints	$6\|\mathcal{S}\|(\|\mathcal{T}\|-1)$
Safety Headway Constraints	$3\|\mathcal{T}\|(\|\mathcal{T}\|-1)(\|\mathcal{S}\|-1)$
Platform Flexibility Constraints	$\|\mathcal{T}\|(\|\mathcal{S}\|-1)$
Train Orders Constraints between Two Connected Stations	$\|\mathcal{T}\|(\|\mathcal{T}\|-1)(\|\mathcal{S}\|-1)$

3-2 Reinforcement Learning Model

In this thesis, as described in the problem overview, the environment is a combination of the railway system and the resulting MILP problem. In the first part of this section, the environment setup will be introduced. Specifically, there are three main components of the reinforcement learning environment: state, action, and reward. The selection of state needs to consider the availability and validity of environmental information together (Zhu et al., 2020). However, the selection of actions should take into account the need of the environment while reducing its dimension as much as possible for training. For the reward function, the design of the reward function should be able to reflect the objective directly. Different reward functions may have significant influence on the performance of agent (Sutton and Barto, 2018). In the second part of this section, the reinforcement learning algorithm used in this thesis and its concrete implementation will be introduced.

3-2-1 Environment Settings

From the figure of the problem overview, it can be noticed that in this thesis, the reinforcement learning environment has two main parts: the railway system and its resulting optimization problem, which is an MILP problem as formulated in the last section. Specifically, the railway model is responsible for formulating the MILP problem and making the transformation to the next step after receiving the complete solution to the MILP problem. While the MILP problem forms states to the reinforcement learning agent and takes its action as the integer solutions, so as to transform the MILP problem into a linear programming problem, which could be solved efficiently. In the following parts of this section, the details of the environment will be discussed, including the update of the environment, the selection of state representation and action, and the reward function.

Railway System Update

In this thesis, the railway system is designed as an event-triggered, time-based system, whose each time step is mathematically described by the MILP problem outlined in the previous section. In practice, trains are operated with a pre-defined, conflict-free timetable. Ideally, if there is no disturbance or disruption at all, trains are able to run smoothly according to the timetable. However, disturbances and even disruptions are inevitable in every railway

system. When a disturbance or disruption happens, rescheduling measures are required to reduce the negative impact of disturbances and disruptions. This thesis focuses specifically on studying disturbances and does not address severe disruptions.

As discussed in the previous section, the rescheduling decisions in this thesis focus on reordering and retiming. In the context of the MILP problem, reordering is accomplished through binary variables, which represent the departure and arrival orders in equation (3-1) and (3-2). On the other hand, retiming is achieved using continuous variables. It is widely known that the MILP problem is much more challenging to solve than addressing linear programming problems due to the involvement of integer variables.

Suppose a station only has one platform, which means that reordering is impossible. In such cases, the MILP problem is simplified to a linear programming problem and has no need to be solved using the reinforcement learning model. Therefore, the starting point of the railway timetable rescheduling problem in this thesis is that a disturbance happened in the railway system and it requires reordering in a station. Since the system is active only when such an event occurs, it is described as an event-triggered system. For a train i arriving at station s, this event becomes the initial event of the railway timetable rescheduling problem if and only if the following condition holds true:

$$(r_{i,s} + a_{i,s} \geq R_{i,s} + A_{i,s}) \wedge (\xi_s > 1) \tag{3-22}$$

The condition above makes two requirements: one is that train i is delayed when it arrives at the station s, and the other is that station s has more than 1 platform, which makes it possible for reordering. Then the initial delay μ_0 of the railway timetable rescheduling problem is given as:

$$\mu_0 = r_{i,s} + a_{i,s} - R_{i,s} - A_{i,s} \tag{3-23}$$

After the initial delay happened, a crucial question is to determine the duration of the railway timetable rescheduling problem. Given that most railway timetables follow a periodic pattern, this question could also be described as determining the number of trains to be considered in the timetable rescheduling process. The total number of trains included in the railway timetable rescheduling problem cannot be too little, otherwise it may not be enough to eliminate the impact of delays. In this thesis, since only disturbances are considered, the number of trains is chosen to be large enough to eliminate the influence of the delay.

After the total time is determined, the next question is to decide the time of each step when updating the railway network. Including all trains in one step may yield two problems: the first is that there may be subsequent delays following the initial delay, causing the previously obtained solution suboptimal. Second, the complete railway timetable rescheduling problem will also formulate a larger MILP problem, which is difficult to solve. Therefore, dividing the complete railway timetable rescheduling problem into multiple consecutive steps is necessary. If the number of trains considered in one step is too small, the feasible solution space may become restricted, making it more challenging to find a satisfactory solution. On the other hand, including too many trains will raise the complexity and size of the optimization problem, which will also increase the difficulty of the learning process for the reinforcement learning agent. In summary, finding the appropriate number of trains considered in one step is vital for achieving a good solution while maintaining computational feasibility.

In order to build the connection between consecutive steps, the concept of $\mathcal{T}^0$ is introduced as follows: $\mathcal{T}^0$ is a set of trains, which includes the most recent departed trains on each track from the initial station s_0. An illustration figure is given as follows:

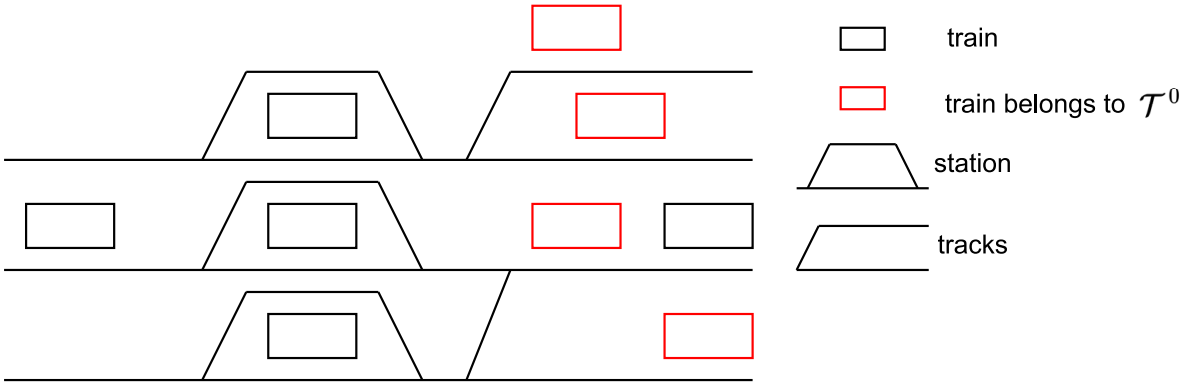

Figure 3-3: Illustration of the Concept of $\mathcal{T}^0$

The figure above represents a station with 3 platforms and 4 departure tracks. Four trains represented by red rectangles on different tracks form the group of $\mathcal{T}^0$ in this situation. Introducing the concept of $\mathcal{T}^0$ is necessary for the reinforcement learning environment to build the connection between steps. Using $\mathcal{T}^0$, the current step could obtain sufficient information about the previous step. From the definition of $\mathcal{T}^0$, it can be seen that the operations of trains belonging to $\mathcal{T}^0$ have already occurred, which means that the departure orders and times related to trains in $\mathcal{T}^0$ cannot be changed. Therefore, for the optimization-based method, some extra constraints are needed to fulfill this requirement.

At the beginning step, trains in $\mathcal{T}^0$ could be selected as the last trains departed on time on each track. The reason for this choice is that the optimal timetable rescheduling solution obviously cannot adjust trains not affected by delays. In another word, it is impossible to change previous trains to reduce the impact of the subsequent train delay. In other steps, trains in $\mathcal{T}^0$ are chosen as the latest departed trains on each track from the last step. In this way, the number of trains in $\mathcal{T}^0$ remains the same across different steps. Consequently, the total number of trains considered in each time step is also the same, which will formulate MILP problems with the same size. This property is very crucial for the reinforcement learning model to learn.

After determining $\mathcal{T}^0$, the subsequent updates of the railway system become straightforward. In this thesis, a single central controller is implemented, which implies that in each step, the action taken by the agent, along with its corresponding continuous solutions will determine the complete operation of the trains involved in this step. Then for the next step, these operations are considered immutable, even though in reality, they may not have taken place yet.

In a short summary, the railway system in the reinforcement learning environment is triggered by a disturbance at the beginning and updated periodically. The number of trains included

at each time step and for the entire problem should be determined carefully, with consideration of railway network structure, timetable details, calculation resources, etc. Finally, the connection between different steps is based on the special setting of $\mathcal{T}^0$.

State Selection

In principle, the state of a reinforcement learning environment should be able to provide all necessary information for the agent to make suitable decisions. However, it is always a challenging task to determine what information is actually "necessary" for the reinforcement learning agent.

In this thesis, the objective of the reinforcement learning agent is to generate accurate integer solutions to the MILP problem derived from the railway timetable rescheduling problem. One natural choice is to consider all parameters in the MILP problem as the state. Theoretically, since these parameters encompass the complete solution of the MILP problem, they should also provide sufficient information for the reinforcement learning agent to learn the integer solutions. However, there are two potential problems with incorporating all parameters of the MILP problem. First, including all parameters may increase the computational burden for the reinforcement learning agent. Although a broader range of states can offer more information to the agent, it also runs the risk of diluting the key factors. Second, some parameters remain constant in different MILP problems. For instance, consider the right sides of constraint (3-14) and (3-16), which consistently have the value of 1 no matter how the MILP problem varies. These kinds of parameters cannot provide much information for the agent. Therefore, including all parameters of the MILP problem as the state for the reinforcement learning model may not be a proper choice.

According to the above considerations, the state of the reinforcement learning model of this thesis will mainly include those parameters that vary when updating the MILP problem. Specifically, the state includes the following parameters:

- Passenger number $p_{i,s}$: From the objective function given in (3-3), it could be known that different passenger numbers represent different weights of trains at their destinations. It is apparent that these passenger numbers will change across the different MILP problems and also have an influence on the integer solutions. Here, the vector $\boldsymbol{p}$ is used to represent all passenger numbers.

- Initial arrival time a_{i,s_0}: As described in the previous part, the starting point of the railway timetable rescheduling problem is that a delay happened at a station where reordering is possible. This specific station is denoted as s_0. Obviously, the arrival time a_{i,s_0} at this station must be known so that the MILP problem could have a solution. In different MILP problems, the initial arrival time will vary. Furthermore, it is also crucial for the entire solution to the MILP problem. The vector $\boldsymbol{a}_{s_0}$ is used to represent the arrival time vector of all trains at the initial station s_0.

- Initial arrival order α_{i,j,s_0}: The initial arrival order could be derived from the initial arrival time a_{i,s_0} using the definition in (3-2). While the arrival order information can be obtained directly from the arrival time, explicitly including it in the state representation

can still be immensely beneficial. This is primarily due to its direct connection with the integer component of the solution, making its inclusion instrumental in capturing essential characteristics. Similarly, the vector $\boldsymbol{\alpha}_{s_0}$ represents the vector of all arrival orders at the initial station s_0.

Besides these parameters from the MILP problem, some other information is also included in the state representation to help the agent learn. First, the existing delays μ will be explicitly given in the state representation. Second, the numbers of $\mathcal{T}^0$ for all trains are also provided to the agent. Theoretically, this information is not part of the MILP problem's parameters. However, both of them are crucial for the agent to understand the environment and its connection between steps. In summary, the state representation is given as follows:

$$\Lambda = (\boldsymbol{p}, \boldsymbol{a}_{s_0}, \boldsymbol{\alpha}_{s_0}, \mu, \mathcal{T}^0) \tag{3-24}$$

Action Selection

From the problem overview, the reinforcement learning agent should give the integer solutions of the MILP problem as its action. Similar to state selection, a natural choice would be to simply add all integer variables in the action space. However, this may introduce several problems:

- Large action space: In the MILP problem, integer variables consist of the arrival order $\alpha_{i,j,s}$ and departure order $\delta_{i,j,s}$ for any two trains i and j at any station s. Suppose there are n trains considered for m stations. For simplicity, it is assumed that all trains will go through all stations. Then, the total number of all integer variables will be about $n \times n \times m \times 2$. However, this is only the number of variables. Since all integer variables are binary in the MILP problem, the actual size of the action space is 2^{2mn^2}. Suppose 10 trains are considered for 5 stations, then the size of action space will be 2^{1000}, which is obviously impossible for any reinforcement learning model to learn.
- Infeasible action space: In the MILP problem given in the first section, there are several constraints related to the integer variables, such as constraint (3-14), (3-16) and (3-20). These constraints significantly limit the size of feasible action space. For the reinforcement learning model, it is impractical to strictly adhere to these constraints. The only possible approach is to incorporate a penalty term in the reward function for the constraint violation. However, since these infeasible solutions cannot be taken by the MILP problem, the updating of the environment cannot be executed as well. To solve this problem, one possible way is to simply stop when an infeasible action is given by the agent. Nevertheless, given the scale of the feasible action space versus the complete action space, it may be very challenging for the agent to find any feasible solution in a short time. Another approach is to keep the state unchanged when the environment received an infeasible action. However, this setting actually requires that the agent needs to give different actions when facing the same state. Although this could be achieved by giving a very high exploration rate, it still makes this entire problem very ill-posed.

From the analysis above, including all integer variables in the action is not a feasible choice. Therefore, reducing the size of action space and consequently increasing the ratio of feasible

action space is necessary and critical for the success of the reinforcement learning model. The ideal scenario is that the action space is completely feasible for the MILP problem. This means that the action can only involve independent integer variables. Denote the independent integer variables as $\sigma \in \{0,1\}^{n_\sigma}$, it can be given as:

$$\sigma = f(\alpha, \delta; \rho) \tag{3-25}$$

where ρ represents the parameters of the MILP problem, α and δ are the arrival and departure orders for all trains at all stations, respectively. The independence of these binary variables is highly related to the layout of the railway network considered. It is noticed that the selection of independent variables may not be unique. Sometimes it may not even be defined among the integer variables of the MILP problem. Even so, it is still possible to prune some binary variables according to explicit constraints. Three basic principles are given as follows:

- The order between two trains on the same track cannot be changed.
- At the intersection point where multiple tracks merge to one track, the order between the train which has already passed this intersection and trains that have not yet passed cannot be changed.
- The conjugate orders are always dependent on each other.

After the reinforcement learning agent gives the independent integer variables, the complete departure and arrival orders could be expressed as:

$$\begin{aligned} \delta &= g_1(\sigma) \\ \alpha &= g_2(\sigma) \end{aligned} \tag{3-26}$$

where δ and α represent all departure and arrival orders, respectively. Function $g_1(\cdot)$ and $g_2(\cdot)$ are two encoding functions that encode the independent binary variable σ into these orders in the railway timetable rescheduling problem. The specific selection approach of independent binary variables and corresponding encoding functions used in this thesis will be introduced in the case study chapter together with the railway network in detail.

Reward Function

For the reinforcement learning agent, the reward function should provide a clear evaluation of its action under the current state. From the perspective of the MILP problem, the objective function is a very natural choice. In fact, the target of the reinforcement learning agent is to give a satisfactory solution of integer variables so as to minimize the objective function of the MILP problem. To some extent, the reinforcement learning agent could be seen as a solver, which is specially designed for solving the MILP problem derived from the railway timetable rescheduling problem. Since in the MILP problem, the objective is to minimize the total delay, for the reward function, the negative value of the objective function given in (3-3) is used. Specifically, the reward function is given as follows:

$$r = -K \sum_{s \in \mathcal{S}} \sum_{i \in \mathcal{T}} p_{i,s}(a_{i,s} - A_{i,s}) \tag{3-27}$$

where K is a scaling constant.

