



A novel Markov model for near-term railway delay prediction[☆]

Jin Xu^a, Weiqi Wang^b, Zheming Gao^{c,*}, Haochen Luo^d, Qian Wu^d

^a School of Management, Huazhong University of Science and Technology, Wuhan, Hubei 430074, China

^b Department of Mathematics and Statistics, Concordia University, Montreal, QC, H3G 1M8, Canada

^c College of Information Science and Engineering, Northeastern University, Shenyang, Liaoning 110819, China

^d Department of Industrial & Systems Engineering, Texas A&M University, College Station, TX 77843, USA



ARTICLE INFO

Keywords:

Railway delay prediction

Markov property test

Non-homogeneous Markov chain

Gaussian kernel density estimate

ABSTRACT

Predicting the near-future delay with accuracy for trains is momentous for railway operations and passengers' traveling experience. This work aims to design prediction models for train delays based on the Netherlands Railway data. We first develop a chi-square test to show that the delay evolution over stations follows a first-order Markov chain. We then propose a delay prediction model based on non-homogeneous Markov chains. To deal with the sparsity of the transition matrices of the Markov chains, we propose a novel matrix recovery approach that relies on Gaussian kernel density estimation. Our numerical tests show that this recovery approach outperforms other benchmark approaches in prediction accuracy. The Markov chain model we propose also shows to be better than other widely-used time series models with respect to both interpretability and prediction accuracy. Moreover, our proposed model does not require a complicated training process, which is capable of handling large-scale forecasting problems.

1. Introduction

Railways are a primary means of transportation that enables the movement of a vast amount of goods and people worldwide, with approximately 10,000 billion freight tonne-kilometers and 3,000 billion passenger-kilometers traveled annually ([Railway statistics 2015 report, 2015](#)). The quality of railway operations and passengers' traveling experience is heavily dependent on punctuality. A train is considered punctual if it arrives or departs at the planned time specified by the timetable. However, railway operations inevitably encounter disturbances and various factors, such as severe weather conditions, unexpected mechanical failures, drivers' and travelers' behavior, and temporary speed restrictions ([Li et al., 2021](#); [Nabian, Alemaazkoor, & Meidani, 2019](#); [Olsson & Haugland, 2004](#)) that can impact the railway system's punctuality. As shown in [Harris, Mjøsund, and Haugland \(2013\)](#), the delay variations across different railway lines in Norway are significant, with the best-performing route achieving 94.4% punctuality and the worst routes only near 80%, against a target of 90%. Punctuality is critical for railway operations, and forecasting delays in real-time can help improve the railway system's efficiency and passenger satisfaction.

Railway control systems rely heavily on accurate online delay prediction. As stated in [Cacchiani et al. \(2014\)](#), [Caimi et al. \(2012\)](#), [Corman and Meng \(2014\)](#), delay forecast is a crucial prerequisite for railway control, enabling railway operators to perform conflict-free disposition schedules or reschedules. By utilizing such forecasts, railway operators can estimate the feasibility of rolling-stock and crew circulation plans ([Corman & Kecman, 2018](#)), improving the reliability of the railway system and reducing operating costs. In addition, providing accurate delay information to passengers can help them adjust their travel plans, leading to a better customer experience. Therefore, accurate delay prediction is critical for the successful operation of railway systems, and the benefits extend to both railway operators and passengers.

In this paper, we aim to develop a data-driven framework that can provide online delay forecasting of trains in a large railway system in the near future. However, developing an accurate online delay forecasting framework for a large railway system is a challenging task, primarily due to the significant heterogeneity in delay distribution across different trains and stations. Each train may experience unique delays at various stations, and trains operating in different regions

[☆] Zheming Gao's research is supported by the NSFC Key Supported Project of the Major Research Plan under Grant 92267206, the National Natural Science Foundation of China under Grant 72201052 and the Fundamental Research Funds for the Central Universities, China under Grant N2204017. Weiqi Wang's research is supported by Concordia University, Canada through the Horizon Postdoctoral Fellowship program.

* Corresponding author.

E-mail addresses: xu_jin@hust.edu.cn (J. Xu), weiqi.wang@concordia.ca (W. Wang), gaozheming@ise.neu.edu.cn (Z. Gao), hcluo@tamu.edu (H. Luo), hi_qianwu@tamu.edu (Q. Wu).

or stations may have distinct delay processes. As a result, training a single unified model to predict delays for all trains and stations is impractical. Instead, it is necessary to build customized models for each train at specific stations. However, this approach requires a large number of models to be trained, which can be time-consuming and computationally intensive. For instance, the Netherlands Railway data that we use in our paper (INFORMS Railway Applications Section, 2018) contain more than 6000 trains that travel 150000 stations in total from the timetable. If a customized model is trained for each station, then we need to train more than 150000 models in total.

Second, building customized models may suffer from inadequate historical records for training. For example, the worst route in Harris et al. (2013) achieves an 80% punctuality rate, which implies that most of the historical data are still punctual. As we will show later in our paper, the Netherlands Railway data that we use have only a small portion that recorded large delays. Moreover, the variety of delay causes makes each type or value of delay at a particular station even less observed. It is challenging to train an accurate prediction model without adequate delay observations.

Third, although machine learning models such as artificial neural networks (ANN) (Huang et al., 2020) can achieve high levels of prediction accuracy, they often lack interpretability, making it difficult for railway operators to understand and act on the insights generated by these models. Additionally, these models typically require large amounts of data for training, and multiple data streams as input, which can be problematic if certain data streams are missing or the training data are inadequate. Furthermore, these models often require significant manual intervention in hyper-parameter tuning, which can be time-consuming and impractical when multiple models need to be built.

To overcome the challenges above, we propose a non-homogeneous Markov chain model to predict the delay over stations. The theoretical and industrial contributions of our work are summarized as follows.

- **Markov property test:** We propose a chi-square test to compute the order of the Markov chain model. The numerical results show the train's delay evolution over stations has the first-order Markov property. This testing method works in the scenario when the transition matrices of the Markov chain are sparse due to the lack of training data. This test method is also applicable in other application scenarios where practitioners need to determine the order of the Markov chain model.
- **Markov prediction model:** After showing that the delay evolution of trains has the first-order Markov property, we build our prediction model by modeling the delay evolution over stations as a non-homogeneous Markov chain. Each transition matrix in our model can characterize the unique pattern of delay evolution between two adjacent stations that a train travels through. The delay evolution over multiple stations can be captured by the Chapman–Kolmogorov equations of the Markov chain model. This prediction model has both prediction accuracy and interpretability.
- **Transition matrix recovery method:** We propose a Gaussian-kernel-based method to recover the empty rows in the transition matrices when the transition matrices are sparse. This recovery method can capture the delay transition probabilities from inadequate historical observations. We show that this recovery method achieves a higher prediction accuracy than other benchmark approaches. This recovery method can be potentially extended to other applications where the transition matrices of the Markov chain are sparse.
- **Accuracy and lightweight:** Our numerical experiments demonstrate that the proposed delay prediction model is effective and efficient. In comparison to benchmark time-series prediction models, our model achieves higher accuracy and can be applied in real-time delay prediction scenarios for railway systems. Furthermore, the model's training process is fast and does not require

complex computations or parameter tuning, making it suitable for systems with a large number of trains and stations. Importantly, the proposed Markov chain model uses only historical delay data for training and does not rely on other features related to railway operations that may not be available in all scenarios. These findings highlight the potential of our model for practical use in railway operations and suggest avenues for future research on predicting and managing delays in transportation systems.

We organize the rest of the paper as follows. We review the related work in Section 2 and describe the railway delay forecasting problem in Section 3. We provide a framework to test the order of the Markov chain model in Section 4. In Section 5, we provide a detailed description of the proposed Markov chain model. We test our model and compare it with other benchmarks in Section 6. We provide the conclusion and discuss our future research in Section 7.

2. Related work

The models for traffic delay prediction have been investigated from different perspectives. Some research focuses on the relationship between railway delay and various factors in railway systems. For instance, Olsson and Haugland (2004) analyze the correlation between train departure delay, number of passengers, and occupancy ratio using Norwegian railway data. Goverde (2005) adopts linear regression to explain the dependencies between train services with a transfer connection and the impact of the bottleneck in a particular station at Eindhoven. Flier et al. (2009) employ linear regression to analyze the delay dependencies on resource conflicts and maintained connections using the Swiss Railways data. Marković et al. (2015) investigate support vector regression (SVR) models that capture the relation between passenger train arrival delays and various characteristics of a railway system, such as the passenger train category, the scheduled time of arrival at the station, the infrastructure influence, the percentage of the journey completed distance-wise, and the traveling distance. All these research studies mainly focus on the dependency between delay and existing system characteristics. They aim to understand the factors influencing railway delays so that to provide guidance to system operators and planners. Performing real-time delay forecasting is not the main focus of these studies.

Recently, machine learning approaches have been used for delay prediction. Lessan, Fu, and Wen (2019) propose a Bayesian-network-based train delay prediction model to characterize the complexity and dependency nature of different train operations. Yaghini, Khoshrafter, and Seyedabadi (2013) use ANN models to predict monthly averaged passenger train delays for Iranian railways, rather than performing real-time delay prediction. Oneto et al. (2018) propose shallow and deep extreme learning algorithms incorporating the types of running day (whether it is a weekday or holiday), dwell times, and the running times for all the other trains running over the same section of the railway network during the day, to predict the delays in Italian railways. Nabian et al. (2019) propose a random forest model by incorporating many features of the railway system, such as distance, number of stops, composite change, and driver change. These models rely on features other than the train's historical delay information for training and may not be applicable in scenarios where these additional features are not recorded. Moreover, training these models could be time-consuming, especially when the number of trains and stations is large.

Some autoregressive (AR) based time series models that only rely on historical delay information have been applied to traffic forecasting. Suwardo, Napiah, and Kamaruddin (2010) employ the Autoregressive Integrated Moving Average (ARIMA) model to predict bus travel time on the Ipoh-Lumut corridor in Malaysia. Pavlyuk (2017) provides a systematic review of multivariate AR-based models in short-term traffic flow forecasting. In forecasting problems with integer-valued time series, integer-valued AR-based models are proposed in the literature (Freeland & McCabe, 2004; Jin-Guan & Yuan, 1991; Kim & Park,

2008). Recently, copula-based time-series models have gained popularity for their capability of characterizing the non-linear dependence in time series (Huang & Emura, 2021; Patton, 2013; Sun, Lee, & Emura, 2020; Sun et al., 2020). However, similar to the AR-based models, most of these copula-based models work in regular time series where the evolution of time series can be captured by a unified model. In the scenario of railway delay forecasting, the delay evolution over stations is not a time series, and it is significantly heterogeneous and complex. So a unified time-series model may not be suitable for railway delay forecasting.

Hybrid statistical time series and machine learning models have been proposed for traffic forecasting. Ma, Antoniou, and Toledo (2020) concatenate a fundamental neural network model with the ARIMA model for network-wide traffic forecasting. Ge et al. (2021) propose a hybrid ARIMA model and fuzzy SVR for high-speed railway passenger traffic forecasting. Besides, the recurrent neural network (RNN) models are also utilized for traffic forecasting (Osipov et al., 2020; Smyl, 2020). However, these learning-based models may lack interpretability, and some statistical assumptions made in these models are hard to justify in practice.

Markov chain is a stochastic model that describes a sequence of random events where each event depends only on the states attained in several previous events. The order of a Markov chain specifies how many previous events the current state depends on. Markov chain has been used for forecasting in many scenarios. For instance, Verma et al. (2018) use Markov chains to forecast wind speed. Carpinone et al. (2015) use first and second-order Markov chains to predict short-term wind power. Zhou et al. (2018) use a Markov-chain-based model to predict the demand in bike sharing systems. However, it is still unknown if the delay evolution over stations for railway systems follows a Markov chain of a certain order.

In summary, there are very few railway delay forecasting models with accuracy, lightweight, and interpretability altogether. Although the Markov chain is used for prediction in multiple scenarios, it is still unclear if the Markov chain model can capture the delay evolution of railway systems. Therefore, in our study, we plan to investigate the Markov property of delays in railway systems and design prediction models that rely on Markov chains without training burdens.

