# Railway Rolling-Stock-Assignment-Scheduling Algorithm for Minimizing Inspection Cost

Tomoe Tomiyama, Tatsuhiro Sato, Kenichirou Okada, Takashi Wakamiya, and Tomohiro Murata

*Abstract*—**Train maintenance regulation requires regular inspections of the rolling stock of railway companies. A planner who makes a schedule of rolling-stock operation should consider both assignments of rolling stock to train services and schedules of periodical maintenance. In Japan, the schedule is manually created for the middle-term, which is for half a month or full month. To reduce spent time to make the schedule, we developed an algorithm for rolling-stock-assignment schedule. To efficiently consider inspection cycles, the proposed algorithm creates assignment patterns of rolling stock which satisfy inspection cycles in advance. And then, the algorithm creates templates of schedule using the patterns, and assigns inspections to the templates. We verified the feasibility and scalability of the proposed algorithm using actual train operation data. The results indicate that the algorithm successfully generates a schedule which satisfies required regular inspection cycles within feasible calculation time.**

*Index Terms*— **Dijkstra method, inspection cycle, scheduling algorithm, rolling stock assignment**

## I. Introduction

TRIN maintenance regulation requires railway companies to regularly inspect their rolling stock at stated cycles in order to ensure safe transportation, and these cycles are based on running distances and inspection periods. Each railway company manages their rolling stock by creating schedules for rolling-stock assignments. These schedules define the assignment of train units to train services and the dates of inspections. In this paper, a train unit means combined vehicles that serve as one train and maintained together.

This scheduling requires much time because they are manually prepared without any decision-support system. The purpose of this research was to support a rolling-stock planner in quickly creating an assignment schedule.

There have been several studies on the problem of rolling-stock assignments without maintenance. The problem of train unit assignment considering combinations of train units was solved [1]-[5]. From another point of view, the usage of train units was minimized [6] and seat demand was considered [7]-[8].

There have been studies considering maintenance in the short term. The rolling-stock assignment and shunting scheduling were solved using two branch-and-bound algorithms [9]. Rolling-stock assignment was modeled as a problem of multi-commodity flow considering the cost of shunting tasks [10]. Combinations of each vehicle were considered [11]. Rolling stock assignment and maintenance scheduling were solved considering the addition of deadhead trains [12].

Whereas most studies were focused on short-term schedules of a few days without inspection cycles, [15] focused on a few days with inspection cycles, which can be applied to a middle- or long-term schedule. This study provides an efficient column-generation algorithm by relaxing constraints across train units.

In Japan, an assignment schedule is created for the middle-term, which is for half a month or full month. We developed an algorithm for a train-unit-assignment schedule with inspection cycles.

We describe the rolling-stock-assignment problem in Section II, present our approach and algorithm in Sections III and IV, respectively, explain the computational results in Section V, and give concluding remarks in Section VI.

## II. Rolling-Stock-Assignment Problem

### A. Scheduling for Rolling-Stock Assignment

Schedules for rolling-stock assignment are normally created in train units. A train unit means a set of combined vehicles that are not divided except in special cases such as breakdowns. The first step in assigning train units is to define sets of train services that are assigned the same train unit (train-circulation). After that, train units are respectively assigned to the train-circulation, and inspection dates are defined in regular cycles (rolling-stock assignment).

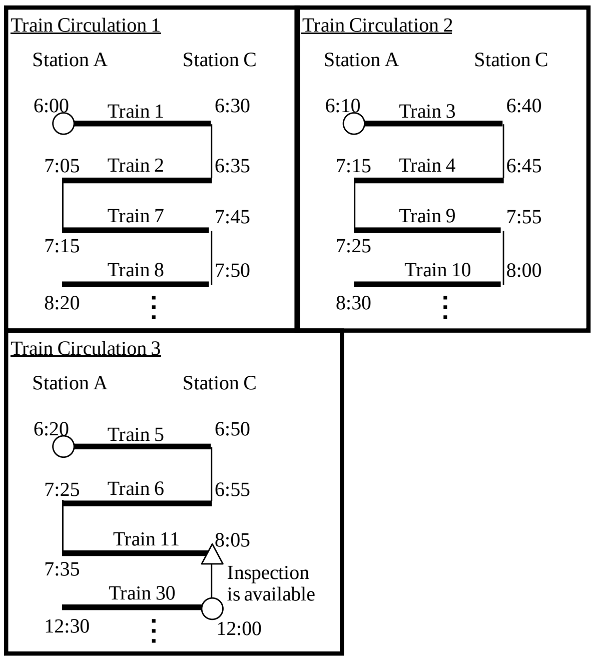
Whereas train-circulations are created in one day, rolling-stock assignments are created in a few days or more because inspection cycles are taken into account. The range of the schedule (planning horizon) depends on the operational policies. In most Japanese railways, except high speed trains, the range is a half a month or full month. Fig. 1 (a) shows an example of a train-circulation. The symbol of circle in the figure indicates the departure from a depot, and the symbol of triangle in the figure indicates the arrival to a depot. In this

Manuscript received October 11, 2017; revised March 26, 2018.