3-2-2 Reinforcement Learning Algorithm

As discussed in the last part, the action space for the reinforcement learning environment is discrete in this thesis. From the introduction of reinforcement learning in Chapter 2, it is known that the value-based method is easy to implement and effective for tasks with discrete action space. However, one of the challenges encountered by the original DQN and other value-based methods is the tendency to overestimate certain action values. To address this issue, the Double DQN (Van Hasselt et al., 2016) technique was introduced. The primary cause of overestimation in DQN arises from utilizing the same Q values for action evaluation and selection. Consequently, there is an increased likelihood of selecting overestimated values, leading to overly optimistic value estimates. The target employed by DQN is defined as follows:

$$y_t = \Psi_{t+1} + \tau \max_{\omega} \hat{Q}(\Lambda_{t+1}, \omega; \boldsymbol{\eta_t^-}) \tag{3-28}$$

where $\boldsymbol{\eta_t^-}$ represents the parameters of the target Q network for DQN. For a clear comparison, the target equation can be rewritten as:

$$y_t = \Psi_{t+1} + \tau \hat{Q}(\Lambda_{t+1}, \arg\max_{\omega} \hat{Q}(\Lambda_{t+1}, \omega; \boldsymbol{\eta_t^-}); \boldsymbol{\eta_t^-}) \tag{3-29}$$

It is easy to notice that the selection of action ω, is still parameterized by the same $\boldsymbol{\eta_t^-}$ as the evaluation Q.

Therefore, the Double DQN method uses two different convolutional neural networks to select and evaluate the action. In this context, the existing online network of DQN would be a natural choice. As mentioned in the last part, DQN uses the target network $\hat{Q}$ parameterized by $\boldsymbol{\eta_t^-}$ to calculate targets and another online network Q parameterized by $\boldsymbol{\eta_t}$ to get updated. The target network will be updated periodically by assigning $\hat{Q} = Q$. Here, Double DQN uses the online network Q to select the action and keeps the target network $\hat{Q}$ to evaluate. Then the target equation could be written as:

$$y_t = \Psi_{t+1} + \tau \hat{Q}(\Lambda_{t+1}, \arg\max_{\omega} Q(\Lambda_{t+1}, \omega, \boldsymbol{\eta_t}); \boldsymbol{\eta_t^-}) \tag{3-30}$$

Here, two networks for selection and evaluation in Double DQN are not fully decoupled, since the target network $\hat{Q}$ remains a periodic copy from the online network Q. This version of Double DQN is considered as the minimal change of DQN towards Double Q-learning.

As mentioned above, in this thesis, the structure of the Q network is designed according to the state feature. Since there is no image information in the railway timetable rescheduling problem, the convolutional layer used in the original paper is not implemented. Specifically, the Q network used in this thesis consists of the following layers:

- Input layer: the same dimension as the state input, used to take the state signal
- Fully connected layer 1, followed by a rectifier non-linearity (ReLU) layer
- Fully connected layer 2, followed by a ReLU layer
- Fully connected layer 3, followed by a ReLU layer

- Output layer, output all action values

An illustration figure for the neural network structure is given as follows:

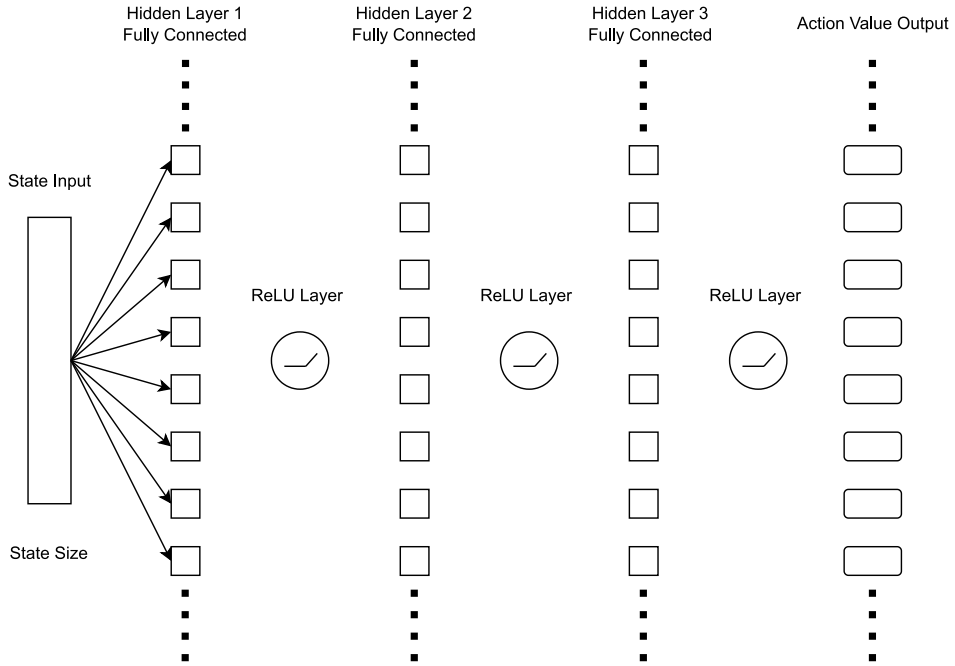

Figure 3-4: Structure of Q-Network

3-3 Complete Solution

In section 3-1, the MILP problem derived from the railway timetable rescheduling is thoroughly explained. Then, section 3-2 explains the dynamics of the environment of reinforcement learning, which includes the update of the railway system, state and action selection, and the reward function. After that, details of the reinforcement learning algorithm implemented in this thesis, Double DQN, are discussed. In this section, all contents mentioned above will be merged together to form the complete reinforcement learning-based solution to the railway timetable rescheduling problem.

From the perspective of the reinforcement learning agent, the complete procedure for solving the railway timetable rescheduling problem could be described as follows:

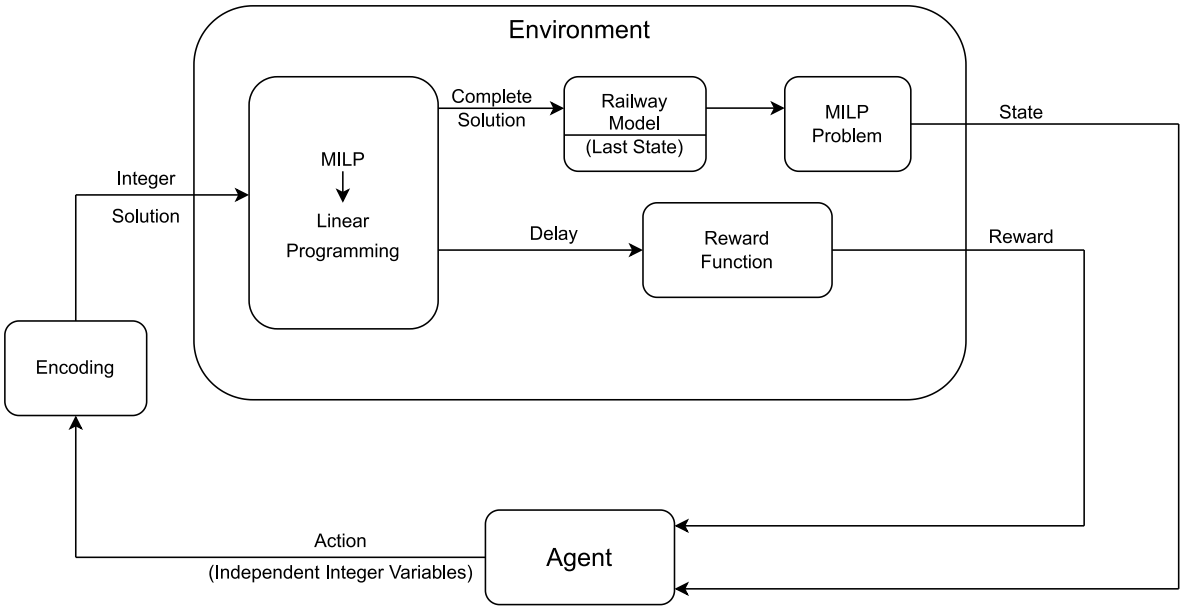

Figure 3-5: Procedure for Solving the Railway Timetable Rescheduling Problem

The following algorithm provides a procedure overview of the proposed solution in this thesis. In the beginning, the railway system and reinforcement learning model are initialized. Then, when a qualified initial delay happens, the railway timetable rescheduling problem starts. First, the MILP problem and $\mathcal{T}^0$ are formulated from the railway system and the initial delay. Then in line 4, the initial state is derived from the MILP problem. After that, from line 5 to the end, the railway timetable rescheduling problem is solved in multiple time steps. In each step, the reinforcement learning model first takes the state Λ and provides the action, which is also the independent binary variable σ. Then, the complete binary variables can be obtained by using g_α and g_δ functions. With α and δ solved, the MILP problem becomes a linear programming problem, which could be solved efficiently. After that, the corresponding reward for this step can also be obtained according to equation (3-27). Then, in lines 11 and 12, the railway model could be updated to the next step. New MILP problem and $\mathcal{T}^0$ can also be formulated from the railway system. After that, the new state Λ' is derived from the new MILP problem. Then, if the procedure happens during the training process, the experience of this step, $(\Lambda, \Omega, \Psi, \Lambda')$ will be put in the experience replay buffer. Then, the agent will be updated according to the Double DQN method mentioned in section 3-2. Finally, no matter training or testing, the state Λ will be updated to the new state Λ', and then move to the next step until the maximum step number is reached.

Algorithm 3 Reinforcement Learning-based Solution to Railway Timetable Rescheduling

Input: Railway system, reinforcement learning agent, initial delay, flag

Formulate the MILP problem and $\mathcal{T}^0$
Formulate the initial state $\Lambda \leftarrow \Lambda_0$ from the MILP problem
for Step = 1, MaxSteps **do**
  Take the state Λ, return action σ from the reinforcement learning agent
  Encode σ: $\begin{cases} \alpha \leftarrow g_\alpha(\sigma) \\ \delta \leftarrow g_\delta(\sigma) \end{cases}$
  Transform the MILP problem into linear programming problem using α and δ
  Solve the linear programming problem
  Calculate the reward Ψ
  Give the complete solution to the railway model
  Update the railway model
  Formulate new MILP problem as (3-21) and $\mathcal{T}^0$
  Formulate new state Λ' as (3-24) from the MILP problem
  if flag == Training **then**
    Store $(\Lambda, \Omega, \Psi, \Lambda')$ in the experience replay buffer
    Update the agent using Double DQN method
  end if
  $\Lambda \leftarrow \Lambda'$
end for

3-4 Conclusions

In this chapter, the reinforcement learning-based solution is proposed to solve the railway timetable rescheduling problem, which is formulated as an MILP problem. The objective of the MILP problem is to minimize the total delay of all passengers. Two different kinds of rescheduling decisions are considered in this thesis: reordering and retiming. In the MILP problem, the reordering is completed by the binary variables, which include the arrival order and departure order. Meanwhile, the retiming is achieved by the continuous variables, which are related to different operation times of trains. The MILP problem also considers multiple constraints, which include train operation constraints, safety headway constraints, and train order constraints.

For the reinforcement learning environment, the railway system is designed as an event-triggered, time-driven system. It starts from a disturbance that fulfills certain conditions. After that, it is updated periodically until the final time is reached. The time step update of the railway system is based on a special group of trains $\mathcal{T}^0$, which is defined as the set of most recent departed trains on each track. Using $\mathcal{T}^0$, the connection between two consecutive steps is conducted.

The state of the reinforcement learning model is selected from the crucial parameters of the MILP problem, which include passenger number, initial arriving time and order, etc. This information from the MILP problem is expected to be enough for the reinforcement learning agent to learn the integer solutions. For the action, not all binary variables are included as the

action. Only independent binary variables formulate the action for the model. Selecting these independent integer variables is strongly correlated with the layout of the railway network, pre-defined timetable, etc. The reward function is simply chosen as the objective function of the MILP problem, multiplied by a constant.

Since the action space of the reinforcement learning environment is discrete, a value-based reinforcement learning algorithm: Double DQN is implemented to train the agent. Compared with the normal DQN algorithm, the Double DQN method can address the problem of overestimation. Because there is no image information in this task, the Q network of the reinforcement learning agent does not use the convolutional layer from the original paper. It is formed as a fully connected neural network, with ReLU layers as non-linear activation.

In the last section, the complete reinforcement learning-based solution is illustrated in both figure 3-5 and algorithm 3. The solution divides the complete railway timetable rescheduling problem into multiple steps and solves each step using the reinforcement learning agent for integer variables and subsequently linear programming for continuous variables. The solution could be used for both training and testing.

Chapter 4

Case Study

The reinforcement learning-based railway timetable rescheduling algorithm is proposed in Chapter 3. In this chapter, the experiment results in this thesis are presented and discussed. As mentioned in Chapter 3, the layout of the railway network has a significant influence on the MILP problem, environment setting, action selection, etc. Therefore, in the first section of this chapter, the railway network used for this thesis and some relevant settings will be given. After that, two case studies with different settings will follow. In the end, the conclusion of case studies is provided.

4-1 Setup

In this section, the basic setup for case studies is introduced. The details of the railway network are presented in the first part. Then, some parameters of the MILP problem formulated from the railway timetable rescheduling problem are given. In the last part, the environment settings of reinforcement learning are discussed and determined.

4-1-1 Railway System

In this thesis, part of the Dutch railway network from Utrecht (Ut) to 's-Hertogenbosch (Ht) is used for the case study. Between Ut and Ht, there are six stations considered: Utrecht Lunetten (Utl), Houten (Htn), Houten Castellum (Htnc), Culemborg (Cl), Geldermalsen (Gdm), and Zaltbommel (Zbm). Among these stations, only stations Htnc and Gdm are available for reordering.

Every half hour there are five trains, including three Intercity trains and two Sprinter trains, operating on the railway lines with a given timetable, and the routes of these trains are given in Figure 4-2. Specifically, the train names are Intercity 8xx, Sprinter 60xx, Intercity 39xx, Sprinter 69xx, and Intercity 35xx, where the last two digits of the train number depend on the specific time. Among these trains, all Intercity (IC) trains only stop at the final destination

Ht. Sprinter 60xx's destination is also Ht, but it will stop at every station. For Sprinter 69xx, the destination is Gdm, and it will also have an intermediate stop at every station between Ut and Gdm. The layout of the railway network and the operation route of these trains are given in Figure 4-2. The details of the timetable are obtained from the real-life timetable of Nederlandse Spoorwegen (NS). An example of the detailed timetable for trains departing from station Ut between 8:00 to 8:30 is given in Table 4-1. In the timetable, Dep refers to the departure time, and Arr is the arrival time.

From the railway network given in Figure 4-2, it can be noticed that reordering is only possible at stations Htnc and Gdm. For other stations, only retiming is possible. Therefore, the initial station s_0 for the railway system is given as station Htnc. The following figure shows the operation of the pre-defined timetable when there is no delay.

