IPSJ Transactions on Mathematical Modeling and Its Applications Vol.12 No.2 92–100 (July 2019)

Regular Paper

Fast Railway Delay Evaluation Method Based on Discrete Distribution Propagation

Tomoshi Otsuki[1,a)] Hideo Sakamoto[1] Hideki Kubo[2]

Received: January 31, 2019, Revised: March 22, 2019,
Accepted: March 31, 2019

Abstract: Accurate evaluation of delay on timetables is crucial for railway companies. There have been some conventional methods utilizing continuous random variables directly. These methods, however, suffer from combinatorial expansion problems, which require complex pruning techniques. In this paper, discretizing the delay distribution on railway networks, we present a method for calculating propagated delay distributions on each event analytically under the assumption of propagated delays' independence. We also show the complexity of the proposed method is $O(M \log MN)$ in the general case and can be reduced to $O(MN)$ for the special cases where the source distribution is the negative binominal distribution, where M denotes the number of quantization levels in discretization and N denotes the total number of events. Finally, computational experiments show that the proposed method is much faster than Monte Carlo simulations (MC), and provides almost the same results as MC when N is small.

Keywords: railway scheduling, train delays, delay propagation, robustness

1. Introduction

Accurate evaluation of delay on timetables is crucial for railway companies, since train delays cause dissatisfaction of passengers, reduction in revenue, and increase in the penalty payment for compensation when the delay exceeds some threshold time.

Thus, there are some buffer time for timetables to prepare for the delay. If we increase buffer time, the frequency of delays can be reduced. However, too much buffer time increases the service time, causing the dissatisfaction of passengers. Thus, a trade-off exists between punctuality and service levels of train operations.

To evaluate timetables, several indicators have been proposed. For example in Refs. [1] and [2], practical indicators of delays are summarized into the combination of (a) expected delay, (b) expected delay in excess of a set value, (c) variance of a delay, and (d) the probability of a delay at most or at least a set value. These indicators can be evaluated if we introduce random variables representing delays on each station in timetables.

Railway timetables consist of multiple sequences, corresponding to respective trains each of which consists of multiple scheduled events such as departures, arrivals, and passing of stations. Train delays occur between two events. For example, the delay while running occurs between the departure event at a station and the arrival event at the following station.

Train delays are categorized into two types. First, we define a *source delay* as a primary delay on each train typically due to technical failures or unexpected passenger's behaviors. Second, we define a *propagated delay* is a secondary delay which is caused by preceding events of a train itself or different trains when accumulating delays exceed some buffer time.

There are various causes of propagated delays. For example, if the delay time of a running train exceeds buffer time, the train arrives late at the next station. Moreover, if the delay of a preceding train exceeds headway buffer time, the delay is propagated to the following train. Actual delays are often caused by the combination of these factors.

In this paper, we propose the way of calculating the propagated delays on each event where source delays are given.

The remainder of this paper is organized as follows. Section 2 provides related work. Then Section 3 gives a problem setting and stochastic modelling of propagated delays. Section 4 gives an algorithm and its complexity. Section 5 reports comparative results of the proposed approach and the conventional Monte Carlo simulation-based approach. Finally, Section 6 provides a conclusion.

2. Related Work

To evaluate propagated delays, there have been mainly three approaches: deterministic approaches, Monte Carlo simulation-based approaches, and stochastic approaches.

First of all, deterministic approaches for timetable simulation have been well studied for a long time. PERT (Program Evaluation and Review Technique), which can find out critical paths of delays, is one of the methods which can be utilized in cases such as diagram simulation [3], rescheduling [4], and shunting scheduling [5]. However, since PERT deals with deterministic variables, it cannot calculate the distribution of delays.

Then Monte Carlo simulation-based approaches (MC) can evaluate propagated delays as well. For example, Ushida et al. [6]

[1] Toshiba Corporation, Kawasaki, Kanagawa 212–8582, Japan
[2] Toshiba Digital Solutions Corporation, Kawasaki, Kanagawa 212–8585, Japan
[a)] tomoshi1.otsuki@toshiba.co.jp

reports a method of evaluating robustness of timetables on a railway company. Then Tatsui et al. [7] proposes a method simulating timetables based on predicted number of passengers by neural network and Nakamura et al. [8] and Takeuchi et al. [9] evaluate the robustness of railway timetable based on statistics of passengers. However, though MC can evaluate timetables accurately if there are the adequate number of iterations, it requires much time to run many iterations. Moreover, for running detailed simulation, we need to collect many kinds of accurate information and to handle missing or error data, which usually takes a lot of time and effort. On the other hand, our proposed method can be executed only by easily-accessible operation records such as run and dwell time statistics or, more simply, by the average of run and dwell time.

Finally, stochastic approaches directly process cumulative distribution functions (CDF) of underlying random variables. To deal with the CDF of propagated delays on the network similar to ours, there have been some probability distribution classes which have the closeness under required operations.

For example, Buker et al. [2] and Kirchhoff et al. [10] propose the distribution classes that consist of the sum of extended exponential polynomials. However, since the number of terms grows exponentially when taking summation or determining the maximum, they require complex pruning techniques such as taking the first three moments of the target distribution function [2] and repeating a term-reduction-process to approximate the function with a few terms [10].

Then Measter et al. [1] proposes a phase-type-distribution based approximation for propagated delays. However, the size of the transition matrix grows exponentially when taking summation or determining the maximum. Thus, they proposed a method approximating the transition matrix with an upper triangular one. They show computational results when the number of nodes is as small as 24.

As another way to approximate distribution functions, Yuan et al. [11], [12] proposes a method of discretizing random variables in part. However, they use the discretization simply for the purpose of integration approximation, which is different from our approach.

3. Problem Setting

In this section, we summarize the delay modelling on a network, most of which is commonly used in various previous researches. See e.g., Ref. [10] for detailed description.