3. Problem description

In this section, we describe the goal of our prediction model and the data that we use in our work.

3.1. Delay prediction

The objective of this research work is to develop a real-time delay prediction model that can forecast each train's delay property in the near future based on the delay at the current station and those that the train has traveled through on that day. Specifically, we aim to predict the delay at the station where the train is scheduled to arrive 20 min later (or near). Predicting into the next 20 min is common and helpful in practice (see Caimi et al., 2012; INFORMS Railway Applications Section, 2018). One of the reasons is that a larger error would occur when forecasting a farther future, as pointed out in Caimi et al. (2012). Another reason is that prediction results are usually used in railway operations. The farther one plans into the future, the more complex the operation task becomes. Hence, we consider 20 min as the prediction horizon.

The model needs to predict each train's delay value in minutes, delay trend (compared with the delay at the current station, whether the delay at the predicted station will decrease or increase), and the delay's jump property (whether the predicted delay will increase or decrease for more than two minutes from the current delay). These predictions will be helpful for railway operators to make operations

and for passengers to make travel plans. A demonstrative graph of the delay prediction is given in Fig. 1.

Note that a train may have different activities in the same station. For example, “V” denotes the departure, “D” denotes passage without stop, “A” denotes arrival, “KV” denotes the departure at a short stop, and “KA” denotes the arrival at a short stop. A train can have “V” and “A” (or “KV” and “KA”) at the same station, as shown in Fig. 1. Since a train may stop at a station longer (or shorter) than scheduled, train delays can be different at the arrival and departure epochs at the same station. Thus, in our analysis, we treat the delays upon the arrival and departure at each station differently. Thus our model will forecast the delay at the station with the activity closest to 20 min later from the scheduled timetable.

3.2. Data description

The data used for our analysis is provided by ProRail, an infrastructure manager responsible for track maintenance and coordinating train operations (INFORMS Railway Applications Section, 2018). The raw dataset contains railway operation history from September 4, 2017, to December 9, 2017, and it consists of a planned timetable and the realization data. The historical data do not contain the last four Tuesdays (2017-11-14, 2017-11-21, 2017-11-28, 2017-12-05). In addition, only data for Mondays, Tuesdays, Thursdays, and Fridays are included in the historical data. The weekend data are not included since the operations on weekends are often altered due to maintenance work. The Wednesday data is also excluded since extra trains are operated on a busy part of the network between Eindhoven and Amsterdam on Wednesdays.

The raw data mainly contains the following information:

- **Planned timetable.** The planned timetable contains the planned arrival and departure times for each train at each station it travels through. The timetable is the same for Mondays, Tuesdays, Thursdays, and Fridays.
- **Actual historical train performance.** The realization data contains the arrival time, departure time, and actual delay (in minutes) for each operated train during the recorded period. Canceled trains are not included in the realization data. We will mainly rely on the realization data to train our prediction model.

Additional information, including crew schedules, rolling stock circulation, infrastructure data, and weather conditions, are also included in the dataset. Our lightweight model uses only past delay data to predict near-term delays and does not rely on these additional features. Thus our model is robust and can be deployed in many other systems when these additional features are not recorded in history.

We provide detailed statistical information on the historical delay information in Table 1, and the histogram of the delay minutes for all the trains in Fig. 2. From Table 1 and Fig. 2 we can see that most of the delay observations from the historical data are concentrated around 0 min. The trains with large absolute delay minutes are less observed. This will make the transition matrices for our Markov model sparse, as we will discuss in Sections 4 and 5. Moreover, earliness (negative delay) is also observed from the historical data. Since our model aims to predict the actual delay minutes, our model will predict the negative delay as well.

4. Markov property test

Before building our Markov chain prediction model, we first answer the questions of whether the delay evolution over stations follows a Markov chain and whether the Markov chain has order one. Answering these questions will help us understand how the delay at the earlier stations influences that at later ones. Moreover, it will provide theoretical support in explaining why the Markov chain model that we propose works. Therefore, in this section, we develop a chi-square

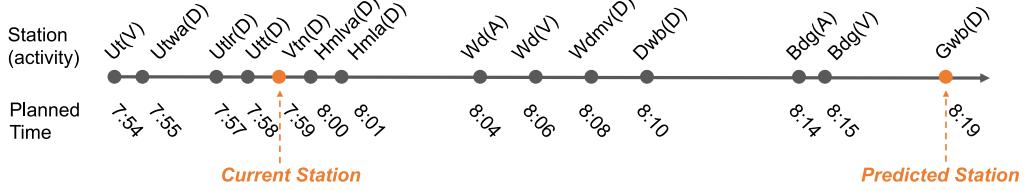


Fig. 1. A demonstrative graph for delay prediction with train number “8820”. Suppose we are at station “Vtn”. We aim to predict the delay at the station that the train is scheduled to arrive 20 min later, which is the station “Gwb”. The delay minutes at and before the current stations are available.

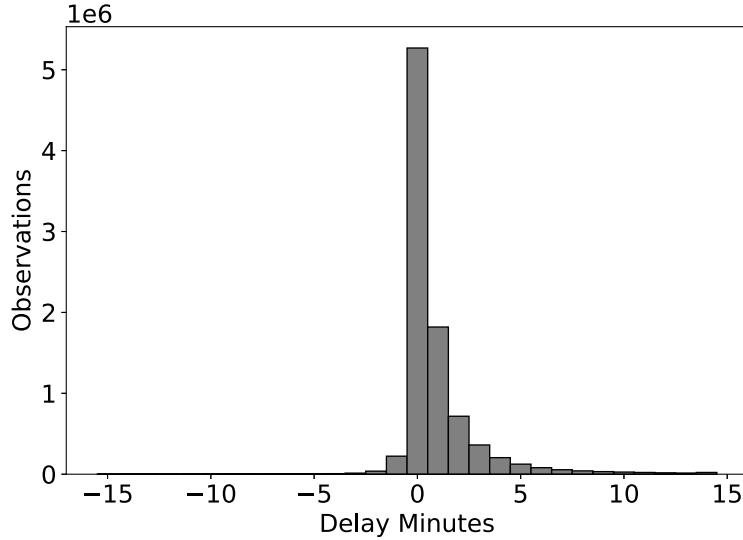


Fig. 2. Delay minutes distribution for all trains from the historical data. Most of the delay observations are near 0. Large absolute delay minutes (e.g., ≥ 10) are less observed. Earliness (negative delay) is also observed from the historical data.

Table 1
Statistical information of the historical data. Large absolute delays are unlikely to observe.

Number of trains	Number of station names	Number of delay records	Non-zero delay percentage	Absolute delay ≥ 2 Percentage	Absolute Delay ≥ 16 Percentage
9304	478	9156970	42.47%	20.16%	0.5853%
Delay Mean	Delay Median	Delay Mode	Delay Variance	Negative Delay (Earliness) Percentage	0.9 Quantile
0.9890	0.0	0.0	10.10	3.2765%	3.0

test to compute the order of the Markov chain. We first introduce the concept of Markov property in Section 4.1, and then discuss data sparsity led by the non-homogeneous modeling of the Markov chain in Section 4.2. We next present the chi-square test for sparse transition matrices in Section 4.3, and then use the Netherlands Railways data to show that the Markov chain has order one in Section 4.4.

4.1. Markov property

We first introduce the concept of Markov property. To facilitate our discussion, we focus on a particular train that travels through multiple stations. We denote the delay at station t by $D(t)$, where $D(t) \in \mathcal{N}$ is a bounded integer random variable and $\mathcal{N} = \{-N, -N + 1, \dots, -1, 0, 1, \dots, N - 1, N\}$ is the integer state space. We consider the integer state space since in the historical data, the delay values are integers (in minutes). Moreover, when announcing the delay information to passengers, usually integer delay values are provided. We assume the state space is symmetric with respect to 0 mainly for the convenience of analysis. The negative delay value means earliness in our paper.

We denote the originated station as station 1, and the destination station as station M . Assume $d(1), d(2), \dots, d(t)$ is a delay series (in

minutes) from station 1 to station t ($t \leq M$). The delay evolution over stations has a k th order Markov property if it satisfies

$$\begin{aligned} & \mathbf{P}(D(t) = d(t) | D(t-j) = d(t-j), \text{ for } j = 1, \dots, t-1) \\ &= \mathbf{P}(D(t) = d(t) | D(t-j) = d(t-j), \text{ for } j = 1, \dots, k), \end{aligned} \quad (1)$$

where $k = 1, \dots, t-1$. In particular, the delays have a zero-order Markov property if

$$\mathbf{P}(D(t) = d(t) | D(t-j) = d(t-j), \text{ for } j = 1, \dots, t-1) = \mathbf{P}(D(t) = d(t)), \quad (2)$$

When the zero-order Markov property holds, $D(t)$ is a random variable independent of the random variables $D(t-1), \dots, D(1)$, which means that the current delay is independent of the historical delays. We aim to use a chi-square test to verify that the delay evolution over stations follows a first-order Markov chain, i.e., $k = 1$.

4.2. Non-homogeneity

The Markov property tests introduced in [Bickenbach and Bode \(2003\)](#), [Tan and Yilmaz \(2002\)](#) are for homogeneous Markov chains, i.e., the transition probabilities $\mathbf{P}(D(t) = d(t) | D(t-j) = d(t-j), \text{ for } j =$

$0, \dots, k$) do not vary for different stations t . We relax this assumption in our work by modeling the delay evolution as non-homogeneous Markov chains, i.e., the transition probabilities $P(D(t) = d(t)|D(t-j) = d(t-j), \dots, k)$ are distinct over different stations t .

There are multiple reasons for the delay being non-homogeneous over stations. First, a train's delay at a certain station could be influenced by other trains that travel through this station. For instance, at some stations, the train can be overtaken by a slower train when it is delayed, making the delay at the following stations more likely to accumulate (Lee, Yen, & Chou, 2016). Moreover, the traveling distances and railway conditions between stations could be distinct, making the delay distributions distinct as well. The third reason could be the change in the rolling stock composition at certain stations (INFORMS Railway Applications Section, 2018), which may result in their delay distributions being different from those without rolling stock change.

A challenge in testing the Markov property of non-homogeneous Markov chains is the data sparsity: since each train at a particular station has a unique transition probability distribution, only the historical delays for this train at this station can be used to fit the distribution. The lack of historical data may result in very sparse transition matrices for $P(D(t) = d(t)|D(t-j) = d(t-j), \dots, k)$. To overcome this challenge, we propose a Markov property testing approach for the non-homogeneous Markov chain, which we will show next.

4.3. Testing procedure

We now introduce the procedure of the Markov property test that aims to determine the order of the Markov chain. The general idea is to test the Markov property from order zero until a certain order of Markov property is accepted. Before introducing the testing procedure, we first define some necessary notations as follows:

- $n_j(t)$: the number of observations in the historical data that the train delays for j minutes at station t ;
- $n_{i,j}(t)$: the number of observations that the train delays for j minutes at station t , and delays for i minutes at station $t-1$;
- $n_{h,i,j}(t)$: the number of observations that the train delays for j minutes at station t , delays for i minutes at station $t-1$, and delays for h minutes at station $t-2$.

Using the defined terms above, we define the maximum likelihood estimate of the state transition probabilities that will be used in the chi-square test as follows:

- $\hat{p}_j(t) = \frac{n_j(t)}{\sum_i n_i(t)}$: the frequency that the train delays for j minutes at station t . The value $\hat{p}_j(t)$ is the *maximal likelihood estimate* (MLE) of the probability $p_j(t) = P(D(t) = j)$.
- $\hat{p}_{i,j}(t) = \frac{n_{i,j}(t)}{\sum_l n_{l,j}(t)}$: the frequency that the train delays for j minutes at station t , given that the train delays for i minutes at station $t-1$. The value $\hat{p}_{i,j}(t)$ is the MLE of the probability $p_{i,j}(t) = P(D(t) = j|D(t-1) = i)$.
- $\hat{p}_{h,i,j}(t) = \frac{n_{h,i,j}(t)}{\sum_l n_{l,i,j}(t)}$: the frequency that the train delays for j minutes at station t , given that the train delays for i minutes at station $t-1$ and h minutes at station $t-2$. The value $\hat{p}_{h,i,j}(t)$ is the MLE of the probability $p_{h,i,j}(t) = P(D(t) = j|D(t-1) = i, D(t-2) = h)$.