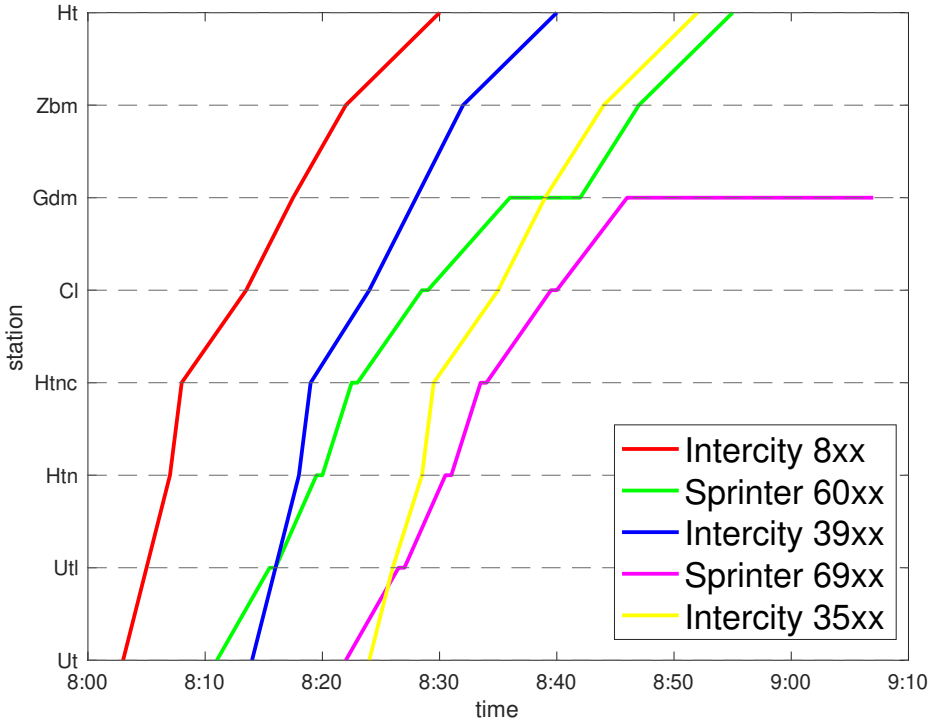

Figure 4-1: Operation of Pre-defined Timetable

Table 4-1: Timetable between Utrecht and Den Bosch (Every 30 minutes)

Station	Ut	Utl		Htn		Htnc		Cl		Gdm		Zbm		Ht
Train	Dep	Arr	Dep	Arr	Dep	Arr	Dep	Arr	Dep	Arr	Dep	Arr	Dep	Arr
IC 8xx	8:03:00	8:05:00	8:05:00	8:07:00	8:07:00	8:08:00	8:08:00	8:13:30	8:13:30	8:17:30	8:17:30	8:22:00	8:22:00	8:30:00
SP 60xx	8:11:00	8:15:30	8:16:00	8:19:30	8:20:00	8:22:30	8:23:00	8:28:30	8:29:00	8:36:00	/	/	/	/
IC 39xx	8:14:00	8:16:00	8:16:00	8:18:00	8:18:00	8:19:00	8:19:00	8:24:00	8:24:00	8:28:00	8:28:00	8:32:00	8:32:00	8:40:00
SP 69xx	8:22:00	8:26:30	8:27:00	8:30:30	8:31:00	8:33:30	8:34:00	8:39:30	8:40:00	8:46:00	8:51:00	8:56:30	8:57:00	09:07:00
IC 35xx	8:24:00	8:26:00	8:26:00	8:28:30	8:28:30	8:29:30	8:29:30	8:35:00	8:35:00	8:39:00	8:39:00	8:44:00	8:44:00	8:52:00

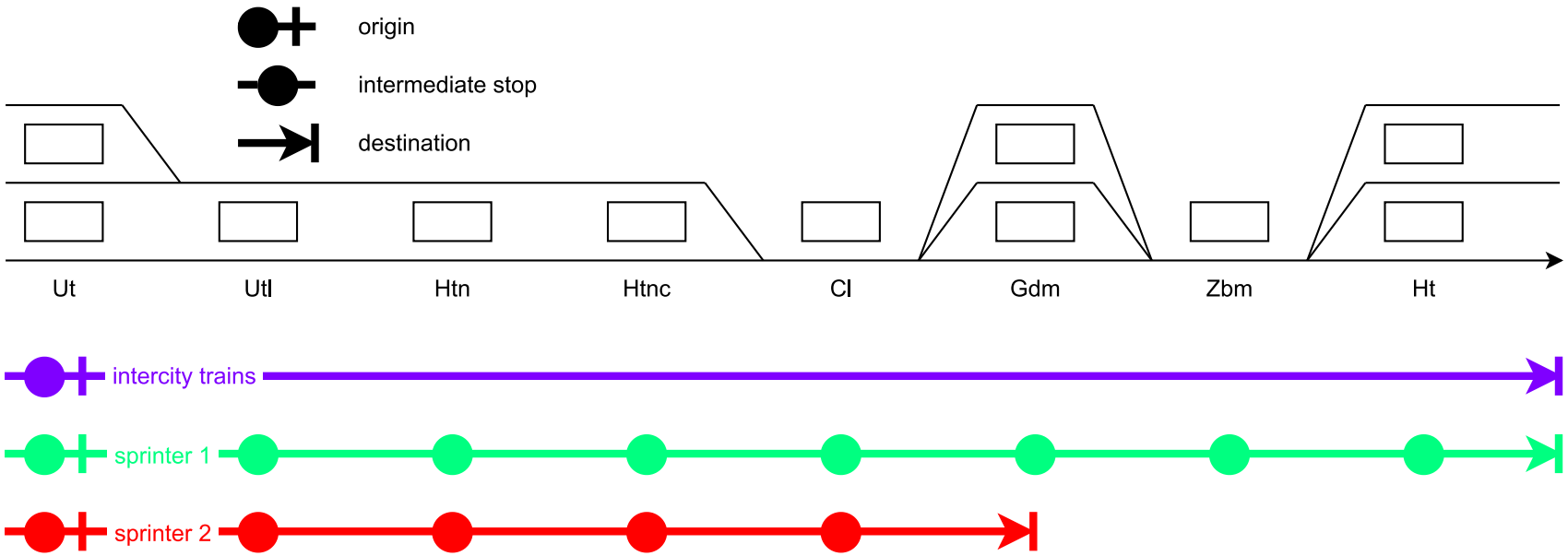

Table 4-2: Layout of the Railway Network

4-1-2 MILP Problem

Given the layout of the railway system in figure 4-2 and the timetable in table 4-1, several parameters of the MILP problem in equation (3-21) could be determined. First, it is obvious that after the Htnc station, there is only one track to operate, which indicates that the arrival order of any station after Htnc should be the same as the departure order from its previous station. Meanwhile, it also requires that the safety headway constraint should be applied to any two consecutive trains. Second, since Cl and Zbm stations only have one platform, reordering is also impossible. Therefore, the departure order should be the same as the arrival order at these two stations.

From the analysis above, some parameters of the MILP problem could be determined as follows:

Table 4-3: Some Parameters of the MILP Problem

Parameter	Notation	Value	Applicable Stations
Departure Situation	$\beta_{i,j,s}$	1	Htnc, Cl, Gdm, Zbm
Arrival Situation	$\theta_{i,j,s}$	1	Cl, Gdm, Zbm, Ht
Minimum Safety Headway	h	1.5	/
Minimum Running Time	$R^{\min}_{i,u,s}$	$0.93R_{i,u,s}$	All stations
Minimum Dwelling Time	$\Gamma^{\min}_{i,s}$	$0.93\Gamma_{i,s}$	Htnc, Cl, Gdm, Zbm
Number of Platforms	ξ_s	1	Cl, Zbm
		2	Htnc, Gdm

In the table above, $R_{i,u,s}$ represents the original running time of train i between station u and s from the timetable, and $\Gamma_{i,s}$ is the original dwelling time of train i at station s. $R_{i,u,s}$, $\Gamma_{i,s}$ and other parameters, such as $D_{i,s}$ and $A_{i,s}$ could be obtained from the pre-defined timetable given in Table 4-1.

4-1-3 Reinforcement Learning Setting

For the reinforcement learning model, some details of the environment settings will be given in this part. First, as discussed in Chapter 3, the selection of $\mathcal{T}^0$ is crucial to the update of the railway system. Since the railway network used in this thesis only has one track after the Htnc station, the $\mathcal{T}^0$ only includes one train at each time step. According to the timetable given in Table 4-1, the operation of trains considered in this thesis has a period of half an hour, which has five trains in total. Therefore, it is natural to take half an hour as the step length for the reinforcement learning algorithm. In this way, the railway timetable rescheduling problem in each step involves five trains. Since $\mathcal{T}^0$ only has one train, for every step four new trains need to be included.

After determining the number of trains for each step, the state of the reinforcement learning model could also be further presented as follows:

- Passenger number $p_{i,s}$: Since the initial station s_0 is Htnc station, only four stations after that need to be considered. Therefore, $5 \times 4 = 20$ passenger numbers will be included in the state. Here, for those stations where the train does not stop, the passenger number is simply set to be 0. In particular, for Sprinter 69xx, since its destination is Gdm, the numbers of passengers arriving at Zbm and Ht are also given as 0.
- Initial arrival time a_{i,s_0}: Since at each step, five trains are considered for the railway timetable rescheduling problem, the initial arrival time should also have five entries in the state representation.
- Initial arrival order α_{i,j,s_0}: Similar to the initial arrival time, the initial arrival order between each pair of all fives trains should be put into the state, which makes it $5 \times 5 = 25$ entries.
- Initial delay μ: As mentioned in Chapter 3, the initial delay is also explicitly given to the reinforcement learning agent in order to provide more information. The dimension of μ depends on the different settings of the following case studies.
- The ID of trains in $\mathcal{T}^0$: To strengthen the connection between two consecutive steps, the ID of $\mathcal{T}^0$ in the small timetable with five trains for one single step is also included in the representation. Since $\mathcal{T}^0$ only consists of one train, the following contents of this thesis directly use $\mathfrak{T}^0$ to represent the corresponding train.

In summary, the state Λ could be given as follows:

$$\Lambda = (\boldsymbol{p}, \boldsymbol{a}_{s_0}, \boldsymbol{\alpha}_{s_0}, \mu, \mathfrak{T}^0) \tag{4-1}$$

For the action of the reinforcement learning model, it is chosen as the independent binary variables as discussed in Chapter 3. For the railway timetable rescheduling problem, all binary variables are related to the reordering decisions. Therefore, the degree of freedom for reordering should be the same as the degree of freedom for the binary variables. From the layout of the railway network given in 4-2, it can be easily noticed that only the Htnc and Gdm stations can be used for reordering.

For the Htnc station, since it is assumed that $\mathfrak{T}^0$ has already departed from Htnc, there are only four trains whose orders need to be determined. It is plain to see that for four trains on two tracks, three binary variables would be enough to determine the entire order. Each binary variable could be used to determine which of the two consecutive trains will depart first. The detailed definition of the independent binary variable σ_k is given as follows:

$$\sigma_k = \begin{cases} 1, & \text{the train arrives first will depart first} \\ 0, & \text{the train arrives later will depart first} \end{cases} \tag{4-2}$$

where k represents the order between these independent binary variables. It is obvious to see that σ_k is closely related to the arrival and departure orders. Suppose that σ_k will determine the order between two trains i and j, it is easy to notice that σ_k is one if and only if the departure order between train i and train j is as same as their arrival order. Mathematically, this relationship can be written as:

$$\begin{cases} \alpha_{i,j,s} = \delta_{i,j,s}, & \text{if } \sigma = 1 \\ \alpha_{i,j,s} \neq \delta_{i,j,s}, & \text{if } \sigma = 0 \end{cases} \tag{4-3}$$

In this thesis, the assignment of tracks is ignored to simplify the problem. The selection of independent binary variables at the Gdm station is similar to the Htnc station. It should be noticed that for $\mathcal{T}^0$, it is assumed that it has already departed from the Htnc station, but there is no assumption or constraint about its relationship with the Gdm station. Therefore, at the Gdm station, all five trains need to be considered for reordering. Therefore, four independent binary variables are needed to completely determine the orders of five trains at the Gdm station. The definition of independent binary variables is the same as above.

In total, there are seven independent binary variables for the railway timetable rescheduling problem at one single step. Then, since they are all independent, the size of action space is given as $2^7 = 128$. In order to reduce the dimension of the action space, these seven independent variables are encoded as one decimal integer, which is used as the final action Ω and has a range from $0 \sim 127$. Here, the seven independent binary variables are given as $\{\sigma_1, \cdots, \sigma_7\}$. Among these variables, σ_1, σ_2 and σ_3 are related to the orders at the Htnc station, where σ_1 is used to determine the order of first two trains, and the other two follow in turn. Similarly, $\sigma_4, \cdots, \sigma_7$ are used to determine the departure order at the Gdm station. Here, σ_4 will be used first, then σ_5, σ_6 and σ_7.

For the training of the reinforcement learning agent, the $\epsilon-$greedy algorithm is implemented to encourage the exploration. The Adam optimizer is used to optimize the Q network of the agent. Specifically, some training parameters for the reinforcement learning agent are given as follows:

Table 4-4: Some Parameters of Reinforcement Learning Agent Training

Parameter	Notation	Value
Size of Hidden Layer 1 in the Q Network	/	State Size $\times$ 256
Size of Hidden Layer 2, 3 in the Q Network	/	256 $\times$ 256
Size of Output Layer in the Q Network	/	256 $\times$ Action Size
Experience Buffer Length	N	10000
Batch Size	/	128
Learning Rate	ζ	0.001
Initial Exploration Rate	ϵ_0	0.8
Epsilon Decay Rate	/	0.001
Discount Factor	τ	0.99
Regularization Factor	/	0.0001
Reward Scaling Constant	K	10^{-4}

4-1-4 Hardware and Software

In this thesis, all case studies are completed on the author's personal laptop. Some hardware and software configurations are given as follows:

- Laptop: Apple MacBook Pro 14.2 2017
- CPU: 3.1 GHz Intel i5

- RAM: 8 GB
- Software: MATLAB R2022b Update 4 (9.13.0.2166757)
- Optimizer: Gurobi Optimizer (v10.0.1rc0 (mac64[x86])) based on YALMIP (20210331)

4-2 Case Study A: Open-loop Control

In this case, the railway timetable rescheduling problem is considered an open-loop control problem. The initial delay of the railway system is given as $\mu_0 = 8 + 4 * U(0, 1)$. Specifically, there are no subsequent delays after the initial disturbance. The delay item μ in the state representation will be the secondary departure delay of $\mathfrak{T}^0$ at station Htnc. Therefore, the aim of the timetable rescheduling is to eliminate the influence of the initial delay. It is expected that the timetable will become normal within 2 hours.

4-2-1 Training Result

In this case study, the agent is trained for 2000 epochs, which take about 2 hours based on previously mentioned configurations. The figure of episode reward during the training process is presented as follows:

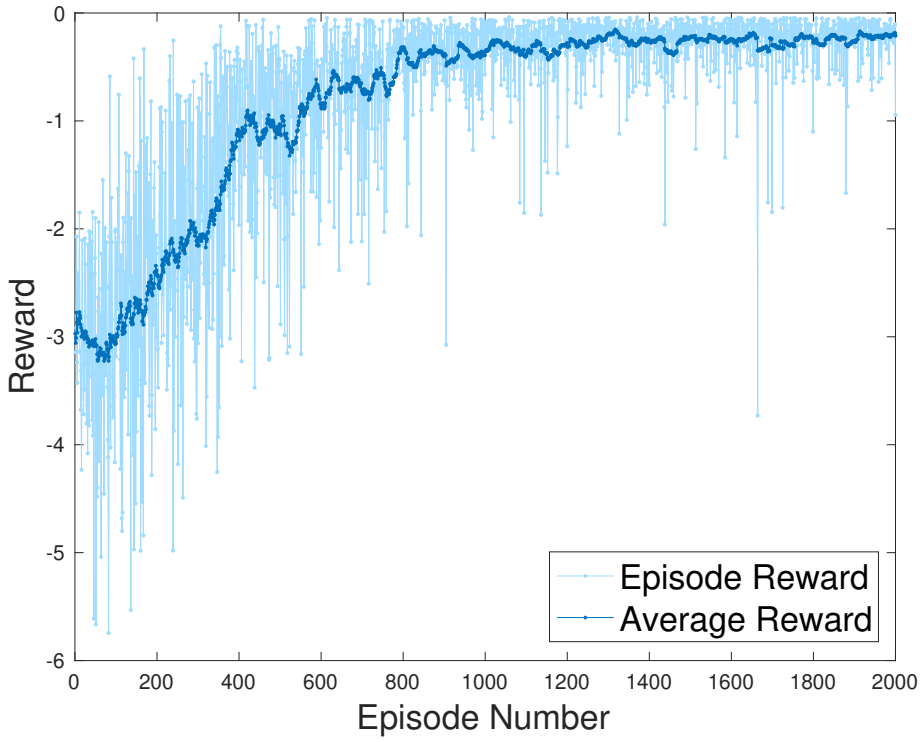

Figure 4-2: Episode Reward during Training Process for Case A

In figure 4-2, the light blue curve is the reward signal for each episode, and the dark blue curve represents the average reward with an average window of 30 episodes. From the figure above, it can be noticed that the curve of average reward becomes stable after about 1200 episodes. The total average reward after 1600 episodes is -0.25.

From the training results above, it can be noticed that the agent learned well in this task. Although there are still some outliers in the curve of episode reward, the average reward is quite stable after 1600 episodes, which shows that the learning result should be satisfactory.

4-2-2 Testing Result

The testing of the trained agent has 30 episodes, which have the exact same setting as the training environment, except that the agent will not be updated anymore. To further study and compare the performance of the reinforcement learning agent, two other methods for solving the railway timetable rescheduling problem are also implemented as the baseline and the global optimal solution.

