Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 59-26 (2025) 383–388

A state reduction approach for learning-based model predictive control for train rescheduling

C.F.O da Silva [*,1] **X. Liu** [*,1] **A. Dabiri** [*] **B. De Schutter** [*]

[*] *Delft University of Technology, The Netherlands (e-mail: {c.f.oliveiradasilva, x.liu-20, a.dabiri, b.deschutter}@tudelft.nl).*

Abstract: This paper proposes a state reduction method for learning-based model predictive control (MPC) for train rescheduling in urban rail transit systems. The state reduction integrates into a control framework where the discrete decision variables are determined by a learning-based classifier and the continuous decision variables are computed by MPC. Herein, the state representation is designed separately for each component of the control framework. While a reduced state is employed for learning, a full state is used in MPC. Simulations on a large-scale train network highlight the effectiveness of the state reduction mechanism in improving the performance and reducing the memory usage.

Keywords: learning for control, urban rail transit systems, model predictive control, hybrid systems, mixed-integer programming

1. INTRODUCTION

Urban rail transit systems play an important role in modern metropolitan mobility due to their high efficiency, capacity, and environmental sustainability. With the growing demand for public transportation and the rapid expansion of urban rail transit networks, ensuring the safety, punctuality, and adaptability of train operations has become increasingly important. Real-time train scheduling improves passenger satisfaction by reducing waiting and transfer times, and helps minimize operational costs under infrastructure and resource restrictions. Therefore, effective real-time train rescheduling approaches are essential to maintain service quality, particularly during disruptions, peak-hour congestion, and unexpected events.

Formulating train scheduling problems typically leads to a mixed-integer programming problem with operational constraints. Model predictive control (MPC) has emerged as an effective framework for real-time train scheduling, due to its ability to incorporate control and state constraints explicitly. De Schutter et al. (2002) applied MPC to train scheduling by formulating a framework that dynamically adjusts transfer connections to minimize overall train delays. Caimi et al. (2012) proposed an MPC framework for coordinating transfers and train allocations of timetable optimization in complex station environments. Liu et al. (2023) developed a simplified passenger flow model by approximating time-varying passenger demands as piecewise constant over fixed intervals, and then a mixed-integer linear programming (MILP)-based MPC approach is introduced for real-time train rescheduling. MPC has also been introduced for real-time train scheduling during various scenarios, including disturbances (Wang et al., 2020), disruptions (Cavone et al., 2020), and uncertain passenger flows (Liu et al., 2024). However, MPC encounters computational challenges in solving large-scale mixed-integer programming problems when applied to real-time train scheduling.

To address the growing complexity and uncertainty in urban rail operations, learning-based approaches have gained increasing attention in the research of train scheduling problems. Gattermann-Itschert et al. (2023) introduced a random forest classifier to replace complex penalty structures in crew scheduling by predicting planners' preferences, achieving higher acceptance rates with minimal cost increase. Ying et al. (2021) applied proximal policy optimization to train scheduling with flexible train composition, using neural networks for policy learning and a masking scheme to enforce constraint satisfaction. Kuppusamy et al. (2020) addressed energy-efficient timetable rescheduling by training a long short-term memory (LSTM) network to select optimal train operation modes under varying conditions. Wang et al. (2022) designed a deep neural network to optimize train dwell times in metro systems by balancing the waiting time on the platforms and the travel time onboard. By leveraging gated recurrent units and graph attention networks, it captures spatio-temporal passenger flow patterns and inter-train interactions. However, learning-based approaches typically have constraint satisfaction issues, which limit their applicability, as safety remains a crucial requirement in railway operations.

Recent studies on mixed-integer programming problems have explored the integration of learning-based and optimization-based approaches, which aim to combine the computational efficiency of learning methods with the constraint satisfaction capabilities of optimization-based approaches. In Cauligi et al. (2022) and Masti et al. (2020), the discrete and continuous decision variables of

[1] These authors contributed equally to this work.

10.1016/j.ifacol.2025.12.065

the mixed-integer programming problem are determined separately by a supervised learning classifier and by the solution of an optimization problem, respectively. In a similar setting, da Silva et al. (2024) applied reinforcement to determine the discrete decision variables and optimization to compute the continuous decision variables. Integrated approaches have also been explored in train scheduling problems. Zhang et al. (2024) combined reinforcement learning with constrained optimization for train rescheduling, where integer variables were determined using a double deep Q-network, after which the MILP problem was transformed into a linear programming problem for efficient solution. However, the proposed learning-based approach has scaling issues and is limited to small-scale cases. Our recent work (Liu et al., 2025) integrated machine learning into MPC to better capture passenger flow patterns and improve rescheduling responsiveness. To enhance computational efficiency, presolve techniques were developed to reduce the integer solution space, and an LSTM network was trained to predict integer variables. Nonetheless, the use of learning introduced a modest degradation in closed-loop performance.