Having defined the necessary notations, we now describe the testing procedure. We first test the null hypothesis that the Markov chain has zero order for a specific station t , i.e., $H_0^{(0)} : \{\forall i, j : p_{i,j}(t) = p_j(t)\}$. Based on the $\hat{p}_{i,j}(t)$ and $\hat{p}_j(t)$ values, we calculate the zero-order likelihood ratio $LR^{(0)}(t)$ (Koch, 1988) and chi-square statistic $Q^{(0)}(t)$ (Pearson, 1900) for hypothesis tests as follows:

$$LR^{(0)}(t) = -2 \ln \frac{\prod_j (\hat{p}_j(t))^{n_j(t)}}{\prod_{i,j} (\hat{p}_{i,j}(t))^{n_{i,j}(t)}} = 2 \sum_{i,j : \hat{p}_{i,j}(t) \neq 0} n_{i,j}(t) \ln \frac{\hat{p}_{i,j}(t)}{\hat{p}_j(t)}, \quad (3)$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 3 & 5 & 0 \\ 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 6 & 3 \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 3 & 5 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 6 & 3 \end{pmatrix}$$

Fig. 3. An example of matrix truncation. The left matrix has a zero row and column, and the right one is the truncated matrix after removing the zero row and column. Markov transition matrices with zero rows and columns are common in the train delay data set (see the matrix in Fig. 4(a) in Section 6 for an example).

$$Q^{(0)}(t) = \sum_{i,j : \hat{p}_{i,j}(t) \neq 0} n_{i,j}(t-1) \frac{(\hat{p}_{i,j}(t) - \hat{p}_j(t))^2}{\hat{p}_j(t)}. \quad (4)$$

The likelihood ratio and chi-square statistic are compared against a chi-square threshold value χ_{a_0, df_0}^2 , where a_1 is the significance level and df_0 is the degree of freedom (see Walpole et al., 1993). The degree of freedom will rely on the non-zero empirical probabilities calculated from the historical data, which is given in the following lemma. Specifically, we let $\mathcal{A}(t) = \{j : n_j(t) > 0\}$ be the index set of the delay minutes observed from the historical data.

Lemma 1. Under the null hypothesis $H_0^{(0)}$, both $LR^{(0)}(t)$ and $Q^{(0)}(t)$ follow an asymptotic chi-square distribution with a degree of freedom

$$df_0 = (|\mathcal{A}(t-1)| - 1)(|\mathcal{A}(t)| - 1). \quad (5)$$

Proof. We leave the detailed proof to Appendix A. \square

Remark 1. In the scenarios where we do not have enough training data, it is likely that $n_{i,j}(t)$ has rows and columns with all the elements being zero. These zero rows and columns do not provide additional information in the test, so they are removed when calculating the degree of freedom. The degree of freedom given in Lemma 1 is thus the one we compute based on the truncated matrix $n_{i,j}(t)$ after removing all the zero rows and columns. Fig. 3 provides a demonstrative graph of the truncated matrix.

After obtaining $LR^{(0)}(t)$ and $Q^{(0)}(t)$, we compare them with the chi-square threshold value χ_{a_0, df_0}^2 . If $LR^{(0)}(t) \geq \chi_{a_0, df_0}^2$ or $Q^{(0)}(t) \geq \chi_{a_0, df_0}^2$, we then reject the hypothesis $H_0^{(1)}$, meaning that the Markov chain's order is at least one. If the zero-order hypothesis is rejected, we further need to test the hypothesis that the Markov chain has order one, i.e., $H_0^{(1)} : \{\forall h, i, j : p_{h,i,j}(t) = p_{i,j}(t)\}$. Similarly, we obtain the first-order likelihood ratio $LR^{(1)}(t)$ and chi-square statistic $Q^{(1)}(t)$ as follows:

$$LR^{(1)}(t) = 2 \sum_{h,i,j : \hat{p}_{h,i,j}(t) \neq 0} n_{h,i,j}(t) \ln \frac{\hat{p}_{h,i,j}(t)}{\hat{p}_{i,j}(t)}, \quad (6)$$

$$Q^{(1)}(t) = \sum_{h,i,j : \hat{p}_{h,i,j}(t) \neq 0} n_{h,i,j}(t-1) \frac{(\hat{p}_{h,i,j}(t) - \hat{p}_{i,j}(t))^2}{\hat{p}_{i,j}(t)}. \quad (7)$$

We compare these two values with the chi-square threshold value χ_{a_1, df_1}^2 , and the following lemma gives the degree of freedom df_1 .

Lemma 2. Under the null hypothesis $H_0^{(1)}$, both $LR^{(1)}(t)$ and $Q^{(1)}(t)$ follow an asymptotic chi-square distribution with a degree of freedom

$$df_1 = (|\mathcal{A}(t-2)| - 1)|\mathcal{A}(t-1)|(|\mathcal{A}(t)| - 1). \quad (8)$$

Proof. We leave the detailed proof to Appendix B. \square

If the first-order hypothesis $H_0^{(1)}$ is rejected, we can then test the null hypothesis that the Markov chain has an order of two. The test

Table 2

Markov property test. The likelihood ratio $LR^{(i)}$ and chi-square statistic $Q^{(i)}$ have $i = 0$ for testing $H_0^{(0)}$ and $i = 1$ for testing $H_0^{(1)}$, with the expressions given in Eqs. (3), (4), (6), and (7). The significance levels α_0 and α_1 are 0.01. Out of all the testing stations, most of them reject $H_0^{(0)}$ and accept $H_0^{(1)}$, which shows that the first-order Markov property is highly likely to hold.

Operating period	Total stations	Statistics	Stations reject $H_0^{(0)}$	Stations reject $H_0^{(1)}$
8:00–8:20	1632	$LR^{(i)}$	1468	2
		$Q^{(i)}$	1571	3
12:00–12:20	1861	$LR^{(i)}$	1630	1
		$Q^{(i)}$	1784	2

Algorithm 1 Markov Property Test

```

1: Obtain the number of observations  $n_j(t)$ ,  $n_{i,j}(t)$ , and  $n_{h,i,j}(t)$  from historical data
2: Obtain the frequencies  $\hat{p}_j(t) = \frac{n_j(t)}{\sum_l n_l(t)}$ ,  $\hat{p}_{i,j}(t) = \frac{n_{i,j}(t)}{\sum_l n_{l,j}(t)}$ , and  $\hat{p}_{h,i,j}(t) = \frac{n_{h,i,j}(t)}{\sum_l n_{h,l,j}(t)}$ .
3: Compute  $Q^{(0)}(t) = \sum_{i,j: \hat{p}_{i,j}(t) \neq 0} n_i(t-1) \frac{(\hat{p}_{i,j}(t) - \hat{p}_{i,j}(t))^2}{\hat{p}_{i,j}(t)}$ .
4: Let  $\alpha_0$  be the error probability (significance level) for the zero order hypothesis  $H_0^{(0)}$ 
5: if  $Q^{(0)}(t) < \chi^2_{\alpha_0, df0}$ , where degree of freedom  $df0 = (|\mathcal{A}(t-1)| - 1)(|\mathcal{A}(t)| - 1)$  then
6:   Do not reject  $H_0^{(0)}$ 
7: else
8:   Reject  $H_0^{(0)}$ 
9:   Compute  $Q^{(1)}(t) = \sum_{h,i,j: \hat{p}_{h,i,j}(t) \neq 0} n_{h,i}(t-1) \frac{(\hat{p}_{h,i,j}(t) - \hat{p}_{i,j}(t))^2}{\hat{p}_{i,j}(t)}$ .
10:  Let  $\alpha_1$  be the error probability for the first order hypothesis  $H_0^{(1)}$ 
11:  if  $Q^{(1)}(t) < \chi^2_{\alpha_1, df1}$ , where degree of freedom  $df1 = (|\mathcal{A}(t-2)| - 1)(|\mathcal{A}(t-1)|)(|\mathcal{A}(t)| - 1)$ . then
12:    Do not reject  $H_0^{(1)}$ 
13:  else
14:    Reject  $H_0^{(1)}$ 
15:  end if
16: end if

```

procedure will be similar, and the degree of freedom can be calculated following the same arguments in Lemmas 1 and 2. For conciseness, we do not provide the details for higher-order tests here since our numerical results in Section 4.4 show that the railway delay can be captured by a first-order Markov chain. Algorithm 1 summarizes the detailed procedure for the Markov property test, where we use the chi-square statistics $Q^{(i)}(t)$ as the testing statistics. One can also use $LR^{(i)}(t)$ as the testing statistics, since Lemmas 1 and 2 show that $Q^{(i)}(t)$ and $LR^{(i)}(t)$ follow the same asymptotic chi-square distribution. Algorithm 1 can also be applied to other practical problems to test the order of Markov chains when the data is sparse.

4.4. Numerical results

We conduct the numerical test to show that the delay evolution follows a first-order Markov chain, based on all the trains that are scheduled to operate during 8:00–8:20 and 12:00–12:20 in the historical data. In our numerical test, we let $N = 15$ since only a few delays (0.5853%) in the historical data are out of the range of \mathcal{N} with $N = 15$.

We calculate the total number of stations that all the trains travel through during these two periods and the number of stations that reject the zero-order and first-order Markov tests. We provide the test results for both likelihood ratio $LR^{(i)}$ and chi-square statistics $Q^{(i)}$ in Table 2. From Table 2 we find that both statistics reject the $H_0^{(0)}$ with a high frequency and reject $H_0^{(1)}$ with a low frequency. This result shows that the delay evolution over stations follows a first-order Markov chain.

5. The Markov chain prediction model

We now introduce how to use the Markov chain framework to predict delay distributions, and then discuss how to recover the transition matrices of the Markov chain from historical data.

5.1. The non-homogeneous Markov chain framework

Since we have shown that the delay evolution over stations follows a first-order Markov chain in Section 4, it is reasonable to utilize the first-order Markov chain model to predict the delay in the near future. We aim to predict the delay for trains at a future station, given the delay at the current station. Without loss of generality, we suppose that the train is currently located at station S with delay $d(S)$ minutes. We assume that from the timetable, the train is supposed to arrive at station T in the future. We aim to predict the delay $d(T)$ at station T .

We predict the delays $d(t)$ using a $1 \times (2N+1)$ dimension probability vector $\mathbf{v}(t)$, where the i th element of $\mathbf{v}(t)$ (denoted as $v_i(t)$) represents the probability that the train is delayed for i minutes at station t in our prediction model. We then have $\sum_{i=-N}^N v_i(t) = 1$ since $v_i(t) = P(D(t) = i)$. The delay at the current station $d(0)$ is a given number, as we already know the current delay. So we have $v_{d(S)}(S) = 1$ and $v_i(S) = 0$ for $i \neq d(S)$. The transition matrix from station $t-1$ to station t that incorporates all the transition probabilities is given by

$$P(t) = \begin{pmatrix} p_{-N,-N}(t) & p_{-N,-N+1}(t) & \dots & p_{-N,N}(t) \\ p_{-N+1,-N}(t) & p_{-N+1,-N+1}(t) & \dots & p_{-N+1,N}(t) \\ \dots \\ p_{N,-N}(t) & p_{N,-N+1}(t) & \dots & p_{N,N}(t) \end{pmatrix}. \quad (9)$$

The delay distribution $\mathbf{v}(S+1)$ at station $S+1$ is then given as $\mathbf{v}(S+1) = \mathbf{v}(S)P(S+1)$. By induction, we have the Chapman–Kolmogorov equation

$$\mathbf{v}(T) = \mathbf{v}(S) \prod_{t=S+1}^T P(t). \quad (10)$$

We then obtain the probability distribution $\mathbf{v}(T)$ for the delay at station T . We will discuss how we use this probability distribution to predict the delay and its trend in Section 6.2.

