A Mixed Integer Linear Programming Approach to a Rolling Stock Rostering Problem with Splitting and Combining

Satoshi Kato [a], Naoto Fukumura [b], Susumu Morito [c]
Koichi Goto [b], Narumi Nakamura [b]
[a] Transport Operation Systems Laboratory
Signalling and Transport Information Technology Division
Railway Technical Research Institute
2-8-38 Hikari-cho, Kokubunji-shi, Tokyo 185-8540, Japan
E-mail: kato.satoshi.58@rtri.or.jp, Phone: +81 (0) 42 573 7311
[b] JR Souken Information Systems
[c] Department of Industrial and Management Systems Engineering, Waseda University

Abstract
Railway operators must schedule resources such as rolling stock and crew in order to operate trains as defined by a timetable. This paper considers scheduling of rolling stock, which is usually done by creating a roster. A roster is a series of trains to be performed by the particular rolling stock. The number of train-sets required to operate a given group of trains is essentially determined by the roster and generation of an efficient roster is essential. Important considerations of the roster generation include maintenance such as pre-departure inspection. On some lines in Japan, splitting and combining are often used to adjust transportation capacity flexibly. Under this type of operation, splitting and combining become necessary. These shunting operations require time and manpower, so it is necessary to reduce the amount of splitting and combining. This paper presents a mixed integer linear programming model so that the amount of splitting and combining is reduced together with the roster length and the distance of empty runs. Results of computational studies will be presented based on real instances of several lines in Japan, indicating the computational effectiveness of the methodology and with respect to the reasonableness of the resultant rosters.

Keywords
Rolling stock rostering, Splitting and combining, Maintenance, Mixed integer linear programming, Travelling salesman problem

1 Introduction

Railway operators must schedule resources such as rolling stock and crew in order to operate trains as defined by a timetable. This paper considers scheduling of rolling stock, which is usually done by creating a roster. A roster is a series of trains to be performed by particular rolling stock, and cyclic execution of the roster theoretically determines which train-sets are assigned to which train. The number of train-sets required to operate a given group of trains is essentially determined by the roster and generation of an efficient roster is essential. Important considerations of the roster generation include maintenance such as pre-departure inspections.

On some lines in Japan, splitting and combining are often used to adjust transportation

capacity flexibly. For example, two train-sets are combined together (say, two 3-car train-sets are combined together, thus effectively forming a 6-car train-set) during morning and evening rush hours, but during the day time, only one train-set is assigned to each train. Under this type of operation, shunting operations of splitting and combining would become necessary. These shunting operations require time and manpower, so it is necessary to reduce the amount of splitting and combining.

Many studies exist on efficient planning and management of rolling stock and a variety of models and algorithms for optimization have been developed. Abbink et al. (2004) give an integer programming model to allocate train types so that shortage of capacity during rush hours is minimized. One fundamental study in the field is Alfieri et al. (2006), presenting a multi-commodity flow model of rolling stock circulation for determining the appropriate number of train units of different types together with their efficient circulation on a single Dutch line. These studies, however, do not consider maintenance.

Giacco et al. (2014) formulate a rolling stock rostering problem with maintenance considerations as a generalized travelling salesman problem (TSP) where a roster is represented as a tour of the associated network. Borndörfer et al. (2016) and Reuther and Schlechte (2018) dealing with the same problem give a mixed integer programming model based on a hypergraph. Morooka et al. (2017) present computational experiences with the Giacco's model and its variants using real instances of several lines in Japan. Nishi et al. (2017) propose a column generation and Lagrangian heuristics for rostering with maintenance considerations.

Regarding research with explicit considerations for splitting and combining, Fioole et al. (2006) present an integer programming model of rolling stock circulation with objective criteria such as operational costs, service quality, and reliability including reduction of shunting movements. Peeters and Kroon (2008) develop a branch-and-price algorithm for a similar problem. These studies, however, do not consider maintenance. Tsunoda et al. (2015) give a multi-commodity network flow model to estimate the required number of train units to meet future traffic demands under a splitting and combining policy, but do not consider maintenance or reduction of shunting operations.

This paper presents an optimization-based methodology to construct a roster with maintenance considerations under the existence of splitting and combining. More specifically, a mixed integer linear programming (MILP) model is proposed based on a travelling salesman problem with multiple arcs between nodes so that the amount of splitting and combining is reduced together with the roster length and the distance of empty runs. Results of computational studies will be presented based on real instances of several lines in Japan with roughly 100 to 200 trains, indicating the effectiveness of the methodology computationally and with respect to the reasonableness of the resultant rosters.

2 Problem Definition

2.1 Rolling Stock Rostering Problem

A rolling stock schedule is produced by covering all trains shown on a train timetable. Since the number of rolling stock required to meet the train timetable requirements is determined by a rolling stock schedule, it is necessary to produce an efficient schedule. A rolling stock schedule must satisfy the minimum time interval between the arrival of a train and its subsequent departure, together with maintenance requirements as described

shortly.

A rolling stock schedule is achieved as a cyclic schedule called a roster. A roster is a finite series of daily schedules performed by a train-set, called duties, which are performed by a particular train-set in a cyclic fashion. The number of train-sets needed to implement a given roster coincides with the number of duties in the roster, which is sometimes called the roster length. Naturally, generating a roster with the shortest roster length is desirable. Rosters are normally created for each type of train-set.