Based on previous work in Liu et al. (2025), the current paper introduces a state reduction approach for learning-based MPC for train rescheduling. Specifically, a multi-resolution state design is integrated into the control framework, where a low-resolution state space is utilized for learning, while the full state space is employed in MPC. For the same case study as in Liu et al. (2025), we demonstrate that the proposed method improves closed-loop performance and reduces computational resource usage.

The remainder of this paper is structured as follows. Section 2 introduces the train scheduling model and the corresponding problem description. Section 3 introduces the proposed state reduction methodology and its integration with the learning-based MPC approach. Section 4 presents a case study to illustrate the effectiveness of the developed approach. Section 5 concludes the paper and outlines future research directions.

2. PROBLEM DESCRIPTION

We apply the passenger flow model developed in Liu et al. (2025) for the train rescheduling problem, where trains depart at regular intervals, time-varying passenger flows are approximated as piecewise constant functions over fixed time windows, and the composition of trains is flexibly adjusted to meet real-time passenger demand. A concise overview of the model and its associated optimization problem is provided below. For comprehensive descriptions and theoretical foundations, readers are referred to Liu et al. (2025).

We consider the problem of minimizing passenger delays and operational costs within the operational constraints. In urban rail transit systems, a train service starts at the origin station, stops at each station along the route, and then either returns to the depot or links with another service at the depot. The total passenger delay relative to platform p and train service k_p is defined as

$$\begin{aligned} J_p^{\text{pass}}(k_p) =& n_p(k_p)\left(d_p(k_p) - d_p^{\text{pre}}(k_p)\right) + \\ &+ n_p^{\text{after}}(k_p)\left(d_p^{\text{pre}}(k_p+1) - d_p(k_p)\right) \end{aligned} \tag{1}$$

where $d_p^{\text{pre}}(k_p)$ represents the predetermined departure time of train service k_p at platform p in the original timetable, and $d_p(k_p)$ is the actual departure time of train service k_p at platform p, $n_p(k_p)$ and $n_p^{\text{after}}(k_p)$ are the number of passengers at platform p at time $d_p^{\text{pre}}(k_p)$ and immediately after $d_p(k_p)$, respectively. The operational costs are described by

$$J_p^{\text{cost}}(k_p) = \ell_p(k_p)E_p^{\text{energy}} + \eta_{k_p,p}E_p^{\text{add}}, \tag{2}$$

where $\ell_p(k_p)$ is the train composition (i.e., the number of train units) of train service k_p when it departs from platform p, E_p^{energy} is the average energy consumption for the train service from platform p to the next platform, $\eta_{k_p,p}$ is a binary variable describing whether the train composition of the train service is changed, and E_p^{add} is the cost for changing the train composition. The cost (1) is nonlinear, but it can be approximated by a linear expression by using upper and lower bounds for the variable $d_p(k_p) \in [d_p^{\text{pre}}(k_p), d_p^{\text{pre}}(k_p+1))$ as follows:

$$\begin{aligned} J_p^{\text{pass}}(k_p) \approx\ & w_3 n_p(k_p)\left(d_p^{\text{pre}}(k_p+1) - d_p^{\text{pre}}(k_p)\right) + \\ &+ n_p^{\text{after}}(k_p)\left(d_p^{\text{pre}}(k_p+1) - d_p^{\text{pre}}(k_p)\right), \end{aligned} \tag{3}$$

where w_3 is a weighing factor used to reduce the approximation error.

The passenger flow model and the corresponding optimization problem are given as:

$$\min J(\kappa_0) := \sum_{p\in\mathcal{P}} \sum_{k_p\in\mathcal{N}_p(\kappa_0)} \left(w_1 J_p^{\text{pass}}(k_p) + w_2 J_p^{\text{cost}}(k_p)\right), \tag{4a}$$

subject to :

$$d_p^{\text{pre}}(k_p) \le d_p(k_p) < d_p^{\text{pre}}(k_p+1), \tag{4b}$$

$$d_p(k_p) = a_p(k_p) + \tau_p(k_p), \tag{4c}$$

$$a_p(k_p+1) = d_p(k_p) + h_p(k_p), \tag{4d}$$

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

$$\begin{aligned} n_p(k_p+1) = n_p(k_p) &+ \rho_p(k_p+1)\left(d_p^{\text{pre}}(k_p+1) - d_p^{\text{pre}}(k_p)\right) \\ &+ n_p^{\text{trans}}(k_p) - n_p^{\text{depart}}(k_p), \end{aligned} \tag{4f}$$