3.1 Network Model

Consider a directed network $G(V, A)$ comprising node set V and arc set A to represent delays on the timetable. The node set V represents events on a timetable and the arc set A represents the possibility of delay propagation between events on both side of the arc. For each node $v \in V$, *scheduled time* t_v^0 is set to represent the pre-determined time of the node v on the timetable.

For simplicity, we deal only three node types in this paper as shown in **Fig. 1** as follows:

- *departure node* representing a departure from one of the stations,
- *arrival node* representing an arrival at one of the stations, and
- *passing node* representing a passing through one of the stations.

Then a directed arc $a_{ij} \in A$ that connects nodes i and j, means the possibility of a delay of the node i can be propagated to the node j. Note that the direction of arcs is determined to satisfy $t_i^0 < t_j^0$.

Though source delays are caused by several factors including boarding passengers, turnarounds, platform tracks, or junctions, we consider only three typical arc types in this paper for simplicity:

- *run arc* representing a running phase of a train between a departure or passing node and the following passing or arrival node.
- *dwell arc* representing a dwell phase of a train between an arrival node and the following departure node.
- *headway arc* representing a delay propagation possibility between different trains, which connects a departure node of one train and the consecutive arrival node of the following train.

Run and dwell arcs are defined between self-train's nodes,

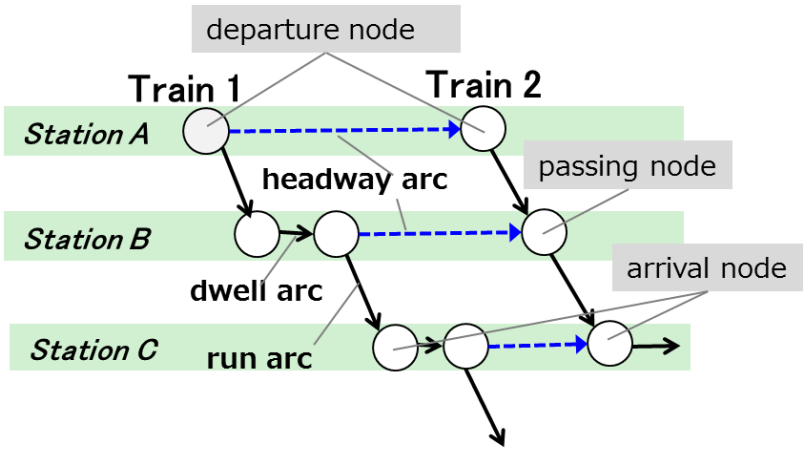

Fig. 1 Directed network representing scheduled events and propagated delays.

while headway arcs are defined between different train's nodes. When we define arcs as above, the number of incoming arcs is 2 or less, as in Fig. 1. Note that, even in the general cases where the number of incoming arcs to a node is 3 or over, the discussion below in this paper holds as well.

3.2 Modelling Delays on the Network

For a node pair i, j where $a_{i,j}$ is defined, we also define *minimum arc time* that represents the minimum required time of the arc $a_{i,j}$ and *buffer time* that represents the marginal time set to $a_{i,j}$.

Then the interval of scheduled times between t_i^0 and t_j^0 is decomposed into minimum arc time $h_{i,j}(> 0)$ and the residual buffer time $b_{i,j}(\geq 0)$ as follows:

$$t_j^0 = t_i^0 + h_{i,j} + b_{i,j}. \tag{1}$$

Then consider actual operations on the graph. Let $d_{i,j}$ be a random variable representing delay time on an arc $a_{i,j}$, which follows a distribution $P_{i,j}$, and let t_j be an *operation time* at which event j actually occurs.

We assume source delays are generated only on run and dwell arcs and no source delays are generated on headway arcs. Thus, on a headway arc, we define $P_{i,j}$ as taking all the probability mass on the delay-zero-point for simplicity.

Moreover, we assume:

(1) The delay of a node j is affected by the delay from multiple nodes $i(\in \mathbf{prev}(j))$, where $\mathbf{prev}(j)$ is a set of nodes immediately before node j. Then the delay of j takes maximum of propagated delays from multiple previous nodes.

(2) On any node j, operation time t_j is never earlier than scheduled time t_j^0, since departing earlier than scheduled time is banned in many railway companies.

Then the operation time t_j on node j is represented as follows:

$$t_j = \max\{t_j^0, \max_{i \in \mathbf{prev}(j)} \{t_i + h_{i,j} + d_{i,j}\}\}, \quad d_{i,j} \sim P_{i,j}. \tag{2}$$

Consider a random variable $X_j = t_j - t_j^0$ representing the delay from scheduled time on node j. Then from Eqs. (1) and (2), we have

$$X_j = [\max_{i \in \mathbf{prev}(j)} \{X_i + d_{i,j} - b_{i,j}\}]^+, \tag{3}$$

where $[x]^+ \equiv \max\{x, 0\}$.

When we evaluate each node in the topological order, we can calculate all the propagated delays recursively by Eq. (3).

In this paper, we consider the problem of calculating the probability distribution of propagated delay X_j on all nodes j given $P_{i,j}$ and $b_{i,j}$.

4. Delay Propagation Algorithm and Its Complexity

4.1 Discretization of Random Variables

In the discussion above, we have regarded propagated delays X as continuous random variables. However, handling continuous random variables analytically suffers from combinatorial expansion problems as shown in Section 2.

Thus, we discretize the random variable X, and deal the probability mass function (PMF) in the range from $k = 1$ to $k = M$. When we set sufficiently large M, the PMF is considered to approach the continuous probability distribution function (PDF).

Note that when the delay time takes negative values, we shift the domain of the distribution to $k \geq 1$ without loss of generality. In this case, let $k = k_0$ be the delay-zero-point in the original distribution.

4.2 Assumption

In our proposed method, we will make the following two assumptions:

S_Independence All the source delay distributions are independent.

P_Independence All the propagated delay distributions from different routes are independent.

The first assumption means the independence of all $P_{i,j}$s. This is natural assumption since small-scale delays usually occur independently.