Important considerations of the roster generation are a type of maintenance that must be performed within a specified interval. Four types of maintenance occur in Japan; namely, pre-departure inspection, regular inspection, bogie inspection, and general inspection. This paper focuses on pre-departure inspections, whose intervals are the shortest. This is because other inspection types often take at least one day, while pre-departure inspections only need a few hours and a train-set that has been inspected or is to be inspected is often assigned to trains within the same calendar day; thus, when to perform inspections must be determined. Throughout the rest of this paper, pre-departure inspections are simply called "inspections".

The scenario considered in this paper often occurs in non-metropolitan cities, and deals with the case where only k-car train-sets are available, where typical values of k are 2 to 4. During morning and evening rush hours, some trains are operated by combining two k-car train-sets, thus effectively yielding a $2\times k$-car train-set. In contrast, except for these rush hours, services are operated by single k-car train-sets. This type of operation gives an effective way to utilize the limited number of rolling stock, and is particularly suited for lines in which rush hour demands differ substantially from those of other time periods.

In order to achieve the above types of operations, splitting and combining of train-sets would become necessary. Here, splitting means to split a combined train-set into two separate train-sets, and combining means to combine two separate train-sets together. Splitting and combining allow flexible adjustments of traffic capacity, but additional time and work of operators will be required, so reducing the amount of splitting and combining is desirable.

Figure 1 shows an example of a rolling stock schedule. In this example, there are four stations, Stations A through D, and nine trains (in service), Trains 1 through 9. Station C is adjacent to a rolling stock depot, where the inspection of a train-set can be performed. A circle means the start of a duty and a triangle means its end. The double lines such as Trains 2, 6, and 9 indicate what we call "double-unit trains" that are operated by two train-sets combined together. On the other hand, single lines such as Trains 1, 3, 4, 5, 7, and 8 indicate what we call "single-unit trains" that are operated by a single train-set.

For those double-unit trains shown by double lines, the left lines indicate the front side of the train-sets, and the right lines indicate the rear side of the train-sets. The two train-sets operating Train 2 are assigned to two distinct trains, Trains 5 and 7, upon arrival at Station A, thus indicating the existence of splitting at Station A. In contrast, combining at Station D would be performed before the departure of Train 6. For this rolling stock schedule, one splitting as well as one combining would be required. Maintenance is performed at the depot next to Station C upon the arrival of Train 4, after which the train-set is deadheaded to Station D.

Figure 2 shows a roster associated with the rolling stock schedule given in Figure 1. There are six duties, Duties 1 through 6, which are assigned to train-sets in this order. After performing Duty 6, a train-set is assigned to Duty 1 again, and this cycle will be repeated. Note that the ending station of a duty coincides with the starting station of the

subsequent duty so that the train-set can continue the schedule cyclically. The number of duties, namely, the roster length, corresponds to the minimum number of train-sets required to achieve the schedule. Duty 3 includes an inspection and this roster contains one inspection every six days, making the schedule feasible if the upper limit of the inspection cycle is six days. The letters F and B shown next to Trains 2, 6, and 9 indicate Front and Back, meaning the front and back halves of the train-sets, respectively. For example, Train 2 is operated by combining the train-sets of Duty 2 and Duty 5.

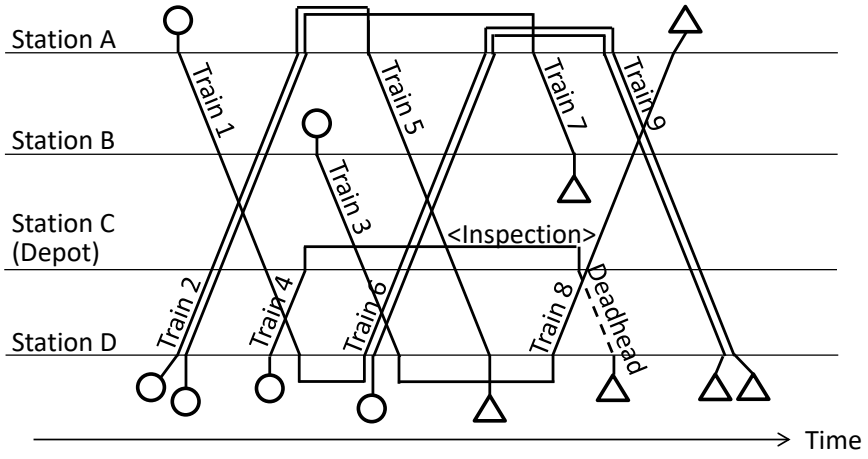

Figure 1. Sample rolling stock schedule

Duty	
1	A ○ Train 1 D Train 6(F) A Train 9(B) △ D
2	D ○ Train 2(F) A Train 5 △ D
3	D ○ Train 4 C <Inspection> Deadhead – – △ D
4	D ○ Train 6(B) A Train 9(F) △ D
5	D ○ Train 2(B) A Train 7 △ B
6	B ○ Train 3 D Train 8 △ A

Figure 2. Sample rolling stock roster

2.2 Splitting and Combining

This paper considers rolling stock rostering with splitting and combining, and seeks the generation of practical rolling stock schedules. For this purpose, it is necessary to identify where splitting and combining occur and to count the exact number of splittings and combinings. Judging when splitting and combining occur, however, is complicated because doing so depends not only on the required number of train-sets before and after train connections, but also on their train positions.