$$n_p^{\text{depart}}(k_p) \le C_p(k_p), \tag{4g}$$

$$C_p(k_p) = \ell_p(k_p)C_{\max}, \tag{4h}$$

$$\ell_{\min} \le \ell_p(k_p) \le \ell_{\max}, \tag{4i}$$

$$n_p^{\text{depart}}(k_p) \le n_p^{\text{before}}(k_p), \tag{4j}$$

$$\begin{aligned} n_p^{\text{before}}(k_p) = n_p(k_p) &+ \rho_p(k_p+1)\left(d_p(k_p) - d_p^{\text{pre}}(k_p)\right) \\ &+ n_p^{\text{trans}}(k_p), \end{aligned} \tag{4k}$$

$$n_p^{\text{arrive}}(k_p) = n_{\text{p}^{\text{pla}}(p)}^{\text{depart}}(k_p), \tag{4l}$$

$$n_p^{\text{after}}(k_p) = n_p^{\text{before}}(k_p) - n_p^{\text{depart}}(k_p). \tag{4m}$$

where $a_p(k_p)$, $\tau_p(k_p)$, $h_p(k_p)$ represent the decision variables for the arrival time, dwell time, and headway of train service k_p at platform p, respectively; $\rho_p(k_p+1)$ is the passenger flow for train service k_p+1 at platform p, $n_p^{\text{depart}}(k_p)$ is the number of passengers that boarded train service k_p, and $n_p^{\text{trans}}(k_p)$ is the number of passengers that transfer to platform p since the departure of train service k_p; $C_p(k_p)$ represents the total capacity of train service k_p, and $C_{\max}$ is the capacity of each train unit. The integer decision variables are the order of the trains at the terminal station of a line, and the number of train compositions $\ell_p(k_p)$ when train service k_p departs from the terminal station.

By using the transformation method in Bemporad and Morari (1999), the finite-horizon optimal control problem

for the MPC controller can finally be defined as (Liu et al., 2025):

$$\min_{\substack{\boldsymbol{x}(\kappa_0),\\ \boldsymbol{\epsilon_c}(\kappa_0),\boldsymbol{\epsilon_d}(\kappa_0)}} J^{(\mathrm{MPC})}(\chi(\kappa_0)) := \sum_{\kappa=\kappa_0}^{\kappa_0+N_p-1} L(x(\kappa),\epsilon_{\mathrm{c}}(\kappa),\epsilon_{\mathrm{d}}(\kappa)) \tag{5a}$$

$$\text{s.t.} \quad x(\kappa+1) = A_\kappa x(\kappa) + B_{1,\kappa}\epsilon_{\mathrm{c}}(\kappa) + B_{2,\kappa}\epsilon_{\mathrm{d}}(\kappa), \tag{5b}$$

$$D_{3,\kappa}x(\kappa) + D_{1,\kappa}\epsilon_{\mathrm{c}}(\kappa) + D_{2,\kappa}\epsilon_{\mathrm{d}}(\kappa) \le D_{4,\kappa}, \tag{5c}$$
$$\kappa = \kappa_0, \cdots, \kappa_0 + N_p - 1,$$

where $J^{(\mathrm{MPC})}(\chi(\kappa_0))$ represents the cost function as defined in (4), $\boldsymbol{\epsilon_c}(\kappa_0) = [\epsilon_{\mathrm{c}}{}^\intercal(\kappa_0), \epsilon_{\mathrm{c}}{}^\intercal(\kappa_0+1), \ldots, \epsilon_{\mathrm{c}}{}^\intercal(\kappa_0+N_p-1)]^\intercal$ represents the continuous decision variables over the prediction horizon, $\boldsymbol{\epsilon_d}(\kappa_0) = [\epsilon_{\mathrm{d}}{}^\intercal(\kappa_0), \epsilon_{\mathrm{d}}{}^\intercal(\kappa_0+1), \ldots, \epsilon_{\mathrm{d}}{}^\intercal(\kappa_0+N_p-1)]^\intercal$ expresses the discrete decision variables over the prediction horizon, and N_p is the prediction horizon. Particularly, $\boldsymbol{\epsilon_d}(\kappa_0)$ represents the train composition, and $\boldsymbol{\epsilon_c}(\kappa_0)$ describes the departure and arrival times for each train service. The system state is expressed as

$$\chi(\kappa) := [x^\intercal(\kappa_0),\ \boldsymbol{\rho}^\intercal(\kappa)]^\intercal, \tag{6}$$

where $\boldsymbol{x}(\kappa_0) = [x^\intercal(\kappa_0), x^\intercal(\kappa_0+1), \ldots, x^\intercal(\kappa_0+N_p-1)]^\intercal$ collects the dynamic component of the state trajectory over the prediction horizon, $\boldsymbol{\rho}(\kappa_0) = [\rho(\kappa_0)^\intercal, \rho(\kappa_0+1)^\intercal, \ldots, \rho(\kappa_0+N_{\mathrm{p}}-1)^\intercal]$ represents a concatenated vector with the expected passenger flows for each platform of the train line over the prediction horizon. More specifically, $x(\kappa_0)$ is composed of the number of passengers at each platform, the train composition for each running train service, and the number of trains in each depot of the line. The optimization program (5) can be either a mixed-integer nonlinear program (MINLP) or a linear mixed program (MILP) depending on whether the accurate nonlinear cost (1) or the approximate linear cost (3) is chosen.

