Transportation Research Part C 191 (2026) 105841

Contents lists available at ScienceDirect

Transportation Research Part C

journal homepage: www.elsevier.com/locate/trc

Learning-based model predictive control for passenger-oriented train rescheduling with flexible train composition

Xiaoyu Liu [a,b,1,*], Caio Fabio Oliveira da Silva [b,1], Azita Dabiri [b], Yihui Wang [c], Bart De Schutter [b]

[a] School of Control Science and Engineering, Shandong University, Jinan, 250061, China
[b] Delft Center for Systems and Control, Delft University of Technology, 2628 CD Delft, The Netherlands
[c] School of Automation and Intelligence, Beijing Jiaotong University, Beijing, 100044, China

ARTICLE INFO

Keywords:
Urban rail transit
Train rescheduling
Time-varying passenger demands
Model predictive control
Long short-term memory networks

ABSTRACT

This paper focuses on passenger-oriented real-time train rescheduling, considering flexible train composition and rolling stock circulation, by integrating learning-based and optimization-based approaches. A learning-based model predictive control (MPC) approach is developed for real-time train rescheduling with flexible train composition and rolling stock circulation to address time-varying passenger demands. In the proposed approach, the values of the integer variables are obtained by pre-trained long short-term memory (LSTM) networks, while the continuous variables are determined through nonlinear constrained optimization. The learning-based MPC approach enables us to jointly consider efficiency and constraint satisfaction by combining learning-based and optimization-based approaches. In order to reduce the number of integer variables, four presolve techniques are developed to prune a subset of integer decision variables. Numerical simulations based on real-life data from the Beijing urban rail transit system are conducted to illustrate the effectiveness of the developed learning-based MPC approach.

1. Introduction

Urban rail transit has become increasingly important in large cities due to its reliability, high capacity, and eco-friendly characteristics. Urban rail transit systems prioritize safe and efficient train operations while providing high-quality service to passengers. Effective real-time train scheduling is essential for enhancing passenger satisfaction and minimizing operational costs within infrastructure limitations. However, the rapid expansion of urban rail transit systems and the increasing passenger demands pose significant challenges to real-time scheduling. Advanced scheduling models and control strategies are required to develop efficient timetables and to improve the overall performance of urban rail transit systems.

1.1. Passenger-oriented train scheduling

In urban rail transit systems, passenger demands vary throughout the day, necessitating adjustments to the train schedules to accommodate these demand variations while considering operational costs. One direction addresses time-varying passenger demands

* Corresponding author .
E-mail addresses: X.Liu-20@tudelft.nl, xiaoyuliu.tudelft@gmail.com (X. Liu), c.f.oliveiradasilva@tudelft.nl (C.F.O. da Silva), a.dabiri@tudelft.nl (A. Dabiri), yihui.wang1@tu-dresden.de (Y. Wang), b.deschutter@tudelft.nl (B. De Schutter).
[1] X. Liu and C.F.O. da Silva Contributed equally.

https://doi.org/10.1016/j.trc.2026.105841
Received 29 July 2025; Received in revised form 9 May 2026; Accepted 25 June 2026
Available online 3 July 2026

by optimizing the departure and arrival times of trains at each station, while considering several attributes of train operations and infrastructure restrictions, e.g., train stopping plans (the set of station stops for each train) (Cacchiani et al., 2020; Qi et al., 2021), rolling stock circulations (Wang et al., 2018), and train speed levels (Hou et al., 2019; Wang et al., 2021). Qi et al. (2021) optimized train stopping plans and timetables of a high-speed railway line considering time-varying passenger demands by formulating a mixed-integer linear programming (MILP) problem, and the aim is to find a solution that considers passenger preferences for departure times while ensuring trains operate within capacity limits. Considering the passenger load of trains, Wu et al. (2021) minimized the passenger waiting time and the energy consumption by developing a heuristic algorithm to solve the resulting nonlinear integer programming problem. Wang et al. (2021) formulated a mixed-integer nonlinear programming (MINLP) problem to minimize train energy consumption and passenger waiting times by optimizing train speed levels and headway deviations. To address train scheduling under uncertain passenger demands, Cacchiani et al. (2020) introduced a protection level on the departure and arrival times, and solved an MILP to obtain the train schedule and the stopping plan. Huang et al. (2025) investigated minimization of operation costs, passenger waiting time, and passenger travel time, and developed a variable neighborhood search approach to solve the resulting problem. However, the above studies optimize train schedules within the fixed transport capacity of a line, such as fixed train compositions or fixed train departure frequencies. As transport capacity directly impacts passenger flows, further research is required to improve passenger satisfaction by including transport capacity as a decision variable.

Another direction for train scheduling problems addresses time-varying passenger demands by optimizing transport capacity explicitly. Several studies focus on optimizing transport capacity by adjusting train departure frequencies, with higher frequencies during peak hours and lower frequencies during off-peak hours (Canca et al., 2016; Liu et al., 2023b, 2024). To handle uncertain passenger demands, Liu et al. (2024) developed a scenario-based distributed control approach, in which a scenario reduction technique is applied to reduce the computational burden. Pu and Zhan (2021) developed a two-step approach in which the first step determines the nominal line plan and the second step adjusts the time plan based on the actual measured passenger demands. Instead of changing the departure frequency, which would significantly affect the schedule, recently, many researchers have focused on optimizing the composition of the train (Ying et al., 2021; Zhao et al., 2023). Pan et al. (2023) developed a column-generation-based approach to optimize the timetable, train composition, and rolling stock circulation plan of an urban rail transit line. Their paper concludes that flexible train composition can provide additional adaptability to better match time-varying passenger demands. Wang et al. (2024) investigated flexible train composition and rolling stock circulation of an urban rail transit line, and solved the resulting MILP problem by developing an approximation approach and a two-stage meta-heuristic algorithm. Yang et al. (2024) investigated the train scheduling problem with flexible train composition for an urban rail transit line, and they applied an adaptive large neighborhood search algorithm to solve the resulting integer programming problem. As the inclusion of train composition optimization and rolling stock circulation planning introduces additional integer variables, the above studies indicate that the online computational complexity is the main challenge of incorporating flexible train composition into the real-time train scheduling problem.

1.2. MPC for real-time train scheduling

Model predictive control (MPC) has been widely adopted in various applications for its ability to handle multivariable constrained control problems (Grüne and Pannek, 2017; Mayne et al., 2000; Qin and Badgwell, 2003). The train scheduling problem is a typical constrained control problem, and many studies have applied MPC for real-time train scheduling. De Schutter et al. (2002) first applied MPC in the train scheduling problem to minimize train delays by adjusting transfer connections. Caimi et al. (2012) developed an MPC algorithm to optimize timetables, transfer connections, and train assignment plans in complex station areas. Cavone et al. (2022) proposed an MPC approach for train rescheduling during disruptions and delays, where in each step the resulting MILP problem is solved by combining bi-level heuristics and distributed optimization. The above studies only handle operator-related factors, leaving an open gap in including passenger demands in real-time train scheduling problems to improve the service quality of urban rail transit systems.

In recent years, several studies have focused on MPC for real-time passenger-oriented train scheduling. Assis and Milani (2004) applied MPC to compute the train timetable of a metro line, considering train headway and passenger load, where a linear programming problem is solved at each step. Based on a state-space model, Li et al. (2016) developed a robust MPC approach to minimize the upper bound of the timetable deviation from the nominal timetable under uncertain disturbances. By solving linear matrix inequalities, they constructed a Lyapunov function to ensure the attenuation of the timetable deviation. An event-triggered MPC approach is further developed by Wang et al. (2022) to reduce the computational burden of updating control variables in each step. However, these studies do not explicitly consider train capacity limitations and time-varying passenger demands, and the results are based on the assumption that the maximum number of passengers does not exceed the maximum train capacity. Chen et al. (2022) investigated real-time train rescheduling and stop-skipping strategies under disturbances and dynamic passenger flows. A mixed-integer quadratic programming (MIQP) problem is solved at each MPC step to minimize the total deviation from the original timetable and the number of waiting passengers. In the follow-up work, Chen et al. (2023) applied distributed MPC to address the real-time train regulation problem, using Dantzig-Wolfe decomposition to solve the resulting optimization problem. Liu et al. (2023a) explicitly incorporated time-varying passenger demands and train capacity limitations into the real-time train scheduling problem. In Liu et al. (2023a), time-varying passenger demands are approximated as piecewise constant functions by dividing the planning time window into several time intervals of equal length, and an MILP-based MPC approach is adopted for real-time train scheduling. The main challenge of applying MPC in real time is the online computational burden. Including additional attributes, such as train capacity, train composition, and rolling stock circulation, will further increase the computational burden. Therefore, further research is required to develop efficient MPC approaches for real-time passenger-oriented train scheduling.

1.3. Learning-based train scheduling

Learning-based approaches are considered effective methodologies for reducing the online computational burden. Learning-based approaches, including deep learning and reinforcement learning (RL), have also been applied in train scheduling problems in recent years (Tang et al., 2022). SL trains models on labeled data to make accurate predictions or classifications, while RL trains an agent to make decisions through trial-and-error using rewards and penalties. Kuppusamy et al. (2020) applied a deep learning approach to an energy-efficient timetable rescheduling problem, where a long short-term memory (LSTM) network is trained to select the optimal operation mode. Šemrov et al. (2016) applied Q-learning in the train rescheduling problem to reduce delays caused by disturbances and disruptions, and they illustrated their method on a single-lane track with three trains. Yin et al. (2016) proposed an approximate dynamic programming approach to address train rescheduling problems, aiming to reduce passenger delays, total travel time, and train energy consumption. In Yin et al. (2016), the states include disturbance information, train arrival times, the number of boarding passengers, and the number of waiting passengers, while the actions include rescheduling the dwell times and running times. Khadilkar (2019) applied RL to determine track allocations and timetables of bidirectional railway lines to minimize the priority-weighted delay. Ying et al. (2021) developed a proximal policy optimization approach for the train scheduling problem in an urban rail transit line, considering flexible train composition. In Ying et al. (2021), the control policy and the value function are parameterized by artificial neural networks, and scheduling constraints are handled by a devised mask scheme. Simulation results show that this approach reduces the computational burden and improves solution quality compared to the genetic algorithm and differential evolution. Nguyen and Chow (2023) introduced an adaptive rollout surrogate approximate dynamic programming (ADP) framework for rail transit network operations, where the value function is approximated by a surrogate function, and the solution is obtained via a cross-entropy-based search, encompassing decisions related to service operations and flexible train unit deployment. Ying et al. (2024) proposed a multi-agent deep reinforcement learning approach for disruption management with short-turning operations in a service corridor, enabling adaptive rescheduling under time-varying passenger demand. Wang et al. (2025) formulated adaptive network dispatching as a partially observable Markov decision process and applied a deep recurrent Q network to derive dispatching policies with flexible fleet deployment. These recent studies have shown that learning-based frameworks can support train scheduling and may incorporate time-varying passenger demands in rail networks. However, explicitly modeling time-varying passenger demands in the train scheduling problem further tightens the coupling among timetable adjustments, capacity allocation via flexible train composition, and rolling-stock circulation. This coupling enlarges the discrete decision space and increases online computational complexity, while operational feasibility still needs to be guaranteed for real-world implementation. Therefore, scalability and constraint satisfaction remain key challenges when developing learning-based train scheduling approaches.

In the above studies, scalability and constraint satisfaction are the main challenges in developing learning-based train scheduling approaches. The train scheduling problem is typically formulated as an MILP or MINLP problem, and the computational complexity increases rapidly as the number of integer variables increases. In recent years, some research has combined learning-based and optimization-based approaches for MILP or MINLP problems by using learning-based approaches to obtain the integer variables (Cauligi et al., 2022; Russo et al., 2023; da Silva et al., 2025). Having the integer variables fixed, the continuous variables are then obtained by solving a linear or nonlinear programming problem. This idea aims to combine the advantages of both learning-based and optimization-based approaches, i.e., the online computational efficiency of learning-based approaches and the feasibility guarantees of constrained optimization (Cauligi et al., 2022; Russo et al., 2023). The promising results of learning-based train scheduling and the recent learning-based approaches for mixed integer programming problems have inspired us to develop an optimization-driven learning-based framework for passenger-oriented real-time train scheduling.

1.4. Paper contributions and structure

The studies introduced in Sections 1.1–1.3 are summarized in Table 1, categorized by network details, train composition, train order, rolling stock circulation, passenger demands, and solution method. From the above literature review, it can be found that flexible train composition combined with rolling stock circulation planning is an effective approach to balance operational costs and service quality. Shorter trains can be deployed during off-peak hours and longer trains during peak hours, thereby balancing operational costs and passenger waiting times. The passenger-oriented train scheduling problem is typically formulated as a mixed-integer programming problem. Various approaches can be applied to solve the resulting scheduling problem. Among them, optimization-based methods often encounter computational challenges for large-scale networks, necessitating additional efforts such as heuristics, two-step frameworks, and distributed approaches to reduce the computational burden. Learning-based methods are considered promising for addressing computational complexity. However, developing learning-based methods that are both scalable and constraint-satisfying for real-time passenger-oriented scheduling remains challenging, especially when time-varying passenger demands are explicitly considered when optimizing flexible train composition and rolling-stock circulation in network settings.