Baseline: First-In-First-Out (FIFO)

The key point of the FIFO method is that it keeps the original order of trains. Therefore, with these orders determined, the railway timetable rescheduling problem will become a linear programming problem. In reality, the FIFO method is widely applied by train dispatchers since it is easy to implement. Compared with the FIFO method, the improvement of the reinforcement learning-based method is defined as:

$$z_{\mathrm{RL}} = \left| \frac{\kappa_{\mathrm{RL}} - \kappa_{\mathrm{F}}}{\kappa_{\mathrm{F}}} \right| \times 100\% \ , \tag{4-4}$$

where z_{RL} is the improvement rate of the reinforcement learning-based method compared with the FIFO method, κ_{F} and κ_{RL} are the corresponding total delay of the FIFO method and the reinforcement learning-based method, respectively.

Global Optimal Solution

Since this case study is an open-loop control problem, the global optimal solution could be obtained by solving the MILP problem which is formulated from the complete railway timetable rescheduling problem with 20 trains. Since it is globally optimal, the result of the reinforcement learning agent cannot be better. Compared with the FIFO method, the improvement of the global optimal solution is defined as:

$$z_{\mathrm{MILP}} = \left| \frac{\kappa_{\mathrm{MILP}} - \kappa_{\mathrm{F}}}{\kappa_{\mathrm{F}}} \right| \times 100\% \ , \tag{4-5}$$

where z_{MILP} is the percentage of the reinforcement learning-based method's performance compared with the global optimal solution, κ_{MILP} represents the total delay calculated using the global optimal solution.

The testing results of the reinforcement learning agent after 30 rounds are given as follows:

Table 4-5: Testing Results for Case A

Method	Average Delay	Improvement	Running Time [s]
Baseline (FIFO)	12290	0%	5.07
Reinforcement Learning-based Method	2157	83.49%	1.42
Global Optimal Solution	1699	86.78%	9.19

From the test results above, it is noticed that compared with the baseline, the reinforcement learning-based agent makes a great improvement, which proves that this method's performance is satisfactory. Compared with the global optimal solution, the reinforcement learning agent only has a gap of about 3%. In fact, in 19 out of 30 rounds, the reinforcement learning-based method actually obtained the global optimal solution.

Although the performance of the reinforcement learning-based approach is not as good as the optimization-based method in this case study, it has an advantage regarding the calculation time. For the reinforcement learning-based method, obtaining the binary variables takes very little time, since they are calculated by the agent. After that, there is only one linear programming problem to solve, which is much easier than the original MILP problem. From the table above, the reinforcement learning-based method took about 1.42 seconds to solve one episode on average, while the MILP solver needs about 9.19 seconds. The reinforcement learning-based method saved about 85% of calculation time. Although for this case study, 9.19 seconds is still acceptable, for the larger railway network, the time advantage of the reinforcement learning-based method will be more significant.

4-3 Case Study B: Closed-loop Control

The main shortcoming of the previous case study is that it does not consider the possible subsequent delays during the timetable rescheduling process. Therefore, in this section, a closed-loop control case is further studied. In this case study, after the initial delay, there is still a probability that some extra disturbances occur in the following steps. It is assumed that in this case study, every step will have at most one extra delay. Therefore, the reinforcement learning agent needs to control the system in a closed-loop scheme.

The delay item μ in the state representation is given as $\mu = (\mu_1, \mu_2)$, where μ_1 represents the initial delay or extra delay, and μ_2 is the secondary delay from the last step. The value of the initial delay and the calculation of the secondary delay is as same as in Case A. The value of the extra delay is given as $\mu_{ex} = 6 + 4 * U(0, 1)$, and the probability of adding this extra delay is 50% for every step. The extra delay may be added to an arbitrary train except $\mathfrak{T}^0$. For the closed-loop control problem, since the extra delay may be added at any step, the agent needs to control the system until the end of the railway timetable rescheduling problem. In this case study, the total time of the railway timetable rescheduling problem is set to 6 hours, which includes 60 trains in total. There is no other stop condition except the agent reaches

the end of the problem. For a single step, the number of trains included remains the same as in the previous case study. Thus, there are 15 steps for each episode.

4-3-1 Training Result

For the closed-loop control case, the agent is trained for 5000 epochs, which takes about 10 hours. Other training settings are as same as the last case study. The figure for the episode and average reward during the training process is given as follows:

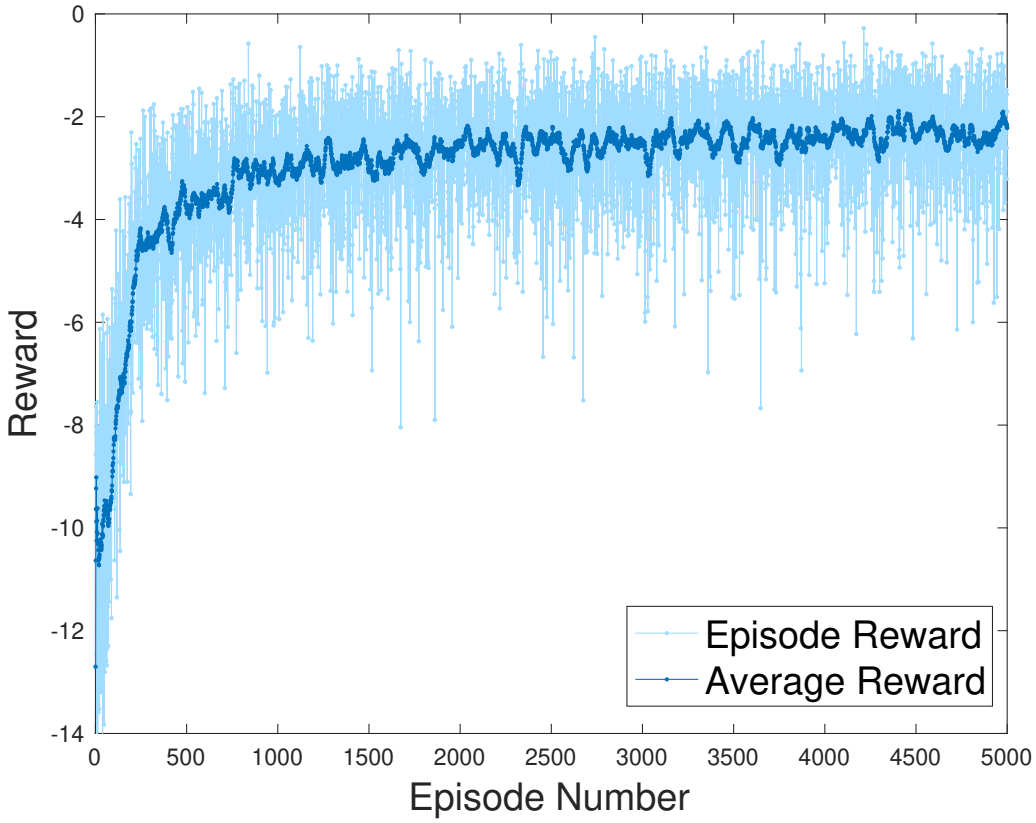

Figure 4-3: Episode and Average Reward during Training Process for Case B

From the training result above, it is noticed that the agent learns quite fast at the beginning. After about 300 epochs, the increase of rewards becomes slower. From about 4000 epochs, the learning curve becomes stable. The average reward after 4000 epochs is -2.3259. Compared with the curve of average reward, the episode rewards have some large outliers. The overall training performance of the reinforcement learning agent in the closed-loop control case is still satisfactory.

4-3-2 Testing Result

For the closed-loop control case, obtaining the global optimal solutions is not realistic anymore, since it is impossible for any algorithms to predict the possible extra delays in the future. In this case study, the baseline is still the FIFO method. Instead of the global optimal solution, a local optimization-based method is introduced to be compared with the reinforcement learning-based approach.

The main idea of this local optimization-based method is simple: an MILP solver will be implemented to solve the exact same MILP problem as the reinforcement learning agent faced, and then use the local optimal solution to control the railway system. Compared with the FIFO method, the improvement of this local optimization-based method could be defined as:

$$z_{\mathrm{LO}} = \left| \frac{\kappa_{\mathrm{LO}} - \kappa_{\mathrm{LO}}}{\kappa_{\mathrm{LO}}} \right| \times 100\% \ , \tag{4-6}$$

where κ_{LO} is the total delay after 15 steps of the local optimization-based method. Higher z_{LO} indicates that the performance of this local optimization-based approach is better.

The testing results for these two methods after 30 rounds are given as follows:

Table 4-6: Testing Results for Case B

Method	Average Delay [min]	Improvement	Running Time [s]
Baseline (FIFO)	42903	0%	36.21
Reinforcement Learning-based Method	16858	61.23%	7.18
Local Optimization-based Method	9740	77.50%	10.98

From the testing results above, it is known that compared with the baseline, both algorithms make significant improvements. However, compared with case A, it can be noticed that the performance of both algorithms drops due to the extra delay. The gap between the improvement of these two approaches is 16.23%.

Similar to the first case study, the reinforcement learning-based method still has the advantage in terms of operation time. From the testing results, the reinforcement learning-based method needs about 7.18 seconds to solve on average. The local optimization-based method used 10.98 seconds to complete one testing round. The reinforcement learning-based approach saves about 34.6% of time in this case. Compared with the last case study, the reinforcement learning-based method still has time advantages, but the size of this advantage is decreasing. The main reason is that in this closed-loop control case study, the optimization-based method is supposed to solve multiple small MILP problems instead of one large MILP problem, which saves a lot of time here. Specifically, the size of the MILP problem in the first case study is roughly 4 times larger than the MILP problem for one step in this case study (according to the number of trains they included), while the time difference is about 12.5 times, which shows that for the optimization-based method, the solving time increases quadratically compared with the increase of the problem size.

4-4 Case Study C: Closed-loop Control with Multiple Extra Delays

In the last case study, there is only one extra delay added to every step with a probability of 50%. In this section, the influence of multiple extra delays will be studied. The reinforcement learning agent is still dealing with a closed-loop control problem, the only difference is that in every step, each new train may get an extra delay simultaneously.

According to the description above, it is clear that all trains except $\mathfrak{T}^0$ may have new delays. Therefore, the delay item μ is given as $\mu = (\mu_1, \mu_2, \mu_3, \mu_4, \mu_5)$, where μ_1 is used to represent the secondary delay of $\mathfrak{T}^0$ after the initial step, μ_2, μ_3, μ_4 and μ_5 are the delays added to the corresponding trains. The IDs of trains are determined according to their original arrival time at the Htnc station except $\mathfrak{T}^0$. The probability of adding extra delays is 50%. The definitions of these delays are given as follows:

Table 4-7: Delay Definitions for Case C

Delay	Initial Step	Other Steps
μ_1	0	$d_{\mathfrak{T}^0,Htnc} - D_{\mathfrak{T}^0,Htnc}$
μ_2	$8 + 4 * U(0,1)$	$1 + 2 * U(0,1)$
μ_3	$4 + 4 * U(0,1)$	$2 + U(0,1)$
μ_4	$5 + 3 * U(0,1)$	$2 + 2 * U(0,1)$
μ_5	$5 + 4 * U(0,1)$	$1 + U(0,1)$

The extra delays $\mu_2, \cdots, \mu_5$ after the initial step are specially designed so that the sum of their expectations ($2 + 2.5 + 2 + 1.5 = 8$) is exactly the same as the expectation of the extra delay in case B. In this way, the influence of multiple delays could be better compared. Other settings, such as the total length and each time step's length of the railway timetable rescheduling problem are as same as in case B.

4-4-1 Training Result

In this case study, the agent is trained for 3500 epochs, which takes about 7 hours. Other training parameters are as same as previously described. The figure for the episode and average reward during the training process is given as follows:

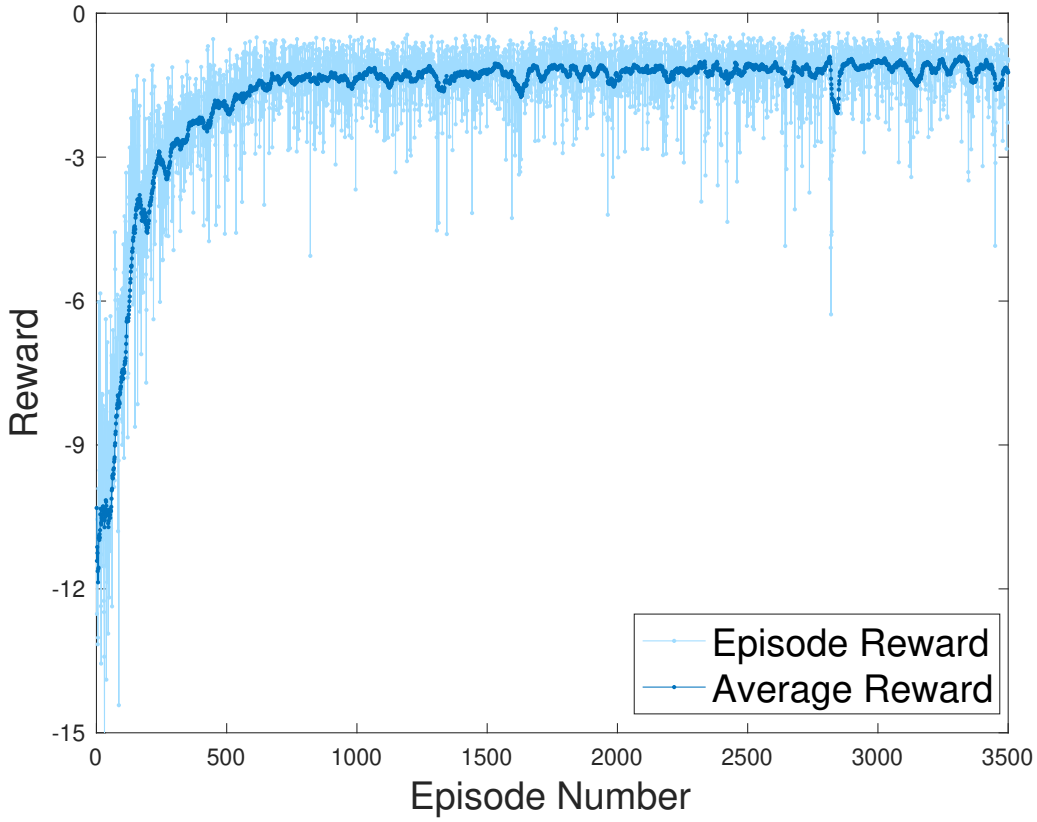

Figure 4-4: Episode and Average Reward during Training Process for Case C

From the figure above, it is noticed that the overall learning performance is good. It can be seen that the agent learned very fast in the first 500 epochs. After about 800 episodes, the learning curve has already become very stable. The average reward after 3000 epochs is -1.1636.

4-4-2 Testing Result

Since this case study is still a closed-loop control problem, the metrics used to show the performance of the reinforcement learning agent are as same as in the last case study. Meanwhile, the local optimization-based method introduced in the last section is also implemented here as a comparison. The testing results are given as follows:

Table 4-8: Testing Results for Case C

Method	Average Delay [min]	Improvement	Running Time [s]
Baseline (FIFO)	38751	0%	41.26
Reinforcement Learning-based Method	9582	75.08%	8.17
Local Optimization-based Method	7139	81.95%	12.54

From the results above, it is noticed that compared with the baseline, both reinforcement learning-based and local optimization-based approaches make a great improvement, which shows the effectiveness of these two algorithms.

Compared with the open-loop control case, both methods' performances dropped, which indicates that the extra delays still have negative influences on the rescheduling performance as in the last case study.

However, compared with the closed-loop control problem with only one extra delay, both methods' performances raise. Specifically, the reinforcement learning-based method increases by about 14%, while the optimization-based approach improves by about 4.5%. Although the performance of the reinforcement learning-based method is still worse than the optimization-based method, the gap between these two approaches is reducing. This property indicates that the reinforcement learning agent learns better when there are multiple small delays instead of one large delay. One possible reason could be that although the extra delays in these two case studies have the same expectations, multiple small delays tend to change the timetable in a more holistic way, while one large delay may cause that the timetable become more unstable.