3. STATE REDUCTION APPROACH FOR LEARNING-BASED MPC

This section focuses on the proposed state reduction approach, which integrates into the learning-based MPC framework presented in Liu et al. (2025). In the following, we describe the decoupling of discrete and continuous decision variables, the design of the state representation with different resolutions for learning and MPC, and the training and inference of the learning-based MPC approach with state reduction.

3.1 Decoupling of the decision variables

Consider a supervised learning classifier $\phi : \chi \mapsto \boldsymbol{\epsilon_d}$ that approximates the mapping from the state χ into the optimal discrete decision variables $\boldsymbol{\epsilon_d^*}$ over the prediction horizon, i.e., $\phi(\chi) \approx \boldsymbol{\epsilon_d^*}(\chi)$, where $\boldsymbol{\epsilon_d^*}(\chi)$ is the solution of (5) in the conditions specified by the state χ. During online operation, the predicted discrete variables $\boldsymbol{\epsilon_d} = \phi(\chi)$ can be used to turn the original problem (5) into a nonlinear (NLP) or linear (LP) problem, depending again on the choice of the performance index. Then the continuous decision variables – the departure and arrival times – can be determined by the solution of the following optimization problem:

$$\min_{\boldsymbol{x}(\kappa_0),\boldsymbol{\epsilon_c}(\kappa_0)} J_{\mathrm{d}}^{(\mathrm{MPC})}(\chi(\kappa_0),\ \boldsymbol{\epsilon_d}) := \sum_{\kappa=\kappa_0}^{\kappa_0+N_p-1} L^{(\epsilon_{\mathrm{d}})}(x(\kappa),\epsilon_{\mathrm{c}}(\kappa)) \tag{7a}$$

$$\text{s.t.} \quad x(\kappa+1) = A_\kappa x(\kappa) + B_{1,\kappa}\epsilon_{\mathrm{c}}(\kappa) + B_{2,\kappa}^{(\epsilon_{\mathrm{d}})}, \tag{7b}$$

$$D_{3,\kappa}x(\kappa) + D_{1,\kappa}\epsilon_{\mathrm{c}}(\kappa) \le D_{4,\kappa}^{(\epsilon_{\mathrm{d}})}, \tag{7c}$$
$$\kappa = \kappa_0, \cdots, \kappa_0 + N_p - 1,$$

where the cost $L^{(\epsilon_{\mathrm{d}})}$ and the matrices $(B_{2,\kappa}^{(\epsilon_{\mathrm{d}})}, D_{4,\kappa}^{(\epsilon_{\mathrm{d}})})$ are introduced to reflect that ϵ_{d} no longer is a decision variable.

3.2 State reduction

State dimensionality reduction in neural networks has several benefits, from training to inference. In a simpler state space, the neural network typically generalizes better, resulting in improved accuracy (Chandrashekar and Sahin, 2014). During training, the usage of computational resources can be lowered, as a lower-dimensional dataset occupies less memory. Alternatively, while maintaining the storage requirement identical, the number of points in the training set can be increased for a lower-dimensional state, which may enhance accuracy. Moreover, a lower-dimensional state space allows for a simpler neural network architecture since fewer neurons are required in each layer, allowing faster inference times. These benefits highlight the importance of proper design of the state space for learning. In what follows, we discuss a state reduction method that is tailored to the train rescheduling problem.

The core idea is to partition the vector $\boldsymbol{\rho}(\kappa)$ into a predetermined number N_{s} of segments, and then compute the average passenger flow for each segment. To illustrate this process, we consider the passenger flow at platform p over a number $M = H \cdot N_{\mathrm{s}}$ of train services $\boldsymbol{\rho}_p = [\rho_p(k_p), ..., \rho_p(k_p+M-1)]$, where H is the length of the segments. Let $[\boldsymbol{\rho}_p]_j$ be the jth element of the vector $\boldsymbol{\rho}_p$ and $[\boldsymbol{\rho}_p]_{i:j}$ a slice of the vector $\boldsymbol{\rho}_p$ between the indices i and j provided that $i < j$. The state reduction procedure can be expressed as

$$[\boldsymbol{\rho}_p^{\mathrm{reduced}}]_k = \mathrm{mean}([\boldsymbol{\rho}_p]_{k\cdot H:(k+1)\cdot H}) \text{ for } k = 0, ..., N_{\mathrm{s}} \tag{8}$$

where $\mathrm{mean}([\boldsymbol{\rho}_p]_{i:j}) = \frac{1}{j-i}\sum_{q=i}^{j-1}[\boldsymbol{\rho}_p]_q$. This operation reduces the number of elements in the original vector by a factor of H. The state reduction preserves the mean of the original vector, thereby ensuring no error in the computation of the total number of passengers in the prediction window. Hereafter, the collection of the passenger flow $\boldsymbol{\rho}_p^{\mathrm{reduced}}$ across all platforms of a train line is denoted by $\boldsymbol{\rho}^{\mathrm{reduced}}$.