T. Tomiyama and T. Sato are with Hitachi, Ltd., Research & Development Group, 292, Yoshida-cho, Totsuka-ku, Yokohama-shi, 244-0817, Japan (e-mail: tomoe.tomiyama.rq@hitachi.com).

K. Okada and T. Wakamiya are Hitachi, Ltd., Social Infrastructure Information Systems Division, 6-27-18, Minamiooi, Shinagawa-ku, Tokyo, Japan.

T. Murata is with Graduate School of Information, Production and Systems, Waseda University, 2-7, Hibikino, Wakamatsu-ku, Kitakyushu-shi, Fukuoka 808-0315, Japan.



(a) Example of train-circulation

| | Sept. 1st | Sept. 2nd | Sept. 3rd | Sept. 4th | Sept. 31st |
|---|---|---|---|---|---|
| Train Unit A | T. C. 1 | T. C. 2 | T. C. 3 | T. C. 1 | T. C. 1 |
| Train Unit B | T. C. 2 | T. C. 3 | T. C. 1 | T. C. 2 | T. C. 2 |
| Train Unit C | T. C. 3 | T. C. 1 | T. C. 2 | T. C. 3 | T. C. 3 |

T. C. : Train Circulation

(b) Example of rolling-stock-assignment

Fig. 1. Schedule examples

case, train-circulation 1 includes trains 1, 2, 7, and 8, and this circulation involves departures and arrivals from station A. In the same way, train-circulation 2 includes trains 3, 4, 9, and 10, and the departure and arrival stations are station A. Train-circulation 3 includes train 5, 6, 11, and 30, and the departure and arrival stations are station A.

In train-circulation 3, a short stay at a depot is included among train 11 and train 30, and an inspection is available during this stay. Fig. 1 (b) shows an example of a rolling-stock-assignment for September. In this case, train unit A is sequentially assigned to train-circulations 1, 2, and 3 from the first day to the last day of the targeted planning horizon.

In theory, rolling-stock-assignment can be automatically created on the basis of the running and inspection results. In practice, it has to be manually managed because unplanned maintenance works cannot be avoided. To keep interactive with a planner, the scheduling algorithm of rolling-stock-assignment is required to quickly response.

## *B. Constraints of Train Connectivity and Inspections*

To create a rolling-stock-assignment schedule, a planner takes into account the following constraints.

(1) Connectivity between train-circulations

It is necessary to maintain consistency of place and time of train-circulations assigned to the same train unit to make a schedule executable. In general, train units are stored at depots or stations near the stations where the train units will depart from the next day. If the departure depot (or station) is different from the arrival depot (or station), a deadhead train is needed. This incurs additional operation cost.

(2) Inspection cycle

An inspection cycle is defined by laws. In our targeted railway company, the cycle is within five days for light inspections such as visual inspection. Regarding heavy inspections, such as overhaul, the date of inspection is designated by another schedule of the rolling-stock factory.

(3) Inspection capacity

The number of inspections in one day is limited because inspections require specific facilities and skilled maintainers. In our targeted railway company, to make it easier to consider the capacity of inspections, train-circulations in which inspections can be conducted are designated.

## *C. Literature*

The researches related to rolling-stock-assignment can be categorized to two types: which deals with train-circulation and rolling-stock-assignment together or which deals with them separately.

In the case of dealing with them together, it is required to consider constraints related to each train such as shunting works for turn-over and the number of vehicles. The algorithm integrating two branch-and-bound algorithms for rolling-stock-assignment and shunting schedule was provided [9]. The penalties against complex shunting works and/or delays caused by shortage of drivers were modeled as evaluation criteria [2] [10].

Combinations of each vehicle were considered [1] [8] [4] [11]. The constraints related to the number of vehicles for each train were modeled as a multi-commodity flow problem [1]. The same constraints were modeled as a mixed integer problem, and were solved by the heuristic algorithm using lagrangian relaxation algorithm and local search [8]. The positions of vehicles composing each train unit were modeled as a multi-commodity flow problem, and the branch-and-bound algorithm applied Dantzing-Wolfe decomposition was provided [4]. The required time to merge train units was considered by the algorithm using branch-and-bound method and column-generation method [11].