The running time of these two algorithms is close to the second case study. For 15 epochs, the reinforcement learning-based method takes about 8.17 seconds on average, while the local optimization-based method takes about 12.53 seconds. The reinforcement learning-based method saves about 34.8% of running time.

4-5 Conclusions

In this chapter, the relevant settings for the case study are introduced first, including the details of the railway system, some parameters of the MILP problem, reinforcement learning agent training and implementation, and some configurations of hardware and software used in this thesis. After that, three case studies are presented and discussed. Case study A is an open-loop problem, where the objective is to eliminate the negative influence of the initial delay. Case study B is a closed-loop control problem, which has an extra delay added with a probability. The main difference between the open-loop and closed-loop control problems is that for the closed-loop control problem, the reinforcement learning agent needs to control the system until the end of the railway timetable rescheduling problem, while for the open-loop problem, the agent can stop when the negative influence of the initial delay is eliminated. Case study C is also a closed-loop control problem. Instead of adding only one extra delay after the initial step, multiple delays could be added in a probabilistic way.

In all three cases, the FIFO method is used as the baseline. For better comparison, some optimization-based are also implemented with the same testing environment. For the first case, since it is an open-loop problem, the global optimal solution could be obtained. For case studies B and C, since they are closed-loop control problems, obtaining the global optimal solution is not practical. Therefore, a local optimization-based method is implemented, which solves the same MILP as the reinforcement learning agent in every step.

Compared with the baseline, the reinforcement learning-based approach makes great im-

provements in all three case studies. However, the performance of the closed-loop control problem is generally lower than the open-loop control problem. It is understandable that the closed-loop control is more difficult to solve than the open-loop control problem.

Compared with the optimization-based method, the reinforcement learning-based method's performance is lower in all three cases but still satisfactory. However, the reinforcement learning-based approach shows an advantage in terms of running time. From the comparison given in Section 4-3, it can be seen that this advantage will become more significant for larger MILP problems.

In case studies B and C, the influence of different numbers of extra delays is studied. The results indicate that compared with one large delay, the reinforcement learning-based method performs much better with multiple small delays, although they may have the same expectations.

Chapter 5

Conclusions and Outlook

In this thesis, a reinforcement learning-based method is proposed to solve the railway timetable rescheduling problem. The main contribution of the proposed method is to combine reinforcement learning with optimization techniques in the railway timetable rescheduling problem. Specifically, a value-based reinforcement learning agent takes the parameters from the MILP problem as the state and determines the binary variables of the problem as the action. With these integer solutions, the MILP problem is transformed into a linear programming problem, which could be easily solved. After solving the linear programming problem, it is equivalent to solving the MILP problem derived from the railway timetable rescheduling problem. Then, the railway system could be updated to the next step, and start this procedure again until the end of the problem. In this chapter, Section 5-1 will conclude the ideas and results of this thesis in detail. Section 5-2 will discuss some potential future work.

5-1 Conclusions

The railway timetable rescheduling problem refers to a problem that requires calculating a feasible and relatively good timetable to reduce the negative influence of disturbance and disruptions as much as possible. The rescheduling of the timetable is achieved by retiming and reordering of trains. In this thesis, the railway timetable rescheduling problem is formulated as an MILP problem, which is a special kind of optimization problem, part of whose variables are set to be integers. Compared with linear programming problems, the MILP problem is much more difficult to solve due to these integer variables. In this thesis, the integer variables are given as departure and arrival orders of trains considered, and continuous variables include the operation times of all trains, such as the departure time, arrival time, running time, and dwelling time. The objective of the MILP problem is to minimize the total arrival delay of all passengers on all trains. Meanwhile, several different constraints are considered for the MILP problem, including train operation constraints, safety headway constraints, station connection constraints, and platform flexibility constraints.

As introduced above, the main idea of this thesis is to implement a reinforcement learning agent to determine the integer variables of the MILP problem, then it will become a linear programming problem, which could be solved efficiently. Therefore, from the perspective of the reinforcement learning agent, both the railway system and its corresponding MILP problem together form the environment for the reinforcement learning task. In this environment, the updating is completed by the update of the railway system, which is designed as an event-triggered, time-based system. To build the connection between two consecutive steps, the concept of $\mathcal{T}^0$ is introduced. The state of the reinforcement learning agent is mainly selected from the parameters of the MILP problem, while the action is determined as all independent binary variables of the MILP problem. In this way, the constraints involving the integer variables in the MILP problem are satisfied. Finding these independent binary variables of the MILP problem is highly dependent on the layout of the railway network considered. Given these independent binary variables, an encoding process is still required to transfer them to the complete binary variables of the MILP problem. Choosing the actions as the independent binary variables not only makes the entire action space feasible but also reduce the size of the action space as much as possible. The reward function is simply given as the negative value of the objective function of the MILP problem. After constructing the environment of this reinforcement learning task, a value-based reinforcement learning algorithm: Double DQN is implemented to train the agent.

In this thesis, three case studies with different problem settings are presented and discussed. Case A is an open-loop problem, which only has one delay at the initial step, the target of the reinforcement learning agent is to eliminate the negative influence of this initial delay. Case B and C are closed-loop control problems, in which there may be extra delays in every step, which require the agent to control the railway system until the end of the entire railway timetable rescheduling problem. Compared the testing results of these three cases, some conclusions can be drawn:

- Compared with the baseline (FIFO), the performances of the reinforcement learning-based method are largely improved in all three cases. In the open-loop control, the reinforcement learning agent shows better performance than in the closed-loop control.
- Compared with the optimization-based methods, the reinforcement learning-based approach cannot reach their performances. Overall, the reinforcement learning-based method can achieve about 60% to 80% of the performance of the optimization-based method under the same problem settings.
- In terms of running time, the reinforcement learning-based approach has an obvious advantage compared with both baseline and optimization-based methods. The larger the railway timetable rescheduling problem size is, the greater the advantage of the reinforcement learning-based algorithm in saving running time.
- For the closed-loop control problems, the proposed method performs better with multiple small delays.

Overall, the reinforcement learning-based approach performs well in the railway timetable rescheduling problem. Especially, it shows an advantage regarding the running time. Now in practice, one of the main reasons that the optimization-based method cannot be implemented

is that it cannot achieve the running time requirement. Therefore, the reinforcement learning-based algorithm may be a potential choice for further implementation.

5-2 Future Work

Although the reinforcement learning-based method achieves satisfactory performance in this thesis, there are still some possible directions for future work:

Larger Railway Network

In this thesis, the railway network used in the case study is a small part of the Dutch railway network. As mentioned above, increasing the size of the railway timetable rescheduling problem may strengthen the advantage of the reinforcement learning-based method in terms of running time. However, a larger railway network may also bring new challenges, such as the larger state and action space, more computation resources required for the training, and more difficulty to model the railway system.

New Reinforcement Learning Algorithms

In this thesis, the Double DQN algorithm is implemented. However, there are many different choices for value-based reinforcement learning algorithms. Most of them could also be used to train the agent for this task. Comparing the results using different reinforcement learning algorithms may also be a potential topic.

More Detailed Passenger Demands

In this thesis, the passenger number $p_{i,s}$ is set as a known parameter. In some countries' railway systems, it is possible to obtain the number of passengers with their destination in real time. However, in other countries like the Netherlands, the destinations of passengers are not available. Therefore, this parameter may only be approximated from the historical data. It is also possible to fit the passenger demand with certain functions. However, it should be noticed that this may change the class of the optimization problem.

Other Applications

In general, the main idea proposed in this thesis could also be applied to other problems, which are formulated as the MILP problem. The only difference is the updating of the environment. As long as the problem is inherently connected through different steps, the reinforcement learning-based method may be implemented.

Appendix A

Paper Draft

In this appendix, a paper that summarizes the main contents of this thesis is provided. The draft is written in the format of IEEE Control Systems Letters.

Integrate Reinforcement Learning and Optimization for Real-time Railway Timetable Rescheduling

Hengkai Zhang[a], Xiaoyu Liu[a], Azita Dabiri[a], Bart De Schutter[a]

***Abstract*— The railway timetable rescheduling problem refers to a problem that requires calculating a feasible and relatively good timetable within a limited time to reduce the negative impact of disturbances or disruptions. A mixed integer linear programming (MILP) problem is typically used to describe the railway timetable rescheduling problem. In this paper, an algorithm that combines both reinforcement learning and optimization approaches is proposed to solve the railway timetable rescheduling problem. Specifically, a value-based reinforcement learning algorithm is implemented to determine the independent integer variables of the MILP problem. Then, the complete solution of the integer variables could be derived from these independent integer variables. With the solution of integer variables, the MILP problem could be transformed into a linear programming problem, which could be solved efficiently. The simulation results show that the proposed method makes a great improvement compared with the baseline regarding reducing the total delay and also gains an obvious advantage regarding time efficiency.**

***Index Terms*— Railway timetable rescheduling, MILP problem, reinforcement learning**

I. Introduction

RAILWAY plays an important role in the modern transportation system. In many countries, railway transport performs a lot of duties such as daily commuting, long-distance travel, and cargo transportation. However, the normal operation of trains is easily affected by disturbances or disruptions. There could be many reasons for railway disturbances and disruptions, such as extreme weather, worker strikes, and system failures. Railway disturbances and disruptions normally require timetable rescheduling in real time to reduce initial delays and prevent further propagation of these delays. This is the so-called railway timetable rescheduling problem [1]. An efficient rescheduling algorithm should minimize the influence of disruption and improve the punctuality of trains. Therefore, developing effective rescheduling algorithms is critical to the operation of the railway system.

Currently, there are several different problem formulations for the railway timetable rescheduling problem, such as mixed logical dynamical (MLD) systems, alternative graph (AG), fuzzy petri net (FPN) and expert system (ES), discrete event model and simulation model, etc [2]. Among all these models, MLD, and AG are the most used formulations to describe the rescheduling problem in the railway network. In fact, most railway timetable rescheduling problems will result in an integer programming (IP) problem or an MILP problem [1], [3]. In this paper, the railway timetable rescheduling problem is formulated as an MILP problem. To address this problem, many optimization-based studies have been conducted. The main advantage of using optimization-based methods is that they are easy to implement and more straightforward. However, the main disadvantage is that most optimization-based methods cannot reach the time requirements for large railway timetable rescheduling problems.

Reinforcement learning (RL) is a special kind of machine learning technique. It refers to a problem that an agent learns how to map states to actions, so as to maximize a numerical reward signal [4]. Specifically, in each time step, the agent first measures the states of the environment and then takes the corresponding action according to its current policy. Depending on different settings, the reward signal may be given at each time step or at the end of the game. The learning target of the agent is to learn an appropriate strategy so that it can maximize the final reward.

There are also a few studies that use reinforcement learning to solve the railway timetable rescheduling problem. Most literature considers a traffic controller as the agent that directly takes actions to control the railway system, which is regarded as the environment. Currently, most existing research chooses to use value-based, model-free reinforcement learning algorithms, like Q-Learning [5]–[7] and Deep Q-Network (DQN) [8]. By using reinforcement learning, the railway timetable rescheduling problem could be solved in a limited time, regardless of the problem size. However, the main disadvantage of machine learning methods is that the optimality of solutions cannot be guaranteed and they are also less robust [9]. Meanwhile, the constraint satisfaction is also a potential problem.

In this paper, our main contribution is that we propose a method that combines both reinforcement learning and optimization techniques to solve the railway timetable rescheduling problem. By selecting the independent integer variables as the action, the constraints involving the integer variables are satisfied.

[a] Delft Center for Systems and Control, Delft University of Technology, 2628 CD Delft, The Netherlands; h.zhang-39@student.tudelft.nl; x.liu-20@tudelft.nl; b.deschutter@tudelft.nl; a.dabiri@tudelft.nl

This paper is structured as follows. The first section provides a brief introduction and some background knowledge of the topic. In the second section, the formulation of the railway timetable rescheduling problem is presented. The third section introduces the reinforcement learning-based railway timetable rescheduling method we proposed. In the fourth section, three case studies are presented and discussed. The last section concludes this paper.

II. Railway Timetable Rescheduling Problem

In this paper, the railway timetable rescheduling problem is formulated as an MILP problem. The decision variables, the model of the railway timetable rescheduling problem and the complete MILP problem are provided in this section.

A. Decision Variables

In this paper, the rescheduling decisions are limited to reordering and retiming. Therefore, the continuous variables are related to the operation times of the train, and the integer variables are given as the order between different trains. Specifically, the continuous variables are defined as follows:

- $a_{i,s}$: arrival time of train i at station s
- $d_{i,s}$: departure time of train i from station s
- $\gamma_{i,s}$: dwelling time of train i in station s
- $r_{i,u,s}$: running time of train i between station u and s

Then, the integer variables consist of the departure and arrival orders between trains. The departure order is defined as follows:

$$\delta_{i,j,s} = \begin{cases} 1, & \text{if } d_{i,s} - d_{j,s} \geq 0 \\ 0, & \text{otherwise} , \end{cases} \tag{1}$$

where $\delta_{i,j,s}$ represents the departure order between train i and j at station s. Then, the arrival order is defined as:

$$\alpha_{i,j,s} = \begin{cases} 1, & \text{if } a_{i,s} - a_{j,s} \geq 0 \\ 0, & \text{otherwise} , \end{cases} \tag{2}$$

where $\alpha_{i,j,s}$ represents the arrival order between train i and j at station s.

B. Model of the Railway Timetable Rescheduling Problem

There are several constraints considered by the railway timetable rescheduling problem, including constraints of train operation, safety headways, and the availability of platforms in all stations, etc. For a train i in the railway system, the train operation should satisfy the following two basic constraints:

$$d_{i,s} = a_{i,s} + \gamma_{i,s} , \tag{3}$$

$$a_{i,s} = d_{i,s_i^-} + r_{i,s_i^-,s} , \tag{4}$$

where s_i^- represents the preceding station of station s on the route of train i. Constraint (3) described the operation of train i within station s, and constraint (4) described the operation of train i between two consecutive stations.

Besides the constraints above, there are also some lower bounds regarding the continuous decision variables. For the departure time $d_{i,s}$ and the arrival time $a_{i,s}$, the train cannot be earlier than the original time from the timetable. The running time $r_{i,u,s}$ and dwelling time $\gamma_{i,s}$ could be shorter than the original time indicated on the timetable, but they also need to have a minimum value to guarantee operation safety. These lower bounds are given by inequalities as follows:

$$\begin{aligned} d_{i,s} \geq D_{i,s} \quad a_{i,s} \geq A_{i,s} \\ r_{i,u,s} \geq R^{\min}_{i,u,s} \quad \gamma_{i,s} \geq \Gamma^{\min}_{i,s} \quad , \end{aligned} \tag{5}$$

where $D_{i,s}$ and $A_{i,s}$ represent the ideal departure time and arrival time of train i at station s from the timetable, respectively. $R^{\min}_{i,u,s}$ is the minimum running time of train i between station u and s, and $\Gamma^{\min}_{i,s}$ is the minimum dwelling time of train i at station s.