The discrete decision variable $\boldsymbol{\epsilon_d}$, which represents the train composition of the train services over the prediction horizon, is relatively insensitive to fast changes in passenger flow over time. In this paper, we consider that train compositions can only be modified at depots located at the terminal stations of the line. Thus, once the composition of the train of a particular train service is established, it has a lasting effect on the optimization problem (7) as it cannot be changed on any of the intermediate platforms. Consequently, the variable $\boldsymbol{\epsilon_d}$ is more sensitive to long temporal patterns of the passenger flow rather than short ones. Therefore, we argue that a reduced passenger flow

vector is more suitable for learning due to its lower dimensionality and its capacity to adequately capture the essential characteristics of the passenger flow required to estimate the optimal train composition. Hence, the state described in (6) is redefined for the classifier ϕ as

$$\chi^{\text{learning}}(\kappa) = [x^\intercal(\kappa),\ (\boldsymbol{\rho}^{\text{reduced}}(\kappa))^\intercal]^\intercal. \tag{9}$$

On the other hand, in the optimization problem (7), the unreduced passenger flow $\boldsymbol{\rho}$ is required to accurately model system dynamics. Furthermore, the continuous decision variable $\boldsymbol{\epsilon_c}$, including departure and arrival times, has a relatively faster effect on (7) and can react to faster changes in passenger flows. Hence, passenger flow $\boldsymbol{\rho}$ with the original resolution is more appropriate to be included in the MPC formulation of (7) that computes $\boldsymbol{\epsilon_c}$.

In essence, we advocate for the use of states with different levels of resolution for learning and optimization, leveraging the fact that the decision variables exhibit different sensitivities to changes in passenger flow. A low-resolution state is employed in learning to determine the discrete actions, while the original state representation is used in the MPC formulation to compute the continuous actions. In this manner, the state representation is tailored to the specific requirements of each component of the framework.

3.3 Training and inference

The procedure for the construction of the training dataset is described in Alg. 1. The training dataset is composed of tuples of states and their optimal solution for the discrete decision variables $\mathcal{D} = \{(\chi^{(k)}_{\text{learning}},\ \boldsymbol{\epsilon}^{(*,k)}_{\mathbf{d}})\}_{k=1}^{N_{\mathcal{D}}}$. To generate a set of states that represent real operation, a number of episodes are started from random states within the operational range, and then the closed-loop trajectory with the optimal solution (7) is simulated for a predetermined number of steps.

Algorithm 1 Data acquisition

Input: training dataset $\mathcal{D} \leftarrow \{\}$
for episode $= 1, ..., N_{\text{episodes}}$ **do**
  Set random state χ
  for step $= 1, ..., N_{\text{steps}}$ **do**
    Get $\boldsymbol{\epsilon}^*_{\mathbf{d}}$ and $\boldsymbol{\epsilon}^*_{\mathbf{c}}$ by solving (5) for state χ
    Get χ_{learning} via state reduction
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\chi_{\text{learning}},\ \boldsymbol{\epsilon}^*_{\mathbf{d}})\}$
    Update χ by applying (5b) with $(\boldsymbol{\epsilon}^*_{\mathbf{d}},\ \boldsymbol{\epsilon}^*_{\mathbf{c}})$
  end for
end for
Output: training set $\mathcal{D}$

With the training set $\mathcal{D}$, the supervised learning classifier in ϕ can then be trained. We consider a set of hyperparameter configurations $\mathcal{P} = \{\mathcal{P}_i\}_{i=1}^{n_{\text{NN}}}$, such that for each $\mathcal{P}_i$, a classifier $\phi_i(\cdot)$ is trained via gradient descent. Finally, the trained neural networks are stored in a set $\mathcal{N} = \{\phi_i\}_{i=1}^{n_{\text{NN}}}$. As the choice of the classifier, we propose the use of recurrent neural networks, which are capable of encoding temporal dependencies in the discrete decision variables through their internal state. In particular, LSTM networks are selected for their ability to represent long-term dependencies in sequential data.

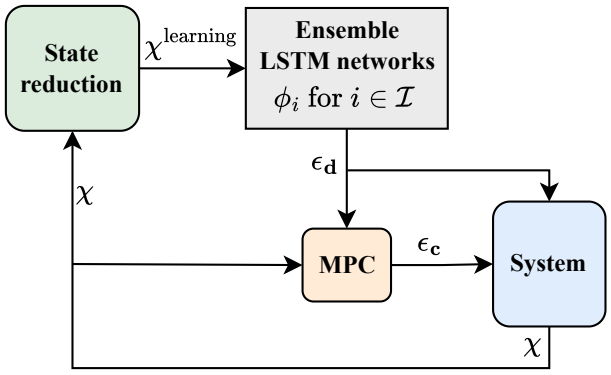

Fig. 1. A representation of the control framework with state reduction.