5.2. The Gaussian-Kernel method for transition matrix recovery

We now need to recover the transition matrix $P(t)$ from the historical data. As described in Section 4, the transition probability is recovered by the empirical probability $\hat{p}_{i,j}(t) = \frac{n_{i,j}(t)}{\sum_{l=1}^N n_{l,j}(t)}$. However, when the historical data are limited, it is possible that for some i , we have $\sum_{l=1}^N n_{i,l}(t) = n_i(t-1) = 0$. This scenario means that there is no observation from the historical data that the train is delayed for i minutes at station $t-1$. In that way, if the observed delay at station $t-1$ is i minutes and $n_i(t-1) = 0$, then it is impossible to predict the delay at station t since there is no corresponding historical record. In this subsection, we propose a matrix recovery method that utilizes the existing observations. The general idea of the proposed matrix recovery approach is to utilize the Gaussian kernel density estimate to recover the joint distribution matrix and then normalize this matrix into a transition matrix.

To recover the joint distribution matrix, we first consider the joint distribution for $D(t-1)$ and $D(t)$. We suppose $f_{D(t-1), D(t)}(x, y) : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ is a two-dimensional probability density function for $D(t)$ and $D(t-1)$ that satisfies $\iint_{\mathbb{R}^2} f_{D(t-1), D(t)}(x, y) dx dy = 1$. To estimate $f_{D(t-1), D(t)}(x, y)$, we suppose $(\mathbf{x}_1(t), \mathbf{x}_2(t), \dots, \mathbf{x}_m(t))$ are m observations for $(D(t-1), D(t))$ from the historical data with mean $\bar{\mathbf{x}}(t)$. We then estimate the probability density function using the Gaussian kernel density estimate provided in Silverman (2017) as follows:

$$\hat{f}_{t,h}(\mathbf{x}) = \frac{1}{mh^2 |\Sigma|^{\frac{1}{2}}} \sum_{i=1}^m \Phi\left(h^{-2}(\mathbf{x} - \mathbf{x}_i(t))^T \Sigma^{-1}(\mathbf{x} - \mathbf{x}_i(t))\right), \quad (11)$$

where $\Phi(u) = \frac{1}{2\pi} e^{-\frac{u}{2}}$, $\Sigma = \frac{1}{m-1} \sum_{i=1}^m (\mathbf{x}_i(t) - \bar{\mathbf{x}}(t))(\mathbf{x}_i(t) - \bar{\mathbf{x}}(t))^T$ is the covariance matrix, and h is a hyper-parameter called window width that determines the fitting smoothness. Since we are fitting a two-dimensional distribution, the optimal window width is chosen as $h = m^{-1/6}$ (Silverman, 2017).

However, if $D(t-1)$ and $D(t)$ are in perfect correlation (the covariance between two random variables equals 1), the covariance matrix Σ is singular. $D(t-1)$ and $D(t)$ are shown to be perfectly correlated when delay change is zero or constant. This phenomenon frequently occurs when the size of the historical data set is small. To resolve this issue, we may add an i.i.d. disturbance to the original data, e.g., we let $(\tilde{\mathbf{x}}_1(t), \tilde{\mathbf{x}}_2(t), \dots, \tilde{\mathbf{x}}_m(t)) = (\mathbf{x}_1(t) + \epsilon_1, \mathbf{x}_2(t) + \epsilon_2, \dots, \mathbf{x}_m(t) + \epsilon_m)$ to be the modified data, where ϵ_i 's are independent two-dimensional disturbances randomly chosen within $[-\epsilon, \epsilon] \times [-\epsilon, \epsilon]$ for a small ϵ . We can thus obtain a non-singular covariance matrix using the modified data.

We now recover the transition matrix using the fitted Gaussian kernel density estimate. Specifically, we denote $\tilde{P}(t)$ as the recovered transition matrix for station t , and each element $\tilde{p}_{i,j}(t)$ within $\tilde{P}(t)$ is given as follows:

$$\tilde{p}_{i,j}(t) = \frac{\hat{f}_{t,h}((i, j))}{\sum_{k=-N}^N \hat{f}_{t,h}((i, k))}. \quad (12)$$

We can then predict the delay distribution $\mathbf{v}(t)$ at station T by substituting $P(t)$ with $\tilde{P}(t)$ in Eq. (10). We summarize our prediction model in Algorithm 2. Specifically, from Step 3 to Step 13 of Algorithm 2, we train the multi-step transition matrix $\tilde{P} = \prod_{t=S+1}^T \tilde{P}(t)$. The training process is lightweight in the sense that it does not require a huge computation. Specifically, the complexity of data modification and kernel computation in Steps 5–9 of Algorithm 2 is $O(m)$. The complexity of the matrix recovery phase (Step 11 of Algorithm 2) is $O(N^2)$. So the complexity of recovering each matrix $\tilde{P}(t)$ is $O(mN^2)$. After training, the practitioners can just store the multi-step transition matrix \tilde{P} for future prediction.

6. Prediction results and discussions

This section presents the numerical tests and discussions for the proposed model. We first introduce the performance measures in Section 6.1, and then discuss the metrics for prediction in Section 6.2. In Section 6.3, we investigate different methods for recovering the transition matrix. We then compare the proposed Markov chain model with other time series models in Section 6.4.

6.1. Performance measures

We evaluate the performance of the prediction models by considering their capability to predict delays at the predicted station:

1. the train's delay trend (decrease, equal, or increase, compared with the current delay);
2. whether there is a delay jump (i.e., the predicted delay increases or decreases for more than two minutes, compared with the current delay);
3. the minutes of delay.

Algorithm 2 The Markov Chain Prediction Model with Gaussian Kernel

```

1: Initialization: Given historical data  $(\mathbf{x}_1(t), \mathbf{x}_2(t), \dots, \mathbf{x}_m(t))$  with  $t = S+1, \dots, T$  and delay  $d(S)$  at station  $S$ .
2: Let  $\mathbf{v}(S) = (0, \dots, 1, \dots, 0)$  and  $\tilde{P} = I$ .
3: for  $t$  from  $S+1$  to  $T$  do  $\triangleright$  Model training process that aims to obtain the multi-step transition matrix  $\tilde{P} = \prod_{t=S+1}^T \tilde{P}(t)$ 
4:   function  $\hat{f}_{t,h}(\mathbf{x})$ 
5:     Randomly generate  $m$  two-dimensional small perturbations  $(\epsilon_1, \epsilon_2, \dots, \epsilon_m)$  from the uniform distribution on  $[-\epsilon, \epsilon] \times [-\epsilon, \epsilon]$ .
6:     Modified data  $(\tilde{\mathbf{x}}_1(t), \tilde{\mathbf{x}}_2(t), \dots, \tilde{\mathbf{x}}_m(t)) = (\mathbf{x}_1(t) + \epsilon_1, \mathbf{x}_2(t) + \epsilon_2, \dots, \mathbf{x}_m(t) + \epsilon_m)$ .
7:     Data average  $\bar{\mathbf{x}}(t) = \frac{1}{m} \sum_{i=1}^m \tilde{\mathbf{x}}_i(t)$ 
8:     Let  $\Phi(u) = \frac{1}{2\pi} e^{-u/2}$ ,  $\Sigma = \frac{1}{m-1} \sum_{i=1}^m (\tilde{\mathbf{x}}_i(t) - \bar{\mathbf{x}}(t))(\tilde{\mathbf{x}}_i(t) - \bar{\mathbf{x}}(t))^T$ , and  $h = m^{-1/6}$ .
9:     return  $\hat{f}_{t,h}(\mathbf{x}) = \frac{1}{mh^2 |\Sigma|^{\frac{1}{2}}} \sum_{i=1}^m \Phi\left(h^{-2}(\mathbf{x} - \tilde{\mathbf{x}}_i(t))^T \Sigma^{-1}(\mathbf{x} - \tilde{\mathbf{x}}_i(t))\right)$ .
10:  end function
11:  Let  $\tilde{P}(t)$  be the transition matrix with  $\tilde{p}_{i,j}(t) = \frac{\hat{f}_{t,h}((i,j))}{\sum_{k=-N}^N \hat{f}_{t,h}((i,k))}$ .
12:  Let  $\tilde{P} = \tilde{P} \tilde{P}(t)$ 
13: end for
14: Obtain the delay distribution  $\mathbf{v}(t)$  at station  $t$  by  $\mathbf{v}(T) = \mathbf{v}(S)\tilde{P}$ 
15: return  $\mathbf{v}(T)$ 
```

We evaluate our model on all the M trains that operate during a randomly selected time window. The model will forecast delay trends, delay jumps, and minutes of delay for each train (we will discuss in detail how these metrics are extracted from the delay distribution in Section 6.2). The model's forecasting scores are calculated based on the prediction results of all the selected trains, as we shall show in the following.

Score for delay trend prediction

We now introduce the delay trend prediction score, and we first discuss the prediction score for delay increase. We define TP_I as the number of true positive predictions for delay increase, i.e., the number of trains whose delay is predicted to increase and the delay actually increases. For those trains whose delay is predicted to increase but actually does not increase, we denoted them as FP_I (false positive). Similarly, for a train whose delay is not predicted to increase, we classify it into FN_I (false negative) if the actual delay increases, and TN_I (true negative) if the actual delay does not increase. The predictions in TP_I and TN_I are correct ones. We next evaluate the model's prediction performance in increasing trend by considering the F1 score defined as

$$F_I = \frac{2TP_I}{2TP_I + FP_I + FN_I}. \quad (13)$$

The F1 score is ranged from 0 and 1, and a high F1 score indicates high classification accuracy (Tharwat, 2021).

Similarly, we can define the model's F1 score for predicting delay decreasing F_D and predicting delay remaining equal F_E . We thus use the F1 score

$$F_{TR} = \frac{1}{3} (F_I + F_D + F_E) \quad (14)$$

as the metric to measure the model's performance in predicting delay trends.

Score for delay jump prediction

Like the F1 score defined for delay trend prediction, we use the F1 score to evaluate the prediction score for delay jump.

Table 3
Statistical information of the testing data.

Testing data	Number of trains	Delay mean	Delay median	Delay mode	Delay variance	Minimum delay	Maximum delay
Test Set 1	174	1.8218	1.0	0.0	8.3091	-1.0	21.0
Test Set 2	222	0.3455	0.0	0.0	0.9395	-1.0	8.0
Test Set 3	561	0.7986	0.0	0.0	1.8969	-1.0	13.0

Score for predicting the minutes of delay

We evaluate the prediction accuracy for minutes of delay using the root weighted mean square error (RWMSE), denoted as Θ . Denote \hat{d}_i as the predicted delay and d_i as the actual (realized) delay for train i . We let the weight be 0.2 when the absolute delay is 0 or 1 min, and the weight for all the other delays be 0.8. Therefore, the RWMSE (Θ) has a greater weight on large delays, which penalizes more if the large delays are not accurately predicted. We then define the root weighted mean square error as

$$\Theta = \sqrt{\sum_{i=1}^M \left(w_1 \mathbf{1}_{|d_i| \leq 1} + w_2 \mathbf{1}_{|d_i| > 1} \right) |\hat{d}_i - d_i|}, \quad (15)$$

where $w_1 = \frac{0.2}{\sum_{i=1}^M \mathbf{1}_{|d_i| \leq 1}}$ and $w_2 = \frac{0.8}{\sum_{i=1}^M \mathbf{1}_{|d_i| > 1}}$. It is easy to verify that

$$\sum_{i=1}^M (w_1 \mathbf{1}_{|d_i| \leq 1} + w_2 \mathbf{1}_{|d_i| > 1}) = \sum_{i=1}^M \left(\frac{0.2}{\sum_{j=1}^M \mathbf{1}_{|d_j| \leq 1}} \mathbf{1}_{|d_i| \leq 1} + \frac{0.8}{\sum_{j=1}^M \mathbf{1}_{|d_j| > 1}} \mathbf{1}_{|d_i| > 1} \right) = 1.$$

So that Eq. (15) is a valid RWMSE.

Total prediction score

We use the total prediction score provided in [INFORMS Railway Applications Section \(2018\)](#) to evaluate the model's general prediction performance. The total prediction score is a linear combination of F_{TR} , F_{JP} , and Θ , which is given as

$$\text{Score} = 10F_{JP} + 5F_{TR} - \Theta. \quad (16)$$

The total prediction score in Eq. (16) values the delay jump prediction more than the trend prediction and RWMSE. The reason is that in reality, being unable to predict the drastic delay change may cause severe negative effects on both the railway system schedulers and passengers. Note that the performance of our proposed model is insensitive to the weights provided in Eq. (16). As we will show later, our proposed model outperforms other models in each of the F_{TR} , F_{JP} , and Θ scores.