In the railway timetable rescheduling problem, safety headway constraints require that there must be a certain distance or time slot between two trains on the same track to guarantee the safe operation of the railway system. In order to investigate the headway constraint, some binary parameters need to be defined. First, the parameter indicating the departure situation is defined as follows:

$$\beta_{i,j,s} = \begin{cases} 1, & \text{if train } i \text{ and train } j \text{ depart from the same} \\ & \text{platform or to the same track at station } s \\ 0, & \text{otherwise} , \end{cases} \tag{6}$$

An illustration figure of $\beta_{i,j,s}$ is given as:

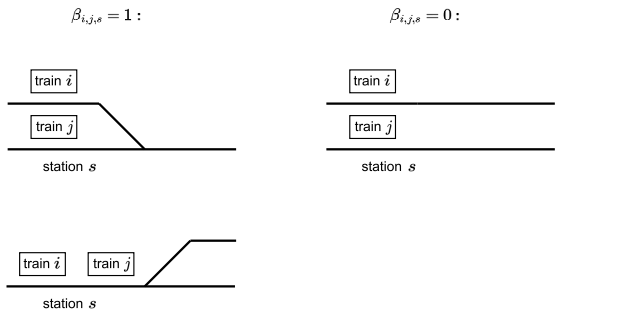

Fig. 1. Illustration Figure of the Departure Situation Parameter $\boldsymbol{\beta_{i,j,s}}$

Similarly, the parameter indicating the arrival situation is defined as follows:

$$\theta_{i,j,s} = \begin{cases} 1, & \text{if train } i \text{ and train } j \text{ arrive from the same} \\ & \text{track or to the same platform at station } s \\ 0, & \text{otherwise} , \end{cases} \tag{7}$$

An illustration figure of $\theta_{i,j,s}$ is given as follows:

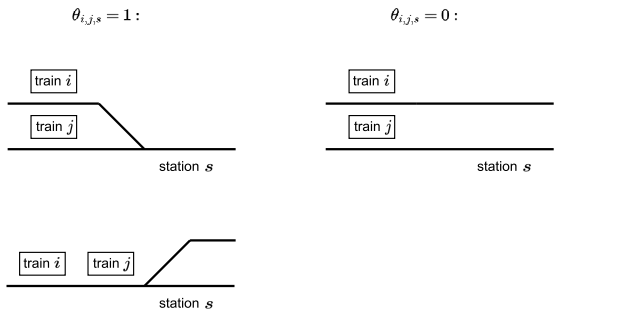

Fig. 2. Illustration Figure of the Arrival Situation Parameter $\boldsymbol{\theta_{i,j,s}}$

The definitions above indicate that the departure and arrival safety headway constraint between train i and j at station s will only need to be considered when $\beta_{i,j,s} = 1$ and $\theta_{i,j,s} = 1$, respectively.

Given definitions of the departure order in (1) and the departure situation parameter in (6), the departure safety headway constraint between any trains i and j at station s could be written as follows:

$$d_{i,s} - d_{j,s} \geq h - M(2 - \beta_{i,j,s} - \delta_{i,j,s}) \ , \tag{8}$$

where h is a time constant, representing the required minimum safety headway in the railway system, and M is a large positive constant. It is noticed that only when $\beta_{i,j,s} = 1$, does the departure headway between train i and train j need to be considered. Then, if $\delta_{i,j,s} = 0$, constraint (8) will also hold automatically. Only when $\beta_{i,j,s} = 1$ and $\delta_{i,j,s} = 1$, the constraint above is applied to guarantee the departure safety headway between train i and j at station s.

From the analysis above, it is noticed that constraint (8) could only demand the safety headway when the train i departs later than train j. In other words, both departure order $\delta_{i,j,s}$ and $\delta_{j,i,s}$ need to be determined in the MILP problem. Naturally, it requires the following constraint:

$$\delta_{i,j,s} + \delta_{j,i,s} = 1 \tag{9}$$

Similar to constraints (8) and (9), the arrival headway constraints are defined as follows:

$$a_{i,s} - a_{j,s} \geq h - M(2 - \theta_{i,j,s} - \alpha_{i,j,s}) \tag{10}$$

$$\alpha_{i,j,s} + \alpha_{j,i,s} = 1 \tag{11}$$

Similar to the analysis of departure safety headway above, only when $\theta_{i,j,s} = 1$ and $\alpha_{i,j,s} = 1$, will constraint (10) guarantee the arrival safety headway.

As mentioned before, in this paper, the rescheduling decisions considered are reordering and retiming. On the same track, it is impossible for the train behind to overtake the train in front. On different tracks, reordering is achieved automatically by retiming without influencing other trains' operations. Therefore, only the reordering happening within the station requires determining the binary variables. However, not all stations can achieve reordering. Define the number of platforms in station s as a positive integer parameter ξ_s, then the constraint of platform flexibility could be written as:

$$\sum_{j \in \mathcal{T}(s) \setminus \{i\}} (\alpha_{i,j,s} - \delta_{i,j,s}) \leq \xi_s - 1 \ , \tag{12}$$

where $\mathcal{T}(s)$ is the set of all trains that pass the station s. The left part of the constraint (12) represents the number of trains that train i overtakes at station s. Since there are only ξ_s platforms in station s, the maximal number of trains that could be overtaken by train i at station s is $\xi_s - 1$. In fact, constraint (12) reflects the relationship between arrival orders and departure orders within a station. The constraint between arrival orders and departure orders between two connected stations is defined as follows:

$$\beta_{i,j,s_i^-}\theta_{i,j,s}\alpha_{i,j,s} = \beta_{i,j,s_i^-}\theta_{i,j,s}\delta_{i,j,s_i^-} \tag{13}$$

The constraint above shows that if and only if train i and train j use the same track to travel from station s_i^- to station s, their arrival order $\alpha_{i,j,s}$ must be as same as their departure order δ_{i,j,s_i^-}.

C. Complete MILP Problem

In this paper, the objective is to minimize delays for all passengers at all stations. The objective function of the railway timetable rescheduling problem is given as follows:

$$\min \quad \sum_{s \in \mathcal{S}} \sum_{i \in \mathcal{T}} p_{i,s}(a_{i,s} - A_{i,s}) \ , \tag{14}$$

where $\mathcal{S}$ and $\mathcal{T}$ are sets of all stations and all trains respectively. Here, $p_{i,s}$ represents the number of passengers on train i with the destination of station s. The $A_{i,s}$ is the ideal arrival time of train i at station s from the pre-defined timetable. The railway timetable rescheduling problem is given as:

$$\begin{aligned}
\min \quad & \sum_{s \in \mathcal{S}} \sum_{i \in \mathcal{T}} p_{i,s}(a_{i,s} - A_{i,s}) \\
\text{s.t.} \quad & d_{i,s} = a_{i,s} + \gamma_{i,s} \\
& a_{i,s} = d_{i,s_i^-} + r_{i,s_i^-,s} \\
& d_{i,s} \geq D_{i,s} \\
& a_{i,s} \geq A_{i,s} \\
& r_{i,u,s} \geq R^{\min}_{i,u,s} \\
& \gamma_{i,s} \geq \Gamma^{\min}_{i,s} \\
& d_{i,s} - d_{j,s} \geq h - M(2 - \beta_{i,j,s} - \delta_{i,j,s}) \\
& \delta_{i,j,s} + \delta_{j,i,s} = 1 \\
& a_{i,s} - a_{j,s} \geq h - M(2 - \theta_{i,j,s} - \alpha_{i,j,s}) \\
& \alpha_{i,j,s} + \alpha_{j,i,s} = 1 \\
& \sum_{j \in \mathcal{T}(s) \setminus \{i\}} (\alpha_{i,j,s} - \delta_{i,j,s}) \leq \xi_s - 1 \\
& \beta_{i,j,s_i^-}\theta_{i,j,s}\alpha_{i,j,s} = \beta_{i,j,s_i^-}\theta_{i,j,s}\delta_{i,j,s_i^-}
\end{aligned} \tag{15}$$

The optimization problem above is an MILP problem.

III. Reinforcement Learning-based Railway Timetable Rescheduling

As discussed in the introduction part, the reinforcement learning process consists of two main components: the environment and the agent. In this section, the construction of the environment will be introduced first. Then the reinforcement learning agent will be presented.

A. Environment Settings

In this paper, the environment is a combination of the railway system and the MILP problem. Specifically, the railway system is responsible for formulating the MILP problem and making the transformation to the next step after receiving the complete solution. The MILP problem forms states to the reinforcement learning agent and encodes its action to the integer solutions, so as to transform the MILP problem into a linear programming problem. After the linear programming problem is solved, the complete solution, i.e. the new timetable, could be given to the railway system.

1) Environment Update: In this paper, the railway system is formulated as an event-triggered, time-based system, whose each time step is mathematically described by the MILP problem outlined in the previous section. For a train i arriving at station s, the railway timetable rescheduling problem is triggered if and only if the following condition holds true:

$$(r_{i,s} + a_{i,s} \geq R_{i,s} + A_{i,s}) \wedge (\xi_s > 1). \tag{16}$$

The condition above makes two requirements: one is that train i is delayed when it arrives at station s, and the other is that station s has more than 1 platform, which makes it possible for reordering.

To build the connection between consecutive steps, the concept of $\mathcal{T}^0$ is introduced as follows: $\mathcal{T}^0$ is a set of trains, which includes the most recent departed trains on each track from the initial station s_0. Using $\mathcal{T}^0$, the current step could obtain sufficient information about the previous step.

After the new $\mathcal{T}^0$ is determined, the subsequent updates of the railway system become straightforward. In the new step, several new trains are added to the railway timetable rescheduling problem. These trains will formulate the new railway timetable rescheduling problem. Since the number of trains in $\mathcal{T}^0$ remains the same in different steps, the number of new trains should also be the same.

2) State: After introducing the update of the environment, the state representation of the reinforcement learning environment is given as follows:

$$\Lambda = (\boldsymbol{p}, \boldsymbol{a}_{s_0}, \boldsymbol{\alpha}_{s_0}, \mu, \mathcal{T}^0), \tag{17}$$

where $\boldsymbol{p}$ is the vector of all passenger numbers, $\boldsymbol{a}_{s_0}$ is the arrival time vector of all trains at the initial station s_0, $\boldsymbol{\alpha}_{s_0}$ represents the vector of all arrival orders at the initial station s_0, the existing delays μ will also be explicitly given in the state representation, and $\mathcal{T}^0$ represents the numbers of trains in $\mathcal{T}^0$.

3) Action with Constraint Satisfication: The action of the reinforcement learning agent is selected as the independent integer variables of the MILP problem. There are two main reasons for not selecting all integer variables as the action. First, the action space may become too large to learn. Second, the action space may not be fully feasible, which will cause further problems with reward design and environment updates. The independent integer variables σ could be given as follows:

$$\sigma = f(\alpha, \delta; \rho), \tag{18}$$

where ρ is the parameters of the MILP problem, δ and α are all departure and arrival orders, respectively. Although finding the independent integer variables is highly related to the layout of the railway network, it is still possible to prune some binary variables according to the following principles:

- The order between two trains on the same track cannot be changed.
- At the intersection point where multiple tracks merge to one track, the order between the train which has already passed this intersection and trains that have not yet passed cannot be changed.
- The conjugate orders are always dependent on each other.

After the independent integer variables are given, the complete departure and arrival orders could be expressed as:

$$\delta = g_1(\sigma) \qquad \alpha = g_2(\sigma), \tag{19}$$

where function $g_1(\cdot)$ and $g_2(\cdot)$ are two encoding functions.

4) Reward Function: For the reward function, the negative value of the objective function given in (14) is used as follows:

$$r = -K \sum_{s \in \mathcal{S}} \sum_{i \in \mathcal{T}} p_{i,s}(a_{i,s} - A_{i,s}), \tag{20}$$

where K is a scaling constant.

B. Reinforcement Learning Algorithm

In this paper, the Double DQN [10] algorithm is implemented to train the agent. As a variant of DQN method [11], the Double DQN uses two different networks to select and evaluate the action in order to address the problem of overestimation. Specifically, the Double DQN uses the target network to evaluate and the online network to select the action. In this way, the Q values for action evaluation and selection become different.

The target employed by DQN is defined as follows:

$$y_t = \Psi_{t+1} + \tau \max_{\omega} \hat{Q}(\Lambda_{t+1}, \omega; \boldsymbol{\eta_t^-}), \tag{21}$$

where $\boldsymbol{\eta_t^-}$ represents the parameters of the target Q network for DQN. For a clear comparison, the target equation can be rewritten as:

$$y_t = \Psi_{t+1} + \tau \hat{Q}(\Lambda_{t+1}, \arg\max_{\omega} \hat{Q}(\Lambda_{t+1}, \omega; \boldsymbol{\eta_t^-}); \boldsymbol{\eta_t^-}). \tag{22}$$

It is easy to notice that the selection of action ω, is still parameterized by the same $\boldsymbol{\eta_t^-}$ as the evaluation Q.

Therefore, the Double DQN method uses two different convolutional neural networks to select and evaluate the action. In this context, the existing online network of DQN would be a natural choice. Here, Double DQN uses the online network Q to select the action and keeps the target network $\hat{Q}$ to evaluate. Then the target equation could be written as:

$$y_t = \Psi_{t+1} + \tau \hat{Q}(\Lambda_{t+1}, \arg\max_{\omega} Q(\Lambda_{t+1}, \omega, \boldsymbol{\eta_t}); \boldsymbol{\eta_t^-}), \tag{23}$$

where two networks for selection and evaluation in Double DQN are not fully decoupled, since the target network $\hat{Q}$ remains a periodic copy from the online network Q. This version of Double DQN is considered as the minimal change of DQN towards Double Q-learning.

In this paper, the Q network consists of the following layers: the first layer is a fully connected input layer with an output dimension of 256, followed by a ReLU layer for non-linearity. After that, two fully connected layers of 256×256 with ReLU activation after each of them are added. In the end, an output layer whose input dimension is also 256 finishes this Q network. An illustration figure for the neural network structure is given as follows:

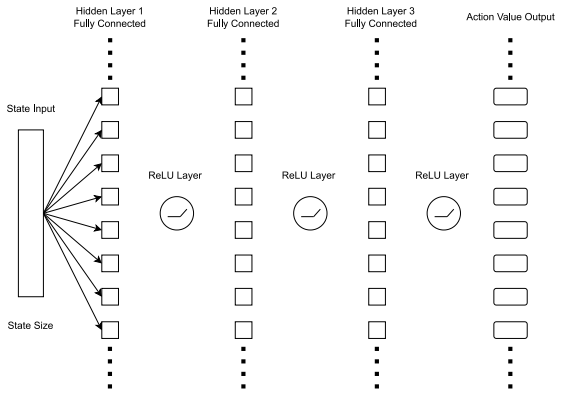

Fig. 3. Structure of the Q-Network

The following algorithm provides a procedure overview of the proposed solution in this paper.

Algorithm 1 Reinforcement Learning-based Solution to Railway Timetable Rescheduling Problem

Input: Railway system, reinforcement learning agent, initial delay, flag

Formulate the MILP problem and $\mathcal{T}^0$
Formulate the initial state $\Lambda \leftarrow \Lambda_0$ from the MILP problem
for Step $= 1$, MaxSteps **do**
  Take the state Λ, return action σ from the reinforcement learning agent
  Encode σ: $\begin{cases} \alpha \leftarrow g_\alpha(\sigma) \\ \delta \leftarrow g_\delta(\sigma) \end{cases}$
  Transform the MILP problem into linear programming problem using α and δ
  Solve the linear programming problem
  Calculate the reward Ψ
  Give the complete solution to the railway model
  Update the railway model
  Formulate new MILP problem as (15) and $\mathcal{T}^0$
  Formulate new state Λ' as (17) from the MILP problem
  if flag == Training **then**
    Store $(\Lambda, \Omega, \Psi, \Lambda')$ in the experience replay buffer
    Update the agent using Double DQN method
  end if
  $\Lambda \leftarrow \Lambda'$
end for

The complete reinforcement learning-based algorithm for the railway timetable rescheduling problem could be described as follows:

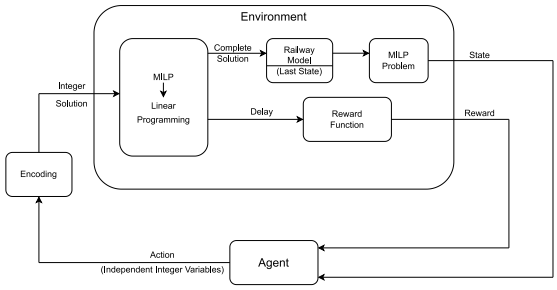

Fig. 4. Procedure for Solving the Railway Timetable Rescheduling Problem

IV. Case Study

In this paper, three case studies are presented, where part of the Dutch railway network from Utrecht (Ut) to 's-Hertogenbosch (Ht) is used for the simulation. The layout of the railway network is given as follows:

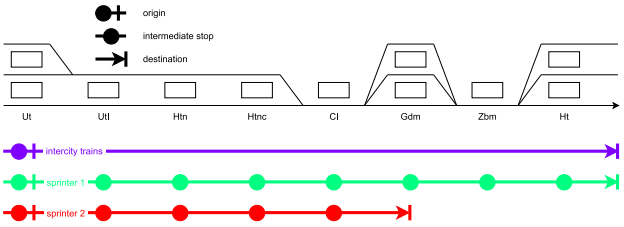

Fig. 5. Layout of the Railway Network

For the timetable, every half hour there are five trains considered. Among these trains, three are Intercity trains, and two are Sprinter trains. From the layout figure above, it can be seen that the initial station s_0 of the railway timetable rescheduling problem is set to be the Htnc station. Therefore, in the following case studies, there are only five stations to be considered: Htnc, Cl, Gdm, Zbm, and Ht.