Splitting will always be necessary to connect from a double-unit train to a single-unit train. Similarly, combining will be necessary to connect from a single-unit train to a double-unit train. The difficulties occur when a double-unit train is connected to another double-unit train, where splitting and combining may or may not occur. When the order of the two train-sets of a double-unit train is the same for the subsequent double-unit train, splitting or combining are not required. On the other hand, there may be a case requiring splitting and combining because the order of the two train-sets of a double-unit train is reversed for the subsequent double-unit train. It is also possible that two train-sets of a double-unit train are connected to two distinct double-unit trains, which require splitting and combining. Therefore, it is necessary to look at train positions, since the existence or non-existence of splitting and combining depends on the train positions.

Figure 3 shows several types of train connections indicating complexities due to splitting and combining. In Figure 3a, Train 2 connects to Train 1 after changing the direction of movement, thus requiring no splitting or combining. In Figure 3b, though, train positions are reversed between the two trains, thus requiring one splitting and one combining. Considering connections between two incoming trains and two outgoing trains, Figure 3c shows a case where Train 2 connects to Train 1 and Train 4 to Train 3 keeping the same train positions, thus requiring no splitting or combining. However, connections as in Figure 3d are possible where splitting will be required after the arrivals of Trains 2 and 4, and also combining before the departures of Trains 1 and 3. Avoiding such connections as in Figure 3d which require many splittings and combinings is desirable.

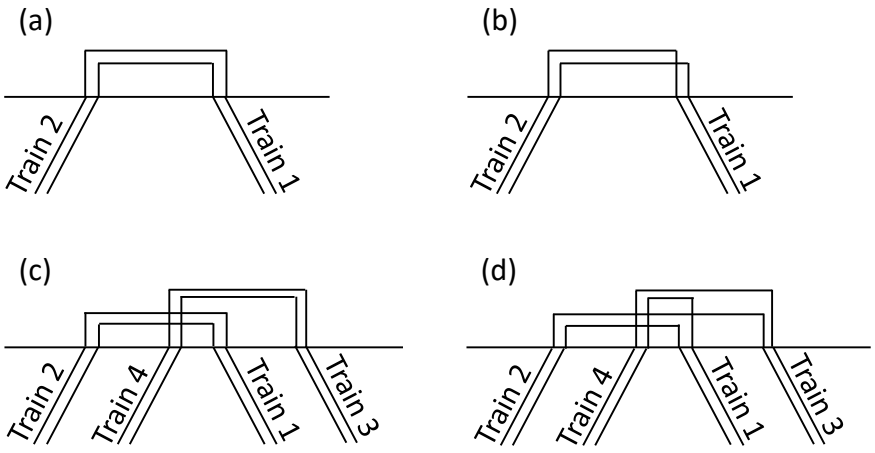

Figure 3. Complication of splitting and combining

We now state our rostering optimization problem with splitting and combining.

1. Given a number of an identical type of train-sets, a single roster would be constructed.
2. Given a set of trains in a given timetable to which train-sets are assigned, the number of train-sets to be assigned to each train is specified as either 1 or 2 in advance. No train requires three or more train-sets. We call those trains operated by a single train-set "single-unit trains", while those trains operated by two train-sets, are called "double-unit trains".
3. Maintenance requirements call for a single type of pre-departure inspection to be performed within minimum and maximum intervals measured in days.
4. The locations and time period during which inspections could be performed are known in advance. Generally, there are multiple locations for maintenance.
5. Empty runs could be inserted as needed.
6. Performance measures include the roster length, the total distance of empty runs, and the amount of splitting and combining, and their weighted sum is to be minimized.

3 Mixed Integer Linear Programming Model for a Rolling Stock Rostering Problem with Splitting and Combining

3.1 Network Model

Our MILP model is based on the roster optimization model of Giacco et al. (2014), where a network is considered in which a node corresponds to each train, and an arc to the connection between trains. We now describe how arcs are drawn in the network. For each connection from node i (its associated train) to node j (its associated train), we check if an arc can be drawn by grouping them into four different types, as follows.

(i) No empty run and no inspection

(ii) Inspection and no empty run

(iii) Empty run and no inspection

(iv) Empty run and inspection

In the following, the details of arc settings for each type will be described. Notations used for the network model are defined in Table 1.

Table 1: Notations for the network model

Notation	Definition
arr_time_i	Arrival time at the destination of the train corresponding to node i
dep_time_i	Departure time at the origin of the train corresponding to node i
arr_sta_i	Destination station of the train corresponding to node i
dep_sta_i	Origin station of the train corresponding to node i
emp_time_{ij}	Time of empty run from destination station of the train corresponding to node i to origin station of the train corresponding to node j
ins_min_time	Earliest possible start time of inspection
ins_max_time	Latest possible completion time of inspection
ins_time	Time required for inspection
$interval\_time$	Minimum time interval between two trains

Type 1: No Empty Run and No Inspection

An arc is drawn if arr_sta_i is equal to dep_sta_j. If $arr\_time_i + interval\_time \leq dep\_time_j$, the arc is set as a same-day arc. That is, the date remains the same after passing through the arc. If $arr\_time_i + interval\_time > dep\_time_j$, the arc is set as a next-day arc. That is, the date changes by one after passing through the arc.