Additionally, as in previous studies including [1], [2], [10], we assume the second assumption, which indicates the independence of all the preceding nodes' delays $X_i(i \in \mathbf{prev}(j))$ when we evaluate X_j in Eq. (3). This assumption can be violated in general since our network is grid-like and all the routes to a node originate from the initial node (e.g., the node at Station A on Train 1 in Fig. 1). We will discuss this issue again in Section 5.

Then in this section, all the random variables but source delay distribution D are discrete variables defined on $k = 1, 2, \ldots, M$. Note that only source delay distribution D takes a value at $k = 0$ so that the range is $k = 0, 1, \ldots, M$. Then all the deterministic values b and d take values on $k = 0, 1, \ldots, M$. Moreover, for simplicity, we denote $\Pr(X = k)$ by $X[k]$.

4.3 Calculation of Propagated Delays in Each Node

The process of calculating X_j in each node j from previous nodes' propagated delays $(\{X_i\}|i \in \mathbf{prev}(j))$ can be written in the following Algorithm 1, where $\mathrm{CONV}(X, D)$ is a convolution of X and D, $\mathrm{SHIFT}(Y, b)$ is a shift operation to Y by a constant value b, $\mathrm{GETMAX}_{i \in \mathbf{prev}(j)}(\{Z_i\})$ is the operation for determining maximum among Z_is, and $\mathrm{FLOOR}(W, k_0)$ is an operation flooring a W by a constant value k_0.

Algorithm 1 The procedure of getting propagated delays on node j

Require: j: current node id
Require: $D_{i,j}$: discrete distribution of source delays on $a_{i,j}$
Require: $b_{i,j}$: buffer time on $a_{i,j}$
Require: $\mathbf{prev}(j)$: node j's previous node set
Ensure: X_j: discrete distribution of propagated delays on j
1: **for all** $i \in \mathbf{prev}(j)$ **do**
2: $Y_i \leftarrow \mathrm{CONV}(X_i, D_{i,j})$
3: $Z_i \leftarrow \mathrm{SHIFT}(Y_i, b_{i,j})$
4: **end for**
5: $W_j \leftarrow \mathrm{GETMAX}_{i \in \mathbf{prev}(j)}\{Z_i\}$
6: $X_j \leftarrow \mathrm{FLOOR}(W_j, k_0)$

When we consider the fact that distribution of the sum of two independent random variables can be represented as convolution, Algorithm 1 strictly corresponds to $X_j = [\max_{i \in \mathbf{prev}(j)}\{X_i + d_{i,j} -$

$b_{i,j}\}]^+$ in Eq. (3).

Here, since X and D are independent under **S_Independence** assumption, the result PMF of $Y \leftarrow \mathrm{CONV}(X, D)$ is obtained as follows:

$$Y[k] = \sum_{i=1}^{k} X[i] \cdot D[k-i] \quad (k = 1, 2, \ldots, M), \tag{4}$$

where $X[k]$ and $D[k]$ are the inputs of the convolution. Note that while the complexity of convolution is $O(M^2)$ when calculating Eq. (4) as it is, it can be reduced to $O(M \log M)$ based on the FFT approach (detailed description is shown e.g., in Section 4.3.3. C on Ref. [13]).

Then the result PMF of $Z \leftarrow \mathrm{SHIFT}(Y, b)$ is

$$Z[k] = Y[\max\{1, k-b\}] \quad (k = 1, 2, \ldots, M), \tag{5}$$

where $Y[k]$ and $b(= 0, 1, 2, \ldots)$ are the inputs of the shift operation.

Moreover, we have the following PMF of $X \leftarrow \mathrm{FLOOR}(W, k_0)$:

$$X[k] = \begin{cases} W[k] & \text{if } k > k_0 \\ \sum_{i=0}^{k_0} W[i] & \text{if } k = k_0 \\ 0 & \text{if } k < k_0 \end{cases} \quad (k = 1, 2, \ldots, M), \tag{6}$$

where W and k_0 are the inputs of the flooring operation.

4.4 Algorithm for Determining the Maximum

Let L be the number of input variables and C_l be CDF of Z_l. Then since all the C_ls are independent under **P_Independence** assumption, the probability that all $\{C_l\}_{l=1}^{L}$ are k or less is represented as $\prod_{l=1}^{L} C_l[k]$. Let $W[k]$ represent the probability that the maximum of $\{C_l\}_{l=1}^{L}$ is just k. Then $W[k]$ is obtained by subtracting the probability that all the variables are under k from the probability under $k+1$. Thus, we obtain $W[k]$ as follows:

$$W[k] = \begin{cases} \prod_{l=1}^{L} C_l[1] & (k = 1) \\ \prod_{l=1}^{L} C_l[k] - \prod_{l=1}^{L} C_l[k-1] & (k = 2, 3, \ldots) \end{cases} \tag{7}$$

Utilizing this fact, we can obtain the following Algorithm 2 to determine the maximum of multiple variables.

Algorithm 2 The procedure of $W \leftarrow \mathrm{GETMAX}_{l \in L}\{Z_l\}$

Require: M: the number of quantization levels in discretization
Require: L: the number of input variables
Ensure: Z_l: input discrete distributions
Ensure: W: result discrete distribution

```
for l = 1, 2, ..., L do
    C_l[1] ← Z_l[1]
    for k = 2, 3, ..., M do
        C_l[k] ← C_l[k − 1] + Z_l[k]
    end for
end for
W[1] ← ∏_{l=1}^{L} C_l[1]
for k = 2, 3, ..., M do
    W[k] ← ∏_{l=1}^{L} C_l[k] − ∏_{l=1}^{L} C_l[k − 1]
end for
```

The complexity of this algorithm is $O(LM)$.