Training and testing data

We evaluate the model performance based on three data sets from the historical data. *Test Set 1* contains 174 trains operated during 8:00–8:20, on November 7, 2017. We use the delay of the train at the predicted station (the one closest to 8:20 from the timetable) as the actual delay in the testing data. *Test Set 2* contains 222 trains operated during 12:00–12:20, on November 9, 2017. *Test Set 3* has 561 trains operated during 8:00–8:20, 12:00–12:20, and 16:00–16:20, on December 8, 2017. [Table 3](#) provides the statistical information of these three testing sets, from which we can find that the delay in *Test Set 1* is more divergent than those in *Test Set 2* and *Test Set 3*.

All of our models are trained based on the historical data that we described in Section 3, excluding the data on the testing date. Since our model is built upon the first-order Markov chain, the input of our prediction model is the delay minute at the current station (the one that the train is scheduled to arrive at right before 8:00, 12:00, or 16:00) in each testing case. In all of our following experiments for the Markov chain, we let $N = 15$, so the state space has 31 values and each transition matrix has 961 elements.

6.2. Delay prediction metrics

[Algorithm 2](#) introduced in Section 5 returns a prediction of distribution $v(T)$ for the delay value $D(T)$. The vector $v(T)$ now represents a conditional delay probability given the delay $D(S)$ at the current station. We then compare the approaches to obtain the predicted delay trend, delay jump, and minutes of delay from the distribution $v(T)$.

We first define the mean, mode, and median for $D(T)$ as follows:

- Mean: $\text{Mean}(D(T)|D(S) = d(S)) = \sum_{i=-N}^N v_i(T) \cdot i$.
- Mode: $\text{Mode}(D(T)|D(S) = d(S)) = \arg \max_i \{v_i(T)\}$.
- Median: $\text{Median}(D(T)|D(S) = d(S)) = \min\{i : \sum_{j=-N}^i v_j(T) \geq \frac{1}{2}\}$.

We also define the probability of delay increasing, decreasing, remaining equal, and delay jump as follows:

- Probability of delay increasing: $P(D(T) > d(S)|D(S) = d(S)) = \sum_{j=d(S)+1}^N v_j(T)$.
- Probability of delay decreasing: $P(D(T) < d(S)|D(S) = d(S)) = \sum_{j=-N}^{d(S)-1} v_j(T)$.
- Probability of delay remaining equal: $P(D(T) = d(S)|D(S) = d(S)) = v_{d(S)}(T)$.

Moreover, we define the probability of delay jump as

$$\bullet P(|D(T) - d(S)| \geq 2|D(S) = d(S)) = 1 - \sum_{i=d(S)-1}^{d(S)+1} v_i(t).$$

We will test the above metrics on the test data and select the best ones to predict the delay trend, delay jump, and minutes of delay.

Delay trend prediction

We now compare four approaches to predict the delay trend based on the mean value, mode, median, and probabilities. When using the mean value to predict the delay trend, we will do the following:

- Return “increase” if $\text{Mean}(D(T)) - d(S) \geq 1$; Return “decrease” if $\text{Mean}(D(T)) - d(S) \leq -1$; Return “equal” otherwise.

The way of using the mode and median to predict the delay trend is similar to that of using the mean. When using the increasing/decreasing/equal probability to predict the delay trend, we do the following:

- Return “increase” if $\sum_{i=d(S)+1}^N v_i(T) > \max\{\sum_{i=-N}^{d(S)-1} v_i(T), v_{d(S)}(T)\}$; Return “decrease” if $\sum_{i=-N}^{d(S)-1} v_i(T) > \max\{\sum_{i=d(S)+1}^N v_i(T), v_{d(S)}(T)\}$; Return “equal” otherwise.

Delay jump prediction

Similar to the delay trend prediction, we compare four approaches to predict whether there is a delay jump based on the mean, mode, median, and probabilities. When using the mean to predict delay jump, we will

- Return “yes” if $|\text{Mean}(D(T)) - d(S)| \geq 2$; Return “no” otherwise.

We will use the same criterion when using the mode and median to predict delay jump. When using the jump probability to predict the delay jump, we will

- Return “yes” when the probability of delay jump $1 - \sum_{i=d(S)-1}^{d(S)+1} v_i(t) \geq 0.5$; Return “no” otherwise.

Table 4

Metrics comparison with the best score marked **bold**. The mean value achieves the smallest θ , and the jump probability provides the highest score for delay jump prediction.

Performance measure	Test data	Mean	Mode	Median	Probability
F_{TR}	Test Set 1	0.56934	0.54502	0.57231	0.58312
	Test Set 2	0.60353	0.73044	0.69949	0.66006
	Test Set 3	0.64599	0.68654	0.70984	0.65818
F_{JP}	Test Set 1	0.48387	0.38596	0.45902	0.56716
	Test Set 2	0.48276	0.59259	0.51852	0.62500
	Test Set 3	0.45517	0.49296	0.489201	0.51701
θ	Test Set 1	2.88631	3.17746	2.96964	N/A
	Test Set 2	2.58319	2.81596	2.74579	N/A
	Test Set 3	1.66623	1.89088	1.75550	N/A

Metrics comparison

For the delay minutes prediction, we compare the performance of using mean, mode, and median as the predictor.

We present the performance score of each metric in Table 4. For delay trend prediction, we find that using the mean, mode, median, and probability in prediction results in similar scores of F_{TR} . Using the mode in prediction results in the lowest F_{TR} score on *Test Set 1*, and the highest score on *Test Set 2*. Using delay increasing/decreasing/equal probability to predict delay trend has the highest score for *Test Set 1*, but its performance on *Test Set 2* ranks only third on *Test Set 2* and *Test Set 3*. We thus choose the median to predict the delay trend, as its performance ranks first on *Test Set 3* and second on the other sets.

We choose the jump probability to predict the delay jump, as its F_{JP} scores for all testing sets are much higher than the other metrics. Using the probability to predict the delay jump is also more reasonable than using the other metrics since both large delay increases and decreases are regarded as delay jumps. An example is when the conditional delay distribution $p_{i,j}(t)$ is Gaussian with a large tail, both large delay decrease and increase have a high probability, but the mean/mode/median value returns a delay minute in the middle, indicating no delay jump.

We choose the mean value to predict the minutes of delay since the mean value achieves the lowest RWMSE for all testing sets. The reason is that the mean value can better characterize the concentration of the delay distribution, and the data with extremely large or small delays are rarely observed in history.

6.3. Comparison of matrix recovery methods

In Section 5.2, we have proposed a Gaussian kernel method to recover the zero elements due to a lack of delay observations. We now compare different benchmark matrix recovery methods with the Gaussian kernel approach.

The historical data lead to rather sparse transition matrices. An example is the trains in *Test Set 2*. There are 222 trains operating in this test set, and these trains travel through 10743 stations in total. We let $N = 15$, so each matrix has 31 rows. Out of the 10743 transition matrices, there are 254062 empty rows, with each matrix having nearly 23.6491 empty rows. Since each matrix has 31 rows, nearly 76.2873% of rows are empty on average. Thus in our case, matrix recovery is necessary.

Diagonal filling

Under the diagonal filling approach, we recover the transition matrix $\tilde{P}(t)$ in the following way.

- If $\sum_{l=-N}^N n_{i,l}(t) \neq 0$, let $\tilde{p}_{i,j}(t) = \frac{n_{i,j}(t)}{\sum_{l=-N}^N n_{i,l}(t)}$.
- If $\sum_{l=-N}^N n_{i,l}(t) = 0$, let $\tilde{p}_{i,i}(t) = 1$ and $\tilde{p}_{i,j}(t) = 0$ for $j \neq i$.

The idea of the diagonal filling approach is that we use the frequency from the historical data to recover the transition probability. If there is no observation that the train is delayed for i minutes at station $t - 1$, i.e., $\sum_{l=-N}^N n_{i,l}(t) = 0$, we assume that the delay at the station t is identical to the delay at the previous station $t - 1$.

Uniform filling

Under the uniform filling approach, we recover the transition matrix $\tilde{P}(t)$ in the following way.

- If $\sum_{l=-N}^N n_{i,l}(t) \neq 0$, we let $\tilde{p}_{i,j}(t) = \frac{n_{i,j}(t)}{\sum_{l=-N}^N n_{i,l}(t)}$.
- If $\sum_{l=-N}^N n_{i,l}(t) = 0$, we let $\tilde{p}_{i,j}(t) = \frac{1}{2N+1}$ for $j \in \{-N, \dots, N\}$.

The idea of uniform recovery is similar to that of the diagonal filling approach, as both of them will use the frequency to recover the transition probability. The difference is that when delay for i minutes at station $t - 1$ is not observed from historical data, under the uniform filling approach, we assume that the delay at station t is uniformly distributed within $\{-N, \dots, N\}$.

A Gaussian regression recovery approach

One phenomenon that we observe from the train delay data is that for each i , the conditional probability $p_{i,j}(t)$ is likely to be concentrated around $p_{i,i}(t)$, i.e., $p_{i,j}(t)$ is greater when j is close to i . This is because the delay jump between two stations is quite rare, and the delay minutes are likely to be similar between two stations. So we can assume that the distribution $p_{i,j}(t)$ follows a Gaussian distribution for each i , and develop the Gaussian regression recovery approach as follows.

1. If $\sum_{l=-N}^N n_{i,l}(t) \neq 0$, we let $\tilde{p}_{i,j}(t) = \frac{n_{i,j}(t)}{\sum_{l=-N}^N n_{i,l}(t)}$.
2. For i such that $\sum_{l=-N}^N n_{i,l}(t) \neq 0$, calculate the mean $\hat{\mu}_i = \frac{\sum_{l=-N}^N n_{i,l}(t) \cdot l}{\sum_{l=-N}^N n_{i,l}(t)}$ and standard deviation $\hat{\sigma}_i = \sqrt{\frac{\sum_{l=-N}^N n_{i,l}(t) (l - \hat{\mu}_i)^2}{\sum_{l=-N}^N n_{i,l}(t) - 1}}$ for the fitted Gaussian distribution.
3. Perform a linear regression based on the fitted mean $\hat{\mu}_i$ and standard deviation $\hat{\sigma}_i$. Obtain two regressed linear functions

$$\mu_i = \alpha + \beta \cdot i,$$

and

$$\sigma_i = \check{\alpha} + \check{\beta} \cdot i,$$

where μ_i and σ_i are the mean and standard deviation for the i th row of $\tilde{P}(t)$, and α , β , $\check{\alpha}$, and $\check{\beta}$ are fitted parameters.

4. For i such that $\sum_{l=-N}^N n_{i,l}(t) = 0$, let

$$\hat{\mu}_i = \alpha + \beta \cdot i,$$

and

$$\hat{\sigma}_i = \check{\alpha} + \check{\beta} \cdot i.$$

Then let $\tilde{p}_{i,j}(t) = \frac{g_{\hat{\mu}_i, \hat{\sigma}_i}(j)}{\sum_{l=-N}^N g_{\hat{\mu}_i, \hat{\sigma}_i}(l)}$, where $g_{\mu, \sigma}(x)$ is a one-dimension probability density function of the Gaussian distribution with mean μ and standard deviation σ .

Comparison and discussion

We now compare the matrix recovery approaches discussed above with the Gaussian kernel approach provided in Section 5.2. We present the Gaussian kernel approach results in Figs. 4, and 5 further provides the results for the matrix recovery methods mentioned above. Both Figs. 4 and 5 are based on the train with the number “519” upon arrival at station “Bl”.