In addition, some researches consider the designed inspection dates [12]-[15]. The dates and the required time to execute inspections were modeled as a mix integer problem, and the algorithms using column-generation algorithm and lagrangian relaxation method were provided [12]-[14].

While these researches assumed that inspection dates were designed, [30] provided an algorithm for scheduling both of assignment of train units and assignment of inspections. In this algorithm, assignment of train units was defined as a sub-problem for column-generation algorithm, and was solved using the algorithm of searching the shortest path based on Bellman-Ford method.

As the case of dealing with train-circulation and rolling-stock-assignment separately, there is a research for long distance trains. The rule-based heuristic algorithm for assigning train units to train-circulations was provided [16]. This research defined a model of ideal cumulative mileage within one inspection cycle for efficiency. To keep real cumulative mileage of each train unit close to the ideal one, the algorithm calculated priorities of train-circulations for each train unit.

## III. Model and Formulation

The constraint of connectivity between train-circulations, which is mentioned in the previous section (constraint (1)), can be considered using a network model. In this model, a task, such as train-circulation, is represented as a node, and only nodes that maintain connectivity are linked by arcs.

Whereas the network model is useful for restricting relations between two nodes, it cannot restrict relations over some nodes such as an inspection cycle. To efficiently consider the connectivity and the inspection cycles, we solve the problem in two steps, i.e., creating templates of train-circulation assignment using the network model, and assigning inspections to the templates considering inspection cycles. The details of the proposed algorithm are given in the next section.

In this section, we explain the network model and mathematical formulation.

### A. *Network Model of Train Unit Operation*

We represent a rolling-stock assignment as a network model to efficiently consider the connectivity of circulations. As previously mentioned, a node in a network model represents a train-circulation. In addition, if a train unit is stocked at a depot for a whole day, this operation is represented as a node. Each node has four attributes: date of circulation, departure depot (or station) of circulation, arrival depot (or station) of circulation, and availability of inspections. In our targeted railway operation system, the availability of inspection is pre-defined in each train-circulation considering the depot capacity. The availability of inspection depends on the time interval of the arrival and departure times at the depot. Therefore, it is defined at the same time as creating a train-circulation.

The nodes are linked by arcs if the train-circulations represented by the nodes maintain connectivity, which is mentioned in Section II (1). Each arc has a weight that represents an order-pattern of train-circulation assignment. This order-pattern defines circulation sequences that are continuously assigned to the same train unit. The detail of the pattern is mentioned in the next section.

A train-unit assignment is represented as a path of the network model. Fig. 2 shows an example of this network model for schedules of September. Train-circulations on 1st September are represented by the left three nodes, train-circulations on 2nd September are represented by the next three nodes, and train-circulations on the other days are represented by the same way. The path indicated in Fig. 2 represents a schedule for train unit A. In the schedule, train unit A is assigned to train-circulation 1, 2, 3, and 1 from 1st September to 4th September.

### B. *Formulation of Rolling-Stock Assignment Problem*

Scheduling rolling-stock assignment consists of two problems:

problem 1) assigning train units to train-circulations,

problem 2) defining dates of train units' inspections.

Assignment of train units is modeled as a problem of extracting a set of paths from the network model and deciding 0-1 values of variables which represent train unit assignments to the extracted paths. Definition of inspections' dates is modeled as a problem of deciding 0-1 values of variables which represent inspection executions satisfying inspection cycles. The evaluation of this problem is minimizing inspection cost.

We defined these problems by using the following elements.

Parameters

$W_r$ : weight of arc r.

$C_k$ : cycle of inspection k.

$FC_{uk}$ : the first cycle of inspection k for train unit u. This parameter is defined depending on running and inspection results of each train unit.

$PC_{kn}$ : cost for executing an inspection k at the train-circulation represented by node n. When the inspection k is not executable at node n, the cost is set high.

Sets

$R$ : arc set in a network model.

$N$ : node set in a network model.

$I$ : date set included in the targeted planning horizon. The first day of the planning horizon is denoted as 0. The following days are sequentially numbered.

$K$ : type of inspection set.