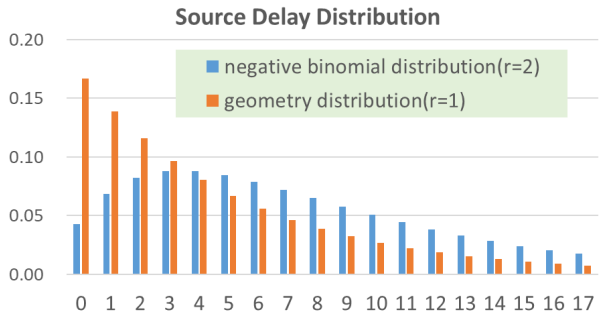

Fig. 2 An example of the geometry distribution ($r = 1$) and the negative binomial distribution ($r = 2$) with these averages at 6.0.

4.5 Accelerated Convolution Algorithm in Case of Negative Binomial Distribution

Since we assume the random variables as discrete in this paper, the distribution of source delays may also be discrete. For the distribution of source delays $D[k]$, it is desirable to have most of the masses on small k domain, and to have long tails.

The negative binomial distribution, which includes the geometric distribution as a special case, is one of such examples as shown in **Fig. 2**.

The PDF of the negative binomial distribution $D_r[k]$ is as follows:

$$D_r[k] = \binom{k+r-1}{k} p^k (1-p)^r \quad (k = 0, 1, 2, \ldots), \tag{8}$$

where p and r are the parameters of the distribution. Note that when $r = 1$, we get the geometric distribution.

Consider the negative binomial distribution as source delays. Then $Z_r[k] = \sum_{i=1}^{k} X[k] \cdot D_r[k-i]$ which is the convolution of $X[k]$ and $D_r[k]$ satisfies the following recurrence (The proof is given in the Appendix):

$$Z_r[k] = \begin{cases} (1-p) \cdot X[1] & (k = 1, r = 1) \\ (1-p) \cdot Z_{r-1}[1] & (k = 1, r = 2, 3, \ldots) \\ p \cdot Z_1[k-1] + (1-p) \cdot X[k] & \\ \qquad (k = 2, 3, \ldots, M, r = 1) & \\ p \cdot Z_r[k-1] + (1-p) \cdot Z_{r-1}[k] & \\ \qquad (k = 2, 3, \ldots, M, r = 2, 3, \ldots) & \end{cases} \tag{9}$$

Utilizing this recurrence, we can obtain the following Algorithm 3 to get the convolution.

Algorithm 3 The procedure of $Z_R \leftarrow \mathrm{CONV}(X, D)$

Require: M: the number of quantization levels in discretization
Require: R: parameter of negative binomial distribution
Ensure: Z_R: result discrete distribution

```
Z_1[1] ← (1 − p) · X[1]
for r = 2, 3, ..., R do
    Z_r[1] ← (1 − p) · Z_{r−1}[1]
end for
for k = 2, 3, ..., M do
    Z_1[k] ← p · Z_1[k − 1] + (1 − p) · X[k]
    for r = 2, 3, ..., R do
        Z_r[k] ← p · Z_r[k − 1] + (1 − p) · Z_{r−1}[k]
    end for
end for
```

The complexity of this Algorithm 3 is $O(MR)$, where R is the parameter of the negative binomial distribution. We can also show that the complexity of $O(M)$ is achieved in the case where D is divided into multiple segments each of which corresponds to the part of the negative binomial distribution's PDF. Having a large degree of freedom, this PDF is useful to approximate actual delays.

Note that we must care truncation errors in Algorithm 3 since the output distribution is terminated in finite k. To deal with this, we implement two measures. First, we calculate $Z_r[k]$ on the domain not only in the range from $k = 1$ to $k = M$ but in the range from $k = 1$ to $k = M + b$, since we need to apply SHIFT(Y, b) immediately after the convolution.

However, even if we consider the extended domain, we still have truncation errors, since the distribution D_k has the mass over $M+b$. Thus, we normalize $Z[k]$ after the operation so that the sum becomes 1.0.

These computational efforts mitigate truncation errors.

4.6 Complexity of the Algorithm

In general case, the complexity of the convolution is $O(M \log M)$ and that of other operations in Algorithm 1 is $O(M)$. Thus, the total complexity of the Algorithm 1 is $O(M \log M)$; therefore, the total complexity of the proposed approach is $O(M \log MN)$ since we have to apply Algorithm 1 once to each node in the topological order, where N is the number of nodes.

On the other hand, in special cases where source delays are the negative binominal distributions, the complexity can be reduced to $O(M)$ as shown in Section 4.5, so that the total complexity of the algorithm is reduced to $O(MN)$, which means linear (means very fast) in both of M and N.

5. Computational Experiments

In this section, we evaluate the proposed discrete delay propagation method (**DDP**) from the viewpoints of accuracy and calculation time.

5.1 Monte Carlo Simulation-Based Approach

To get propagated delay distributions, Monte Carlo simulation (**MC**) can also be used in the following manner. **MC** performs multiple iterations to get histograms of propagated delays in each node. In each iteration, we evaluate nodes in the topological order like **DDP**. However, unlike **DDP**, **MC** regards source delays $d_{i,j}$ as not random variables but deterministic values. As a result, all the propagated delays (X_js) are also deterministic variables. Thus, each node's process with Eq. (3) is simpler in **MC** than in **DDP** since **MC** only needs to generate $d_{i,j}$ from $P_{i,j}$ and to perform scalar operations after that.

The bottle neck of each node's process in **MC** is the procedure that generates source delays from the negative binomial distribution. We utilize binary search in the range 0 to M to find out the point where CDF of the binomial distribution exceeds a value generated from the uniform distribution in (0, 1). Thus, the complexity of each node process is $O(\log(M))$.

MC repeats this iteration S times to get histograms of bins $1, 2, \ldots M$ corresponding to the range of discrete variables in **DDP**. Then we obtain the propagated delay distribution by dividing frequencies in each bin by S.