Type 2: Inspection and No Empty Run

An arc is drawn if arr_sta_i is equal to dep_sta_j, and also if arr_sta_i is a station capable of performing an inspection.

1. Case where an inspection is performed the same day
   An arc is drawn as a same-day arc if the following condition is satisfied:

$$\max\{arr\_time_i, ins\_min\_time\} + ins\_time \leq \min\{dep\_time_j, ins\_max\_time\} \quad (1)$$

   Judge if an inspection is possible within the same day.
2. Case where an inspection is performed the next day
   An arc is drawn as a next-day arc if the following condition is satisfied:

$$\max\{arr\_time_i, ins\_min\_time\} + ins\_time > \min\{dep\_time_j, ins\_max\_time\} \wedge ins\_min\_time + ins\_time \leq dep\_time_j \quad (2)$$

   Here an inspection can be started at ins_min_time the next day.

Type 3: Empty Run and No Inspection

An arc is drawn when arr_sta_i is different from dep_sta_j. The arc is set as a same-day arc if the following condition is satisfied. Otherwise, the arc is set as a next-day arc.

$$arr\_time_i + 2 * interval\_time + emp\_time_{ij} \leq dep\_time_j \quad (3)$$

Type 4: Empty Run and Inspection

An arc is drawn when arr_sta_i and dep_sta_j are different, and also if either arr_sta_i or dep_sta_j is a station capable of performing an inspection (we assume that this station is adjacent to the rolling stock depot).

1. Case where an inspection is performed at arr_sta_i the same day
   If the following condition is satisfied, the arc is set as a same-day arc.

$$\max\{arr\_time_i, ins\_min\_time\} + ins\_time \leq \min\{dep\_time_j - emp\_time_{ij}, ins\_max\_time\} \quad (4)$$

   Judge if an inspection is possible within the same day.

2. Case where an inspection is performed at arr_sta_i the next day
   If the following condition is satisfied, the arc is set as a next-day arc.

$$\max\{arr\_time_i, ins\_min\_time\} + ins\_time > \min\{dep\_time_j - emp\_time_{ij}, ins\_max\_time\} \wedge ins\_time\_min + ins\_time \leq dep\_time_j - emp\_time_{ij} \quad (5)$$

   Here an inspection can be started at ins_time_min the next day.

3. Case where an inspection is performed at dep_sta_j the same day
   If the following condition is satisfied, the arc is set as a same-day arc.

$$\max\{arr\_time_i + emp\_time_{ij}, ins\_min\_time\} + ins\_time \leq \min\{dep\_time_j, ins\_max\_time\} \quad (6)$$

   Judge if an inspection is possible within the same day.

4. Case where an inspection is performed at dep_sta_j the next day
   If the following condition is satisfied, the arc is set as a next-day arc.

$$\max\{arr\_time_i + emp\_time_{ij}, ins\_min\_time\} + ins\_time > \min\{dep\_time_j, ins\_max\_time\} \wedge ins\_time\_min + ins\_time \leq dep\_time_j \quad (7)$$

   Here empty run is inserted on the same day the train arrives at arr_sta_i and then an inspection can be started at ins_time_min the next day.

3.2 Modelling as a Travelling Salesman Problem

Since several arcs in type 1 to 4 are set between nodes, the network generally includes multiple arcs between nodes. The model of Giacco et al. tries to find an optimal Hamiltonian path on the network in such a way that maintenance requirements are satisfied. Note that the problem becomes a generalized TSP on the (directed) network with multiple arcs. With regard to performance measures, the roster length could be measured by assigning a weight of 1 to the arc when the transition between nodes corresponds to date change; 0, otherwise. Similarly, the empty distance could be measured by assigning an empty distance to the arc when the transition between nodes includes an empty run.

Figure 4 shows a network for the services of five trains and the nodes correspond to these trains. Four possible types of arcs exist between a pair of nodes, and only feasible arcs are drawn.

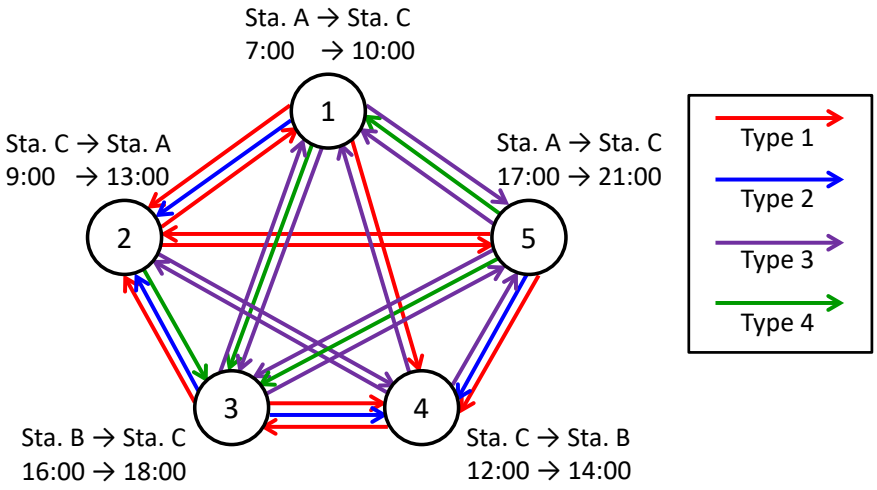

Figure 4: An example of the network model

3.3 Model Extensions for Splitting and Combining

Basic Idea

In our problem, there are two different types of trains, namely, single-unit trains and double-unit trains. This is modeled within the framework of the Giacco model by "dualizing" nodes for the double-unit trains. That is, two distinct nodes are prepared for those double-unit trains.