Fig. 4(a) presents the original transition matrix that we obtained by simply letting $\hat{p}_j(t) = \frac{n_j(t)}{\sum_l n_l(t)}$ if $\sum_l n_l(t) \neq 0$. We find that many rows of the transition matrix are zeros because the negative delay in this station was not observed. Fig. 4(a) also shows that the recorded delays are likely to concentrate near the diagonal of the transition matrix. Only

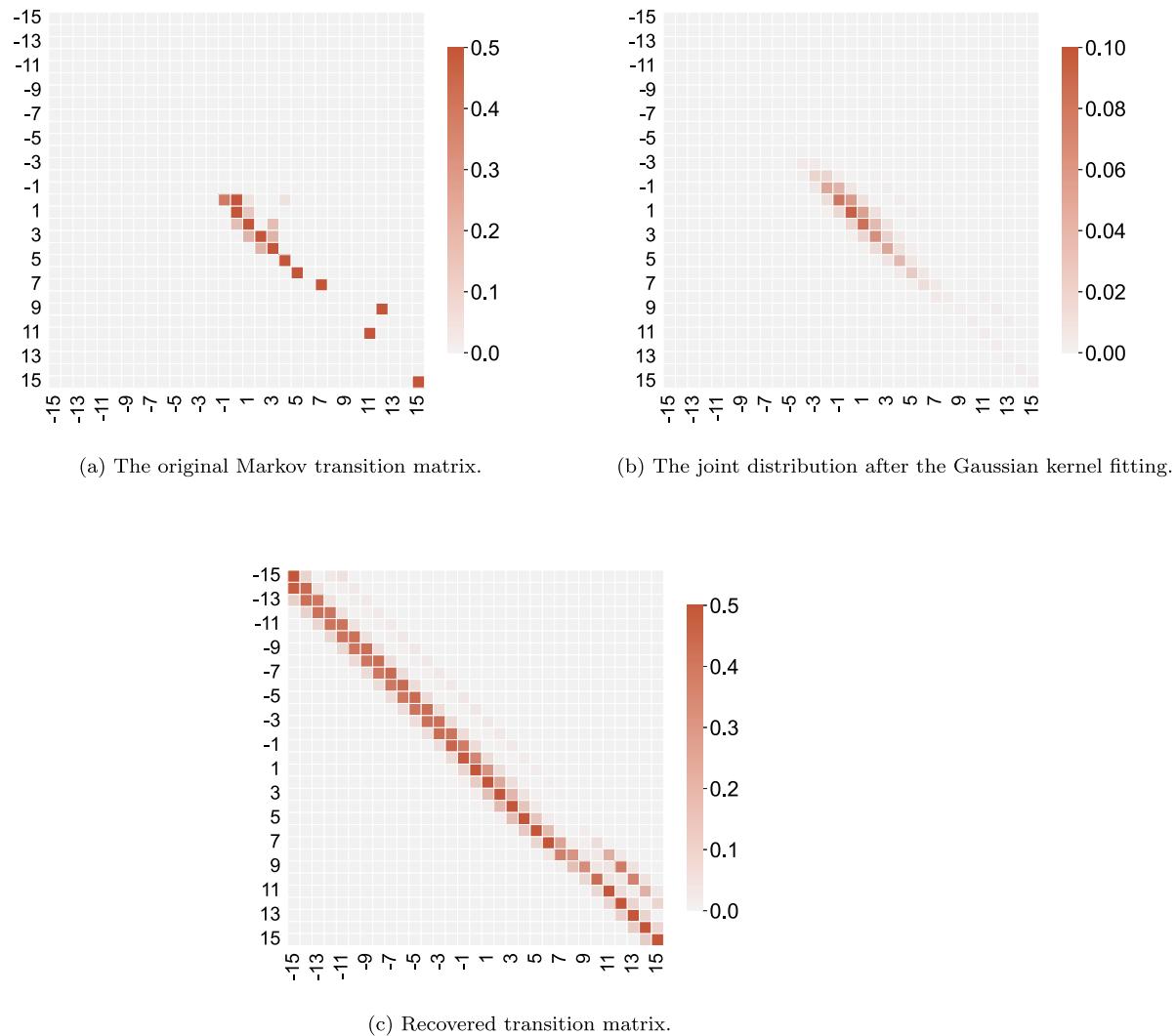


Fig. 4. An illustration of using the Gaussian kernel method to approach the Markov transition matrix (The train number is “519”, and the arrival station is “BL”). (a) The original Markov transition matrix obtained simply from historical delay data. (b) The joint distribution after applying Gaussian kernel fitting in Fig. (a). (c) Normalize the joint distribution in Fig. (b) and recover the new Markov transition matrix without zero rows.

in a few cases do we see that the delay has a jump. For instance, the delay jumped from 9 min to 12 min with a probability 1 in the historical record.

Fig. 4(b) plots the two-dimensional joint distribution density after the Gaussian kernel fitting. The delay density is also concentrated around the diagonal, showing that most historical delay values are small. Based on this joint distribution matrix, we further recover the transition matrix in Fig. 4(c) by normalizing each row in Fig. 4(b). Thus Fig. 4(c) is a valid transition matrix as the summation of all values within a row is 1. From Fig. 4(c), we can see that the Gaussian kernel method recovers most features of the recorded values in Fig. 4(a). For instance, the recovered transition matrix shows that the next delay is likely to stay unchanged when the current delay is small. Moreover, when the current delay is around 9 min, the recovered matrix indicates a high probability that the next delay will jump.

We plot the matrix by diagonal filling approach in Fig. 5(a), and the one by uniform filling approach in Fig. 5(b). These two approaches do not rely on the available data within the original transition matrix. We can expect their performance to degrade when the training data size becomes smaller.

We plot the matrix by the Gaussian regression recovery approach in Fig. 5(c). We find that this approach does not retain the features of the original matrix. Using linear regression to fit the parameters μ_i and

σ_i can result in a large σ on one side and a small one on the other, as we observe from Fig. 5(c). When the current delay i is small, the fitted σ_i is large. Thus we see that the distribution is more dispersed. When the current delay is large, we even have negative fitted σ_i values in the numerical study. For rows with negative fitted σ_i values, we let its diagonal be 1. So eventually, this approach leads to a skewed transition matrix, as we see in Fig. 5(c).

Table 5 provides the performance comparison for the matrix recovery approaches. Again, we select the median to predict the delay trend, the jump probability to predict the delay jump, and the mean to predict the minutes of delay for these matrix recovery approaches. We find from Table 5 that the Gaussian kernel method achieves the lowest Θ and the highest total score. Its F_{JP} score is also higher than that for other approaches on Test Set 1 and Test Set 2, and its F_{TR} score is close to the best score by other filling approaches. This show that in general, the Gaussian kernel recovery approach outperforms the other approaches in terms of prediction accuracy.

6.4. Comparison with other time series models

We now compare our proposed model with the following benchmark time series models.

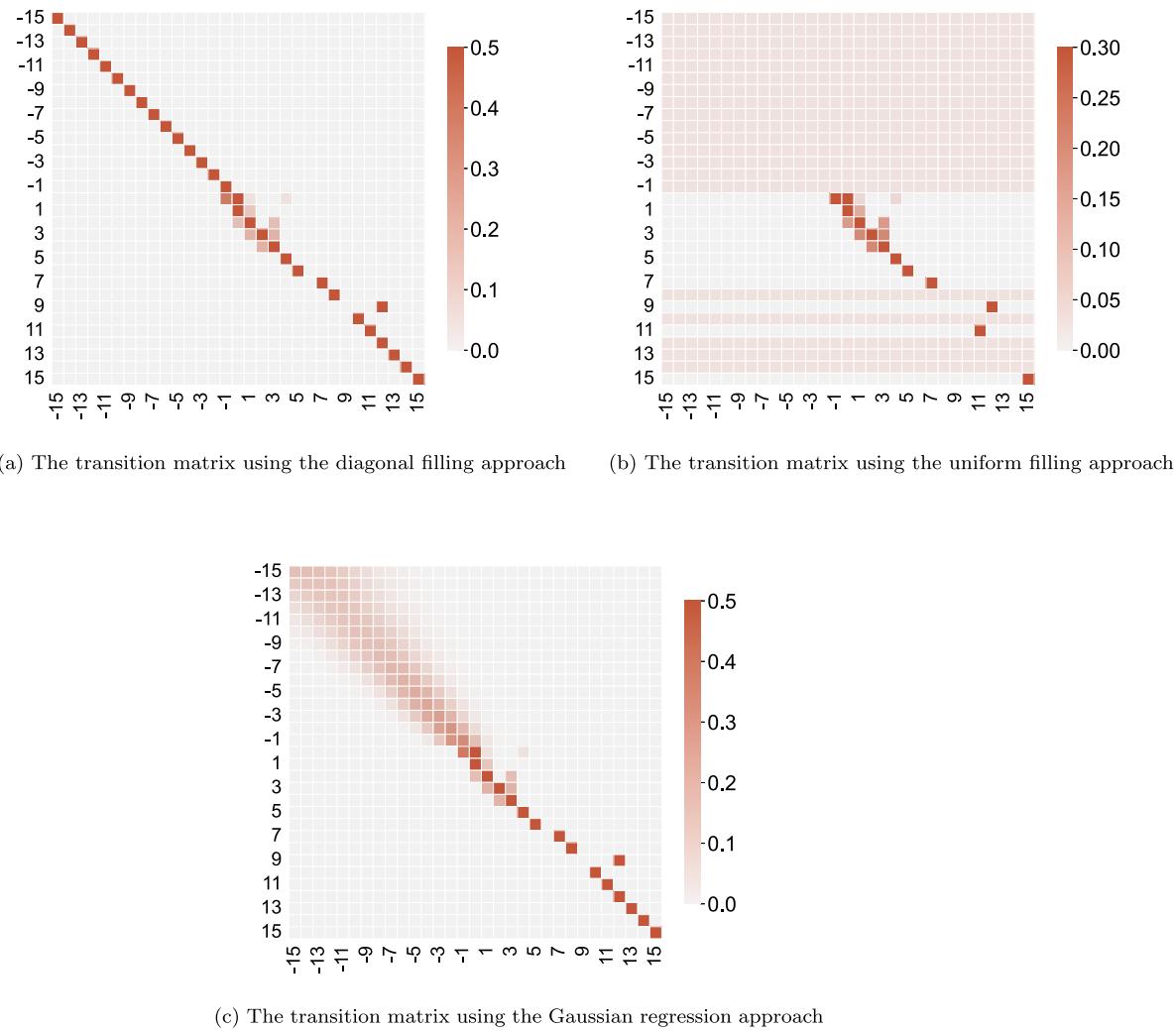


Fig. 5. Comparison of the transition matrices using different matrix recovery approaches. If the i th row is empty: (a) Diagonal filling: we fill 1 on the diagonal of the empty rows. (b) Uniform filling: we fill $\frac{1}{2N+1}$ on each element of the empty rows. (c) Gaussian regression: we fit the data using a Gaussian distribution and fill the empty rows based on the probability density of the Gaussian distribution.

Naive prediction approach

A naive forecasting approach is to assume the delay at station T is identical to the current delay, i.e., $d(T) = d(S)$. This approach does not rely on historical data, and is simple to implement in reality.

Probability distribution

We can predict the delay at station T by simply using its delay distribution without incorporating the current delay. Specifically, we compute $v(T) = (\frac{n_{-N}(T)}{\sum_{l=-N}^N n_l(T)}, \dots, \frac{n_N(T)}{\sum_{l=-N}^N n_l(T)})$ for station T . We then use the median to predict delay trend, use jump probability to predict delay jump, and delay expectation to predict the minutes of delay.

Integer-valued autoregressive process with signed binomial thinning

The autoregressive (AR) model is a time-series forecasting model that relies on past period values to predict the current one. Since the railway delays are in minutes, we implement an integer-valued autoregressive model with signed binomial thinning (INARS) to predict the delay. We use *Test Set 1* as an example to demonstrate how we develop the INARS model. For each train, we use the delays recorded before the current station on the testing date (i.e., 8:00, 12:00, or 16:00 with respect to each test set) as the training sets for the INARS model. Then we use the trained model to predict the delay at the predicted station. The maximum likelihood estimation is applied to estimate the parameters during the training process. A detailed description of how

Table 5

Performance comparison for matrix recovery approaches. Our proposed Gaussian kernel approach achieves the best Θ and the total prediction score, and relatively good scores for F_{TR} and F_{JP} .