The current paper, therefore, addresses the learning-based train rescheduling problem in rail networks, considering time-varying passenger demands, flexible train composition, and rolling stock circulation. Unlike existing train rescheduling methods in the literature, the proposed approach integrates optimization and deep learning to combine the constraint satisfaction advantages of optimization-based methods with the fast online computation advantages of learning-based approaches. The main contributions of the paper are listed as follows:

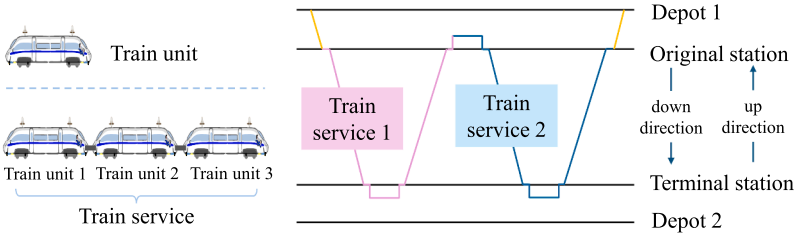

Fig. 1. Illustration of definitions used in this paper.

1. A passenger-oriented train rescheduling model in Liu et al. (2023a) is extended to include time-varying sectional passenger demands, flexible train composition, and rolling stock circulation. The time-varying passenger demands can be approximated as a piecewise constant function by dividing the prediction horizon into several time intervals.
2. Four presolve techniques are developed to streamline optimization processes by pruning a subset of integer decision variables. After implementing the presolve techniques, long short-term memory (LSTM) networks are applied to obtain the remaining integer variables with reduced dimensions.
3. A deep-learning-based MPC approach is developed for real-time train rescheduling. To improve the online computational efficiency of MPC, the learning-based approach is applied to obtain integer variables, while the detailed timetable is obtained by solving a constrained optimization problem with the fixed integer variables.

The remaining part of the paper is organized as follows: Section 2 provides the problem description and general explanations. Section 3 introduces the passenger-oriented train rescheduling model. In Section 4, the problem formulation and the MINLP-based MPC approach are presented. In Section 5, we propose a learning-based MPC approach for real-time train rescheduling. Section 6 provides an illustrative case study. Section 7 concludes the paper.

2. Problem description and explanations

In this section, we present the problem description of train rescheduling with flexible train composition. Some general explanations and assumptions used in the problem formulation are summarized as follows.

(1) In urban rail transit, a *line* is defined as a fixed route that connects an ordered sequence of stations and is served by trains operating in both directions.
(2) A *train unit* is the minimum rolling-stock element that can operate independently and that can be coupled/decoupled; multiple train units form a train composition.
(3) A *train service* is defined as a train departure from its origin station, visiting every station in the line, and finally returning to the depot (or connecting to other train services at the depot). As illustrated in Fig. 1, a train service consists of one or several train units, and the composition can be changed at the station connected to a depot by adding or removing train units.
(4) Since passenger volumes are typically large, the approximation error incurred by relaxing passenger counts to continuous variables is relatively small (Liu et al., 2023a). Hence, variables representing the number of passengers are modeled as nonnegative real-valued variables.

In this paper, we jointly optimize train compositions and timetables considering time-varying passenger demands and rolling stock circulation. We consider the regular departure of trains and focus on the train rescheduling problem that jointly optimizes departure times, arrival times, and train composition. In this context, both train travel times and transport capacity are adjusted to accommodate the time-varying passenger demands. We aim to minimize the total waiting time of passengers and the train energy consumption, and the control actions are the train composition, train service orders at the platforms connected to the same depot, and departure/arrival times. Flexible train composition and rolling stock circulation relate to the set of integer variables, while train timetables relate to the set of continuous variables, such as departure and arrival times. Trains operate under several constraints, including train capacity constraints, train availability in a depot, and headway constraints. By applying MPC, we solve the resulting mixed-integer programming problem in a moving horizon manner for real-time train rescheduling. To improve the online computational efficiency of MPC, we use the learning-based approach to obtain integer variables, i.e., train compositions and train orders; then, we optimize the detailed timetable with the fixed integer variables by solving a constrained optimization problem.

3. Mathematical formulation for passenger-oriented train rescheduling

In this section, we develop a passenger-oriented train rescheduling model for urban rail transit systems. The notations for the model formulation are provided in Section 3.1. The train operation constraints of the model are introduced in Section 3.2. In Section 3.3, the rolling stock circulation constraints related to the model are introduced. In Section 3.4, passenger flow constraints of the model are presented.

Table 1
Summary of relevant studies on passenger-oriented train rescheduling.

Publications	Infrastructure	Train composition	Train order	Rolling stock circulation	Passenger demands	Solution method	Objective (s)
Šemrov et al. (2016)	single line	fixed	yes	no	no	Q learning	minimize the total delay of trains
Li et al. (2016)	single line	fixed	no	no	yes	robust MPC	minimize the deviations form the nominal timetable
Khadilkar (2019)	single line	fixed	yes	no	no	Q learning	minimize the priority-weighted delay
Cacchiani et al. (2020)	single line	fixed	yes	no	yes	robust MPC	minimize train travel time and unsatisfied passenger demands
Kuppusamy et al. (2020)	single line	fixed	no	no	no	deep learning	minimize energy consumption
Pu and Zhan (2021)	single line	fixed	no	no	yes	Lagrangian relaxation	minimize operational costs and total passenger travel time
Wang et al. (2021)	single line	fixed	no	yes	yes	MILP	minimize train energy consumption and passenger waiting times
Ying et al. (2021)	single line	flexible	no	yes	yes	deep reinforcement learning	minimize the operation cost and passenger waiting times
Cavone et al. (2022)	network	fixed	yes	no	no	bi-level heuristics, MPC	minimize train delays
Wang et al. (2022)	single line	fixed	no	no	yes	event-triggered MPC	minimize the deviations from the nominal timetable
Chen et al. (2022)	single line	fixed	no	yes	yes	MPC	minimize total train deviation an the number of waiting passengers
Liu et al. (2023a)	network	fixed	no	no	yes	MPC	minimize the waiting time of passengers
Pan et al. (2023)	single line	flexible	yes	yes	yes	column-generation-based heuristic	minimize passenger waiting times and the usages of train units
Liu et al. (2023b)	network	fixed	yes	yes	yes	bi-level MPC	minimize passenger travel times and the total operation costs
Nguyen and Chow (2023)	network	flexible	no	yes	yes	rollout surrogate-ADP	minimize total waiting time of passengers
Ying et al. (2024)	single line	flexible	no	yes	yes	multi-agent deep RL	minimize passenger travel times and the total operation costs
Liu et al. (2024)	network	fixed	no	yes	yes	scenario-based distributed MPC	minimize passenger travel times and the total operation costs
Wang et al. (2024)	single line	flexible	no	yes	yes	two-stage heuristic	reduce the number of stranded passengers, the number of rolling stocks, and the operational costs
Yang et al. (2024)	single line	flexible	no	yes	yes	adaptive large neighborhood search	minimize total waiting time of passengers and the number of train units used

Table 1
continued.

Publications	Infrastructure	Train composition	Train order	Rolling stock circulation	Passenger demands	Solution method	Objective (s)
Wang et al. (2025)	network	flexible	no	yes	yes	deep RL	minimize passenger travel times and the total operation costs
Huang et al. (2025)	single line	fixed	no	yes	yes	variable neighborhood search	minimize passenger waiting time, travel time and operation costs
Current paper	network	flexible	yes	yes	yes	deep-learning-based MPC	minimize passenger waiting times and operation costs

Table 2
Indices and input parameters.

Notations	Definition
p	Index of platforms, $p \in \mathcal{P}$; $\mathcal{P}$ is the set of platforms
k_p	Index of train services at platform p, $k_p \in \mathcal{I}_p$; $\mathcal{I}_p$ is the set of train services departing from platform p
z	Index of depots, $z \in \mathcal{Z}$; $\mathcal{Z}$ is the set of depots
$\mathrm{p}^{\mathrm{pla}}(p)$	Predecessor platform of platform p
$\mathrm{s}^{\mathrm{pla}}(p)$	Successor platform of platform p
$d_p^{\mathrm{pre}}(k_p)$	Predetermined departure time of train service k_p at platform p
$h_p^{\min}$	Minimum departure-arrival headway at platform p
$r_p^{\min}$	Minimum running time of trains from platform p to its succeeding platform
$r_p^{\max}$	Maximum running time of trains from platform p to its succeeding platform
$C_{\max}$	Maximum capacity of a train unit
$\ell^{\min}$	Minimum number of train units allowed to be included in any train service
$\ell^{\max}$	Maximum number of train units allowed to be included in any train service
σ_p	Parameter indicating whether the train composition can be adjusted at platform p, i.e., if train composition can be adjusted at platform p, then, $\sigma_p = 1$, otherwise, $\sigma_p = 0$.
$\tau_p^{\min}$	Minimum dwell time of a train service at platform p
t_p^{comp}	Time required for changing the train composition at platform p
$t_{p,p'}^{\mathrm{roll}}$	Time for train units from platform p to other platform p' corresponding to the same depot
$\mathrm{dep}(z)$	Set of platforms directly connected with depot z
$\mathrm{pla}(p)$	Set of platforms belonging to the same station as platform p
$\rho_p(k_p)$	Passenger demands from platform p to its successor platform during $d_{k-1,p}^{\mathrm{pre}}$ to $d_{k,p}^{\mathrm{pre}}$
N_z^{train}	Total number of train units available at depot z
$\chi_{k_q,q,k_p,p}$	Binary parameter, which can be determined based on the predetermined timetable; if train k_q at platform q has transfer connection with train k_p at platform p, $\chi_{k_q,q,k_p,p} = 1$; otherwise, $\chi_{k_q,q,k_p,p} = 0$.
$\beta_{q,p}$	Transfer rate from platform q to platform p
t_q^{trans}	Average transfer time from platform q to the platforms at the same station
E_p^{energy}	Average energy consumption for a train unit running from platform p to its successor platform
E_p^{add}	Additional cost of changing train composition at platform p

Table 3
Decision variables.

Notations	Definition
$d_p(k_p)$	Departure time of train service k_p at platform p
$a_p(k_p)$	Arrival time of train service k_p at platform p
$\ell_p(k_p)$	Number of train units included in train service k_p at platform p, $\ell_p(k_p) \in \mathbb{Z}_+$
$\xi_{k_p,k_{p'},p,p'}$	Binary variable; if the train units from train service $k_{p'}$ at platform p' can be used for train service k_p at platform p, $\xi_{k_p,k_{p'},p,p'} = 1$; otherwise, $\xi_{k_p,k_{p'},p,p'} = 0$.
$\eta_{k_p,p}$	Binary variable; if the composition of train service k_p is changed at platform p, $\eta_{k_p,p} = 1$; otherwise, $\eta_{k_p,p} = 0$.

3.1. Notations

Tables 2–4 list the indices and input parameters, the decision variables, and the output variables of the model formulations, respectively.

Table 4
Output variables.

Notations	Definition
$\tau_p(k_p)$	Dwell time of train service k_p at platform p
$h_p(k_p)$	Departure-arrival headway between train service k_p and train service k_p+1 at platform p
$r_p(k_p)$	Running time of train service k_p from platform p to its successor platform
$r_p^{\text{turn}}(k_p)$	Turnaround time of train service k_p at platform p
$\tau_p^{\text{add}}(k_p)$	Additional time required for changing the composition of train service k_p at platform p
$y_p(k_p)$	Number of train units coming to/from the depot for train service k_p, $y_p(k_p) \in \mathbb{Z}$
$n_p(k_p)$	Number of passengers waiting at platform p at time $d_p^{\text{pre}}(k_p)$
$n_p^{\text{trans}}(k_p)$	Number of transfer passengers arriving at platform p for train k_p
$n_p^{\text{depart}}(k_p)$	Number of passengers departing from platform p with train service k_p
$n_p^{\text{arrive}}(k_p)$	Number of passengers arriving at platform p with train k_p from the predecessor platform $\text{p}^{\text{pla}}(p)$
$n_p^{\text{before}}(k_p)$	Number of passengers waiting at platform p immediately before the departure of train service k_p
$C_p(k_p)$	Total capacity of train service k_p at platform p
$n_p^{\text{after}}(k_p)$	Number of passengers waiting at platform p immediately after train service k_p departs from platform p.

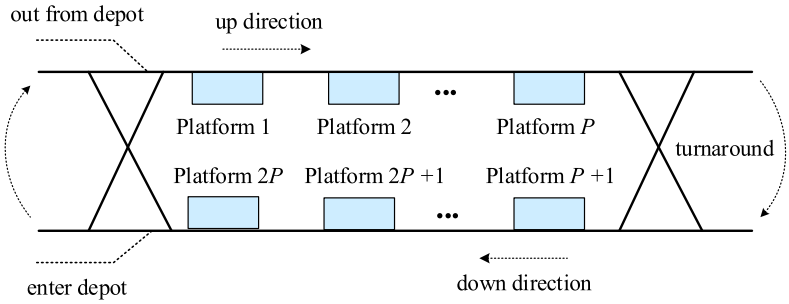

Fig. 2. Layout of a bidirectional urban rail transit line.

3.2. Train operation constraints

In urban rail transit systems, each line typically comprises two directions, i.e., the up direction and the down direction, as shown for a line of P stations in Fig. 2. For each line, a train service can be defined as a train running from the starting platform to the terminal platform, e.g., from Platform 1 to Platform $2P$ in Fig. 2.

Trains generally operate following a predetermined timetable, and the predetermined departure time of train service k_p at platform p is represented by $d_p^{\text{pre}}(k_p)$.

In practice, premature departure is usually not permitted; thus, the actual departure time cannot be earlier than the predetermined departure time. In general, passengers prefer a regular timetable at each platform so that they can conveniently plan their trips. Therefore, in this paper, we keep the planned service schedule unchanged, i.e., we neither insert nor cancel train services when adjusting the timetable and train compositions. In this context, we define a cycle, and each cycle includes only one departure at each platform. Hence, the actual departure time is constrained by

$$d_p^{\text{pre}}(k_p) \le d_p(k_p) < d_p^{\text{pre}}(k_p+1), \tag{1}$$

where $d_p(k_p)$ is the actual departure time of train service k_p at platform p. Instead of adjusting the departure frequency, we optimize train composition under a regular timetable with constant departure frequency.[1] In this setting, (1) defines a *cycle* as the time interval between two consecutive predetermined departures, and each cycle contains exactly one departure at each platform. The departure time $d_p(k_p)$ is determined by

$$d_p(k_p) = a_p(k_p) + \tau_p(k_p), \tag{2}$$

where $a_p(k_p)$ and $\tau_p(k_p)$ are the arrival time and dwell time of train service k_p at platform p.