Whether splitting and/or combining operations are needed is partially judged by examining two consecutive nodes on the selected tour. If two adjacent nodes on the tour are both single-unit trains, then there is no shunting operation between them, but if the adjacent nodes call for a different number of train-sets, at least one shunting operation will definitely be needed between them. However, when the transition goes from one double-unit train to another double-unit train, it is possible that splitting and combing operations would be needed despite the fact that the same composition of two train-sets of the previous train may remain the same for the next train without shunting operations.

Train Position

In order to consider splitting and combining, we now define what we call "train position", which is sometimes simply called "position". The train position of a single-unit train is defined to be 0, with the train position of a double-unit train being 1 and 2. Here, the train position of a double-unit train represents an absolute geographic position. For example, the east side represents 1, and the west side 2, if the line extends in the east-west direction. Then, the train position of east-bound double-unit train 1M will be 1 for the front half of the train-sets, and 2 for the rear half. On the other hand, the train position of west-bound double-unit train 2M will be 2 for the front half of the train-sets, and 1 for the rear half. Assigning train positions allows us to judge the existence of splitting and combining during connections of double-unit trains. In the above example, if train 1M connects to

train 2M with the same train positions, no splitting and combining would occur as the connection keeps the same train composition. However, if train positions change after the connection from 1M to 2M in such a way that the positions are reversed (i.e., 1→2, 2→1), splitting is required first, then reversing the order of two train-sets, and finally combining the two train-sets together, thus requiring one splitting and one combining.

In our modification of the original Giacco model to consider splitting and combining, the network model is revised in such a way that each double-unit train is represented by two nodes, as mentioned above. In particular, for each double-unit train, we prepare one node with train position 1, and another node with train position 2. On the other hand, each single-unit train is represented by a single node with train position 0, just like the original Giacco model.

3.4 Counting the Amount of Splitting and Combining Based on Train Positions of Connection

Following the basic approach described above, we explicitly count in our model the amount of splitting and combining by judging the existence or non-existence of splitting and combining based on the information of train positions of trains before and after connections.

To do so, we separate the case of splitting from the case of combining. From a network viewpoint, this corresponds to separating considerations of the predecessor node of an arc from considerations of the successor node of the arc.

Splitting

(A) Connection from train position 0
This corresponds to a connection from a single-unit train, and therefore there is no splitting.

(B) Connection from train position 1
Here situations differ depending on the train positions after the connection.

(B1) Connection to train position 0
This case corresponds to a connection from a double-unit train to a single-unit train, so splitting occurs. We thus increment the number of splittings by one.

(B2) Connection to train position 1
This is the connection from position 1 to position 1, so there is no splitting, provided that the train-set of position 2 (before connection) connects to the same train as the train of position 1 (after connection). However, if the train-set of position 2 connects to a different train from the train of position 1 (after connection), splitting will be required. Therefore, the number of splittings will be either 0 or 1 depending on the status of position 2.

(B3) Connection to train position 2
The trains before and after connection are both double-unit trains in this case. As described in Section 3.3, whenever a train position changes, splitting is always required, so we increment the number of splittings by one.

(C) Connection from train position 2
For each train with position 2, there is always the same train with position 1. When splitting occurs, it will be counted in case B, above, for position 1. To avoid double counting, we assume that no splitting occurs for a connection from train position 2.

Combining

(D) Connection to train position 0
This corresponds to a connection to a single-unit train, so there is no combining.

(E) Connection to train position 1
Here situations differ depending on the train positions before the connection.

(E1) Connection from train position 0
This case corresponds to a connection from a single-unit train to a double-unit train, so combining occurs. We increment the number of combinings by one.

(E2) Connection from train position 1
This is the connection from position 1 to position 1, so there is no combining, provided that the train-set of train position 2 (after connection) connects from the same train as the train of position 1 (before connection). However, if the train-set of position 2 connects from a different train from the train of position 1 (before connection), combining will be required. Therefore, the number of combinings will be either 0 or 1 depending on the status of position 2.

(E3) Connection from train position 2
The trains before and after connection are both double-unit trains in this case. As described in Section 3.3, whenever a train position changes, combining is always required, so we increment the number of combinings by one.

(F) Connection to train position 2
For each train with position 2, there is always the same train with position 1. When combining occurs, it will be counted in case E above for position 1. To avoid double counting, we assume that no combining occurs for a connection from train position 2.

Application to Our Model

What is described above will be combined with the network model of Section 3.1 to derive the modified model, as will be described in detail below:

1. The case of B1, B3, E1, and E3
   Add cost to the corresponding arc.
2. The case of B2 and E2
   Existence or non-existence of splitting and combining depends on position 2, so the amount of splitting and combining is counted based on the position 2 after the connection. In order to count splitting and combining, we adopt several logical conditions. The details of this method will be described in the next section.