Method	Test data	F_{TR}	F_{JP}	RWMSE	Θ	Total score
Diagonal Filling	Test Set 1	0.56947	0.48485	3.04482	4.65101	
	Test Set 2	0.71579	0.57143	2.86607	6.42719	
	Test Set 3	0.70815	0.53503	2.06683	6.82422	
Uniform Filling	Test Set 1	0.57084	0.46877	3.04449	4.49696	
	Test Set 2	0.71665	0.48485	2.86613	5.56580	
	Test Set 3	0.70145	0.51948	2.06327	6.63879	
Gaussian Regression	Test Set 1	0.58557	0.47889	5.99850	1.72371	
	Test Set 2	0.57463	0.38724	2.65238	4.09323	
	Test Set 3	0.63532	0.46429	1.97997	5.83950	
Gaussian Kernel	Test Set 1	0.57231	0.56716	2.88631	5.64684	
	Test Set 2	0.69949	0.62500	2.58319	7.16426	
	Test Set 3	0.70984	0.52349	1.66689	7.11717	

to use the INARS model to predict several time units into the future can be found in [Kim and Park \(2008\)](#).

Simple recurrent neural network model

Machine learning methods have been widely applied to solving time-series forecasting problems, and one of the best-known is the

Table 6

Time series model comparison. Our proposed model achieves the best score for F_{TR} and the total score, and a relatively decent score for F_{JP} and Θ .

Method	Test data	F_{TR}	F_{JP}	RWMSE Θ	Total score	Computation time per train (min)
Naive Forecasting	Test Set 1	0.18156	0.00476	2.86511	-1.95253	0.0170
	Test Set 2	0.24182	0.00111	2.45632	-1.23613	0.0206
	Test Set 3	0.21435	0.00000	1.89412	-0.82214	0.0507
Probability Distribution	Test Set 1	0.43879	0.53488	4.00108	3.54168	0.0152
	Test Set 2	0.64149	0.64849	2.58138	7.11094	0.0205
	Test Set 3	0.54174	0.45854	2.42547	4.86859	0.0509
INARS	Test Set 1	0.39343	0.48780	4.13695	2.70822	0.2148
	Test Set 2	0.60548	0.50000	3.20359	4.82382	0.1611
	Test Set 3	0.52720	0.43386	2.93270	4.04193	0.2539
RNN	Test Set 1	0.38971	0.48101	4.14002	2.61867	0.5478
	Test Set 2	0.62547	0.62069	2.50282	6.83139	0.4928
	Test Set 3	0.52345	0.40884	2.68192	4.02374	0.5451
Our Model	Test Set 1	0.57231	0.56716	2.88631	5.64684	0.0611
	Test Set 2	0.69949	0.62500	2.58319	7.16426	0.0661
	Test Set 3	0.70984	0.52349	1.66689	7.11717	0.0949

RNN model. A simple RNN model with one hidden layer is built and implemented to predict the delay. For each train, the training sets selected for RNN are similar to those for the INARS model mentioned above.

Model comparison

We compare the prediction scores of the time series model mentioned above in Table 6. Although the naive forecasting approach has relatively smaller Θ scores, its F_{TR} and F_{JP} scores are the worst since it does not utilize the historical data for training. Using probability distribution to predict delay can be promising when the delay is not varied. For instance, its performance for *Test Set 2* is close to the performance of our Markov chain model. However, for *Test Set 1* and *Test Set 3* whose delays are varied and unstable, the performance of using probability distribution is much worse than our model. The reason is that the probability distribution is only obtained from historical data. This approach does not utilize the delays at the past stations on the particular predicted date.

The INARS and RNN models only use the historical data on the predicted date for training without using the other historical data. When the predicted train's current station S is not far from its starting station 1, the training set could be small so that not many patterns can be learned from history. We can find from Table 6 that the INARS and RNN models perform worse than the method of using probability distribution due to its training set being small.

The Markov chain model we proposed in this work has a better overall performance than the other models described above. The reason is that the Markov chain model is trained based on historical data and also takes the delay at the current station as the input. Our model has the largest score in F_{TR} . Its F_{JP} score is the best for *Test Set 1* and *Test Set 3* where the delays are divergent. Its Θ scores are also small compared with other approaches.

Table 6 also provides the computation time for the benchmark models. The computation time includes the time for data reading, training, and predicting. Our experiment is coded in Python and runs on a Lenovo Thinkpad with an Intel i7-8550U core, 1.80 GHz CPU, 24 GB RAM, and Windows 10 operating system. Specifically, we use the “gaussian_kde” function from “scipy” package in Python to facilitate our computation of the Gaussian kernel. The Python package “statsmodels” is adopted to implement the INARS model. We use the “Sequential” model of “Tensorflow” in Python to build the RNN model. To obtain the prediction result for one train, our model takes less than 6 s for all the computations. This computation time is not significantly larger than that of the naive forecasting and probability distribution method (which require almost no training), and much smaller than those of

the INARS and RNN models. In fact, one can store the trained models (transition matrices) offline. Whenever a prediction is needed, one can just fetch the corresponding transition matrix, and apply it to the current delay to produce a forecast. This process can be finished in near real-time since the time complexity in searching the corresponding matrix in a list is just linear (Cormen et al., 2022).

7. Conclusion and future research

In conclusion, this study presents a novel, accurate, and efficient model for predicting railway delays using a Markov-chain-based framework. By rigorously investigating the theoretical properties of the model and conducting numerical experiments using the Netherlands Railways data, we demonstrate that the proposed model achieves higher prediction accuracy and efficiency compared to other benchmark time series prediction models. The major findings of this paper are summarized below:

- We propose a non-homogeneous Markov chain to characterize the delay process over stations. To test the order of the Markov chain, we propose and conduct a chi-square Markov property test with the Netherlands Railways data. The results show that the delays over stations of the same train follow a first-order Markov chain.
- We develop a Gaussian-kernel-based method to recover the transition matrices for the Markov chain model when the size of the training data is small. The Markov chain model equipped with the proposed recovery method achieves a higher prediction accuracy than being with other matrix recovery methods.
- We conduct numerical experiments on the real-world Netherlands Railways data and measure the prediction performance of each implemented model using a certain score calculated with the delay trend, the delay jump, and the root weighted mean square error. The proposed model provides a higher prediction score than other benchmark time series prediction models.

Our investigation of the Markov-chain-based delay prediction model leads to several potential research works. First, it is possible to consider interval state spaces (i.e., the delay falls into different intervals) using the Dempster-Shafer evidence theory (see He & Jiang, 2018) or model the uncertain probabilities using the fuzzy Markov chains (see Avrachenkov & Sanchez, 2002). Second, we aim to incorporate the forecasting model into the railway operation framework, and use forecasting results to make railway schedules. Third, we can migrate the proposed Markov property test, Markov chain model, and matrix recovery method to other application areas where forecasting is needed, such as bus delay forecasting (see Amberg, Amberg, & Kliewer, 2019), wind forecasting (see Carpinone et al., 2015; Verma et al., 2018), and air temperature forecasting (see Cifuentes et al., 2020).

CRediT authorship contribution statement

Jin Xu: Conceptualization, Methodology, Validation, Software, Writing – review & editing. **Weiqi Wang:** Conceptualization, Methodology, Software, Writing – review & editing. **Zheming Gao:** Conceptualization, Methodology, Software, Writing – review & editing. **Haochen Luo:** Data curation, Software. **Qian Wu:** Data curation, Software.

Data availability

The authors do not have permission to share data.

Acknowledgment

The authors are grateful to the editorial board and anonymous reviewers. Their comments and suggestions have significantly improved the content and presentation of this work.

Appendix A. Proof of Lemma 1

Proof. It has been proven in [Anderson and Goodman \(1957\)](#), [Bickenbach and Bode \(2003\)](#) that $LR^{(0)}(t)$ and $Q^{(0)}(t)$ follow the asymptotic chi-square distributions with the identical degree of freedom. We thus discuss the degree of freedom for the chi-square distribution focusing on $Q^{(0)}(t)$.

Using a similar argument in [Anderson and Goodman \(1957\)](#), we can show that $\sum_{j:\hat{p}_{i,j}(t)\neq 0} n_i(t-1) \frac{(\hat{p}_{i,j}(t)-\hat{p}_j(t))^2}{\hat{p}_j(t)}$ has an asymptotic chi-square distribution with a maximal degree of freedom $|\mathcal{A}(t)| - 1$. To further derive the degree of freedom of $Q^{(0)}(t)$, we only need to compute how many rows of i such that $n_i(t) \neq 0$. Here, we define the indicator function as

$$\mathbf{1}_A = \begin{cases} 1 & \text{if condition } A \text{ is true,} \\ 0 & \text{if condition } A \text{ is false.} \end{cases}$$

Since $|\mathcal{A}(t)| = \sum_{j=-N}^N \mathbf{1}_{n_j(t)\neq 0}$ and $n_j(t) = \sum_{i=-N}^N n_{i,j}(t)$, we have $|\mathcal{A}(t)| \geq \sum_{j=-N}^N \mathbf{1}_{n_{i,j}(t)\neq 0}$ for any $i \in \mathcal{N}$. We define $\mathcal{B}_i(t) = \{j : n_{i,j}(t) > 0\}$, and we compute the number of non-zero \mathcal{B}_i s as follows:

$$\begin{aligned} \sum_{i=-N}^N \mathbf{1}_{|\mathcal{B}_i(t)|\neq 0} &= \sum_{i=-N}^N \mathbf{1}_{\sum_{j=-N}^N n_{i,j}(t)\neq 0} \\ &= \sum_{i=-N}^N \mathbf{1}_{n_i(t-1)\neq 0} \\ &\leq |\mathcal{A}(t-1)|. \end{aligned}$$

Therefore, there is no more than $|\mathcal{A}(t-1)|$ number of \mathcal{B}_i . Moreover, since $\sum_{j=-N}^N p_j(t) = 1$, we should also subtract $(|\mathcal{A}(t)| - 1)$ number of degree of freedom from the summation. Therefore, the degree of freedom under $H_0^{(0)}$ is given by

$$|\mathcal{A}(t-1)|(|\mathcal{A}(t)| - 1) - (|\mathcal{A}(t)| - 1) = (|\mathcal{A}(t-1)| - 1)(|\mathcal{A}(t)| - 1).$$

Hence proved. \square

Appendix B. Proof of Lemma 2

Proof. We prove this lemma following a similar argument to the proof for [Lemma 1](#) by focusing on the degree of freedom for $Q^{(1)}(t)$. For each fixed h and i , we have $\sum_{j=-N}^N \mathbf{1}_{n_{h,i,j}(t)\neq 0} \leq |\mathcal{A}(t)|$, the maximum degree of freedom is then $|\mathcal{A}(t)| - 1$. We now consider the number of h and i such that $n_{h,i}(t-1) \neq 0$, then $n_{h,i,j}(t) \leq n_{h,i}(t-1)$. Summing over these h and i , we have

$$\sum_{h=-N}^N \sum_{i=-N}^N \mathbf{1}_{n_{h,i,j}(t)\neq 0} \leq \sum_{h=-N}^N \sum_{i=-N}^N \mathbf{1}_{n_{h,i}(t-1)\neq 0}$$

$$\begin{aligned} &\leq \sum_{h=-N}^N \mathbf{1}_{\sum_{i=-N}^N n_{h,i}(t-1)\neq 0} |\mathcal{A}(t-1)| \\ &\leq \sum_{h=-N}^N \mathbf{1}_{n_h(t-2)\neq 0} |\mathcal{A}(t-1)| \\ &\leq |\mathcal{A}(t-2)| |\mathcal{A}(t-1)|. \end{aligned}$$

Now the total degree of freedom is $|\mathcal{A}(t-2)| |\mathcal{A}(t-1)| (|\mathcal{A}(t)| - 1)$. We subtract it by the number of degree of freedom $|\mathcal{A}(t-1)| |\mathcal{A}(t)| - 1$ we lose by imposing $\sum_j p_{i,j}(t-1) = 1$. Therefore, the degree of freedom under H_0 is given by

$$\begin{aligned} &|\mathcal{A}(t-2)| |\mathcal{A}(t-1)| (|\mathcal{A}(t)| - 1) - |\mathcal{A}(t-1)| (|\mathcal{A}(t)| - 1) \\ &= (|\mathcal{A}(t-2)| - 1) |\mathcal{A}(t-1)| (|\mathcal{A}(t)| - 1). \end{aligned}$$