For the safe operation of trains, the headway constraint should be satisfied:

$$a_p(k_p+1) = d_p(k_p) + h_p(k_p), \tag{3}$$

$$h_p(k_p) \ge h_p^{\min}, \tag{4}$$

where $h_p(k_p)$ is headway of train service k_p at platform p , and $h_p^{\min}$ denotes the minimum headway.

[1] For a regular timetable, longer trains can be used during peak hours and shorter trains during off-peak hours. In this way, passengers can still have short waiting times even in off-peak periods, which is preferable to reducing departure frequency under a fixed train length.

The arrival time of train service k_p at the successor platform of platform p should also satisfy

$$a_{s^{pla}(p)}(k_p) = d_p(k_p) + r_p(k_p), \tag{5}$$

$$r_p^{min} \le r_p(k_p) \le r_p^{max}, \tag{6}$$

where $r_p(k_p)$ is the running time of train service k_p from platform p to platform $s^{pla}(p)$, and r_p^{min} and r_p^{max} are the minimum and maximum running time from platform p to platform $s^{pla}(p)$.

3.3. Rolling stock circulation constraints

At the terminal station, a turnaround action is required for the continuation of the train service. The turnaround constraints can be formulated as:

$$a_{s^{pla}(p)}(k_{s^{pla}(p)}) = d_p(k_p) + r_p^{turn}(k_p), \tag{7}$$

$$r_p^{turn,min} \le r_p^{turn}(k_p) \le r_p^{turn,max}, \tag{8}$$

where $r_p^{turn}(k_p)$ represents the turnaround time of train service k_p at platform p, and $r_{p,min}^{turn}$ and $r_{p,max}^{turn}$ denote the minimum and maximum turnaround times at platform p.

An urban rail transit line typically has a limited number of train units that either operate on the line or are stored in the depot. The train composition can be adjusted at the platform that is linked with the depot, and the number of train units $\ell_p(k_p) \in \mathbb{Z}_+$ for train service k_p at platform p is determined by

$$\ell_p(k_p) = \ell_{p^{pla}(p)}(k_p) + \sigma_p y_p(k_p), \tag{9}$$

$$\ell_{min} \le \ell_p(k_p) \le \ell_{max}, \tag{10}$$

where σ_p is a parameter related to the network layout indicating whether the train composition can be adjusted at platform p, i.e., if the train composition can be adjusted at platform p, then $\sigma_p = 1$, otherwise, $\sigma_p = 0$. Moreover, $y_p(k_p) \in \mathbb{Z}$ represents the number of train units coming to/from the depot for train service k_p; specifically, if $y_p(k_p) > 0$, then, $y_p(k_p)$ extra train units will come from the depot to be added to train service k_p; if $y_p(k_p) < 0$, train service k_p will be decomposed, and $|y_p(k_p)|$ train units will return to the depot; if $y_p(k_p) = 0$ the composition of train service k_p will not be changed at platform p. Furthermore, ℓ_{min} and ℓ_{max} represent the minimum and the maximum number of train units allowed to be included in any train service.

Remark 1. If the depot is linked with the terminal platform (e.g., Platform 1 in Fig. 2) and train service k_p is performed by the turnaround train units of the previous train service, then (9) becomes $\ell_1(k_1) = \ell_{2P}(k_{2P} - 1) + \sigma_1 y_1(k_1)$, i.e., we set $\ell_1(k_1) = \ell_{2P}(k_{2P} - 1)$.

To capture the changes of train composition at platform p, we introduce a binary variable $\eta_{k_p,p}$ as

$$\eta_{k_p,p} = \begin{cases} 1, & \text{if} \quad |y_p(k_p)| > 0; \\ 0, & \text{otherwise}, \end{cases} \tag{11}$$

where $\eta_{k_p,p} = 1$ indicates the composition of train service k_p is changed at platform p; otherwise, $\eta_{k_p,p} = 0$.

In general, additional dwell time is required when changing the train composition; thus, the dwell time $\tau_p(k_p)$ of train service k_p at platform p should satisfy:

$$\tau_p(k_p) \ge \tau_p^{min} + \sigma_p \tau_p^{add}(k_p), \tag{12}$$

where τ_p^{min} is the minimum dwell time at platform p, and $\tau_p^{add}(k_p)$ represents the additional time required for changing the composition of train service k_p at platform p, which can be determined by

$$\tau_p^{add}(k_p) = \eta_{k_p,p} \cdot t_p^{comp}, \tag{13}$$

where t_p^{comp} represents the time required for changing the train composition at platform p.

A depot typically connects to at least one platform, and the departure order of train services at the corresponding platforms influences the number of available train units in a depot, i.e., if a newly arriving train service requires changing its composition, the total number of train units in the corresponding depot will change. To represent the relation of departure time of train services corresponding to the same depot, we define a binary variable $\xi_{k_p,k_{p'},p,p'}$ as

$$\xi_{k_p,k_{p'},p,p'} = \begin{cases} 1, & \text{if} \quad d_p(k_p) \ge d_{p'}(k_{p'}) + t_{p',p}^{roll}; \\ 0, & \text{otherwise}, \end{cases} \tag{14}$$

where $d_p(k_p)$ is the departure time of train service k_p at platform p, $d_{p'}(k_{p'})$ is the departure time of train service $k_{p'}$ at platform p', and $t_{p',p}^{roll}$ is the time for trains from platform p' to platform p corresponding to the same depot. In (14), $\xi_{k_p,k_{p'},p,p'} = 1$ indicates train units assigned to train service $k_{p'}$ from platform p' can be reassigned (i.e., are available) for train service k_p at platform p; otherwise, $\xi_{k_p,k_{p'},p,p'} = 0$.

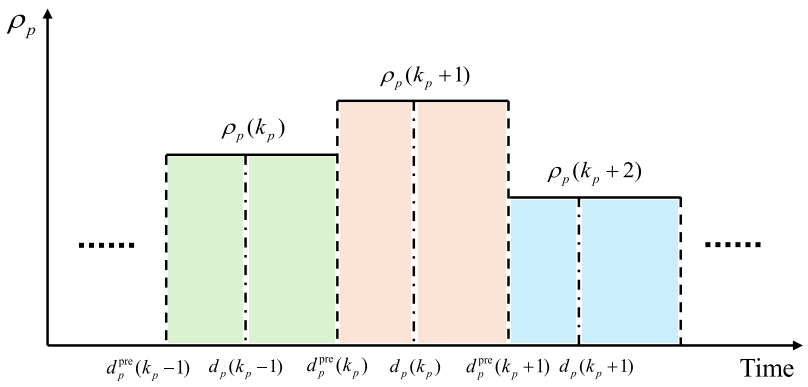

Fig. 3. Illustration of piecewise approximation of passenger arrival rates for platform p.

Let $\mathcal{I}_p$ denote the set of train services departing from platform p, and let $\mathcal{I}_p(k_p) = \{k \in \mathcal{I}_p \mid k \le k_p\}$ denote the set of services at platform p up to and including service k_p. The rolling-stock circulation constraint is

$$\sum_{k'_p \in \mathcal{I}_p(k_p)} y_p(k'_p) + \sum_{p' \in \mathrm{dep}(z)\setminus\{p\}} \sum_{k_{p'} \in \mathcal{I}_{p'}} \xi_{k_p,k_{p'},p,p'}\, y_{p'}(k_{p'}) \le N_z^{\mathrm{train}}. \tag{15}$$

Here, z is the index of the depot, $\mathrm{dep}(z)$ defines the set of platforms directly connected with depot z, and N_z^{train} denotes the total number of train units in depot z. Eq. (15) ensures that, for each service k_p, the depot inventory remains feasible under both dispatching ($y_p(k_p) > 0$) and returning ($y_p(k_p) < 0$) of train units, and the total number of train units in depot z is bounded by N_z^{train}. In this context, rolling stock can be reused across different services within the same line through the circulation constraints. At depot-linked platforms, a train unit can either return to the depot or be assigned to subsequent services on the same line as long as the rolling stock circulation constraints are satisfied.

3.4. Passenger flow constraints

A predetermined timetable is generally designed based on time-varying passenger demands to guide daily train operations, and passenger flows are updated over consecutive predetermined departure times at each platform, i.e., over each cycle at each platform. At each platform, the predetermined timetable naturally divides the planning time window into several time intervals, with the predetermined train departure times at the platforms as the partition points. Following the previous research in Liu et al. (2023a), which demonstrated that the piecewise constant approximation of the time-varying passenger demands can significantly reduce computational complexity while preserving comparable accuracy, in this paper, we also approximate the passenger arrival rates as piecewise constant functions. The resulting piecewise approximation is shown in Fig. 3 where $d_p^{\mathrm{pre}}(k_p)$ is the predetermined departure time of train service k_p; and $\rho_p(k_p)$ denotes the passenger arrival rate during $d_p^{\mathrm{pre}}(k_p - 1)$ to $d_p^{\mathrm{pre}}(k_p)$ for passengers aiming to leave platform p with train service k_p.

The number of passengers waiting at a platform immediately after the predetermined departure time of train service $k_p + 1$ at platform p can be calculated by

$$n_p(k_p + 1) = n_p(k_p) + \rho_p(k_p + 1)\big(d_p^{\mathrm{pre}}(k_p + 1) - d_p^{\mathrm{pre}}(k_p)\big) + n_p^{\mathrm{trans}}(k_p) - n_p^{\mathrm{depart}}(k_p), \tag{16}$$

where $n_p(k_p)$ represents the number of passengers waiting at platform p immediately before the predetermined departure time $d_p^{\mathrm{pre}}(k_p)$; the term $\rho_p(k_p{+}1)\big(d_p^{\mathrm{pre}}(k_p{+}1) - d_p^{\mathrm{pre}}(k_p)\big)$ gives the number of passengers arriving at platform p between $d_p^{\mathrm{pre}}(k_p)$ and $d_p^{\mathrm{pre}}(k_p{+}1)$; $n_p^{\mathrm{trans}}(k_p)$ denotes the number of transfer passengers arriving at platform p who intend to board train service k_p, and $n_p^{\mathrm{depart}}(k_p)$ denotes the number of passengers departing from platform p on train service k_p.

Since the departure time of each train service is adjusted according to (1), there exists one departure in each time interval. As shown in Fig. 3, the number of passengers $n_p^{\mathrm{before}}(k_p)$ waiting at platform p immediately before the departure of train service k_p can be calculated by

$$n_p^{\mathrm{before}}(k_p) = n_p(k_p) + \rho_p(k_p + 1)\big(d_p(k_p) - d_p^{\mathrm{pre}}(k_p)\big) + n_p^{\mathrm{trans}}(k_p). \tag{17}$$

The number of passengers $n_p^{\mathrm{depart}}(k_p)$ departing from platform p with train service k_p should satisfy

$$n_p^{\mathrm{depart}}(k_p) \le n_p^{\mathrm{before}}(k_p), \tag{18a}$$

$$n_p^{\mathrm{depart}}(k_p) \le C_p(k_p), \tag{18b}$$

where $C_p(k_p)$ is the total capacity of train service k_p at platform p. Constraint (18) provides a standard linear approximation of $n_p^{\mathrm{depart}}(k_p) = \min\{n_p^{\mathrm{before}}(k_p),\, C_p(k_p)\}$, which is commonly used in related studies (e.g., Luan and Corman, 2022; Pan et al., 2023; Wang et al., 2024).

The total capacity of train service k_p at platform p is computed by

$$C_p(k_p) = \ell_p(k_p) C_{\max}, \tag{19}$$

where $\ell_p(k_p)$ is the number of train units composing train service k_p at platform p.

The number of passengers $n_p^{\text{arrive}}(k_p)$ arriving at platform p with train k_p from the predecessor platform $\text{p}^{\text{pla}}(p)$ can be calculated by

$$n_p^{\text{arrive}}(k_p) = n_{\text{p}^{\text{pla}}(p)}^{\text{depart}}(k_p). \tag{20}$$

Then, the number of passengers $n_p^{\text{trans}}(k_p)$ transferring to platform p for train k_p is computed by

$$n_p^{\text{trans}}(k_p) = \sum_{q \in \text{pla}(p)} \sum_{k_q \in \mathcal{I}_q} \chi_{k_q,q,k_p,p} \beta_{q,p} n_q^{\text{arrive}}(k_q), \tag{21}$$

where $\text{pla}(p)$ defines the set of platforms belonging to the same station as platform p, $\beta_{q,p}$ is the parameter defines the transfer rate from platform q to platform p, and $\chi_{k_q,q,k_p,p}$ is the binary parameter denoting transfer connection between train k_q at platform q and train k_p at platform p, which is defined as

$$\chi_{k_q,q,k_p,p} = \begin{cases} 1, & \text{if } d_p^{\text{pre}}(k_p - 1) < d_q^{\text{pre}}(k_q) + t_q^{\text{trans}} \le d_p^{\text{pre}}(k_p); \\ 0, & \text{otherwise}, \end{cases} \tag{22}$$

where t_q^{trans} represents the average transfer time from platform q to the corresponding platforms at the same station. To improve the accuracy of the passenger flow model, (21) can be directly extended by making the transfer-rate parameter $\beta_{q,p}$ time-dependent, i.e., using $\beta_{q,p}(k_p)$, which can be estimated online from real-time passenger flow data.