Following the basic approach described above, we explicitly count in our model the amount of splitting and combining by judging the existence or non-existence of splitting and combining based on the information of positions of trains before and after connections.

3.5 MILP Formulation

We now formulate the MILP problem. Notations for the MILP formulation are described in Table 2. Constraints can be classified into the four categories: assignment constraints, subtour elimination constraints, inspection constraints, forcing constraints for splitting and combining, and other constraints.

Table 2. Notation for MILP formulation

Notation	Definition
V	Set of nodes (Set of trains i, Index ranges over $0,1,\dots,\lvert V\rvert-1$)
V^1	Set of nodes with train position 1
A	Set of arcs
A^1	Set of arcs with no empty run and no inspection
A^2	Set of arcs with inspection and no empty run
A^3	Set of arcs with empty run and no inspection
A^4	Set of arcs with empty run and inspection
K	Set of arc types
c_{ij}^k	1 if the date changes when the arc from node i to node j of type k is selected, and 0 otherwise
d_{ij}^k	Distance of an empty run between nodes i,j of type k (will be positive when k is 3 or 4, and 0 when k is 1 or 2)
e_{ij}^k	Additional cost for splitting and combining between nodes i,j of type k
l	Lower limit of the inspection interval (in days)
m	Upper limit of the inspection interval (in days)
p_i	Node with train position 2 which shares the same train as node i with train position 1
x_{ij}^k	1 if the arc between nodes i,j of type k is selected, and 0 otherwise
y_{ij}	Order of the arc between nodes i,j on the selected tour
z_{ij}^k	Number of days since previous inspection of the arc between nodes i,j of type k on the selected tour
r_i	1 if connection from node i with train position 1 requires splitting, 0 otherwise
s_i	1 if connection to node i with train position 1 requires combining, 0 otherwise

Assignment Constraints

$$\sum_{k\in K}\sum_{j:(i,j,k)\in A} x_{ij}^k = 1,\ \ \forall i \in V \tag{8}$$

$$\sum_{k\in K}\sum_{i:(i,j,k)\in A} x_{ij}^k = 1,\ \ \forall j \in V \tag{9}$$

Equation (8) ensures that only one arc which emanates from node i is selected, and equation (9) ensures that only one arc which enters node j is selected.

Subtour Elimination Constraints

$$\sum_{j\in V} y_{ij} = \sum_{h\in V} y_{hi} + 1,\ \ \forall i \in V/\{0\} \tag{10}$$

$$y_{ij} \le |V| \sum_{k\in K} x_{ij}^k,\ \ \forall i \in V, \forall j \in V \tag{11}$$

$$\sum_{j \in V} y_{0j} = 1 \tag{12}$$

Subtours of the TSP would be eliminated by equations (10) to (12).

Inspection Constraints

$$\sum_{k \in K} \sum_{j:(i,j,k) \in A} z_{ij}^k = \sum_{k \in K} \sum_{j:(i,j,k) \in A^1 \cup A^3} \left(c_{ij}^k x_{ij}^k + z_{ij}^k\right) + \sum_{k \in K} \sum_{j:(i,j,k) \in A^2 \cup A^4} c_{ij}^k x_{ij}^k, \forall i \in V \tag{13}$$

$$l x_{ij}^k \leq z_{ij}^k, \qquad \forall (i,j,k) \in A^2 \cup A^4 \tag{14}$$

$$m x_{ij}^k \geq z_{ij}^k, \qquad \forall (i,j,k) \in A \tag{15}$$

The number of days since a previous inspection on a tour is calculated by equation (13). The first term corresponds to the case without inspection, and the second term the case with inspection. Lower and upper limits of the inspection interval are ensured by equations (14) and (15), respectively. With equations (13) to (15), inspections will be performed within the specified inspection intervals.

Forcing Constraints for Splitting and Combining

We now create logical conditions which say that a splitting would occur if for each double-unit train, position 1 is connected, and also if position 2 is not connected to the identical train, splitting will occur.

$$\sum_{k \in K} x_{ij}^k = 1 \ \wedge \ \sum_{k \in K} x_{p_i p_j}^k = 0 \longrightarrow r_i = 1, \qquad \forall i \in V^1, \forall j \in V^1 \tag{16}$$

The above logical conditions will be transformed into the following linear constraints:

$$\sum_{k \in K} x_{ij}^k - \sum_{k \in K} x_{p_i p_j}^k \leq r_i, \qquad \forall i \in V^1, \forall j \in V^1 \tag{17}$$

The case of combining would be similar to that of splitting. Logical conditions which say that combining would occur if for each double-unit train, position 1 is connected, and also if position 2 is not connected to the identical train, combining will occur.

$$\sum_{k \in K} x_{ji}^k = 1 \ \wedge \ \sum_{k \in K} x_{p_j p_i}^k = 0 \longrightarrow s_i = 1, \qquad \forall i \in V^1, \forall j \in V^1 \tag{18}$$

The above logical conditions will be transformed into the following linear constraints:

$$\sum_{k \in K} x_{ji}^k - \sum_{k \in K} x_{p_j p_i}^k \leq s_i, \qquad \forall i \in V^1, \forall j \in V^1 \tag{19}$$