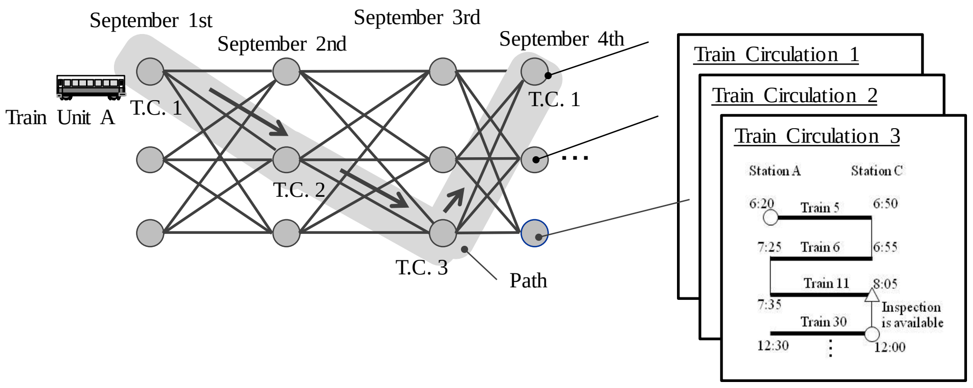


Fig. 2. Network model of train-unit operation (cited from [17])

$U$ : train unit set.

$J$ : path set. This set is created on the basis of the result of problem 1.

Variables

$a_{kn} \in \{0,1\}$: $a_{kn} = 1$ if inspection k is executable at node n, or $a_{kn} = 0$ if not.

$t_{in} \in \{0,1\}$: $t_{in} = 1$ if node n denotes a train-circulation at the i-th day, or $t_{in} = 0$ if not.

$o_{rn} \in \{0,1\}$: $o_{rn} = 1$ if node n is the origin of arc r, or $o_{rn} = 0$ if not.

$d_{rn} \in \{0,1\}$: $d_{rn} = 1$ if node n is the destination of arc r, or $d_{rn} = 0$ if not.

$s_n \in \{0,1\}$: $s_n = 1$ if node n denotes a train-circulation at the first day of the targeted planning horizon, or $s_n = 0$ if not.

$e_n \in \{0,1\}$: $e_n = 1$ if node n denotes a train-circulation at the last day of the targeted planning horizon, or $e_n = 0$ if not.

$p_{jn} \in \{0,1\}$: $p_{jn} = 1$ if node n is included in path j, or $p_{jn} = 0$ if not. This value is decided by the result of problem 1.

Decision variables

$x_r^j \in \{0,1\}$ : $x_r^j = 1$ if arc r is included in a solution for path j, or $x_r^j = 0$ if not.

$y_{kn} \in \{0,1\}$ : $y_{kn} = 1$ if inspection k is assigned to node n, or $y_{kn} = 0$ if not.

$z_{ju} \in \{0,1\}$ : $z_{ju} = 1$ if train unit u is assigned to path j, or $z_{ju} = 0$ if not.

The rolling-stock-assignment model is defined as follows.

Objective functions

$$\min \sum_{n\in N} \sum_{k\in K} y_{kn} PC_{kn} \tag{1}$$

A solution is evaluated by the total inspection cost. As the mentioned above, $PC_{kn}$ is set high if inspection k is not executable at node n. Therefore, if all nodes assigned inspections have availability of inspection, objective value is minimized.

Constraints

$$\sum_{j\in J} x_r^j \le 1 \qquad \forall r \in R \tag{2}$$

$$\sum_{j\in J} \sum_{r\in R} (o_{rn} x_r^j + e_n d_{rn} x_r^j) = 1 \qquad \forall n \in N \tag{3}$$

$$(\sum_{r\in R} d_{nr} x_r^j - \sum_{r\in R} o_{nr} x_r^j)\overline{e_n}\,\overline{s_n} = 0 \qquad \forall n \in N, \forall j \in J \tag{4}$$

$$p_{jn} = \sum_{r\in R} x_r^j (o_{rn} + e_n d_{rn}) \qquad \forall n \in N, \forall j \in J \tag{5}$$

$$\sum_{j\in J} z_{ju} = 1 \qquad \forall u \in U \tag{6}$$

$$\sum_{u\in U} z_{ju} = 1 \qquad \forall j \in J \tag{7}$$

$$\sum_{n\in N} z_{ju} s_n = 1 \qquad \forall j \in J \tag{8}$$

$$\sum_{n\in N} z_{ju} e_n = 1 \qquad \forall j \in J \tag{9}$$

$$\sum_i^{i+FC_{uk}} \sum_{n\in N} t_{in} p_{jn} y_{kn} - z_{ju} \ge 0 \qquad \forall i \in I, \forall k \in K, \forall j \in J, \forall u \in U \tag{10}$$

$$\sum_i^{i+C_k} \sum_{n\in N} t_{in} p_{jn} y_{kn} \ge 1 \qquad \forall i \in I, \forall k \in K, \forall j \in J \tag{11}$$

Equation (2) is the constraint restricting that each arc is covered by only one path.