After train service k_p departs from platform p, the number of passengers waiting at the platform can be calculated by

$$n_p^{\text{after}}(k_p) = n_p^{\text{before}}(k_p) - n_p^{\text{depart}}(k_p), \tag{23}$$

where $n_p^{\text{after}}(k_p)$ represents the number of passengers waiting at platform p immediately after train service k_p departs from platform p.

In summary, passenger flows are updated over consecutive predetermined departure times at each platform. For platform p, considering a cycle defined by two consecutive predetermined departure times $d_p^{\text{pre}}(k_p)$ and $d_p^{\text{pre}}(k_p{+}1)$, the number of arriving passengers in this interval is given by $\rho_p(k_p{+}1)\big(d_p^{\text{pre}}(k_p{+}1) - d_p^{\text{pre}}(k_p)\big)$. Consequently, the number of passengers waiting before train service $k_p{+}1$ equals the number of passengers remaining after service k_p plus the arrivals in this interval. The number of boarded (departing) passengers $n_p^{\text{depart}}(k_p)$ is limited by the available capacity implied by the train composition decision; any unboarded passengers remain waiting and are carried over to the next departure, which is captured by $n_p^{\text{after}}(k_p)$. For interchange platforms, the number of transfer passengers is explicitly represented by $n_p^{\text{trans}}(k_p)$, and their impact is captured by the same update and reflected in the passenger-oriented objective. This cycle-based update is used in the time-driven control implementation in the next section.

4. Model predictive control for real-time train rescheduling

4.1. Problem formulation

Based on the developed model, we can formulate the problem to minimize passenger delays and operational costs. The passenger delays corresponding to train service k_p at platform p can be formulated as

$$J_p^{\text{pass}}(k_p) = n_p(k_p)\big(d_p(k_p) - d_p^{\text{pre}}(k_p)\big) + n_p^{\text{after}}(k_p)\big(d_p^{\text{pre}}(k_p + 1) - d_p(k_p)\big). \tag{24}$$

As shown in (16), $n_p(k_p)$ is a dynamic queue state, representing passengers waiting immediately before $d_p^{\text{pre}}(k_p)$. It includes passengers left behind by previous services and passengers arriving during the previous cycle. After service k_p, unboarded passengers are carried over through $n_p^{\text{after}}(k_p)$, and new arrivals are added to the queue of the next service. Therefore, Eq. (24) does not represent the full continuous-time integral of waiting time. It measures the additional waiting time relative to predetermined departure events, while passenger dynamics are captured through a cycle-based manner. Specifically, $n_p(k_p)\big(d_p(k_p) - d_p^{\text{pre}}(k_p)\big)$ measures the additional delay of passengers waiting at platform p immediately before $d_p^{\text{pre}}(k_p)$ due to the rescheduled departure time $d_p(k_p)$. In addition, $n_p^{\text{after}}(k_p)\big(d_p^{\text{pre}}(k_p{+}1) - d_p(k_p)\big)$ measures the additional waiting time of passengers who intend to board service k_p but cannot due to limited capacity and thus have to wait for the next predetermined departure $d_p^{\text{pre}}(k_p{+}1)$. Passengers arriving after $d_p^{\text{pre}}(k_p)$ are accounted for through the queue update at subsequent predetermined departures.

Assigning more train units to a train service can increase the capacity for transporting passengers while leading to higher energy consumption. Furthermore, changing the train composition may require additional workload from operators and thus lead to additional costs. The operational costs of train service k_p running from platform p to its successor platform can be expressed as

$$J_p^{\text{cost}}(k_p) = \ell_p(k_p) E_p^{\text{energy}} + \eta_{k_p,p} E_p^{\text{add}}, \tag{25}$$

where E_p^{energy} represents the average energy consumption for a train unit running from platform p to its successor platform, $\ell_p(k_p) E_p^{\text{energy}}$ denotes the approximate energy consumption for $\ell_p(k_p)$ train units running from platform p to its successor platform, and E_p^{add} denotes the additional cost for changing the train composition of a train service at platform p.

In urban rail transit systems, a train service typically departs from a depot, visiting each platform along a line before returning to the depot. Based on the predetermined departure times at each platform, we adopt a cycle-based MPC scheme and update the controller across cycles. The length of the control step, denoted as T_{ctrl}, is defined as the time interval of a cycle, and κ denotes the cycle counter[2]

Therefore, the optimization problem for train rescheduling over the control step κ_0 is

$$\begin{aligned} &\min_{\boldsymbol{g}(\kappa_0)} \ J(\kappa_0) := \sum_{p\in\mathcal{P}} \sum_{k_p\in\mathcal{N}_p(\kappa_0)} \Big(w_1 J_p^{\text{pass}}(k_p) + w_2 J_p^{\text{cost}}(k_p) \Big), \\ &\text{s.t.} \quad (1)-(25), \end{aligned} \tag{26}$$

where $\boldsymbol{g}(\kappa_0)$ represents a vector collecting all the variables of problem (26), $\mathcal{P}$ is the set collecting all the platforms of the line, $\mathcal{N}_p(\kappa_0)$ is the set indices of trains departing from platform p within the prediction time window starting at step κ_0, and w_1 and w_2 are weights balancing two objectives.

4.2. MINLP-based MPC for real-time train rescheduling

Problem (26) contains piecewise constant ("if-then") constraints in (11) and (14). We apply the following transformation properties to convert (11) into a mixed logical dynamical (MLD) system (Williams, 2013).

Transformation property 4.1. *If we introduce an auxiliary continuous variable $o_{k,p}$ and an auxiliary binary variable $\gamma_{k_p,p}$ with $\gamma_{k_p,p} = 1 \Leftrightarrow o_{k_p,p} = y_p(k_p)$ and $\gamma_{k_p,p} = 0 \Leftrightarrow o_{k_p,p} = -y_p(k_p)$, and then $o_{k_p,p} = |y_p(k_p)|$ is equivalent to*

$$\begin{cases} o_{k_p,p} - y_p(k_p) \geq 0, \\ o_{k_p,p} - y_p(k_p) \leq 2Y_{\max}(1-\gamma_{k_p,p}), \\ o_{k_p,p} + y_p(k_p) \geq 0, \\ o_{k_p,p} + y_p(k_p) \leq 2Y_{\max}\gamma_{k_p,p}, \end{cases} \tag{27}$$

where $Y_{\max}$ denotes the maximum value of $y_p(k_p)$.

Transformation property 4.2. *Based on Transformation property 4.1, (11) is equivalent to $\eta_{k_p,p} = \begin{cases} 1, & \text{if} \quad o_{k_p,p} > 0; \\ 0, & \text{otherwise}, \end{cases}$, which can be converted to*

$$\begin{cases} o_{k_p,p} \leq \eta_{k_p,p} O_{\max}, \\ o_{k_p,p} \geq \epsilon + (1-\eta_{k_p,p})(O_{\min} - \epsilon). \end{cases} \tag{28}$$

where $O_{\max}$ and $O_{\min}$ are the minimum and maximum values of $o_{k_p,p}$, respectively, and ϵ is a sufficiently small number, typically representing machine precision.

Transformation property 4.3. *If we define m_{a} and M_{a} as the minimum and maximum values of $a_{p'}(k_{p'})$, respectively, then following the transformation property in Bemporad and Morari (1999), (14) is equivalent to the following inequalities*

$$\begin{cases} a_{p'}(k_{p'}) - a_p(k_p) \leq \Big(1 - \xi_{k_{p'},k_p,p',p}\Big)\big(M_{\text{a}} - a_p(k_p)\big), \\ a_{p'}(k_{p'}) - a_p(k_p) \geq \epsilon + \xi_{k_p,k_{p'},p,p'}\big(m_{\text{a}} - a_p(k_p) - \epsilon\big). \end{cases} \tag{29}$$

Based on the transformations described above, we can convert problem (26) into an MINLP problem. The nonlinearity in the MINLP arises from the nonlinear objective function. The integer variables in this problem encompass the train composition variables ($y_p(k_p)$ and $\ell_p(k_p)$), the train ordering variable ($\xi_{k_p,k_{p'},p,p'}$), and auxiliary binary variables ($\gamma_{k_p,p}$ and $\eta_{k_p,p}$).

For compactness, we rewrite the resulting MINLP problem in the following form:

$$\min_{\boldsymbol{x}(\kappa_0),\boldsymbol{u}(\kappa_0),\boldsymbol{\delta}(\kappa_0)} J(\kappa_0) := \sum_{\kappa=\kappa_0}^{\kappa_0+N-1} L(x(\kappa), u(\kappa), \delta(\kappa)) \tag{30a}$$

$$\text{s.t.} \quad x(\kappa+1) = A_\kappa x(\kappa) + B_{1,\kappa} u(\kappa) + B_{2,\kappa}\delta(\kappa), \tag{30b}$$

$$D_{3,\kappa} x(\kappa) + D_{1,\kappa} u(\kappa) + D_{2,\kappa}\delta(\kappa) \leq D_{4,\kappa}, \tag{30c}$$

$$\kappa = \kappa_0, \dots, \kappa_0 + N - 1,$$

where N is the total number of control steps, $x(\kappa)$, $u(\kappa)$, and $\delta(\kappa)$ collect all the independent variables, continuous decision variables, and discrete decision variables for control step κ, respectively, and $\boldsymbol{x}(\kappa_0) = [x^{\intercal}(\kappa_0), x^{\intercal}(\kappa_0+1), \dots, x^{\intercal}(\kappa_0+N-1)]^{\intercal}$, $\boldsymbol{u}(\kappa_0) = [u^{\intercal}(\kappa_0), u^{\intercal}(\kappa_0+1), \dots, u^{\intercal}(\kappa_0+N-1)]^{\intercal}$, and $\boldsymbol{\delta}(\kappa_0) = [\delta^{\intercal}(\kappa_0), \delta^{\intercal}(\kappa_0+1), \dots, \delta^{\intercal}(\kappa_0+N-1)]^{\intercal}$. In (30), $L(x(\kappa), u(\kappa), \delta(\kappa))$ represents the nonlinear objective function for control step κ, (30b) collects all equality constraints, and (30c) collects all inequality constraints.

[2] In this paper, the planned departure intervals are identical across lines.

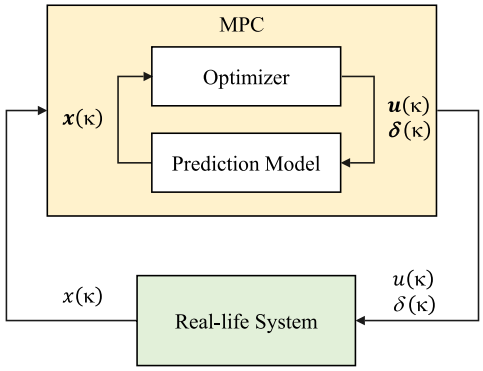

Fig. 4. Model predictive control for real-time train rescheduling.

Table 5
Numbers of variables and constraints in problem (30).

Variables or constraints	Maximum possible total number
Continuous variables	$(12 \cdot \lvert\mathcal{P}\rvert + 4 \cdot \lvert\mathcal{Z}\rvert) \cdot N$
Binary variables	$(6 + N) \cdot \lvert\mathcal{Z}\rvert \cdot N$
Integer variables (excluding binaries)	$2 \cdot N \cdot \lvert\mathcal{Z}\rvert$
Constraints	$(33 + N) \cdot N \cdot \lvert\mathcal{Z}\rvert$

The MPC update index κ is cycle-based. At step κ, the state contains the depot inventory and the availability of train units, and circulation/depot constraints are enforced over the prediction horizon. Thus, the impact of a decision is propagated to later services via train units availability. As each cycle includes only one departure at each platform, the transition from κ to κ+1 depends only on the current state and current decisions at control step κ. Solving (30) leads to a series of continuous decision variables and discrete decision variables, and only the decision variables at control step κ_0 are applied. At the next step, the prediction time window is shifted for one step, and a new optimization is formulated. The framework is depicted in Fig. 4.

Lemma 4.1. *(*Recursive Feasibility*): If problem (30) is feasible at control step κ with initial state $x(\kappa)$, then problem (30) is also feasible at control step $\kappa + 1$.*

Proof. The proof relies on finding a feasible solution for control step $\kappa + 1$. Recall that the planning time window at each platform is divided into several intervals of equal length, with a train service departing from the platform at each interval according to (1). In general, there are two types of depots: (i) depots connected to the terminal platform (e.g., Platform 1 in Fig. 2), and (ii) depots connected to intermediate platforms (e.g., Platform 2 in Fig. 2).

i) For a depot connected to the terminal platform: if the problem (30) is feasible at control step κ, then at each control step, a train service returns to the depot from the opposite direction of the line. In this context, the new train service departing from the terminal platform at control step $\kappa + 1$ can directly utilize the train units by performing a turnaround from the opposite direction of the line following (7). Moreover, utilizing one train unit when performing the turnaround action is always feasible.
ii) For any depot connected to an intermediate platform: a feasible solution is obtained by maintaining the composition of each train service the same as it was at control step κ.

□

Remark 4.1. (Complexity Analysis). There are three categories of variables in (30), i.e., continuous variables, binary variables, and integer variables (excluding binaries). The total number of variables and constraints is listed in Table 5, where $\mathcal{P}$ and $\mathcal{Z}$ are the set of platforms and depots, respectively, and $|\cdot|$ denotes the cardinality of a set.

4.3. MILP-based MPC for real-time train rescheduling

In Section 4.1, a nonlinear objective function has been defined in (24) to calculate the passenger delays. This nonlinear objective function results in the MINLP-based MPC approach formulated in Section 4.2. In general, the nonlinear term significantly increases the computational burden. In what follows, we simplify the nonlinear objective function to reduce the computational burden.