In the MILP problem, some parameters are defined as follows. All departure parameters $\beta_{i,j,s} = 1$ and all arrival parameters $\theta_{i,j,s} = 1$. The minimum safety headway $h = 1.5$. Only Htnc and Gdm stations have two platforms, for all other stations $\xi_s = 1$. In this paper, a margin of 7% is accepted for the running time and dwelling time. These parameters are calculated as follows:

$$R^{\min}_{i,u,s} = 0.93 R_{i,u,s} \qquad \Gamma^{\min}_{i,s} = 0.93 \Gamma_{i,s} \ , \tag{24}$$

where $R_{i,u,s}$ represents the original running time of train i between station u and s from the timetable, and $\Gamma_{i,s}$ is the original dwelling time of train i at station s. These two parameters and other original times, such as $D_{i,s}$ and $A_{i,s}$, could be obtained from the pre-defined timetable.

Since there is only one track after the Htnc station, $\mathcal{T}^0$ only consists of one train, which is denoted as $\mathfrak{T}^0$. For every step, five trains are considered. From the layout of the railway network, it is clear that reordering is only possible at the Htnc and Gdm stations. For the Htnc station, since $\mathfrak{T}^0$ has already departed, there are only four trains' orders that need to be decided. Since there are two tracks, 3 independent binary variables could determine the orders of these four trains. Similarly, for the Gdm station, there are five trains and still

two platforms, 4 independent binary variables are enough. In total, there are seven independent binary variables for the railway timetable rescheduling problem at one single step. Since they are all independent, the size of action space is given as $2^7 = 128$. To reduce the dimension of the action space, these seven independent variables are encoded as one decimal integer, which is used as the final action Ω and has a range from $0 \sim 127$. For the reward function, the constant K is set to be 10^{-4}.

A. Open-loop Control

In the first case study, the railway timetable rescheduling problem is considered as an open-loop control problem. Specifically, there are no subsequent delays after the initial disturbance. Therefore, the agent only needs to eliminate the influence of the initial delay, which is given as $\mu_0 = 8 + 4 * U(0,1)$, where $U(0,1)$ represents a uniformly distributed random variable on $[0,1]$. The delay item μ in the state representation is defined as μ_0 at the initial step, and becomes the departure delay of $\mathfrak{T}^0$ at the Htnc station afterward.

The figure of episode reward during the training process is presented as follows:

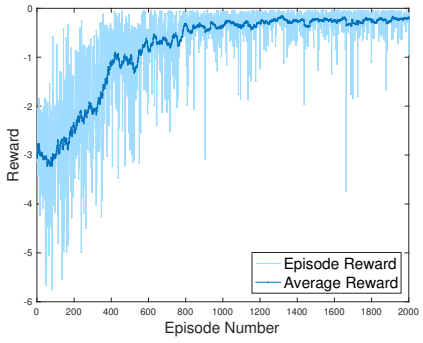

Fig. 6. Episode Reward during Training Process for Case A

From the training results above, it can be noticed that the agent learned well in this task. The total average reward after 1600 episodes is -0.25.

The testing of the trained agent has 30 episodes, which have the exact same setting as the training environment, except that the agent will not be updated anymore. For better comparison, the First-In-First-Out (FIFO) method is used as the baseline. Compared with the FIFO method, the improvement of the reinforcement learning-based method is defined as:

$$z_{\mathrm{RL}} = \left| \frac{\kappa_{\mathrm{RL}} - \kappa_{\mathrm{F}}}{\kappa_{\mathrm{F}}} \right| \times 100\% \ , \tag{25}$$

where κ_{F} and κ_{RL} are the corresponding total delay of the FIFO method and the RL-based method, respectively. Since this case is an open-loop problem, the global optimal result could also be obtained. Similarly, the improvement of the global optimal solution is defined as

$$z_{\mathrm{MILP}} = \left| \frac{\kappa_{\mathrm{MILP}} - \kappa_{\mathrm{F}}}{\kappa_{\mathrm{F}}} \right| \times 100\% \ , \tag{26}$$

κ_{MILP} is the total delay of the global optimal solution.

The testing results of the reinforcement learning agent after 30 rounds are given as follows:

TABLE I
TESTING RESULTS FOR CASE A

Method	Average Delay	Improvement	Running Time $[s]$
Baseline (FIFO)	12290	0%	5.07
RL-based Method	2157	83.49%	1.42
Optimal Solution	1699	86.78%	9.19

In 19 out of 30 testing rounds, the reinforcement learning agent obtained the global optimal solution. Compared with the baseline, the reinforcement learning agent makes a great improvement of more than 80%. Compared with the global optimal solution, the RL-based solution has a gap of about 3%.

During testing, the RL-based method took about 1.42 seconds to solve one episode on average, while the MILP solver needs about 9.19 seconds. The RL-based method saved about 85% of calculation time.

B. Closed-loop Control

The main shortcoming of the previous case study is that it does not consider the possible subsequent delays during the timetable rescheduling process. In this case study, after the initial delay, there is still a probability that some extra disturbances occur in the following steps. It is assumed that every step will have at most one extra delay.

The delay item μ is given as $\mu = (\mu_1, \mu_2)$, where μ_1 is the initial delay or the extra delay in different steps, and μ_2 is the secondary departure delay of $\mathfrak{T}^0$ at the Htnc station from the last step. The possible extra delay is given as $\mu_{ex} = 6 + 4 * U(0,1)$. The probability of adding this delay is 50%.

The figure of episode reward during the training process is presented as follows:

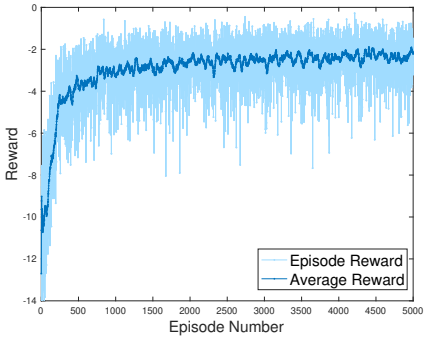

Fig. 7. Episode Reward during Training Process for Case B

From the figure above, the training result is satisfactory. The average reward after 4000 epochs is -2.3259.

Since this case is a closed-loop control problem, obtaining the global optimal solution is not practical. Therefore, a local optimization-based method is also introduced for better comparison. The main idea of this local optimization-based method is that an MILP solver will be implemented to solve

the exact same MILP problem as the reinforcement learning agent faced at every step, and then use the local optimal solution to control the railway system. Compared with the FIFO method, the improvement of this local optimization-based method is defined as:

$$z_{\mathrm{LO}} = \left| \frac{\kappa_{\mathrm{LO}} - \kappa_{\mathrm{LO}}}{\kappa_{\mathrm{LO}}} \right| \times 100\% \ , \tag{27}$$

where κ_{LO} is the total delay after 15 steps of the local optimization-based method. The testing results for these two methods after 30 rounds are given as follows:

TABLE II
TESTING RESULTS FOR CASE B

Method	Average Delay	Improvement	Running Time $[s]$
Baseline (FIFO)	12290	0%	36.21
RL-based Method	16858	61.23%	7.18
Optimization Method	9740	77.50%	10.98

From the testing results above, it is known that compared with the baseline, both algorithms make significant improvements. However, compared with Case A, the performance of both algorithms drops due to the extra delay. The gap between the improvement of these two approaches is 16.23%. Similar to the first case study, the RL-based method still has the advantage in terms of operation time. For one epoch, the RL-based method needs about 7.18 seconds to solve on average. The local optimization-based method used 10.98 seconds. The RL-based approach saves about 34.6% of time in this case.

C. Closed-loop Control with Multiple Extra Delays

In this case study, the influence of multiple extra delays will be studied. Every new train may get an extra delay in the new step. The delay item μ in the state should be a five-dimensional vector as $\mu = (\mu_1, \mu_2, \mu_3, \mu_4, \mu_5)$, where μ_1 is used to represent the secondary delay of $\mathfrak{T}^0$ after the initial step, μ_2, μ_3, μ_4 and μ_5 are the delays added to the corresponding trains. The probability of adding extra delay for each train is 50%.

The figure for the episode and average reward during the training process is given as follows:

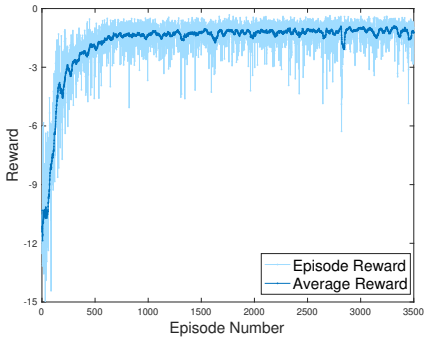

Fig. 8. Episode Reward during Training Process for Case C

From the figure above, it is noticed that the overall learning performance is good. It can be seen that the agent learned very fast in the first 500 epochs. After about 800 episodes, the learning curve has already become very stable. The average reward after 3000 epochs is -1.1636.

The test settings in this section are as same as in the last case study. The testing results are given as follows:

TABLE III
TESTING RESULTS FOR CASE C

Method	Average Delay	Improvement	Running Time $[s]$
Baseline (FIFO)	38751	0%	41.26
RL-based Method	9582	75.08%	8.17
Optimization Method	7139	81.95%	12.54

Compared with the baseline, both approaches make a great improvement. Also, compared with Case B, both methods' performances raise. Specifically, the RL-based method increases by about 14%, while the optimization-based approach improves by about 4.5%. Although the performance of the RL-based method is still worse than the optimization-based method, the gap between these two approaches is reducing. This property indicates that the reinforcement learning agent learns better when there are multiple small delays instead of one large delay. For one epoch, the RL-based method takes about 8.17 seconds on average, while the local optimization-based method takes about 12.53 seconds. The RL-based method saves about 34.8% of running time.

V. CONCLUSION

In this paper, a reinforcement learning-based method is proposed to solve the railway timetable rescheduling problem. The main idea of the proposed method is to combine reinforcement learning with optimization techniques to solve the MILP problem efficiently. The simulation results in three different settings show that the proposed method improves the performance a lot compared with the baseline. Also, the reinforcement learning method has an obvious advantage in saving running time.

REFERENCES

[1] X. Luan, Y. Wang, B. De Schutter, L. Meng, G. Lodewijks, and F. Corman, "Integration of real-time traffic management and train control for rail networks-part 1: Optimization problems and solution approaches," *Transportation Research Part B: Methodological*, vol. 115, pp. 41–71, 2018.

[2] W. Fang, S. Yang, and X. Yao, "A survey on problem models and solution approaches to rescheduling in railway networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 6, pp. 2997–3016, 2015.

[3] A. D'Ariano, D. Pacciarelli, and M. Pranzo, "A branch and bound algorithm for scheduling trains in a railway network," *European Journal of Operational Research*, vol. 183, no. 2, pp. 643–657, 2007.

[4] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT press, 2018.

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

[6] H. Khadilkar, "A scalable reinforcement learning algorithm for scheduling railway lines," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 2, pp. 727–736, 2019.

[7] Y. Zhu, H. Wang, and R. M. Goverde, "Reinforcement learning in railway timetable rescheduling," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2020, pp. 1–6.

[8] L. Ning, Y. Li, M. Zhou, H. Song, and H. Dong, "A deep reinforcement learning approach to high-speed train timetable rescheduling under disturbances," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 3469–3474.

[9] R. Tang, L. De Donato, N. Besinović, F. Flammini, R. M. Goverde, Z. Lin, R. Liu, T. Tang, V. Vittorini, and Z. Wang, "A literature review of artificial intelligence applications in railway systems," *Transportation Research Part C: Emerging Technologies*, vol. 140, p. 103679, 2022.

[10] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.

[11] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

Appendix B

Experiment Results without the Margin for Running and Dwelling Time

In the main context of the thesis, the running and dwelling time have a margin of 7% when there is a disturbance happening. This number is obtained from the practical operation of NS. In this appendix, some experiment results without this margin are presented for comparison. The only difference between these experiments and the experiments in the main context is that constraint (3-8) and (3-9) are given as follows:

$$r_{i,u,s} \geq R_{i,u,s} \tag{B-1}$$

$$\gamma_{i,s} \geq \Gamma_{i,s} \tag{B-2}$$

All other training and testing settings remain the same.

B-1 Case Study A: Open-loop Control

The figure of episode reward during the training process is presented as follows:

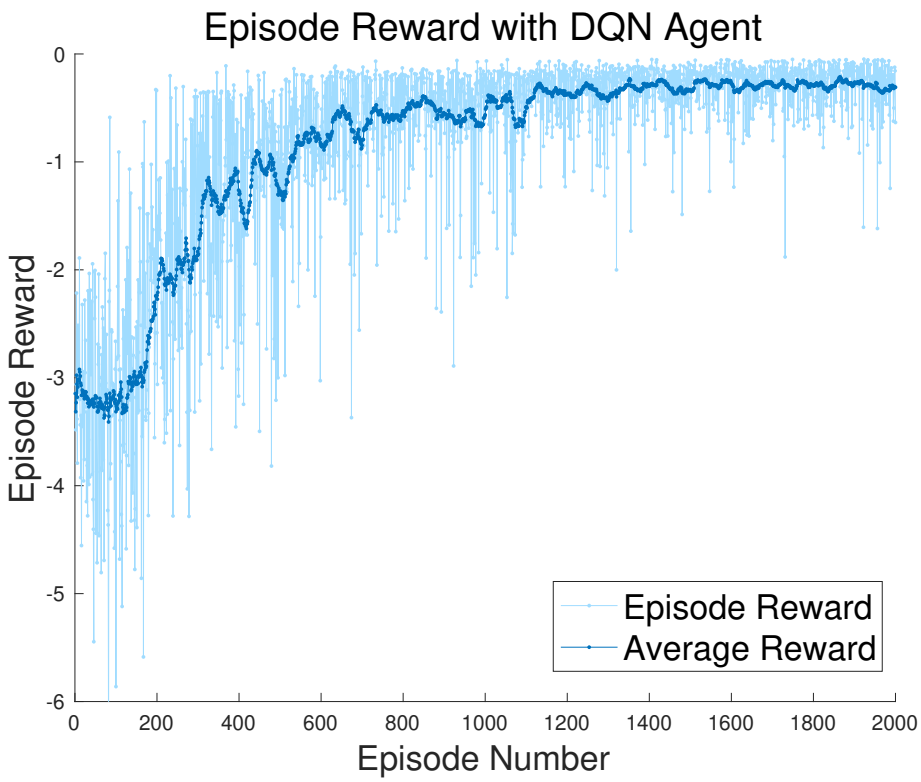

Figure B-1: Episode Reward during Training Process for Case A

The average reward after 1600 epochs is -0.2921.