Thus, the total complexity of **MC** is $O(S \log(M))$. Therefore, though calculation time of each iteration in **MC** is shorter than **DDP**, the total time of **MC** may be longer for large S. We set $S = 10{,}000$ based on preliminary test results.

Note that **MC** does not assume **P_Independence**, so that **MC** can evaluate propagated delays under more natural assumptions. However, as S tends to infinity, **MC** results converge to the distribution which is different from **DDP**'s.

5.2 Evaluation Using Small Network

For evaluating proposed method, we first used an small artificial three-train-five-station network.

5.2.1 Experimental Setting

The network consists of 22 nodes as shown in **Fig. 3**. Then we set $M = 720$ ($= 60 \times 60/5$), under assumption that we evaluate delays at most 60 minutes with accuracy of five seconds. On the network, we evaluate the influence of a trigger delay on the initial node to the final nodes. Then we consider four scenarios. We set trigger delays and buffer times $b_{i,j}$ for each scenario as follows:

- Scenario 1 (SN1): We set no delay on the initial node. Then buffer times for all the headway arcs are from one to three minutes as shown in Fig. 3, while those for residual arcs are 0.5 minute.
- Scenario 2 (SN2): We set no delay on the initial node. Then buffer times for headway arcs are the values obtained by adding 3 minutes to those for SN1, while those for residual arcs are the same as those for SN1. That is, we assume the situation where we move back the whole sequence of train 2 and 3 by 3 and 6 (= 3 + 3) minutes for SN1, respectively.
- Scenario 3 (SN3): We set 10-minute delay on the initial node, and buffer times are same as SN1.
- Scenario 4 (SN4): We set 10-minute delay on the initial node, and buffer times are same as SN2.

We set source delays in the initial node occur at $k = 0$(in SN1 and SN2) or $k = 120$ (in SN3 and SN4) with a probability of 1.0. Then we assume source delays on each arc follow the negative binominal distribution ($r = 2$) with average $\mu_{i,j}$. Here, to make 100 instances for each scenario, we generate $\mu_{i,j}$ from independent discrete uniform distributions in the range from 1 to 24, meaning the range from 5 to 120 seconds.

The experiments were all performed on a computer with an Intel Xeon E5-2697v3 2.60 GHz Processor and 264 GB RAM.

5.2.2 Computational Results

Table 1 shows the comparative results of **DDP** and **MC** from SN1 to SN4 by evaluating the propagated delay distribution of the final node. The third to sixth columns of the table corresponds to the average of mean delay time in minutes (m) and the probability that delays are k minutes or over ($q_k : k = 5, 10, 15$) of the distribution in 100 instances.

Results show that the delay times in SN1 and SN3 are longer than those in SN2 and SN4 due to the shorter buffer times in SN1 and SN3, while delay times in SN3 and SN4 are shorter than those in SN1 and SN2 due to trigger delays in SN3 and SN4.

Moreover, in the cases where buffer times between trains are

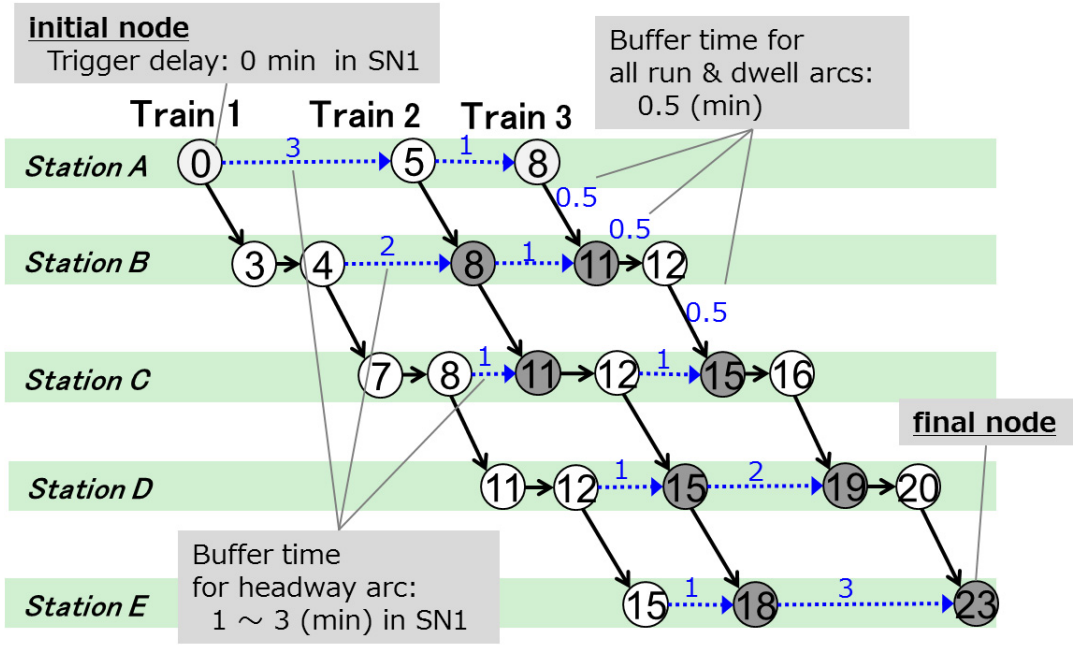

Fig. 3 An example of the network that we use in the evaluation. The number in each node shows scheduled time of the node, while the number on each arc shows the buffer time of the arc in SN1.

Table 1 Comparative results in **DDP** and **MC** from SN1 to SN4. We show mean delays in minutes (m) and the probability that delays are k minutes or over ($q_k : k = 5, 10, 15$). All results are the average of 100 instances.