The learning-based MPC approach becomes infeasible when the approximator fails to provide a discrete decision variable $\boldsymbol{\epsilon_d} = \phi_i(\chi_{\text{learning}})$ that renders the problem (7) feasible. To mitigate this issue, we employ an ensemble of neural networks trained with heterogeneous hyperparameter configurations. Typically, after hyperparameter optimization, only the most accurate neural network is retained, while the others are discarded. In contrast, our approach evaluates each of the neural networks through closed-loop simulations of the system, as shown in Alg. 2. Based on this evaluation, a set $\mathcal{I} \subseteq \{1, ..., n_{\text{NN}}\}$ of the indices of the best-performing neural networks is selected and sorted from lowest to highest closed-loop total cost.

Algorithm 2 Formation of the ensemble

Input: set of trained neural networks $\mathcal{N}$
for $i = 1, ..., n_{\text{NN}}$ **do**
  for episode $= 1, ..., N_{\text{test}}$ **do** ▷ *Closed-loop test*
    Set random state χ
    for step $= 1, ..., N_{\text{steps}}$ **do**
      Get χ_{learning} via state reduction
      $\boldsymbol{\epsilon_d} \leftarrow \phi_i(\chi_{\text{learning}})$
      if (7) is feasible for $(\chi,\ \boldsymbol{\epsilon_d})$ **then**
        Retrieve $\boldsymbol{\epsilon_c}$
      else
        Use heuristics to find a feasible $\boldsymbol{\epsilon_d}$
        Get $\boldsymbol{\epsilon_c}$ by solving (7)
      end if
      Update χ by applying (7b) with $(\boldsymbol{\epsilon_d},\ \boldsymbol{\epsilon_c})$
    end for
  end for
end for
Output: $\mathcal{I} \subseteq \{1, ...n_{\text{NN}}\}$: ordered set of indices of the networks arranged from lowest to highest total cost.

The online inference procedure is described in Alg. 3 and Fig. 1. The core idea is to sequentially evaluate the trained neural networks in the ensemble, i.e., $\phi_i(\chi_{\text{learning}})$ for $i \in \mathcal{I}$, until a feasible solution for the problem (7) is found. If a feasible solution is not found by the supervised learning classifier, then heuristics described in Liu et al. (2025) can be used to restore feasibility, e.g., maintaining the same train composition of the previous time step or selecting the minimum number of train units for every train service. By construction, feasibility also implies that all the constraints are satisfied and that the continuous decision variables $\boldsymbol{\epsilon_c}$ are optimal with respect to the choice

of $\epsilon_{\mathbf{d}}$. As a result, integrating a learning-based approach with MPC can combine fast online evaluation with constraint satisfaction and optimality. Although the neural network ensemble is evaluated sequentially due to its lower computational cost, the proposed approach is flexible to other types of evaluation, e.g., parallel evaluation followed by selection of the most common action or the one with the lowest cost.

Algorithm 3 Online Inference

Input: set of indices $\mathcal{I}$, system state $\chi(\kappa)$
$\chi \leftarrow \chi(\kappa)$
`solution_found` $\leftarrow False$
Get χ_{learning} via state reduction
for $i \in \mathcal{I}$ **do**
  $\epsilon_{\mathbf{d}} \leftarrow \phi_i(\chi_{\text{learning}})$
  if (7) is feasible for (χ, $\epsilon_{\mathbf{d}}$) **then**
    Retrieve $\epsilon_{\mathbf{c}}$
    `solution_found` $\leftarrow True$
    Break for loop
  end if
end for
if `solution_found == False` **then**
  Use heuristics to find a feasible $\epsilon_{\mathbf{d}}$
  Get $\epsilon_{\mathbf{c}}$ by solving (7)
end if
Output: $\epsilon_{\mathbf{d}}$, $\epsilon_{\mathbf{c}}$

4. CASE STUDY

A three-line railway network in Beijing, shown in Fig. 2, is used to assess the performance of the proposed approach. The network has 3 bidirectional lines, 45 stations, and 3 transfer stations (ZXZ, XEQ, HY). There is a depot connected to each line: CPX for Changping Line, XZM for Line 13, and ZXZ for Line 8. Moreover, the passenger flow data are based on real-world data. The full description of the case study is provided in Liu et al. (2025). The experiments were conducted on Python with Intel XEON E5-6248R CPUs using PyTorch and Gurobi for machine learning and optimization, respectively. The code and datasets used in this work are publicly available [2].