Since we divide the planning time window as indicated in Fig. 3 and the actual departure time $d_p(k_p)$ is constrained by $d_p^{\mathrm{pre}}(k_p) \le d_p(k_p) < d_p^{\mathrm{pre}}(k_p + 1)$, by using the upper bound and lower bound of $d_p(k_p)$ we approximate the nonlinear objective function (24) by

$$J_p^{\mathrm{pass}}(k_p) \approx w_3 n_p(k_p)\big(d_p^{\mathrm{pre}}(k_p + 1) - d_p^{\mathrm{pre}}(k_p)\big) + n_p^{\mathrm{after}}(k_p)\big(d_p^{\mathrm{pre}}(k_p + 1) - d_p^{\mathrm{pre}}(k_p)\big), \tag{31}$$

where w_3 is a weight used to balance the approximated errors. In particular, w_3 can be defined as

$$w_3 = \frac{1}{\sum_{p\in\mathcal{P}} |I_p|} \sum_{p\in\mathcal{P}} \sum_{k_p\in I_p} \frac{\bar{d}_p(k_p) - d_p^{\mathrm{pre}}(k_p)}{d_p^{\mathrm{pre}}(k_p+1) - \bar{d}_p(k_p)}, \tag{32}$$

where $|I_p|$ represents the cardinality of I_p, and $\bar{d}_p(k_p)$ represents the average value of $d_p(k_p)$ from historical data.

Other settings are identical to those of Section 4.2. With this approximation, a linear objective function (31) is obtained, and due to the objective function and constraints being all linear, we obtain an MILP-based MPC approach for real-time train rescheduling. Since Section 4.3 modifies only the objective function and keeps the constraint set identical to that of Section 4.2, Lemma 4.1 *(recursive feasibility)* applies to MILP-based MPC as well.

5. Learning-based MPC for real-time train rescheduling

For the MINLP-based MPC and MILP-based MPC approaches introduced in Section 4, a mixed-integer optimization problem must be solved at each control step. The computational complexity, mainly due to the integer variables, can make these approaches unsuitable for real-time implementation. To address this issue but in a different context, Cauligi et al. (2022) proposed PRISM, in which presolve techniques and an LSTM network are used to generate integer decisions, followed by solving a convex optimization problem to compute the continuous variables. Motivated by PRISM, we develop a learning-based MPC approach with presolve techniques for train rescheduling. In the proposed method, presolve techniques and a neural network ensemble with feedforward layers and an LSTM network are used to assign the discrete (integer) variables of the MPC problem. As a result, only a continuous nonlinear or linear optimization problem needs to be solved online to compute the continuous variables and to update the timetable. The main differences with PRISM lie in the design of the presolve techniques, the use of a neural network ensemble to improve feasibility, and the offline training procedure tailored to the train rescheduling problem.

5.1. Presolve techniques

Presolve techniques streamline optimization processes by pruning a subset of decision variables with values predetermined by other coupled variables and constraints, setting them to predefined values. In this paper, we develop the following presolve techniques.

Presolve technique 5.1: As we adjust the departure time of train service k_p at platform p according to $d_p^{\mathrm{pre}}(k_p) \le d_p(k_p) < d_p^{\mathrm{pre}}(k_p+1)$, the departure order between some trains has already been determined: If $d_p^{\mathrm{pre}}(k_p) \ge d_{p'}^{\mathrm{pre}}(k_{p'}+1) + t_{p'}^{\mathrm{roll}}$, then $\xi_{k_p,k_{p'},p,p'} = 1$. If $d_p^{\mathrm{pre}}(k_p+1) \le d_{p'}^{\mathrm{pre}}(k_{p'}) + t_{p'}^{\mathrm{roll}}$, then $\xi_{k_p,k_{p'},p,p'} = 0$.

Presolve technique 5.2: According to the definition of $\xi_{k_p,k_{p'},p,p'}$ in (14), the order of trains at the same platform should be kept consistent, i.e., $\xi_{k_p+1,k_{p'},p,p'} \ge \xi_{k_p,k_{p'},p,p'}$.

Presolve technique 5.3: The train composition cannot be changed at a station that is not linked with a depot: If $\sigma_p = 0$, then $y_p(k_p) = 0$.

Presolve technique 5.4: Let $t_0 = \kappa_0 T$ represent the current time. If train service k_p has already departed from station p at time t_0, the composition cannot be changed at that station: If $d_p(k_p) \le t_0$, $y_p(k_p) = y_p^*(k_p)$, $\forall p \in \mathcal{P}$, where $y_p^*(k_p)$ is the value of $y_p(k_p)$ obtained before train k departs from platform p and $\mathcal{P}$ denotes the set of all platforms of the line.

5.2. Environment setting

The environment of the learning-based MPC algorithm includes the system and an MPC optimization problem. The state and variables that interact with the environment are defined as follows:

State $\boldsymbol{s}(\kappa) \in \mathcal{S}$: The system state is defined to encompass all necessary information required to characterize the system at control step κ and to serve as input to the neural network. Hence, the state at the control step κ is defined as

$$\boldsymbol{s}(\kappa) = [\boldsymbol{n}^\intercal(\kappa), \boldsymbol{\rho}^\intercal(\kappa), \boldsymbol{N}^\intercal(\kappa)]^\intercal, \tag{33}$$

where $\boldsymbol{n}(\kappa)$ includes the variables $n_p(k_p)$ for each train service k_p at its corresponding platform p with $\kappa T \in [d_p^{\mathrm{pre}}(k_p-1), d_p^{\mathrm{pre}}(k_p))$, $\boldsymbol{\rho}(\kappa)$ collects the passenger arrival rates $\rho_p(k_p)$ for all train services k_p departing from all platforms p from time $t = \kappa T$ to the end of the prediction time window, and $\boldsymbol{N}(\kappa)$ collects the number of available trains for all depots at time $t = \kappa T$.

Discrete variables: The variable $\boldsymbol{\delta}(\kappa) \in A$ represents the discrete variables of the MPC optimization problem (30) or (31) at control step κ.

Continuous variables: The variable $\boldsymbol{u}(\kappa) \in U$ denotes the continuous variables of the MPC optimization problem (30) or (31) at control step κ.

5.3. Offline training

We formulate the learning problem as a multi-class classification task. Each feasible combination of discrete control actions encountered during data acquisition is mapped to a unique class label. In the forward pass, one class label is predicted for each time step of the prediction horizon from a label set. This formulation enables the a priori exclusion of infeasible, suboptimal, or internally inconsistent discrete action combinations, as only labels corresponding to admissible MPC solutions are included in the label set.

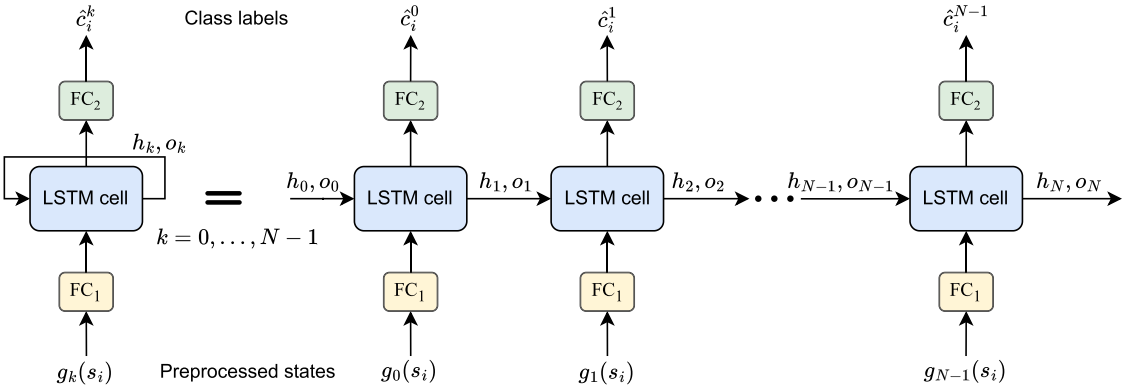

Fig. 5. Unrolled LSTM architecture with feedforward input layer FC_1 and output layer FC_2 over the prediction horizon.

The goal is to learn a parametric mapping $\phi_\theta : \mathcal{S} \to \mathcal{L}^N$ from the state set $\mathcal{S}$ to the label set $\mathcal{L}^N$, with $\theta \in \mathbb{R}^{n_\theta}$ being the parameter vector. At control step κ, the input is the system state $\boldsymbol{s}(\kappa)$, and the output is a label sequence of length N, with one label for each prediction step, and each label is selected from the label set $\mathcal{L}$. The label sequence is then decoded into the corresponding discrete decision variables. In this way, ϕ_θ approximates only the discrete-decision component of the MPC solution, including train composition and ordering decisions.

We use supervised learning to train $\phi_\theta(\cdot)$ by adjusting the parameter vector θ. The training data consist of labeled states and the corresponding discrete decisions obtained from simulations of the MPC policy, where (30) or (31) is solved only offline to generate the training labels. In this way, the combinatorial complexity associated with the discrete variables is shifted from online evaluation to offline training. During online evaluation, the trained model $\phi_\theta(\cdot)$ assigns the discrete variables, and the MPC optimization problem in (30) or (31) reduces to an NLP or an LP by fixing the discrete variables.

For various realizations of the initial system state $\boldsymbol{s}(\cdot)$, we generate the training dataset by solving the MINLP in (30) or the MILP in (31). Let

$$\mathcal{D} = \{(\boldsymbol{s}_i,\ C_i)\}_{i=1}^{N_{\text{data}}}, \tag{34}$$

denote the resulting dataset with size N_{data}, where $\boldsymbol{s}_i \in \mathcal{S}$ is a sampled realization of $\boldsymbol{s}(\kappa_i)$ from the training set, and $C_i := \{c_i^k\}_{k=0}^{N-1} \in \mathcal{L}^N$ is the corresponding label sequence associated with the discrete variables in the solution of the MPC problem. To improve robustness under unobserved operating conditions, we augment the dataset with noise to reflect variability in passenger demands and delay profiles. In this learning framework, the MINLP (or MILP) formulation of MPC serves as an offline label generator providing the discrete decisions for each state. Before training, the dataset is split into three disjoint subsets, i.e., $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{val}}$, $\mathcal{D}_{\text{test}}$ for training, validation, and testing, respectively, with $\mathcal{D} = \mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{val}} \cup \mathcal{D}_{\text{test}}$.

In practice, the train schedules across consecutive time intervals are interdependent due to the headway relation between trains and the physical connections between stations. The neural network should be able to capture and retain essential information over sequences of time intervals. Therefore, a long short-term memory (LSTM) network (Hochreiter and Schmidhuber, 1997; Cauligi et al., 2022) is selected as the parametric model $\phi_\theta(\cdot)$. As a deep recurrent neural network (RNN), the LSTM architecture enables the network to remember the dynamic interdependencies within train schedules, ensuring effective adaptation and learning in response to evolving temporal dynamics.

The architecture of the neural network for the parametric model $\phi_\theta(\cdot)$ is shown in Fig. 5, where each block represents the identical LSTM cell with shared parameters across time steps. The input at step k is the pre-processed state vector $g_k(\boldsymbol{s}_i)$, where $k = 0, \ldots, N-1$ indexes the prediction horizon. Each input is first mapped into a latent representation, i.e., a compressed representation of input data, by the feedforward layer FC_1. The latent features are then processed by an LSTM that maintains a hidden state $h_k \in \mathbb{R}^{n_h}$ and a cell state $o_k \in \mathbb{R}^{n_h}$, both initialized to zero at the beginning of the forward pass. In this paper, we use zero-padding, with $g_0(\boldsymbol{s}_i) = \boldsymbol{s}_i$ and $g_k(\boldsymbol{s}_i) = \boldsymbol{0}$ for $k = 1, \ldots, N-1$, and the rationale is to place all relevant information on the system state and forecasted passenger demands at the initial step, so that the LSTM can propagate it across the prediction horizon via its internal states. Finally, the output of the LSTM network is passed through the feedforward layer FC_2, whose output neurons correspond to the possible combinations of discrete decision variables, yielding a categorical probability distribution over the label set at each time step through a softmax layer.

Each target label $c_i^k \in \{0,1\}^{|\mathcal{L}|}$ corresponds to one discrete sequence and is represented by one-hot encoding, where $|\mathcal{L}|$ is the cardinality of the label set $\mathcal{L}$. The output at prediction step k is $\hat{c}_i^k = [\phi_\theta(\boldsymbol{s}_i)]_k \in \mathbb{R}^{|\mathcal{L}|}$ denoting a categorical distribution over $\mathcal{L}$ for state $\boldsymbol{s}_i$ at step k, with $[\cdot]_k$ extracting the kth element. We train the network by stochastic gradient descent Amari (1993) and minimize the mean squared error (MSE) between predicted and target labels over the prediction horizon. Let $\mathcal{J} \subset \{1, \ldots, N_{\text{data}}\}$ denote a mini-batch with $|\mathcal{J}| = N_{\text{batch}}$, sampled randomly at each training iteration. Mini-batches reduce the per-iteration cost and yields an unbiased stochastic estimate of the full-dataset loss. The loss function is defined as

$$L(\theta) = \sum_{j \in \mathcal{J}} \sum_{k=0}^{N-1} \|c_j^k - \hat{c}_j^k\|_2^2. \tag{35}$$

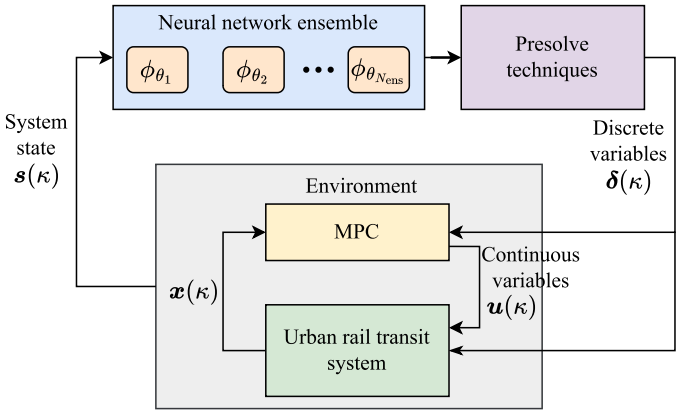

Fig. 6. Closed-loop implementation of the proposed learning-based MPC framework.