SN	Method	m	q_5	q_{10}	q_{15}
SN1	**DDP**	6.0	61.0	8.0	0.4
	MC	5.3	49.3	6.2	0.4
SN2	**DDP**	4.1	32.3	3.0	0.1
	MC	4.0	31.3	3.0	0.1
SN3	**DDP**	15.0	100.0	98.7	46.8
	MC	13.9	100.0	93.1	32.9
SN4	**DDP**	9.5	97.9	39.4	3.9
	MC	8.8	93.8	30.7	3.1

relatively large as in SN2, the result values are almost same in both **DDP** and **MC**. On the other hand, when buffer times are relatively small as in SN1, SN3, or SN4, there are gaps between **DDP** and **MC**. This is because **DDP** assume **P_Independence** while **MC** does not. In case where buffer times are small, **P_Independence** usually does not hold since the correlation in delays from different arcs gets larger. Though we have not proven yet, mean delays of **DDP** tends to be slightly larger than those of **MC** in many cases as shown in the difference of m in Table 1.

Then **Fig. 4** shows both linear and log-scale plots of the final nodes' distributions from SN1 to SN4 in **DDP** and **MC**.

As shown in the results in Fig. 4, we confirm the curve of **DDP** roughly agrees with that of **MC**, where the probability mass is over 10^{-4}.

However, **MC** cannot calculate the values under 10^{-4} since the number of iterations in **MC** is 10^4, while **DDP** can provide smooth curves when the probability mass is under 10^{-4}. Moreover, results of **MC** seem to be unstable even in the region where the probability mass is slightly larger than 10^{-4}. When we analyze propagated delays, we usually focus on rare events. Therefore, we believe that **DDP** which can evaluate rare events has the advantage over **MC**.

5.3 Evaluation using real network model

Then we perform additional experiments on real network data in a UK Train Operation Company on May 2018.

5.3.1 Experimental Setting

First we can configure node set V from the departure, arrival, and passing events based on the morning rush hour timetable. As a result, the number of nodes is 1,472. Then, we span arcs between nodes where minimum times are set for security reasons. For example, in addition to the cases we have shown in Section 3.1, we span headway arcs between nodes where a train uses the different platform from the preceding train for connecting or overtaking. Thus, the number of incoming arcs is at most three.

For each headway arc between different trains, we use the minimum time defined on the timetable for security reasons for $h_{i,j}$. On the other hand, for each run and dwell arc, we use the minimum time calculated from run curves on the timetable for $h_{i,j}$. Note that the $h_{i,j}$ of arcs differs depending on whether the types of nodes on both sides of the arc are departure, arrival, or passing nodes. We can calculate the buffer time $b_{i,j}$ by taking the difference between $h_{i,j}$ and the time difference in a timetable as shown in Section 3.2.

Moreover, we obtain the average time of source delay in each run or dwell arc from the historic records of train lateness available in Ref. [14]. In the following experiment, this value was multiplied by α ($\alpha = 0.125, 0.25, 0.5$ and 1.0) to evaluate various delay patterns. We express the source delay distribution as a negative binomial distribution ($r = 2$) with this value as the average.

Finally we set $M = 1{,}500$ ($= (120 + 5) \times 60/5$), under assumption that we evaluate propagated delays from -5 minutes to 120 minutes with accuracy of five seconds. Note that propagated delays in real network may take negative values on the event where arriving or passing earlier than scheduled time is permitted.