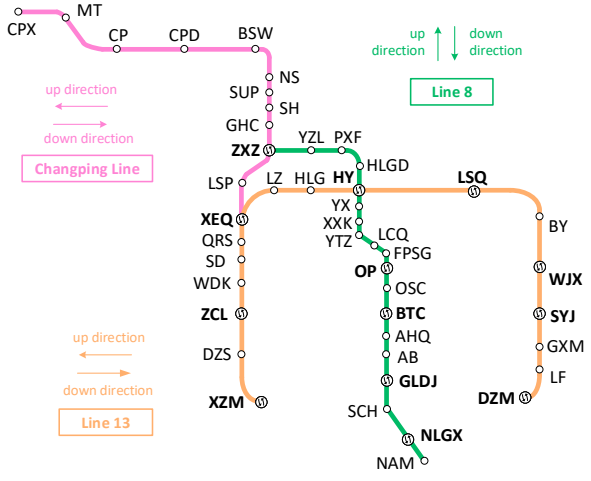

Fig. 2. Three-line railway network in Beijing.

[2] `https://github.com/fabcaio/railway_conf`

The dataset characteristics for the full and reduced states, obtained by Alg. 1, are represented in Table 1. In our case, we had the two originally contrasting goals of (i) improving accuracy by enlarging the dataset and (ii) reducing memory requirements, which was one of the computational bottlenecks in our setup. Both goals were achieved by applying the state reduced approach (8) with $N_{\text{s}} = 4$ and $H = 9$, which were adjusted via grid-search with validation data. Consequently, the state dimension was reduced from 4002 to 748. Moreover, the number of data points could be increased by a factor of 2.65, while decreasing the memory usage by a factor of 1.5. The datasets were then used for the training of $|\mathcal{P}| = 64$ LSTM networks. Subsequently, a subset of $|\mathcal{I}| = 15$ networks was selected to form the neural network ensemble by applying Alg. 2 with validation data. Each hyperparameter configuration $\{\mathcal{P}_i\}_{i=1}^{64}$ included distinct values for the learning rate, number of hidden neurons, dropout rate, and two Boolean variables indicating whether output masking, and learning rate scheduling were applied.

Table 1. Dataset information.

State	State dimension	Number of points	Memory (GB)
Full	4002	94871	5.00
Reduced	748	252207	3.29

The comparison of the closed-loop operation of seven different methods averaged over 7864 samples is shown in Table 2. The control time step and the MPC prediction horizon are set to $T = 4$ minutes and $N_p = 40$, respectively. The initial state of each closed-loop simulation is randomly chosen within the operational range and it is consistent across the methods. The maximum simulation length is chosen to be 80 time steps. In methods (I)–(III), both the discrete and continuous variables are determined by optimization. The benchmark (I) corresponds to the solution of (5) with a maximum solution time of 10 minutes, rendering the method unachievable since the MPC controller must find a solution within the control time step. Hence, benchmark (I) is used solely to define the optimality gap, measured as the percentage difference between the performance of a given method and that of benchmark (I). The methods (II) and (III) represent the solution of (5) with a maximum solution time of 4 minutes – equal to the control time step – and nonlinear and linear performance indices, respectively. In contrast, methods (IV)–(VII) employ the decoupling procedure described in Section 3.1, where the discrete variables are determined by an ensemble of neural networks, as described by Alg. 3 and Fig. 1, and the continuous variables are computed by the solution of the optimization problem (7) with a maximum solution time of 4 minutes. The approaches (IV) and (V) use the original state representation, while (VI) and (VII) employ the reduced state described in (9). Although methods (V) and (VII) use the linear performance index (3), their solutions are evaluated on the nonlinear index (1), ensuring consistency in the evaluation of all the methods.

As illustrated by Table 2, the optimality gaps of the learning-based approaches with unreduced state (IV) and (V) are significantly reduced by the proposed approaches (VI) and (VII). More specifically, comparing (IV) to (VI) the optimality gap dropped from 13.30% to 0.73%, and comparing (V) to (VII) the optimality gap decreases from

13.06% to 0.51%, showing the significant effect of the state reduction procedure (8). The learning-based approaches (IV)–(VII) reduce the computation time considerably compared to the optimization-based approaches (I)–(III). Particularly, the learning-based approach (VII) with the linear performance index (3) has the best overall performance considering optimality and computation time. In methods (IV) and (VI), the NLP (7) is not solved to optimality, as an early termination criterion is employed to reduce the CPU time. This heuristic considerably reduces the computation at the expense of a minimal increase in the optimality gap. This accounts for the slightly larger optimality gap of (VI) when compared to that of (VII).