As a common limitation of learning-based methods, including Cauligi et al. (2022), the parametric model $\phi_\theta(\cdot)$ cannot guarantee the feasibility of the resulting continuous optimization problem. To mitigate infeasibility, we use a neural network ensemble defined as $\{\phi_\theta^i\}_{i=1}^{N_{\text{ens}}}$ and evaluate the networks sequentially until a feasible discrete assignment is found, where N_{ens} is the ensemble size. If the ensemble fails to produce feasible discrete variables, we apply the heuristic strategy of Lemma 4.1 to guarantee feasibility during online operation.

In the learning context, classification accuracy and performance are correlated. However, the aim of this paper is control performance, namely minimizing passenger delays and operational costs. Therefore, instead of using the validation loss, we evaluate each trained model on the validation set D_{val} by the average open-loop objective value obtained by the corresponding MPC problem. Specifically, for each state $\boldsymbol{s}_i \in D_{\text{val}}$, we compute $\phi_\theta(\boldsymbol{s}_i)$ and use the predicted outputs to assign the discrete variables in the MINLP or MILP formulation. With the discrete variables fixed, we solve the resulting continuous MPC optimization problem and obtain the objective value $J(\kappa_i; \boldsymbol{s}_i, \phi_\theta(\boldsymbol{s}_i))$, and κ_i denotes the control step associated with $\boldsymbol{s}_i$. The average open-loop objective value over all $\boldsymbol{s}_i \in D_{\text{val}}$ is denoted by $\bar{J}_{\phi_\theta}$.

This performance metric is used in the hyperparameter optimization stage. We train multiple neural networks with different weight initializations and hyperparameter settings. Let $\Phi = \{\phi_{\theta_1}, \dots, \phi_{\theta_{N_{\text{train}}}}\}$ denote the set of trained networks obtained from the hyperparameter optimization procedure. For each $\phi_{\theta_i} \in \Phi$, we compute its average open-loop objective value $\bar{J}_{\phi_{\theta_i}}$ on D_{val} and rank the models accordingly. We then select the N_{ens} models with the lowest $\bar{J}_{\phi_{\theta_i}}$ to form the ensemble, denoted by $\Phi^{\text{ens}} \subseteq \Phi$. In this manner, the ensemble is obtained as a byproduct of the hyperparameter optimization process. Finally, the states in D_{test} are used to evaluate both open-loop and closed-loop performance of the learning-based framework.

5.4. Online evaluation

The outcome of the training process is a neural network ensemble $\Phi_{\text{ens}} = \{\phi_{\theta_1}, \dots, \phi_{\theta_{N_{\text{ens}}}}\}$. During online operation, the measured system state is provided to the ensemble as an input. The networks are evaluated sequentially, and the $(i{+}1)$th network is evaluated only if the ith network fails to produce a feasible assignment of the discrete variables. The ensemble size N_{ens} is fixed and relatively small, and the computational cost of each forward pass is negligible compared to that of solving the continuous MPC problem. As a result, even in the worst-case scenario where all neural networks are evaluated, the additional computational overhead remains compatible with real-time operation.

Before applying the discrete actions suggested by the ensemble, the presolve techniques in Section 5.1 are used to prune infeasible discrete variables by fixing them to predetermined values. The resulting discrete assignments are then fixed in the MPC formulation, and the corresponding continuous optimization problem is solved to obtain the continuous decision variables for updating the timetable. Although feasibility can be improved by a neural network ensemble, it cannot be guaranteed by design. If no feasible assignment is obtained, we apply the heuristic strategy in Lemma 4.1 to ensure recursive feasibility. The closed-loop implementation is illustrated in Fig. 6.

6. Case study

6.1. Simulation setup

In this section, we illustrate the proposed approaches based on real-life data from a network of three lines in the Beijing urban rail transit network. As shown in Fig. 7, the network includes 3 bi-directional lines with 45 stations. There are 3 transfer stations, i.e., Station ZXZ, Station XEQ, and Station HY, where passengers can transfer from one line to another. For each line, there is a depot connected with the starting station of the line, i.e., Station CPX for Changping Line, Station XZM for Line 13, and Station ZXZ for Line

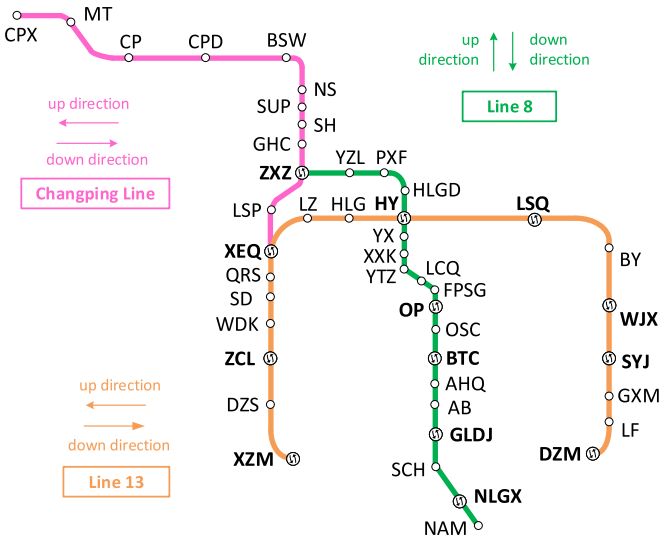

Fig. 7. The layout of the considered urban rail transit network (with 3 lines).

8. The values of parameters for the case study are given in Table 6. The original timetable is generated based on the regular headway, regular dwell time, and average running times in Table 6. According to the definition, the length of a control step is the time interval between two consecutive predetermined train departures at a platform, i.e., the sum of the regular headway and the regular dwell time. The total number of train units N_z^{train} in the depot for each line has been selected as a random integer number with the value varying within the range given in Table 6. The constraint (15) becomes binding when the number of train units assigned from depot z, after considering train units that can be reused through rolling-stock circulation, reaches N_z^{train}. In this situation, no additional train unit can be dispatched from the depot, even if a larger train composition could reduce passenger waiting time. Therefore, the optimizer has to keep the current composition, reuse train units released by previous services, or allow more passengers to remain for the next cycle. Thus, (15) prevents infeasible overuse of train units and couples current composition decisions with future rolling-stock availability. In practice, time required for changing the train composition t_p^{comp} may vary across platforms. In this case study, we set $t_p^{\text{comp}} = 60$ s for all platforms, which does not affect the comparison since all methods use the same parameter setting.

The length between every two consecutive stations is openly accessible on the website of Beijing Subway.[3] In the case study, the average running time and the average energy consumption of a train between every two consecutive stations are calculated using the method in Wang et al. (2015) with the maximum acceleration of 0.75 m/s^2, the maximum deceleration of 0.7 m/s^2, and the cruising speed of 70 km/h, respectively. The sectional passenger demands are obtained based on real-life passenger flow data from the Beijing urban rail transit network, collected in January 2020. We have selected data from 6:00 AM to 10:00 PM for simulation, so the data contains both peak hours and off-peak hours. At interchange stations, the passengers include both passengers entering from outside the system and transfer passengers arriving from other lines. Hence, transfer impacts are reflected in the passenger-oriented results through the rescheduled departure times and the available capacity implied by train composition decisions. For training agents and simulation, we have generated passenger demands based on a Poisson distribution by using real-life passenger flow data as the expected value.

The simulations have been conducted using Python as a programming language, PyTorch as the machine learning library, and `gurobi` to solve optimization problems. Adam (Kingma and Ba, 2014) is applied in the offline training process to minimize MSE, and dropout (Srivastava et al., 2014) is used to handle the overfitting issue. Moreover, the experiments were conducted on a computing cluster with Intel XEON E5-6248R CPUs. The dataset consists of 96000 states and the corresponding reference solutions, and it has been built using 60 CPU cores and 240GB of RAM in 24 h. The training and hyperparameter tuning processes have been conducted using 144 CPU cores, 864GB of RAM, with more than 200,000 iterations for 24 h. The code used to generate the results in this paper is available online.[4]

To reduce the solution time of the MINLP solver without significantly compromising optimality, a simple early termination criterion was employed: if the reference objective gap does not decrease by 0.5% within 10 s, the solution process terminates, and the solver outputs the best solution found. Moreover, as the MILP-based approach typically has a significantly shorter solution time than the MINLP-based approach, the integer variables generated by the MILP-based approach are used as a warm-start rule for the MINLP-based approach.

[3] https://www.bjsubway.com/station/zjgls/
[4] https://github.com/fabcaio/railwaynet_learning/

Table 6
Main parameters for the case study.

Parameter	Symbol	Value
Regular headway	h_p^{regular}	180 s
Minimum headway	$h_p^{\min}$	120 s
Regular dwell time	τ_p^{regular}	60 s
Minimum dwell time	$\tau_p^{\min}$	30 s
Average turnaround time	r_p^{turn}	52.9 s
Minimum running time	$r_p^{\min}$	$0.8 \cdot r_p^{\text{avrg}}$
Maximum running time	$r_p^{\max}$	$1.2 \cdot r_p^{\text{avrg}}$
Time for changing train composition	t_p^{comp}	60 s
Time for rolling stock circulation	$t_{p,p'}^{\text{roll}}$	240 s
Transfer rate at a transfer station	$\beta_{q,p}$	10%
Capacity of a train unit	$C_{\max}$	400 persons
Regular train composition of a train service	ℓ_p^{regular}	2 train units
Minimum number of train units included in a train service	$\ell_p^{\min}$	1 train unit
Maximum number of train units included in a train service	$\ell_p^{\max}$	4 train units
Weighted term	w_1	10^{-4}
Weighted term	w_2	10^{-1}
Weighted term	w_3	10^{-1}
Number of train units for Changping Line	N_z^{train}	[55, 75]
Number of train units for Line 13	N_z^{train}	[70, 90]
Number of train units for Line 8	N_z^{train}	[60, 80]
Prediction horizon of MPC	N	40

Table 7
Comparison of the open-loop performance of different approaches for real-time train rescheduling.

Approach	Warm-start	Reference objective gap (%)				CPU time (s)			Feasibility rate
		mean	std	min	max	mean	min	max	
Benchmark	yes	–	–	–	–	222.18	3.67	600.10	100%
MINLP	no	7.22	14.14	0	100.81	239.84	52.47	240.14	100%
Warm-start MINLP	yes	0.04	3.14	–0.24	12.96	96.71	3.58	240.10	100%
MILP	no	0.47	1.11	–33.73	1.54	8.77	0.37	240.01	100%
Learning + NLP	no	–0.11	1.18	–34.28	3.48	6.89	1.80	112.22	98.94%
Learning + LP	no	0.22	1.10	–33.42	1.73	0.13	0.11	0.25	98.55%

In this section, the developed train rescheduling approaches, i.e., MINLP-based MPC, warm-start-MINLP-based MPC, MILP-based MPC, learning-NLP-based MPC, and learning-LP-based MPC, are evaluated. As defined in Section 3, the length of a control step is 240 s. Hence, to ensure that a solution can be obtained for each step, we set the maximum solution time for each approach as 240 s. In addition, as a longer solution time typically yields a better objective function value, we use the MINLP approach with a warm-start and a longer maximum solution time, i.e., 600 s, as a benchmark to evaluate the performance of the developed approaches.

To improve feasibility, we use an ensemble of 15 neural networks trained with different weight initializations and hyperparameter settings. During online operation, the networks are evaluated sequentially, as described in Section 5, until a feasible assignment of the discrete variables is obtained. If the ensemble fails to provide feasible discrete variables, we apply the strategy of Lemma 4.1 to ensure feasibility.

We perform hyperparameter optimization by grid search over a predefined set of candidate values for size of the hidden layer, dropout rate, learning rate scheduling, and output masking. Although alternative methods, such as Bayesian optimization, can be more efficient for large search spaces, grid search is effective and reliable for the low-dimensional search space considered in this work. The neural network ensemble is obtained as a byproduct of the hyperparameter optimization procedure. The resulting elements of the ensemble are trained under the following hyperparameter configurations: size of the hidden layer $\in \{512, 1024\}$, dropout rate $\in \{0, 0.5\}$, learning-rate scheduling $\in \{\text{ON}, \text{OFF}\}$, and output masking $\in \{\text{ON}, \text{OFF}\}$. The input dimension of FC_1 equals the state dimension, and its output dimension equals the LSTM size of the hidden layer. The output dimension of FC_2 equals the cardinality of the shared label set $|\mathcal{L}|$, and its input dimension equals the LSTM size of the hidden layer.

Dropout is applied to both weight connections from FC_1 to the LSTM and from the LSTM to FC_2. When learning-rate scheduling is enabled, the learning rate is halved when a plateau in the validation loss is detected, i.e., whenever the validation loss does not improve for five consecutive epochs. When output masking is enabled, the network is prevented from assigning labels that were not observed during data acquisition for each time step.