Another way to prove the lemma is to follow a similar argument in [Anderson and Goodman \(1957\)](#): We consider the statistic

$$\chi_i^2 = \sum_{h,j:\hat{p}_{h,i,j}(t)\neq 0} n_{h,i}(t-1) \frac{(\hat{p}_{h,i,j}(t) - \hat{p}_{i,j}(t))^2}{\hat{p}_{i,j}(t)}.$$

The statistic χ_i^2 has the degree of freedom $(|\mathcal{A}(t-2)| - 1) (|\mathcal{A}(t)| - 1)$ due to $\sum_j \hat{p}_{h,i,j}(t) = 1$ and $\sum_j \hat{p}_{i,j}(t) = 1$. Therefore, $Q^{(1)}(t) = \sum_{i:\sum_{h,j}\hat{p}_{h,i,j}(t)\neq 0} \chi_i^2$ has the degree of freedom $(|\mathcal{A}(t-2)| - 1) |\mathcal{A}(t-1)| (|\mathcal{A}(t)| - 1)$. Hence proved. \square

References

- Amberg, Bastian, Amberg, Boris, & Kliewer, Natalia (2019). Robust efficiency in urban public transportation: Minimizing delay propagation in cost-efficient bus and driver schedules. *Transportation Science*, 53(1), 89–112. <http://dx.doi.org/10.1287/trsc.2017.0757>.
- Anderson, Theodore W., & Goodman, Leo A. (1957). Statistical inference about Markov chains. *The Annals of Mathematical Statistics*, 28(1), 89–110, URL: <https://www.jstor.org/stable/2237025>.
- Avrachenkov, Konstantin E., & Sanchez, Elie (2002). Fuzzy markov chains and decision-making. *Fuzzy Optimization and Decision Making*, 1, 143–159. <http://dx.doi.org/10.1023/A:1015729400380>.
- Bickenbach, Frank, & Bode, Eckhardt (2003). Evaluating the, Markov property in studies of economic convergence. *International Regional Science Review*, 26(3), 363–392. <http://dx.doi.org/10.1177/0160017603253789>.
- Cacchiani, Valentina, et al. (2014). An overview of recovery models and algorithms for real-time railway rescheduling. *Transportation Research, Part B (Methodological)*, 63, 15–37. <http://dx.doi.org/10.1016/j.trb.2014.01.009>.
- Caimi, Gabrio, et al. (2012). A model predictive control approach for discrete-time rescheduling in complex central railway station areas. *Computers & Operations Research*, 39(11), 2578–2593. <http://dx.doi.org/10.1016/j.cor.2012.01.003>.
- Carpinone, A., et al. (2015). Markov chain modeling for very-short-term wind power forecasting. *Electric Power Systems Research*, 122, 152–158. <http://dx.doi.org/10.1016/j.epwr.2014.12.025>.
- Cifuentes, Jenny, et al. (2020). Air temperature forecasting using machine learning techniques: a review. *Energies*, 13(16), 4215, URL: <https://www.mdpi.com/1996-1073/13/16/4215>.
- Corman, Francesco, & Kecman, Pavle (2018). Stochastic prediction of train delays in real-time using, Bayesian networks. *Transportation Research Part C (Emerging Technologies)*, 95, 599–615. <http://dx.doi.org/10.1016/j.trc.2018.08.003>.
- Corman, Francesco, & Meng, Lingyun (2014). A review of online dynamic models and algorithms for railway traffic management. *IEEE Transactions on Intelligent Transportation Systems*, 16(3), 1274–1284. <http://dx.doi.org/10.1109/TITS.2014.2358392>.
- Cormen, Thomas H., et al. (2022). *Introduction to algorithms*. MIT Press: URL: <https://mitpress.mit.edu/9780262046305/introduction-to-algorithms/>.
- Flier, Holger, et al. (2009). Mining railway delay dependencies in large-scale real-world delay data. In *Robust and online large-scale optimization* (pp. 354–368). Springer, http://dx.doi.org/10.1007/978-3-642-05465-5_15.
- Freeland, R. Keith, & McCabe, Brendan P. M. (2004). Forecasting discrete valued low count time series. *International Journal of Forecasting*, 20(3), 427–434. [http://dx.doi.org/10.1016/S0169-2070\(03\)00014-1](http://dx.doi.org/10.1016/S0169-2070(03)00014-1).
- Ge, Meng, et al. (2021). ARIMA-FSVR hybrid method for high-speed railway passenger traffic forecasting. *Mathematical Problems in Engineering*, 2021(9961324), <http://dx.doi.org/10.1155/2021/9961324>.
- Goverde, Rob M. P. (2005). *Punctuality of railway operations and timetable stability analysis*. (Doctoral thesis), The Netherlands TRAIL Research School, URL: <http://resolver.tudelft.nl/uuid:a40ae4f1-1732-4bf3-bbf5-fdb8df635e7>.
- Harris, Nigela G., Mjøsund, Christiana S., & Haugland, Hans (2013). Improving railway performance in, Norway. *Journal of Rail Transport Planning & Management*, 3(4), 172–180. <http://dx.doi.org/10.1016/j.jrtpm.2014.02.002>.

- He, Zichang, & Jiang, Wen (2018). A new belief Markov chain model and its application in inventory prediction. *International Journal of Production Research*, 56(8), 2800–2817. <http://dx.doi.org/10.1080/00207543.2017.1405166>.
- Huang, Xin-Wei, & Emura, Takeshi (2021). Model diagnostic procedures for copula-based, Markov chain models for statistical process control. *Communications in Statistics. Simulation and Computation*, 50(8), 2345–2367. <http://dx.doi.org/10.1080/03610918.2019.1602647>.
- Huang, Ping, et al. (2020). A deep learning approach for multi-attribute data: A study of train delay prediction in railway systems. *Information Sciences*, [ISSN: 0020-0255] 516, 234–253. <http://dx.doi.org/10.1016/j.ins.2019.12.053>.
- INFORMS Railway Applications Section (2018). RAS problem solving competition: Train delay forecasting. URL: <https://connect.informs.org/railway-applications/new-item3/problem-solving-competition681>.
- Jin-Guan, Du, & Yuan, Li (1991). The integer-valued autoregressive (INAR(p)) model. *Journal of Time Series Analysis*, 12(2), 129–142, URL: <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9892.1991.tb00073.x>.
- Kim, Hee-Young, & Park, Younsung (2008). A non-stationary integer-valued autoregressive model. *Statistical Papers*, 49(3), 485–502, URL: <https://link.springer.com/article/10.1007/s00362-006-0028-1>.
- Koch, Karl-Rudolf (1988). Parameter estimation and hypothesis testing in linear models. (p. 306). <http://dx.doi.org/10.1007/978-3-662-03976-2>.
- Lee, Wei-Hsun, Yen, Li-Hsien, & Chou, Chien-Ming (2016). A delay root cause discovery and timetable adjustment model for enhancing the punctuality of railway services. *Transportation Research Part C (Emerging Technologies)*, [ISSN: 0968-090X] 73, 49–64. <http://dx.doi.org/10.1016/j.trc.2016.10.009>.
- Lessan, Javad, Fu, Liping, & Wen, Chao (2019). A hybrid, Bayesian network model for predicting delays in train operations. *Computers & Industrial Engineering*, 127, 1214–1222. <http://dx.doi.org/10.1016/j.cie.2018.03.017>.
- Li, ZhongCan, et al. (2021). Near-term train delay prediction in the Dutch railways network. *International Journal of Rail Transportation*, 9(6), 520–539. <http://dx.doi.org/10.1080/23248378.2020.1843194>.
- Ma, Tao, Antoniou, Constantinos, & Toledo, Tomer (2020). Hybrid machine learning algorithm and statistical time series model for network-wide traffic forecast. *Transportation Research Part C (Emerging Technologies)*, 111, 352–372. <http://dx.doi.org/10.1016/j.trc.2019.12.022>.
- Marković, Nikola, et al. (2015). Analyzing passenger train arrival delays with support vector regression. *Transportation Research Part C (Emerging Technologies)*, 56, 251–262. <http://dx.doi.org/10.1016/j.trc.2015.04.004>.
- Nabian, Mohammada Amin, Alemazkoor, Negin, & Meidani, Hadi (2019). Predicting near-term train schedule performance and delay using bi-level random forests. *Transportation Research Record*, 2673(5), 564–573. <http://dx.doi.org/10.1177/0361198119840339>.
- Olsson, Nilsa O. E., & Haugland, Hans (2004). Influencing factors on train punctuality—results from some Norwegian studies. *Transport Policy*, 11(4), 387–397. <http://dx.doi.org/10.1016/j.trapol.2004.07.001>.
- Oneto, Luca, et al. (2018). Train delay prediction systems: a big data analytics perspective. *Big Data Research*, 11, 54–64. <http://dx.doi.org/10.1016/j.bdr.2017.05.002>.
- Osipov, Vasiliy, et al. (2020). Urban traffic flows forecasting by recurrent neural networks with spiral structures of layers. *Neural Computing and Applications*, 32(18), 14885–14897.
- Patton, Andrew (2013). Chapter 16 - Copula methods for forecasting multivariate time series. In *Handbook of economic forecasting*, vol. 2 (pp. 899–960). Elsevier, <http://dx.doi.org/10.1016/B978-0-444-62731-5.00016-6>.
- Pavlyuk, Dmitry (2017). Short-term traffic forecasting using multivariate autoregressive models. *Procedia Engineering*, 178, 57–66. <http://dx.doi.org/10.1016/j.proeng.2017.01.062>.
- Pearson, Karl (1900). On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 50(302), 157–175. <http://dx.doi.org/10.1080/14786440009463897>.
- Railway Statistics 2015 Report (2015). International union of railways. URL: https://uic.org/IMG/pdf/synopsis_2015_print_5_.pdf.
- Silverman, Bernarda W. (2017). Density estimation for statistics and data analysis. <http://dx.doi.org/10.1201/9781315140919>.
- Smyl, Slawek (2020). A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting*, 36(1), 75–85.
- Sun, Li-Hsien, Lee, Chang-Shang, & Emura, Takeshi (2020). A, Bayesian inference for time series via copula-based, Markov chain models. *Communications in Statistics. Simulation and Computation*, 49(11), 2897–2913. <http://dx.doi.org/10.1080/03610918.2018.1529241>.
- Sun, Li-Hsien, et al. (2020). *Copula-based Markov models for time series: Parametric inference and process control*.
- Suwardo, W., Napiah, Madzlan, & Kamaruddin, Ibrahim (2010). ARIMA models for bus travel time prediction. *Journal of the Institute of Engineers Malaysia*, 49–58, URL: <https://www.semanticscholar.org/paper/ARIMA-MODELS-FOR-BUS-TRAVEL-TIME-PREDICTION-Suwardo-Madzlan/b2c16e5eb4efba7f8ca3f93eb1a13f317677>.
- Tan, Baris, & Yilmaz, Kamil (2002). Markov chain test for time dependence and homogeneity: an analytical and empirical evaluation. *European Journal of Operational Research*, 137(3), 524–543, URL: <https://ideas.repec.org/a/eee/ejores/v137y2002i3p524-543.html>.
- Tharwat, Alaa (2021). Classification assessment methods. *Applied Computing and Informatics*, 17(1), 168–192. <http://dx.doi.org/10.1016/j.aci.2018.08.003>.
- Verma, Samidhaa Mridul, et al. (2018). Markov models based short term forecasting of wind speed for estimating day-ahead wind power. In *2018 International conference on power, energy, control and transmission systems* (pp. 31–35). IEEE, <http://dx.doi.org/10.1109/ICPECTS.2018.8521645>.
- Walpole, Ronald E., et al. (1993). *Probability and statistics for engineers and scientists*, vol. 5. Macmillan New York.
- Yaghini, Masoud, Khoshrafter, Mohammad M., & Seyedabadi, Masoud (2013). Railway passenger train delay prediction via neural network model. *Journal of Advanced Transportation*, 47(3), 355–368. <http://dx.doi.org/10.1002/atr.193>.
- Zhou, Yajun, et al. (2018). A, Markov chain based demand prediction model for stations in bike sharing systems. *Mathematical Problems in Engineering*, 2018(8028714), <http://dx.doi.org/10.1155/2018/8028714/>.