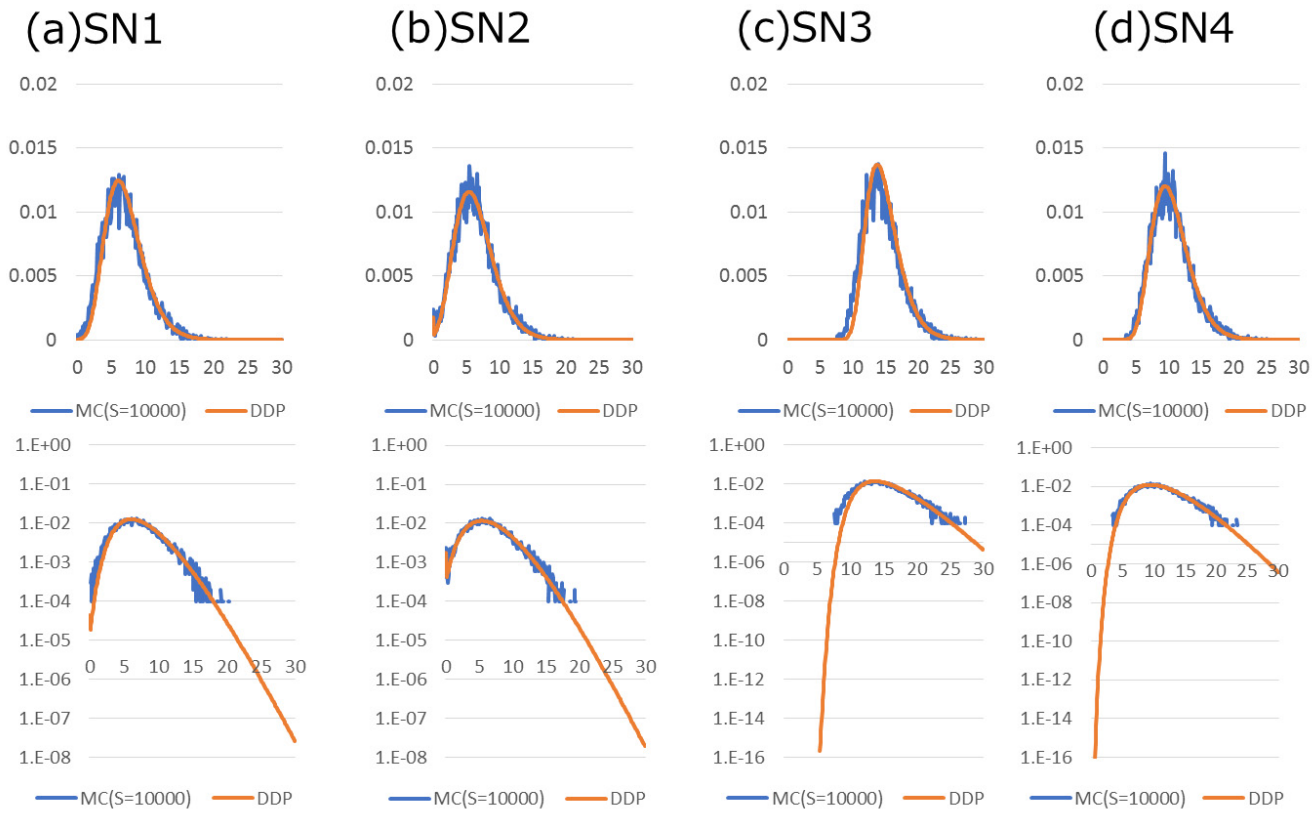

Fig. 4 Examples of final node's propagated delay distributions in **DDP** and **MC** in (a) SN1, (b) SN2, (c) SN3 and (d) SN4. The horizontal axis in each plot represents the delay time in minutes. The vertical axis is linear in the top figure, while log-scale in the bottom figure for each scenario.

Table 2 Comparison of mean delays of the final node (m) and computation time for **DDP** and **MC** for $\alpha = 0.125, 0.25, 0.5$, and 1.0.

α	method	m (minute)	conputation time (s)
1.0	**DDP**	50.7	0.021
	MC	43.7	4.139
0.5	**DDP**	31.8	0.022
	MC	29.1	3.282
0.25	**DDP**	23.8	0.021
	MC	22.5	2.544
0.125	**DDP**	19.9	0.021
	MC	19.2	1.922

5.3.2 Computational Results

Table 2 shows the comparative results of **DDP** and **MC** for $\alpha = 0.125, 0.25, 0.5, 1.0$ by evaluating mean delays of the final node and the computation times. This result shows that computation time in **DDP** is over 90 times faster than that in **MC**. However, there are gaps between the mean delays in **DDP** and **MC**. Especially in the case where the amount of delays is larger (e.g., in $\alpha = 1.0$), the gaps get larger.

In order to investigate the cause of gaps, we examine the accumulation degree of propagated delays in each case. Figure 2 shows the average of propagated delays in each nodes in all eight cases shown above. In this figure, the nodes in horizontal axis are sorted by propagated delay time in **DDP** ($\alpha = 1.0$) case. In fact, this is roughly the topological order.

This result shows that delays are accumulated constantly and the amount of the propagated delay increases almost monotonically, representing the tendency that the propagated delay accumulates in the rush hour. Moreover, results show that while differences between the mean delay time in **DDP** and that in **MC** are also small when N is small, the difference gradually expands as N becomes larger. In other words, the slight difference seen in small problems as shown in Section 5.2.2 gets larger as N becomes larger, since delays are propagated without interruption in the rush hour.

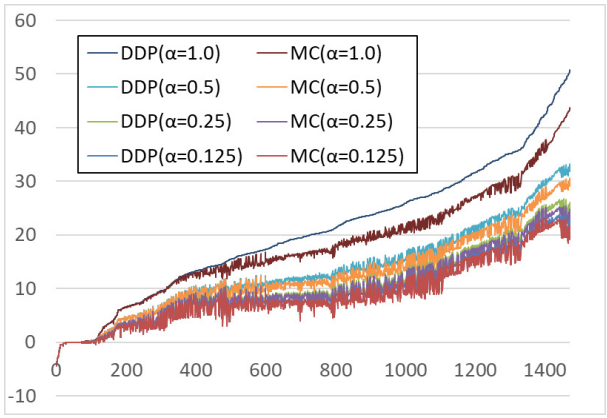

Fig. 5 Plots of propagated delay time in **DDP** and **MC** for $\alpha = 0.125, 0.25, 0.5, 1.0$. The vertical axis shows the mean propagated delays in minutes. The horizontal axis represents the nodes, and the order of the nodes are sorted by the delay in **DDP**($\alpha = 1.0$) case.

6. Conclusion

In this paper, we have proposed a novel discrete distribution propagation method (**DDP**) to evaluate delay distributions on railway timetables. We have shown the way of calculating the propagated delay distribution on each event analytically under the as-

sumption of independence of source and propagated delays. Then we have shown the complexity of **DDP** is $O(M \log MN)$ in the general case and $O(MN)$ in the special cases where source distributions are the negative binominal distributions, where M denotes the number of quantization levels in discretization and N denotes the total number of events. Finally, computational experiments show that the proposed method is much faster than Monte Carlo simulations (MC), and provides almost the same results as MC when N is small. Moreover, **DDP** also has the advantage that it can be executed only by easily-accessible operation records such as the average of run and dwell time in each node, unlike conventional detailed simulation-based approaches.

However, **DDP** requires the unnatural assumption of the propagated delay independence. As a result, there is a gap between the result of **DDP** and that of **MC** in cases where N is large or the amount of propagated delays is large. At this moment, we need further analysis regarding the level of error arising from this assumption. Thus, evaluating the effect of the error and improving **DDP** to cope with the problem need to be performed in the future work.

References

[1] Meester, L. and Muns, S.: Stochastic Delay Propagation in Railway Networks and Phase-type Distributions, *Transportation Research Part B*, Vol.41, pp.218–230 (2007).

[2] Buker, T. and Seybold, B.: Stochastic Modelling of Delay Propagation in Large Networks, *Journal of Rail Transport Planning and Management*, Vol.2, pp.34–50 (2012).

[3] Abe, K. and Araya, S.: Train Traffic Simulation Using the Longest Path Method, *Trans. Information Processing Society of Japan*, Vol.27, No.1, pp.103–111 (1986).

[4] Nagasaki, Y., Takano, M. and Koseki, T.: Train Rescheduling Evaluation and Assistance System with Passengers' Behavior Simulation based on Graph Theory, *2004 Technical Meeting on Systems and Control*, pp.25–29, IEE Japan (2004).