Other Constraints

Other constrains are as follows.

$$x_{ij}^k \in \{0,1\}, \qquad \forall (i,j,k) \in A \tag{20}$$

$$y_{ij} \geq 0, \text{integer}, \qquad \forall i \in V, \forall j \in V \tag{21}$$

$$z_{ij}^k \geq 0, \qquad \forall (i,j,k) \in A \tag{22}$$

Objective Function

Under constraints (8) to (15), (17), and (19) to (22), the following objective function is minimized:

$$\alpha \sum_{(i,j,k)\in A} c_{ij}^{k} x_{ij}^{k} + \beta \sum_{(i,j,k)\in A} d_{ij}^{k} x_{ij}^{k} + \gamma \left\{ \sum_{(i,j,k)\in A} e_{ij}^{k} x_{ij}^{k} + \sum_{i\in V^{1}} r_i + s_i \right\} \tag{23}$$

The first term in equation (23) indicates the roster length (the number of days required to complete the roster, which is equivalent to the number of train-sets required to perform the roster), and the second term shows the total distance of empty runs. The third and fourth terms mean the amount of splitting and combining. The third one indicates the sum of additional costs for splitting and combining in the case of B1, B3, E1, and E3 described in Section 3.4. The fourth one is the sum of splitting and combining, which is calculated by the logical conditions (16) and (18) in the cases of B2 and E2 described in Section 3.4. Here, α, β, and γ are weight parameters.

4 Case Study

4.1 Lines and Settings

The proposed methodology is evaluated based on real instances of several lines in Japan with roughly 100 to 200 train services. Table 3 shows the details of each railway line, namely, the number of trains (sum of the single-unit trains and the double-unit trains), the number of the double-unit trains, the numbers of nodes and arcs in the network, together with the line length. The four lines are designated as A, B, C, D and for Lines B and C, problem instances with the reduced numbers of nodes by fixing some more or less obvious connections are added, namely Instances B-133 and B-147 for the original Instance B-161, and Instance C-177 for the original Instance C-256. In total, seven instances are tested. The numbers of trains range roughly from 100 to 200 which are typical in Japanese railways, and splitting and combining are performed in all these lines. Even though Line D involves fewer trains than Line C, the number of nodes of Instance D-258 is slightly more than those of Instance C-256 because Line D has more double-unit trains.

In our experiments, the proposed approach was evaluated based on the roster length, the total distance of empty runs, the amount of splitting and combining, and the computational time. We also analysed how solutions change when parameter weights are adjusted. The proposed algorithm was tested on a PC with Windows 10 Professional (64 bit), Core i7-8700K, and 64 GB RAM. In addition, Gurobi Optimizer 8.1.0 was used to solve the MILP model.

Table 3: Details of actual railway lines

Instance	No. of trains	No. of double-unit trains	No. of nodes	No. of arcs	Line length (km)
A-121	89	32	121	20,173	148.6
B-133	114	19	133	24,743	112.8
B-147	128	19	147	30,025	112.8
B-161	142	19	161	35,021	112.8
C-187	153	34	187	49,622	206.1
C-256	217	39	256	88,370	206.1
D-258	178	80	258	83,434	197.5

4.2 Results of Computational Experiments

Table 4 summarizes results of our computational experiments. Weight parameters were set at $\alpha = 1$, $\beta = 0.001$, and $\gamma = 0.01$ in these experiments. Computations were terminated after the maximum CPU time of 10,800 seconds, and the best results obtained at termination were shown if the maximum CPU time was reached before optimality.

Table 4 indicates that the optimal solutions were obtained except for Instance D-258. The total distances of the empty runs also appear to be reasonably small compared to line lengths. The number of splittings and combinings were either 8 or 10, which are small if we consider the numbers of the double-unit trains. Generally, good practical schedules were judged to be obtained for all instances. Instance D-258, however, could not be solved to optimality after the time limit of 10,800 seconds, even though the solution quality seemed to be good enough. Considering the fact that Instance D-258 required substantially more CPU time than Instance C-256 whose number of nodes and arcs are comparable to those of Instance D-258, it appears that the more double-unit trains we have, the more difficult is solving the instance, provided that the number of nodes is approximately the same.

Tables 5 and 6 show the effects of changing weight parameters β and γ (by fixing $\alpha = 1$) for bigger Instances C-256 and D-258, respectively. Since the roster length is constant regardless of whether the parameter is β or γ, the roster length is considered to be minimized in each instance. Under the weight parameter $\gamma = 0$, the amount of splitting or combining jumps up unreasonably high, which implies that in order to obtain practically reasonable results, counting and reducing the amount of splitting and combining must be included in the model mechanisms. We found that raising the value of γ even just a little reduced the amount of splitting and combining, and making β relatively larger is more effective. It should also be noted that optimal solutions are obtained quickly when $\gamma = 0$, but as γ is increased above zero, optimality could not be reached within the set CPU time limit in many instances. This could be attributed to the fact that logical conditions were introduced into the model to count the amount of splitting and combining, which makes the problem difficult to solve. Considering the fact that practically good upper bounds are obtained, CPU times could be reduced by improving the LP lower bounds with the generation of effective cuts.