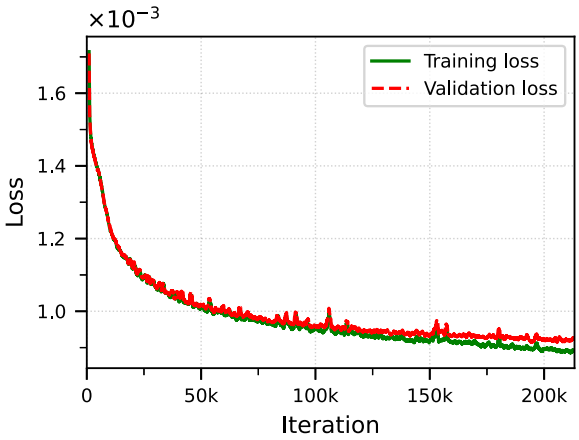

Fig. 8. Training and validation loss as a function of the mini-batch gradient descent iterations.

6.2. Simulation results

We conduct simulations to generate training data and train the learning algorithm. The training process of one neural network in the ensemble is shown in Fig. 8, where a 1000-step moving average is used to smooth the loss curves. The loss decreases rapidly during the first 10^5 iterations and then gradually converges over the remaining iterations.

We perform simulations for both open-loop optimization, where the MPC optimization problem is solved for one step, and closed-loop MPC, involving real-time MPC optimization over 30 steps. For open-loop optimization, we have performed simulations using the developed learning-based approaches on 10 separate cores for 24 h (resulting in 1054 scenarios). For comparison, simulations have also been performed with the benchmark approach, the MINLP approach, the warm-start MINLP approach, and the MILP approach, where the benchmark approach corresponds to the MINLP approach with warm-start and a maximum solution time of 600 s.

To compare the proposed learning-based approach with the MINLP- and MILP-based MPC controllers, we report three metrics: relative objective gap, CPU time, and feasibility rate. Since the benchmark is obtained by solving the warm-started MINLP with a finite time limit, it is not guaranteed to be globally optimal. Therefore, we use the term reference objective gap, which is defined as the percentage difference between the average open-loop objective value of an approach and that of a benchmark. The benchmark is the MINLP-based MPC problem solved with a time limit of 10 min, which exceeds the control step duration of 4 min. Therefore, the benchmark is used as a reference rather than an online strategy, since the longer time limit generally allows the solver to find a better local solution. In contrast, all other methods are implemented online under the 4-min control step. The feasibility rate is defined as the percentage of instances in which the neural network ensemble produces discrete actions that lead to a feasible continuous MPC optimization problem.

The reference objective gap, CPU time, and feasibility rate obtained from the simulations are given in Table 7. The MINLP approach without warm-start has the worst reference objective gap and the worst CPU times. In general, the nonlinear objective function and the presence of integer decision variables significantly influence the performance of MINLP. By applying the warm start, the reference objective gap and the solution time are reduced; however, the maximum solution time and the average solution time of the warm-start MINLP approach are still large with values of 240.10 s and 96.71 s, respectively. By approximating the nonlinear objective function, the solution time of the MILP approach is further reduced without significant performance degradation. However, the MILP also reached the maximum allowed solution time in some cases, i.e., the maximum CPU time of MILP in Table 7 still reached 240 s. The MILP controller is a strong optimization-based baseline. The MILP controller is a strong optimization-based baseline. For a fair comparison, the reference objective gaps of all methods are computed using the original nonlinear objective after the corresponding solutions are obtained. The small reference objective gap and low mean CPU time show that the MILP performs well in many cases. However, its worst-case CPU time may still approach the control-step limit in some rolling-horizon instances. Therefore, the MILP is used as a strong baseline, while the learning-based LP/NLP methods are designed to obtain faster and more stable online computation after the discrete decisions are predicted.

In the open-loop optimization, the developed learning-based NLP approach (Learning + NLP) and learning-based LP approach (Learning + LP) achieve comparable performance with an average reference objective gap of -0.11% and 0.22% among the feasible cases, while significantly reducing the solution time to an average of 6.89 s and 0.13 s, respectively. Furthermore, by approximating the nonlinear objective function, the learning-based LP approach further reduces the solution time, enabling the optimized timetable to be obtained in under 1 s. Both the learning-based NLP and LP approaches demonstrate high feasibility rates, at 98.94% and 98.55%, respectively. Negative reference objective gaps are possible when an alternative approach attains a lower objective value than the benchmark, since the benchmark is computed under a fixed solver time limit and is not guaranteed to be globally optimal. During online operation, the system state may deviate from the training distribution due to time-varying passenger demands and unforeseen disruptions, which can degrade the performance of the neural network ensemble. To improve the generalization under distribution

Table 8
Comparison of the closed-loop performance of different approaches for real-time train rescheduling.

Approach	Warm-start	Reference objective gap (%)				CPU time (s)			Feasibility rate
		mean	std	min	max	mean	min	max	
Benchmark	yes	–	–	–	–	227.18	0.05	600.12	100%
Warm-start MINLP	yes	–0.22	3.42	–60.95	13.03	51.36	0.05	240.06	100%
MILP	no	0.46	1.53	–35.61	21.18	6.83	0.06	240.05	100%
Learning + NLP	no	4.92	10.21	–11.02	93.43	6.50	1.84	91.20	100%
Learning + LP	no	5.50	11.75	–9.99	126.71	0.15	0.11	0.31	100%

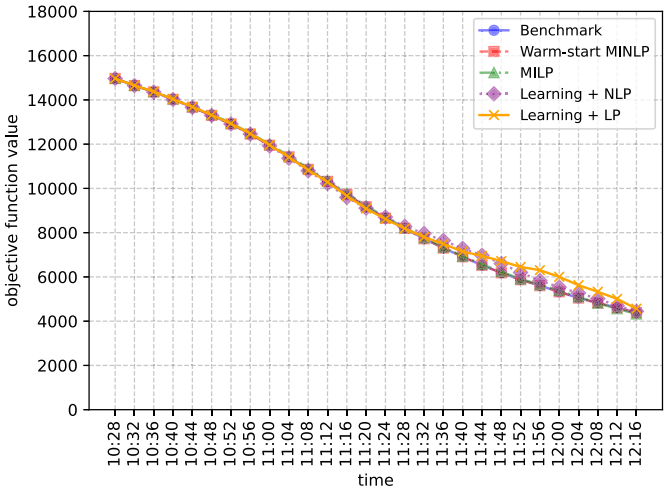

Fig. 9. Objective function value at each MPC step for one episode.

shift, we use data augmentation, dropout, and an ensemble of neural networks during training. For long-term changes in passenger behavior, the ensemble can be periodically retrained with recent data, and imitation learning can be used for fine-tuning. Under severe disruptions that differ substantially from the training conditions, the ensemble performance may still deteriorate. In such cases, the feasible suboptimal actions of Lemma 4.1 or backup controllers can be applied to ensure feasibility.

To further demonstrate the performance of the learning-based MPC approach, we conduct closed-loop control tests on 10 separate cores over 24 h, with each episode consisting of 30 control steps. If the integer variables generated by the learning agents are infeasible, the heuristic described in Lemma 4.1 is applied to obtain the train decomposition and to generate a feasible timetable. The simulation results are presented in Table 8. It can be observed that after applying the heuristic rule described in Lemma 4.1, the feasibility rate reached 100%. Compared to the warm-start MINLP and MILP approaches, the optimality of learning-based approaches is reduced in some cases because heuristic rules typically generate suboptimal solutions. In the closed-loop case, the maximum solution time for traditional approaches, i.e., warm-start MINLP and MILP, reaches 240 s, making them unsuitable for real-life applications where operators require an optimized solution as soon as possible for real-time timetable rescheduling. In contrast, the solution times for learning-based NLP and LP approaches are significantly lower, with average solution times of 6.50 s and 0.13 s, respectively. Table 8 indicates that learning-based approaches significantly enhance solution efficiency, with an acceptable optimality sacrifice of approximately 5%.

As an illustrative example, Fig. 9 presents the simulation results for one episode, showing the objective function values at each step for the benchmark, warm-start MINLP, MILP, learning-based NLP, and learning-based LP approaches. The episode begins at 10:30 and spans 30 MPC steps. Fig. 9 indicates that the developed learning-based approaches achieve comparable performance concerning objective function values. The timetables for the Changping Line, covering from Station CPX to Station XEQ, are depicted in Figs. 10–14 for the benchmark approach, the warm-start MINLP, MILP, learning-based NLP, and learning-based LP approaches, respectively, where the line width indicates the number of train units included in each train service. From Figs. 10–14, it can be observed that all approaches deploy long trains from 10:30 to 10:50 to transport passengers. After 10:50, the benchmark, warm-start MINLP, and MILP approaches continue using long trains, whereas the learning-based NLP approach initially uses short trains before switching to long trains, resulting in a slight difference in the objective function value shown in Fig. 9. In contrast, the learning-based LP approach continues using short trains until 11:40, resulting in the worst performance in Fig. 9.

In general, the rail operator expects to generate a timetable as soon as possible to ensure an efficient rolling-stock circulation that satisfies depot/turnaround constraints and limits unnecessary coupling/decoupling operations, thereby enabling effective management of real-time traffic conditions and passenger flows. The simulation results indicate that the developed learning-based MPC approach significantly reduces solution times with an acceptable trade-off in objective function performance. The developed learning-

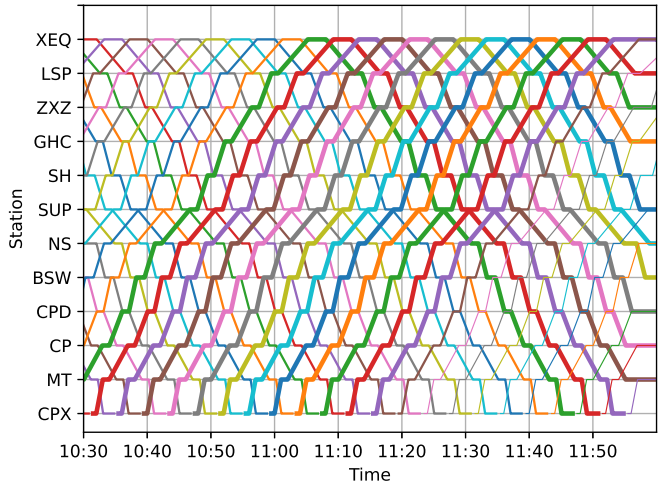

Fig. 10. Timetable for Changping Line of the benchmark approach.

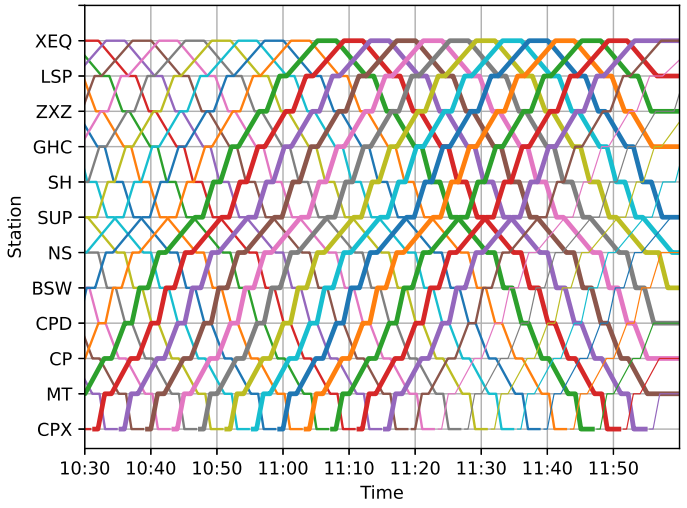

Fig. 11. Timetable for Changping Line of the warm-start MINLP approach.

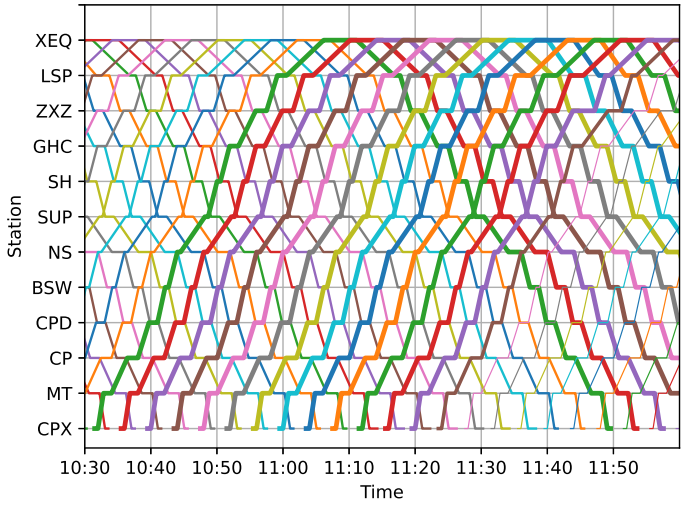

Fig. 12. Timetable for Changping Line of the MILP approach.

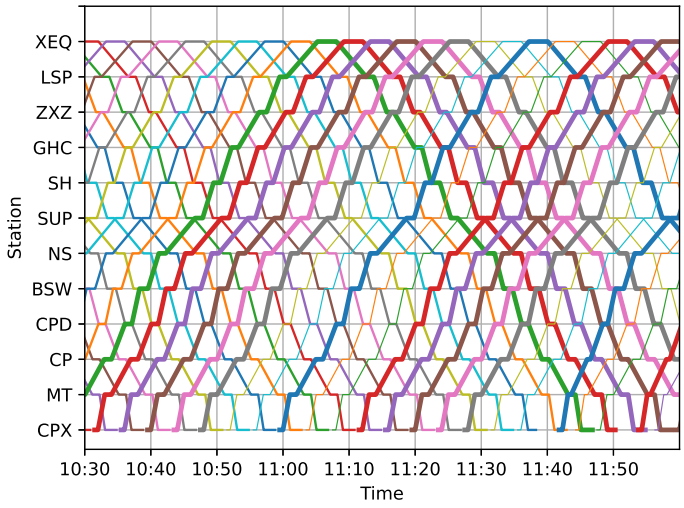

Fig. 13. Timetable for Changping Line of the learning-based NLP approach.