The testing results of the reinforcement learning agent after 30 rounds are given as follows:

Table B-1: Testing Results for Case A

Method	Average Delay	Improvement	Running Time $[s]$
Reinforcement Learning-based Method	2399	81.85%	1.19
Global Optimal Solution	2043	84.32%	7.27

B-2 Case Study B: Closed-loop Control

The figure of episode reward during the training process is presented as follows:

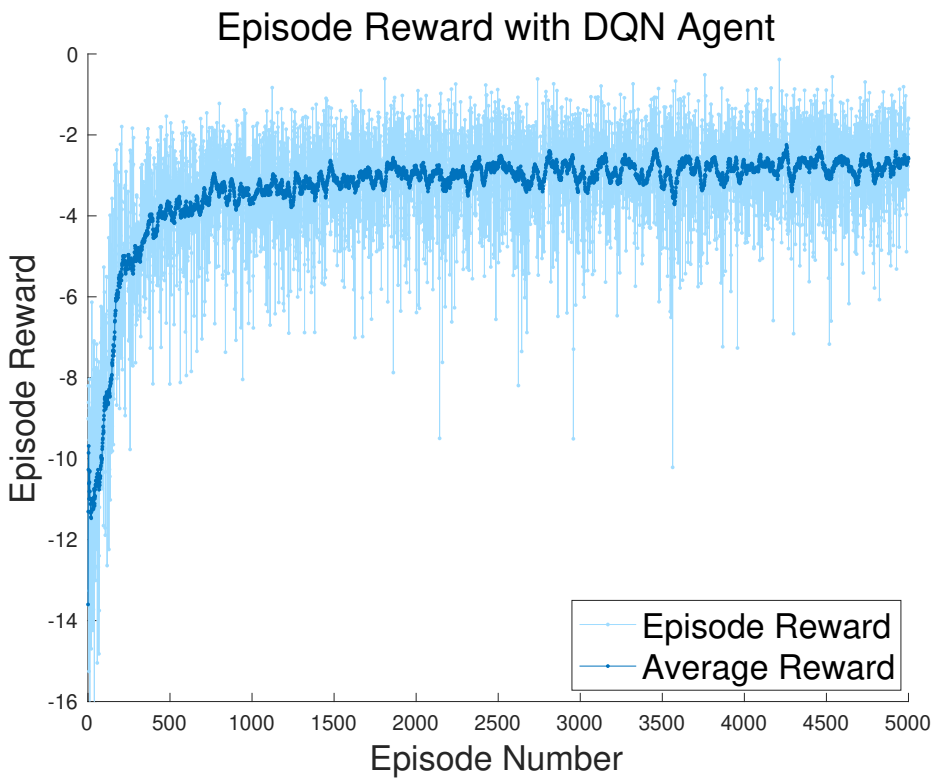

Figure B-2: Episode Reward during Training Process for Case B

The average reward after 4000 epochs is -2.7607.

The testing results of the reinforcement learning agent after 30 rounds are given as follows:

Table B-2: Testing Results for Case B

Method	Average Delay	Improvement	Running Time $[s]$
Reinforcement Learning-based Method	20567	54.26%	7.22
Local Optimization-based method	15764	67.11%	10.63

B-3 Case Study C: Closed-loop Control with Multiple Delays

The figure of episode reward during the training process is presented as follows:

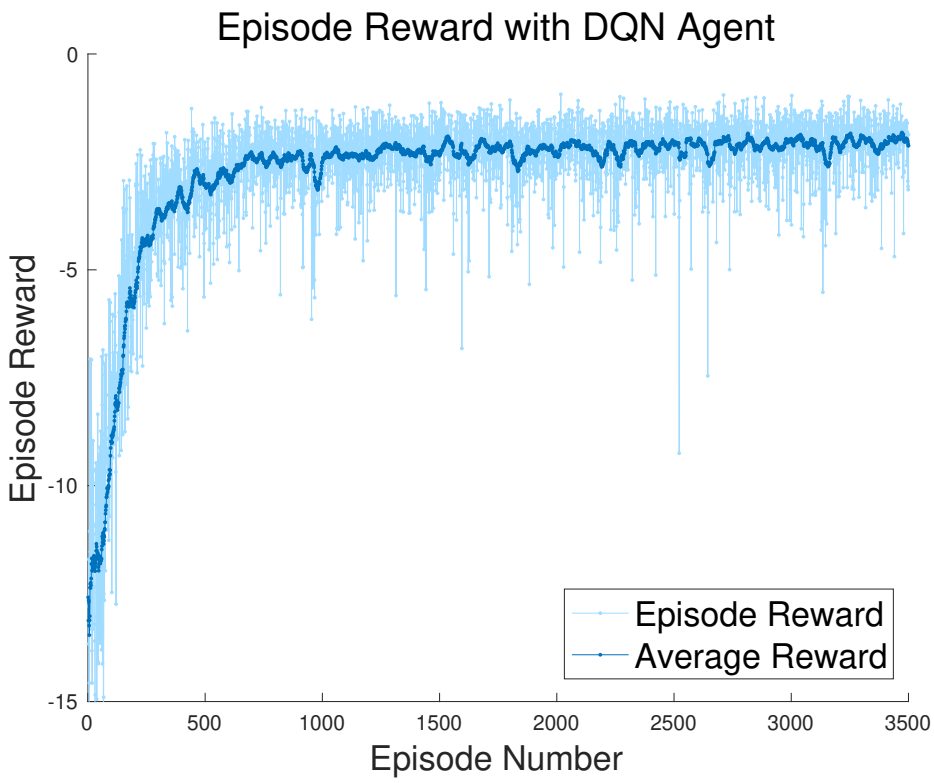

Figure B-3: Episode Reward during Training Process for Case C

The average reward after 3000 epochs is -2.0707.

The testing results of the reinforcement learning agent after 30 rounds are given as follows:

Table B-3: Testing Results for Case C

Method	Average Delay	Improvement	Running Time $[s]$
Reinforcement Learning-based Method	20428	57.36%	9.27
Local Optimization-based method	11973	73.63%	12.19

From the experiments above, it is noticed that introducing the margin of running time and dwelling time could significantly reduce the total delay. Meanwhile, the gap between the reinforcement learning-based method and the optimization-based method is also decreasing due to this margin. One possible reason is that the margin provides more fault tolerance for the reinforcement learning-based method.

Bibliography

Achiam, J., 2018. Spinning Up in Deep Reinforcement Learning .

Acuna-Agost, R., Michelon, P., Feillet, D., Gueye, S., 2011. Sapi: Statistical analysis of propagation of incidents. a new approach for rescheduling trains after disruptions. European Journal of Operational Research 215, 227–243.

Binder, S., Maknoon, Y., Bierlaire, M., 2017. The multi-objective railway timetable rescheduling problem. Transportation Research Part C: Emerging Technologies 78, 78–94.

van den Boom, T.J., Weiss, N., Leune, W., Goverde, R.M., De Schutter, B., 2011. A permutation-based algorithm to optimally reschedule trains in a railway traffic network. IFAC Proceedings Volumes 44, 9537–9542.

Cacchiani, V., Huisman, D., Kidd, M., Kroon, L., Toth, P., Veelenturf, L., Wagenaar, J., 2014. An overview of recovery models and algorithms for real-time railway rescheduling. Transportation Research Part B: Methodological 63, 15–37.

Corman, F., D'Ariano, A., Hansen, I.A., Pacciarelli, D., 2011. Optimal multi-class rescheduling of railway traffic. Journal of Rail Transport Planning & Management 1, 14–24.

Corman, F., D'Ariano, A., Pacciarelli, D., Pranzo, M., 2010. A tabu search algorithm for rerouting trains during rail operations. Transportation Research Part B: Methodological 44, 175–192.

Corman, F., D'Ariano, A., Pacciarelli, D., Pranzo, M., 2012. Bi-objective conflict detection and resolution in railway traffic management. Transportation Research Part C: Emerging Technologies 20, 79–94.

Cplex, I.I., 2021. V20.1.0: User's manual for cplex. International Business Machines Corporation 46, 157.

Davis, L., 1991. Handbook of Genetic Algorithms. New York: Van Nostrand Reinhold.

Dollevoet, T., Huisman, D., Schmidt, M., Schöbel, A., 2012. Delay management with rerouting of passengers. Transportation Science 46, 74–89.

Dorigo, M., Birattari, M., Stutzle, T., 2006. Ant colony optimization. IEEE Computational Intelligence Magazine 1, 28–39.

D'Ariano, A., Pacciarelli, D., Pranzo, M., 2007. A branch and bound algorithm for scheduling trains in a railway network. European Journal of Operational Research 183, 643–657.

Fang, W., Yang, S., Yao, X., 2015. A survey on problem models and solution approaches to rescheduling in railway networks. IEEE Transactions on Intelligent Transportation Systems 16, 2997–3016.

Gendreau, M., Potvin, J.Y., et al., 2010. Handbook of Metaheuristics. volume 2. Springer.

Ghasempour, T., Heydecker, B., 2019. Adaptive railway traffic control using approximate dynamic programming. Transportation Research Procedia 38, 201–221. Journal of Transportation and Traffic Theory.

Ghasempour, T., Nicholson, G.L., Kirkwood, D., Fujiyama, T., Heydecker, B., 2019. Distributed approximate dynamic control for traffic management of busy railway networks. IEEE Transactions on Intelligent Transportation Systems 21, 3788–3798.

Glover, F., Laguna, M., 1997. Tabu Search. Kluwer Academic Publishers.

Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. MIT press.

Kecman, P., Corman, F., D'Ariano, A., Goverde, R.M., 2013. Rescheduling models for railway traffic management in large-scale networks. Public Transport 5, 95–123.

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

Krasemann, J.T., 2012. Design of an effective algorithm for fast response to the re-scheduling of railway traffic during disturbances. Transportation Research Part C: Emerging Technologies 20, 62–78.

Lamorgese, L., Mannino, C., Piacentini, M., 2016. Optimal train dispatching by benders'-like reformulation. Transportation Science 50, 910–925.

Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D., 2015. Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971 .

Luan, X., De Schutter, B., Meng, L., Corman, F., 2020. Decomposition and distributed optimization of real-time traffic management for large-scale railway networks. Transportation Research Part B: Methodological 141, 72–97.

Luan, X., Wang, Y., De Schutter, B., Meng, L., Lodewijks, G., Corman, F., 2018. Integration of real-time traffic management and train control for rail networks-part 1: Optimization problems and solution approaches. Transportation Research Part B: Methodological 115, 41–71.

Min, Y.H., Park, M.J., Hong, S.P., Hong, S.H., 2011. An appraisal of a column-generation-based algorithm for centralized train-conflict resolution on a metropolitan railway network. Transportation Research Part B: Methodological 45, 409–429.

Mladenović, N., Hansen, P., 1997. Variable neighborhood search. Computers & Operations Research 24, 1097–1100.

Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., Kavukcuoglu, K., 2016. Asynchronous methods for deep reinforcement learning, in: International Conference on Machine Learning, PMLR. pp. 1928–1937.

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., 2015. Human-level control through deep reinforcement learning. Nature 518, 529–533.

Mu, S., Dessouky, M., 2011. Scheduling freight trains traveling on complex networks. Transportation Research Part B: Methodological 45, 1103–1123.

Nederlandse-Spoorwegen, 2021. NS Annual Report 2021.

Ning, L., Li, Y., Zhou, M., Song, H., Dong, H., 2019. A deep reinforcement learning approach to high-speed train timetable rescheduling under disturbances, in: 2019 IEEE Intelligent Transportation Systems Conference (ITSC), IEEE. pp. 3469–3474.

Pachl, J., 2002. Railway Operation and Control. VTD Rail Publishing.

Pellegrini, P., Marlière, G., Pesenti, R., Rodriguez, J., 2015. Recife-milp: An effective milp-based heuristic for the real-time railway traffic management problem. IEEE Transactions on Intelligent Transportation Systems 16, 2609–2619.

Pellegrini, P., Marlière, G., Rodriguez, J., 2014. Optimal train routing and scheduling for managing traffic perturbations in complex junctions. Transportation Research Part B: Methodological 59, 58–80.

Rummery, G.A., Niranjan, M., 1994. On-line Q-learning Using Connectionist Systems. volume 37. Citeseer.

Šama, M., Pellegrini, P., D'Ariano, A., Rodriguez, J., Pacciarelli, D., 2016. Ant colony optimization for the real-time train routing selection problem. Transportation Research Part B: Methodological 85, 89–108.

Schrijver, A., 1998. Theory of Linear and Integer Programming. John Wiley & Sons.

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

Sutton, R.S., Barto, A.G., 2018. Reinforcement Learning: An Introduction. MIT press.

Tang, R., De Donato, L., Besinović, N., Flammini, F., Goverde, R.M., Lin, Z., Liu, R., Tang, T., Vittorini, V., Wang, Z., 2022. A literature review of artificial intelligence applications in railway systems. Transportation Research Part C: Emerging Technologies 140, 103679.

Törnquist, J., Persson, J.A., 2005. Train traffic deviation handling using tabu search and simulated annealing, in: Proceedings of the 38th Annual Hawaii International Conference on System Sciences, IEEE. pp. 73a–73a.

Törnquist, J., Persson, J.A., 2007. N-tracked railway traffic re-scheduling during disturbances. Transportation Research Part B: Methodological 41, 342–362.

Van Hasselt, H., Guez, A., Silver, D., 2016. Deep reinforcement learning with double q-learning, in: Proceedings of the AAAI Conference on Artificial Intelligence.

Watkins, C.J., Dayan, P., 1992. Q-learning. Machine Learning 8, 279–292.

Williams, R.J., 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning 8, 229–256.

Xu, P., Corman, F., Peng, Q., Luan, X., 2017. A train rescheduling model integrating speed management during disruptions of high-speed traffic under a quasi-moving block system. Transportation Research Part B: Methodological 104, 638–666.

Zhu, Y., Goverde, R.M., 2019. Dynamic passenger assignment for major railway disruptions considering information interventions. Networks and Spatial Economics 19, 1249–1279.

Zhu, Y., Wang, H., Goverde, R.M., 2020. Reinforcement learning in railway timetable rescheduling, in: 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC), IEEE. pp. 1–6.

Glossary

List of Acronyms

AG	alternative graph
ADP	approximate dynamic programming
B&B	branch and bound algorithm
DQN	deep Q network
ES	expert system
FCFS	first come first served
FIFO	first in first out
FLFS	first leave first served
FPN	fuzzy patri net
FRFS	first rescheduled first served
FSFS	first scheduled first served
MDP	Markov decision process
MILP	mixed integer linear programming
MLD	mixed logical dynamical systems
NS	Nederlandse Spoorwegen
ReLU	rectified linear unit
RL	reinforcement learning

List of Symbols

$A_{i,s}$	arrival time of train i at station s from the timetable
C	updating steps
$D_{i,s}$	departure time of train i at station s from the timetable
E	dataset of experience
G_t	cumulative future reward at time step t

M	large positive constant for the safety headway constraint
N	capacity of the experience replay buffer
Q	action value function, Q-network
Q^*	optimal action value function
$R_{i,u,s}$	running time of train i between station u and s
$R^m_{i,u,s}$	minimum running time of train i between station u and s
$\Gamma_{i,s}$	dwelling time of train i at station s from the timetable
$\Gamma^m_{i,s}$	minimum dwelling time of train i at station s from the timetable
Λ_t	state at time step t
Ω_t	action at time step t
Ψ_t	reward at time step t
$\vec{\alpha}_{s_0}$	arrival order vector of trains at the initial station s_0
$\vec{a}_{s_0}$	arrival time vector of trains at the initial station s_0
$\vec{p}$	vector of the number of passengers
$\alpha_{i,j,s}$	arrival order between train i and train j at station s
$\beta_{i,j,s}$	departure relationship between train i and j at station s
$\delta_{i,j,s}$	departure order between train i and train j at station s
ϵ	probability of taking random actions
η	weights of the Q network
η^-	weights of the target Q network
$\gamma_{i,s}$	dwelling time of train i at station s
$\hat{Q}$	target Q network
λ	state
$\mathcal{S}$	set of stations
$\mathcal{T}(s)$	set of all trains that pass station s
$\mathcal{T}^0$	set of trains, which includes the most recent departed trains on each track from the initial station s_0
μ	delay for each step
μ_0	initial delay
ω	action
ϕ	data preprocessing layer
π	policy
ψ	reward
ρ	parameters of the MILP problem
σ	independent integer variables of the MILP problem
τ	discount factor
$\theta_{i,j,s}$	arrival relationship between train i and j at station s
ξ_s	number of platforms at station s
ζ	learning rate
$a_{i,s}$	arrival time of train i at station s
$d_{i,s}$	departure time of train i at station s

e_t	experience of state transformation at time step t
h	minimum safety headway
$i,\ j$	train
k	constant for the reward function
$p_{i,s}$	number of passengers on train i with destination of station s
$q_\pi(\lambda, \omega)$	the value of taking action ω in state λ under policy π
$r_{i,u,s}$	running time of train i between station u and s
s	station
s_0	initial station
s_i^-	the preceding station of station s on the route of train i
$v_\pi(\lambda)$	the value of state λ under policy π
y_t	target at time step t