The results in Tables 5 and 6 indicate that reduction of splitting and combining would be essential to obtain a reasonable roster. Otherwise, the resultant rosters would be unrealistic and may not be practically acceptable due to too many splittings and combinings. Computational feasibility is confirmed for the range of instances tested even

though CPU time increases when we consider reduction of splitting and combining operations.

Table 4. Results of computational experiments

Instance	Roster length	Empty run (km)	No. of splittings and combinings	CPU time (sec.)
A-121	17	39.5	8	67
B-133	13	4.8	8	49
B-147	13	4.8	8	102
B-161	13	4.8	8	94
C-187	22	75.4	10	1,845
C-256	22	75.4	10	3,254
D-258	29	131.6	8	*10,800

* indicates termination before optimality due to maximum time limit.

Table 5. Influence of change of the weight parameters (Instance C-256)

β	γ	Roster length	Empty run (km)	No. of splittings and combinings	CPU time (sec.)
0	0	22	3257.2	76	819
0	0.01	22	1506.8	8	*10,800
0.001	0	22	75.4	68	804
0.001	0.00001	22	75.4	10	*10,800
0.001	0.0001	22	75.4	10	*10,800
0.001	0.01	22	75.4	10	3,254
0.001	0.1	22	252.4	6	*10,800

* indicates termination before optimality due to maximum time limit.

Table 6. Influence of change of weight parameter (Instance D-258)

β	γ	Roster length	Empty run (km)	No. of splittings and combinings	CPU time (sec.)
0	0	29	5335.6	140	1,082
0	0.01	29	5368.8	2	*10,800
0.001	0	29	131.6	134	778
0.001	0.00001	29	131.6	10	10,675
0.001	0.0001	29	131.6	8	*10,800
0.001	0.01	29	131.6	8	*10,800
0.001	0.1	29	192.8	4	*10,800

* indicates termination before optimality due to maximum time limit.

5 Conclusions

In this paper, we focused on railway rolling stock rostering problems with maintenance considerations. Splitting and combining are used to adjust transportation capacity flexibly in Japanese railways. On the other hand, it is desirable that the amount of splitting and combining be minimized because these shunting operations require time and manpower. This paper proposes an MILP model based on a TSP with multiple arcs between nodes so that the amount of splitting and combining is reduced. Numerical experiments based on actual lines in Japan show that the proposed model incorporating the mechanisms to count

and reduce the number of shunting operations can generate practically good rosters with a reduced number of splittings and combinings. Computational feasibility is confirmed for the range of instances tested even though CPU time increases when we consider reduction of splitting and combining.

Possible future work includes:

- reduction of CPU time
- extensions of the model to the cases where three or more train-sets are assigned to some trains

Reduction of CPU time could be achieved by improving LP lower bounds with the generation of effective cuts. Considerations of trains to which three or more train-sets are assigned, may be included into the model by expanding the idea of train positions, but logical conditions would be expected to become very complicated which may increase CPU burden, so alternative approaches may also need to be considered.

References

Abbink, E., Van den Berg, B., Kroon, L., Salomon, M., 2004. "Allocation of railway rolling stock for passenger trains", *Transportation Science*, vol. 38, pp. 33-41.

Alfieri, A., Groot, R., Kroon, L., Schrijver, A., 2006. "Efficient circulation of railway rolling stock", *Transportation Science*, vol. 40, no. 3, pp. 378-391.

Borndörfer, R., Reuther, M., Schlechte, T., Waas, K., Weider, S., 2016. "Integrated optimization of rolling stock rotations for intercity railways", *Transportation Science*, vol. 50, pp. 863-877.

Giacco, G, L., D'Ariano, A., Pacciarelli, D., 2014. "Rolling stock rostering optimization under maintenance constraints", *Journal of Intelligent Transportation Systems: Technology, Planning, and Operations*, vol. 18, pp. 95-105.

Fioole, P.-J., Kroon, L., Maroti, G., Schrijver, A., 2006. "A rolling stock circulation model for combining and splitting of passenger trains", *European Journal of Operational Research*, vol. 174, pp. 1281-1297.

Morooka, Y., Fukumura, N., Takayuki, S., Imaizumi, J., Morito, S., 2017. "Rolling stock optimization based on the model of Giacco et al.: computational evaluation and model extensions", In: *Proceedings of 7th International Conference on Railway Operations Modelling and Analysis (RailLille2017)*, Lille, France.

Nishi, T., Ohno, A., Inuiguchi, M., Takahashi, S., Ueda, K., 2017. "A combined column generation and heuristics for railway short-term rolling stock planning with regular inspection constraints", *Computers and Operations Research*, vol. 81, pp. 14–25.

Peeters, M., Kroon, L., 2008. "Circulation of railway rolling stock: a branch-and-price approach", *Computers and Operations Research*, vol. 35, pp. 538-556.

Reuther, M., Schlechte, T., 2018. "Optimization of rolling stock rotations", In: Borndörfer, R., et al. (eds), *Handbook of Optimization in the Railway Industry*, International Series in Operations Research & Management Science 268, Springer, Switzerland.

Tsunoda, M., Imaizumi, J., Morito, S., 2015. "A model for estimating the required number of train units under split-and-merge policy for decision making in railways –a mathematical formulation by integer multi-commodity network flow-", In: *Proceedings of 6th International Conference on Railway Operations Modelling and Analysis (RailTokyo2015)*, Narashino, Japan.