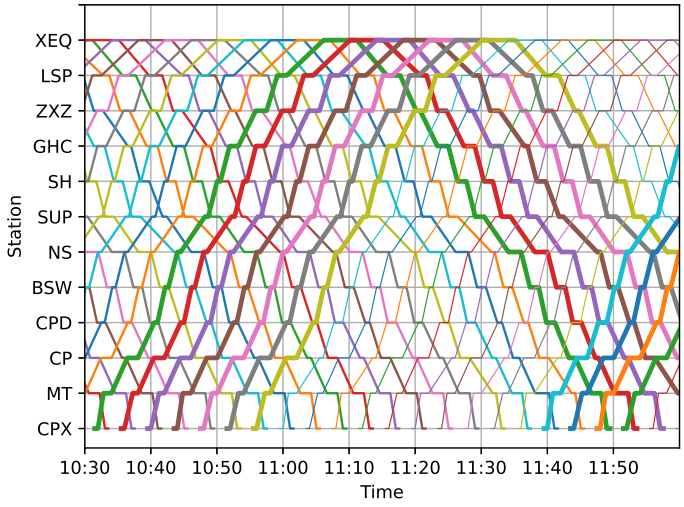

Fig. 14. Timetable for Changping Line of the learning-based LP approach.

based MPC approach can generate timetables in real time with the train composition and rolling stock circulation plan generated by learning, and the detailed departure and arrival times generated by nonlinear or linear programming. It should be noted that solving MINLP or MILP problems typically requires commercial solvers, such as `gurobi` and `cplex`, to generate solutions in a computationally efficient way. By generating integer variables with well-trained learning agents at each MPC step, the resulting continuous linear or nonlinear programming problems can be efficiently solved using various available solvers and algorithms.

As the dataset is divided into disjoint training, validation, and testing subsets, the LSTM models are trained only on the training set, and the tests are conducted on held-out samples. In this paper, the rail network topology and operational parameters are fixed. The testing samples correspond to unseen dynamic passenger flows within the same rail network, which is consistent with our goal of evaluating robustness to dynamic passenger flows. Transfer to networks with different topology or parameters is beyond the scope of this paper. The reported CPU time does not include offline data generation or LSTM training. It refers to the online computation at each MPC step, including LSTM prediction and the subsequent continuous optimization.

7. Conclusions

In this paper, we have investigated the passenger-oriented train rescheduling problem considering flexible train composition and rolling stock circulation. The passenger-oriented train rescheduling model of Liu et al. (2023a) has been extended to include train compositions and rolling stock circulation, considering time-varying passenger demands. To improve the online computational efficiency of model predictive control, we have combined the optimization-based and learning-based approaches, where the learning-based approach obtains integer variables, i.e., train compositions and train orders, by using pre-trained long short-term memory

networks; then, the detailed timetables are optimized by solving a constrained optimization problem with the fixed integer variables. We have developed several presolve techniques to prune the subset of integer decision variables. Simulation results indicate that the developed learning-based framework can achieve comparable performance compared to the exact approach with an acceptable loss of feasibility, while the solution time is significantly reduced.

In the future, we will investigate the integration of reinforcement learning (instead of deep learning) and model predictive control for real-time train rescheduling to improve the learning ability of the approach. A state reduction approach can also be applied to improve performance and reduce memory usage. Among several directions, multi-agent learning-based approaches can also be a promising research direction for handling large-scale urban rail transit networks.

CRediT authorship contribution statement

Xiaoyu Liu: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Investigation, Formal analysis, Data curation; **Caio Fabio Oliveira da Silva:** Writing – review & editing, Validation, Software, Methodology, Investigation, Formal analysis; **Azita Dabiri:** Writing – review & editing, Supervision, Investigation, Conceptualization; **Yihui Wang:** Writing – review & editing, Resources, Investigation, Data curation; **Bart De Schutter:** Writing – review & editing, Supervision, Resources, Project administration, Methodology, Funding acquisition, Conceptualization.

Acknowledgements

This research has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 101018826 - CLariNet). The work of X. Liu is also supported by the China Scholarship Council under Grant 202007090003.

Data availability

Data will be made available on request.

References

Amari, S.-i., 1993. Backpropagation and stochastic gradient descent method. Neurocomputing 5 (4-5), 185–196.
Assis, W.O., Milani, B. E.A., 2004. Generation of optimal schedules for metro lines using model predictive control. Automatica 40 (8), 1397–1404.
Bemporad, A., Morari, M., 1999. Control of systems integrating logic, dynamics, and constraints. Automatica 35 (3), 407–427.
Cacchiani, V., Qi, J., Yang, L., 2020. Robust optimization models for integrated train stop planning and timetabling with passenger demand uncertainty. Transp. Res. B Methodol. 136, 1–29.
Caimi, G., Fuchsberger, M., Laumanns, M., Lüthi, M., 2012. A model predictive control approach for discrete-time rescheduling in complex central railway station areas. Comput. Oper. Res. 39 (11), 2578–2593.
Canca, D., Barrena, E., De-Los-Santos, A., Andrade-Pineda, J.L., 2016. Setting lines frequency and capacity in dense railway rapid transit networks with simultaneous passenger assignment. Transp. Res. B Methodol. 93, 251–267.
Cauligi, A., Chakrabarty, A., Di Cairano, S., Quirynen, R., 2022. PRISM: recurrent neural networks and presolve methods for fast mixed-integer optimal control. In: Learning for Dynamics and Control Conference. PMLR, pp. 34–46.
Cavone, G., van den Boom, T., Blenkers, L., Dotoli, M., Seatzu, C., De Schutter, B., 2022. An MPC-based rescheduling algorithm for disruptions and disturbances in large-scale railway networks. IEEE Trans. Autom. Sci. Eng. 19 (1), 99–112.
Chen, Z., Li, S., D'Ariano, A., Yang, L., 2022. Real-time optimization for train regulation and stop-skipping adjustment strategy of urban rail transit lines. Omega 110, 102631.
Chen, Z., Li, S., Zhang, H., Wang, Y., Yang, L., 2023. Distributed model predictive control for real-time train regulation of metro line based on dantzig-wolfe decomposition. Transp. B Transp. Dyn. 11 (1), 408–433.
De Schutter, B., Van den Boom, T., Hegyi, A., 2002. Model predictive control approach for recovery from delays in railway systems. Transp. Res. Rec. 1793 (1), 15–20.
Grüne, L., Pannek, J., 2017. Nonlinear Model Predictive Control. Springer.
Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Comput. 9 (8), 1735–1780.
Hou, Z., Dong, H., Gao, S., Nicholson, G., Chen, L., Roberts, C., 2019. Energy-saving metro train timetable rescheduling model considering ATO profiles and dynamic passenger flow. IEEE Trans. Intell. Transp. Syst. 20 (7), 2774–2785.
Huang, Y., Zhou, W., Xu, G., Deng, L., 2025. Integrated demand-oriented and energy-efficiency train timetabling and rolling stock circulation planning for an urban rail transit line. Transp. Res. C Emerg. Technol. 171, 104993.
Khadilkar, H., 2019. A scalable reinforcement learning algorithm for scheduling railway lines. IEEE Trans. Intell. Transp. Syst. 20 (2), 727–736.
Kingma, D.P., Ba, J.L., 2014. Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980.
Kuppusamy, P., Venkatraman, S., Rishikeshan, C.A., Reddy, Y.P., 2020. Deep learning based energy efficient optimal timetable rescheduling model for intelligent metro transportation systems. Phys. Commun. 42, 101131.
Li, S., De Schutter, B., Yang, L., Gao, Z., 2016. Robust model predictive control for train regulation in underground railway transportation. IEEE Trans. Control Syst. Technol. 24 (3), 1075–1083.
Liu, X., Dabiri, A., Wang, Y., De Schutter, B., 2023a. Modeling and efficient passenger-oriented control for urban rail transit networks. IEEE Trans. Intell. Transp. Syst. 24 (3), 3325–3338.
Liu, X., Dabiri, A., Xun, J., De Schutter, B., 2023b. Bi-level model predictive control for metro networks: integration of timetables, passenger flows, and train speed profiles. Transp. Res. E Logist. Transp. Rev. 180, 103339.
Liu, X., Dabiri, A., Wang, Y., De Schutter, B., 2024. Real-time train scheduling with uncertain passenger flows: a scenario-based distributed model predictive control approach. IEEE Trans. Intell. Transp. Syst. 25 (5), 4219–4232.
Luan, X., Corman, F., 2022. Passenger-oriented traffic control for rail networks: an optimization model considering crowding effects on passenger choices and train operations. Transp. Res. B Methodol. 158, 239–272.
Mayne, D.Q., Rawlings, J.B., Rao, C.V., Scokaert, P. O.M., 2000. Constrained model predictive control: stability and optimality. Automatica 36 (6), 789–814.
Nguyen, H. T.M., Chow, A. H.F., 2023. Adaptive rail transit network operations with a rollout surrogate-approximate dynamic programming approach. Transp. Res. C Emerg. Technol. 148, 104021.

Pan, H., Yang, L., Liang, Z., 2023. Demand-oriented integration optimization of train timetabling and rolling stock circulation planning with flexible train compositions: a column-generation-based approach. Eur. J. Oper. Res. 305 (1), 184–206.
Pu, S., Zhan, S., 2021. Two-stage robust railway line-planning approach with passenger demand uncertainty. Transp. Res. E Logist. Transp. Rev. 152, 102372.
Qi, J., Cacchiani, V., Yang, L., Zhang, C., Di, Z., 2021. An integer linear programming model for integrated train stop planning and timetabling with time-dependent passenger demand. Comput. Oper. Res. 136, 105484.
Qin, S.J., Badgwell, T.A., 2003. A survey of industrial model predictive control technology. Control Eng. Pract. 11 (7), 733–764.
Russo, L., Nair, S.H., Glielmo, L., Borrelli, F., 2023. Learning for online mixed-integer model predictive control with parametric optimality certificates. IEEE Control Syst. Lett. 7, 2215–2220.
Šemrov, D., Marsetič, R., Žura, M., Todorovski, L., Srdic, A., 2016. Reinforcement learning approach for train rescheduling on a single-track railway. Transp. Res. B Methodol. 86, 250–267.
da Silva, C. F.O., Dabiri, A., De Schutter, B., 2025. Integrating reinforcement learning and model predictive control for mixed-logical dynamical systems. IEEE Open J. Control Syst. .
Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. 15 (1), 1929–1958.
Tang, R., De Donato, L., Besinović, N., Flammini, F., Goverde, R. M.P., Lin, Z., Liu, R., Tang, T., Vittorini, V., Wang, Z., 2022. A literature review of artificial intelligence applications in railway systems. Transp. Res. C Emerg. Technol. 140, 103679.
Wang, P., Xiao, X., Nießen, N., 2024. Flexible rolling stock composition strategy in urban rail transit lines: the influences of various train units and station capacities. Transp. Res. C Emerg. Technol. 162, 104609.
Wang, S.-y., Chow, A. H.F., Ying, C.-s., 2025. Adaptive and flexible rail transit network service dispatching as a partially observable markov decision process. Transp. Res. C Emerg. Technol. 179, 105286.
Wang, X., Li, S., Tang, T., Yang, L., 2022. Event-triggered predictive control for automatic train regulation and passenger flow in metro rail systems. IEEE Trans. Intell. Transp. Syst. 23 (3), 1782–1795.
Wang, Y., D'Ariano, A., Yin, J., Meng, L., Tang, T., Ning, B., 2018. Passenger demand oriented train scheduling and rolling stock circulation planning for an urban rail transit line. Transp. Res. B Methodol. 118, 193–227.
Wang, Y., Ning, B., Tang, T., van den Boom, T. J.J., De Schutter, B., 2015. Efficient real-time train scheduling for urban rail transit systems using iterative convex programming. IEEE Trans. Intell. Transp. Syst. 16 (6), 3337–3352.
Wang, Y., Zhu, S., D'Ariano, A., Yin, J., Miao, J., Meng, L., 2021. Energy-efficient timetabling and rolling stock circulation planning based on automatic train operation levels for metro lines. Transp. Res. C Emerg. Technol. 129, 103209.
Williams, H.P., 2013. Model Building in Mathematical Programming. John Wiley & Sons.
Wu, X., Dong, H., Chi, K.T., 2021. Multi-objective timetabling optimization for a two-way metro line under dynamic passenger demand. IEEE Trans. Intell. Transp. Syst. 22 (8), 4853–4863.
Yang, L., Gao, Y., D'Ariano, A., Xu, S., 2024. Integrated optimization of train timetable and train unit circulation for a y-type urban rail transit system with flexible train composition mode. Omega 122, 102968.
Yin, J., Tang, T., Yang, L., Gao, Z., Ran, B., 2016. Energy-efficient metro train rescheduling with uncertain time-variant passenger demands: an approximate dynamic programming approach. Transp. Res. B Methodol. 91, 178–210.
Ying, C., Chow, A. H.F., Yan, Y., Kuo, Y.-H., Wang, S., 2024. Adaptive rescheduling of rail transit services with short-turnings under disruptions via a multi-agent deep reinforcement learning approach. Transp. Res. B Methodol. 188, 103067.
Ying, C.-S., Chow, A. H.F., Wang, Y.-H., Chin, K.-S., 2021. Adaptive metro service schedule and train composition with a proximal policy optimization approach based on deep reinforcement learning. IEEE Trans. Intell. Transp. Syst. 23 (7), 6895–6906.
Zhao, Y., Li, D., Yin, Y., Zhao, X., 2023. Integrated optimization of demand-driven timetable, train formation plan and rolling stock circulation with variable running times and dwell times. Transp. Res. E Logist. Transp. Rev. 171, 103035.