[5] Tomii, N., Zhou, L.J. and Fukumura, N.: An Algorithm for Shunting Scheduling Problems Combining Probabilistic Local Search and PERT, *IEEJ Trans. Electronics, Information and Systems*, Vol.119, No.3 (1999).

[6] Ushida, K., Makino, S. and Tomii, N.: Increasing Robustness of Dense Timetables by Visualization of Train Traffic Record Data and Monte Carlo Simulation, *Proc. World Congress on Railway Research* (*WCRR*) (2011).

[7] Tatsui, D., Nakabasami, K. and Kunimatsu, T.: Predicting Method of Train Delay and Train Congestion Using Neural Network, RTRI Report, Vol.31, No.10, pp.29–34 (2017).

[8] Nakamura, Y.: Simulation Model to Analyze Delay in Commuter Train Schedule, Master Thesis (2004).

[9] Takeuchi, Y., Tomii, N. and Hirai, C.: Evaluation Method of Robustness for Train Schedules, *Quarterly Report of RTRI*, Vol.48, No.4, pp.197–201 (2007).

[10] Kirchhoff, F. and Kolonko, M.: Modelling Delay Propagation in Railway Networks Using Closed Family of Distributions, available from ⟨http://bisec.bjut.edu.cn/en/Events/Colloquium_&_Seminar/201798/15048610866411526_1.html⟩ (accessed 2019-03-15).

[11] Yuan, J.: Stochastic Modelling of Train Delays and Delay Propagation in Stations, Trail Series T2006/6, Eburon Academic Publishers (2006).

[12] Yuan, J. and Hansen, I.A.: Optimizing Capacity Utilization of Stations by Estimating Knock-on Train Delays, *Transportation Research Part B*, Vol.41, pp.202–217 (2007).

[13] Knuth, D.: *Seminumerical Algorithms* (3rd. ed.), Reading, Massachusetts: Addison-Wesley (1997).

[14] available from ⟨https://raildar.co.uk/⟩ (accessed 2019-03-15).

Appendix

A.1 Proof of Eq. (9)

Initially in the case of $k = 1$, when we consider the fact that $Z_r[1] = X[1] \cdot D_r[0] = X[1] \cdot (1-p)^r$,

$$Z_r[1] = \begin{cases} (1-p) \cdot X[1] & (k=1, r=1) \\ (1-p) \cdot Z_{r-1}[1] & (k=1, r \geq 2) \end{cases} \tag{A.1}$$

holds, so that we have shown Eq. (9) in the case of $k = 1$.

Secondly in the case of $k \geq 2, r = 1$,

$$\begin{aligned} Z_1[k] &= \sum_{i=1}^{k} X[i] \cdot D_1[k-i] \\ &= \left\{ \sum_{i=1}^{k-1} X[i] \cdot p^{k-i} \cdot (1-p) \right\} + X[k] \cdot (1-p) \\ &= p \cdot \left\{ \sum_{i=1}^{k-1} X[i] \cdot p^{k-i-1} \cdot (1-p) \right\} + (1-p) \cdot X[k] \\ &= p \cdot Z_1[k-1] + (1-p) \cdot X[k] \end{aligned} \tag{A.2}$$

holds, so that we have shown Eq. (9) in the case of $k \geq 2, r = 1$.

Finally, in the case of $k \geq 2, r \geq 2$, consider the following two recurrences. First, when $k \geq 1, r \geq 2$, the following recurrence

$$\begin{aligned} D_r[k] &= \binom{k+r-1}{k} p^k (1-p)^r \\ &= \left\{ \binom{k+r-2}{k-1} + \binom{k+r-2}{k} \right\} \\ &\quad p^k (1-p)^r \\ &= p \cdot \binom{(k-1)+r-1}{k-1} p^{k-1} (1-p)^r \\ &\quad + (1-p) \cdot \binom{k+(r-1)-1}{k} p^k (1-p)^{r-1} \\ &= p \cdot D_r[k-1] + (1-p) \cdot D_{r-1}[k] \end{aligned} \tag{A.3}$$

holds. Then in the case of $k = 0, r \geq 2$, the following recurrence

$$D_r[0] = (1-p) \cdot D_{r-1}[0] \tag{A.4}$$

holds. Defining $D_r[-1] = 0$ for convenience, we summarize Eqs. (A.3) and (A.4) to have

$$D_r[k] = p \cdot D_r[k-1] + (1-p) \cdot D_{r-1}[k], \tag{A.5}$$

when $k \geq 0$ and $r \geq 2$. Using this Equation, we obtain

$$\begin{aligned} Z_r[k] &= \sum_{i=1}^{k} X[i] \cdot D_r[k-i] \\ &= \sum_{i=1}^{k} X[i] \cdot \left\{ p \cdot D_r[k-i-1] + (1-p) \cdot D_{r-1}[k-i] \right\} \\ &= \sum_{i=1}^{k-1} X[i] \cdot p \cdot D_r[k-i-1] \\ &\quad + \sum_{i=1}^{k} X[i] \cdot (1-p) \cdot D_{r-1}[k-i] \} \\ &= p \cdot Z_r[k-1] + (1-p) \cdot Z_{r-1}[k], \end{aligned} \tag{A.6}$$

so that we have shown Eq. (9) in the case of $k \geq 2, r \geq 2$.

Under the discussion above, we have proven Eq. (9) in all the cases.

Tomoshi Otsuki was born in 1979. He received his Ph.D. (Information Science and Technology) from the University of Tokyo. He is currently a senior research scientist at Toshiba Corporation. His research interest is optimization and scheduling. He is a member of the Operations Research Society of Japan.

Hideo Sakamoto was born in 1967. He received his M.S. degree from Tokyo Institute of Technology in 1992. His research interest is applied probability models. He is a member of the ORSJ.

Hideki kubo was born in 1965. He received his Bachelor of Engineering from Doshisha University in 1989. He joined the TOSHIBA Corporation in 1989 and received the certificate of Professional Engineer, Japan (Information Engineering) in 2010. He is currently designing software for transportation systems.