Method	Optimality gap (%)			CPU time (s)		
	mean	max	std	mean	max	std
(i) Benchmark	0.00	0.00	0.00	258.18	600.41	287.56
(ii) MINLP	0.51	22.98	2.27	41.07	240.11	47.53
(iii) MILP	0.92	65.05	4.94	3.66	145.10	10.75
(iv) Learning + NLP	13.30	53.82	8.79	7.04	79.50	6.47
(v) Learning + LP	13.06	40.04	7.44	0.15	0.43	0.03
(vi) Learning + NLP with state reduction	**0.73**	**5.11**	**0.79**	**7.24**	**101.61**	**6.36**
(vii) Learning + LP with state reduction	**0.51**	**4.39**	**0.66**	**0.15**	**0.52**	**0.03**

Table 2. Comparison of the closed-loop performance for several approaches, with the proposed approach is highlighted in bold.

5. CONCLUSIONS

This paper has proposed a state reduction scheme for learning-based MPC in the context of train rescheduling, where different levels of resolution are used to represent the system state for learning and control. Simulation experiments conducted on a large-scale railway network revealed a dramatic decrease in suboptimality when the proposed state reduction was employed, thereby confirming its effectiveness. Future research will explore alternative state reduction techniques such as principal component analysis and autoencoders.

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).

REFERENCES

Bemporad, A. and Morari, M. (1999). Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3), 407–427.

Caimi, G., Fuchsberger, M., Laumanns, M., and Lüthi, M. (2012). A model predictive control approach for discrete-time rescheduling in complex central railway station areas. *Computers & Operations Research*, 39(11), 2578–2593.

Cauligi, A., Chakrabarty, A., Cairano, S.D., and Quirynen, R. (2022). PRISM: Recurrent neural networks and presolve methods for fast mixed-integer optimal control. In *Proceedings of the 4th Annual Learning for Dynamics and Control Conference*, 34–46. PMLR.

Cavone, G., van den Boom, T., Blenkers, L., Dotoli, M., Seatzu, C., and De Schutter, B. (2020). An MPC-based rescheduling algorithm for disruptions and disturbances in large-scale railway networks. *IEEE Transactions on Automation Science and Engineering*, 19(1), 99–112.

Chandrashekar, G. and Sahin, F. (2014). A survey on feature selection methods. *Computers & electrical engineering*, 40(1), 16–28.

da Silva, C.F.O., Dabiri, A., and De Schutter, B. (2024). Integrating reinforcement learning and model predictive control with applications to microgrids. *arXiv preprint arXiv:2409.11267*. Submitted for publication.

De Schutter, B., van den Boom, T., and Hegyi, A. (2002). Model predictive control approach for recovery from delays in railway systems. *Transportation Research Record*, 1793(1), 15–20.

Gattermann-Itschert, T., Poreschack, L.M., and Thonemann, U.W. (2023). Using machine learning to include planners' preferences in railway crew scheduling optimization. *Transportation Science*, 57(3), 796–812.

Kuppusamy, P., Venkatraman, S., Rishikeshan, C., and Reddy, Y.P. (2020). Deep learning based energy efficient optimal timetable rescheduling model for intelligent metro transportation systems. *Physical Communication*, 42, 101131.

Liu, X., da Silva, C.F.O., Dabiri, A., Wang, Y., and De Schutter, B. (2025). Learning-based model predictive control for passenger-oriented train rescheduling with flexible train composition. *arXiv preprint arXiv:2502.15544*. Submitted for publication.

Liu, X., Dabiri, A., Wang, Y., and De Schutter, B. (2023). Modeling and efficient passenger-oriented control for urban rail transit networks. *IEEE Transactions on Intelligent Transportation Systems*, 24(3), 3325–3338.

Liu, X., Dabiri, A., Wang, Y., and De Schutter, B. (2024). Real-time train scheduling with uncertain passenger flows: A scenario-based distributed model predictive control approach. *IEEE Transactions on Intelligent Transportation Systems*, 25(5), 4219–4232.

Masti, D., Pippia, T., Bemporad, A., and De Schutter, B. (2020). Learning approximate semi-explicit hybrid MPC with an application to microgrids. *IFAC-PapersOnLine*, 53(2), 5207–5212. 21st IFAC World Congress.

Wang, X., Li, S., Tang, T., and Yang, L. (2020). Event-triggered predictive control for automatic train regulation and passenger flow in metro rail systems. *IEEE Transactions on Intelligent Transportation Systems*, 23(3), 1782–1795.

Wang, Z., Pan, Z., Chen, S., Ji, S., Yi, X., Zhang, J., Wang, J., Gong, Z., Li, T., and Zheng, Y. (2022). Shortening passengers' travel time: A dynamic metro train scheduling approach using deep reinforcement learning. *IEEE Transactions on Knowledge and Data Engineering*, 35(5), 5282–5295.

Ying, C.S., Chow, A.H., Wang, Y.H., and Chin, K.S. (2021). Adaptive metro service schedule and train composition with a proximal policy optimization approach based on deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, 23(7), 6895–6906.

Zhang, H., Liu, X., Sun, D., Dabiri, A., and De Schutter, B. (2024). Integrated reinforcement learning and optimization for railway timetable rescheduling. *IFAC-PapersOnLine*, 58(10), 310–315.