Equation (3) is the constraint of the train-circulation coverage. It makes each node covered by only one path. When an arc is included in a solution, the nodes out-linked by the arc are included in a solution at the same time. However the last day of the targeted planning horizon, nodes have no out-linked arc. Therefore, Eq. (3) restricts the coverage of both of the origin nodes of all arcs and the destination nodes of arcs in the last day.

To make paths from the first day to the last day of the targeted planning horizon, Eq. (4) restricts the number of in-linked and out-linked arcs with a node to the same number of each other. The first nodes in the targeted planning horizon have only out-linked arcs, and the last nodes in the targeted planning horizon have only in-linked arcs. Therefore, the left term of Eq. (4) is multiplied by $\overline{s_n}$ and $\overline{e_n}$ to except the first nodes and the last nodes.

Equation (5) defines nodes included in the paths extracted as a solution.

Equation (6) is the constraint of the train unit coverage. Each train unit is restricted to be assigned to only one path.

Equation (7) is the constraint of the path coverage. Each path is restricted to be assigned to only one train unit.

Equations (8) and (9) are the constraints of the coverages of nodes in the first day and the last day of the targeted planning horizon. To make each path start from the first day and make it finish at the last day, this constraint makes each path include only one node respectively in the first day and the last day.

The constraint of an inspection cycle, which is mentioned in Section II (constraint (2)), is expressed by Eqs. (10) and (11). Eq. (10) is for the first inspection cycle. The fist cycle depends on the running and inspection results of each train unit. Therefore, Eq. (10) needs to be considered on the basis of the train unit assignment, that is controlled by the decision variable $z_{ju}$. Eq. (11) is for all inspection cycles.

The constraint of inspection capacity, which is mentioned in Section II (constraint (3)), is expressed by variable $a_{kn}$.

## IV. Proposed Algorithm

### A. Overview

We propose an algorithm for searching schedules for all train units using the network model mentioned in section III. The reason of using the network model is to apply order-patterns for efficient search. Order-pattern is created by using a kind of know-how of rolling-stock-assignment planner, it defines a partial order of train-circulations which satisfies an inspection cycle (see step 0 mentioned below).

Fig.3 shows the overview of the way to use the network model. The algorithm uses the network model in the following two phases.

[Phase1]: Process of Creating Templates of Schedule

The algorithm creates templates of schedule for each train unit by selecting feasible paths from the network model considering Eqs. (2)-(9). Template is a rough schedule for a train unit, it defines assignment of train unit to each train-circulation. To increase the possibility of success of the next phase (Phase 2), the order-patterns are reflected in the network model.

[Phase 2]: Process of Setting Inspections

The algorithm assigns inspections to the paths selected in phase 1 considering Eqs. (10) and (11) to satisfy required inspection cycles. The results of this phase are defined as a rolling-stock-assignment schedule.

If there are any violations of inspection cycles, a template is selected to be canceled. The network model is modified by pushing back the nodes included in the canceled template. And then, the procedures of phase 1 and 2 are repeated until a feasible solution, which has no violation of constraints mentioned in section II, is found.

### B. The Algorithm

The flow of the proposed algorithm is shown in Fig. 4. At the beginning of the flow, the network model is created (step 0). And the algorithm creates a schedule by searching partial solution for each train unit (steps 1-5). And then, to get a better solution, the algorithm repeats the steps 1-5 within time limit by randomly changing processing order (step 6).

The details of steps 0-5 are illustrated below.

[Step 0]: Process of Creating Order-patterns

In step0, the algorithm creates order-patterns by changing a part of an operation pattern. An operation pattern is a sequence of train-circulations satisfying inspection cycles, which is created together with train-circulations. In an operation pattern, more than one inspection is included at the interval of inspection cycle. The algorithm creates order-patterns by dividing an operation pattern by inspection cycle and partly changing the divided patterns while keeping satisfaction of inspection cycle. The detailed processes are the following.

Step A: Divides an operation pattern at the interval of inspection cycle, and defines them as basic patterns. Fig. 5 (a) shows an example of basic patterns. In this example, inspection cycle is three days. The sequence of train-circulations lined up from T.C.1 to T.C.9 is an operation pattern. Inspection is available at T.C.1, 3, 5, and 7.The sequences from T.C.1 to 3, from T.C.4 to 6, and from T.C.7 to 9 are respectively created as basic patterns (BPs).

Step B: Exchange train-circulations included in the basic patters among each other, and define them as order-patterns. The target of this exchange is only train-circulations with availability of inspection in order to keep satisfaction of

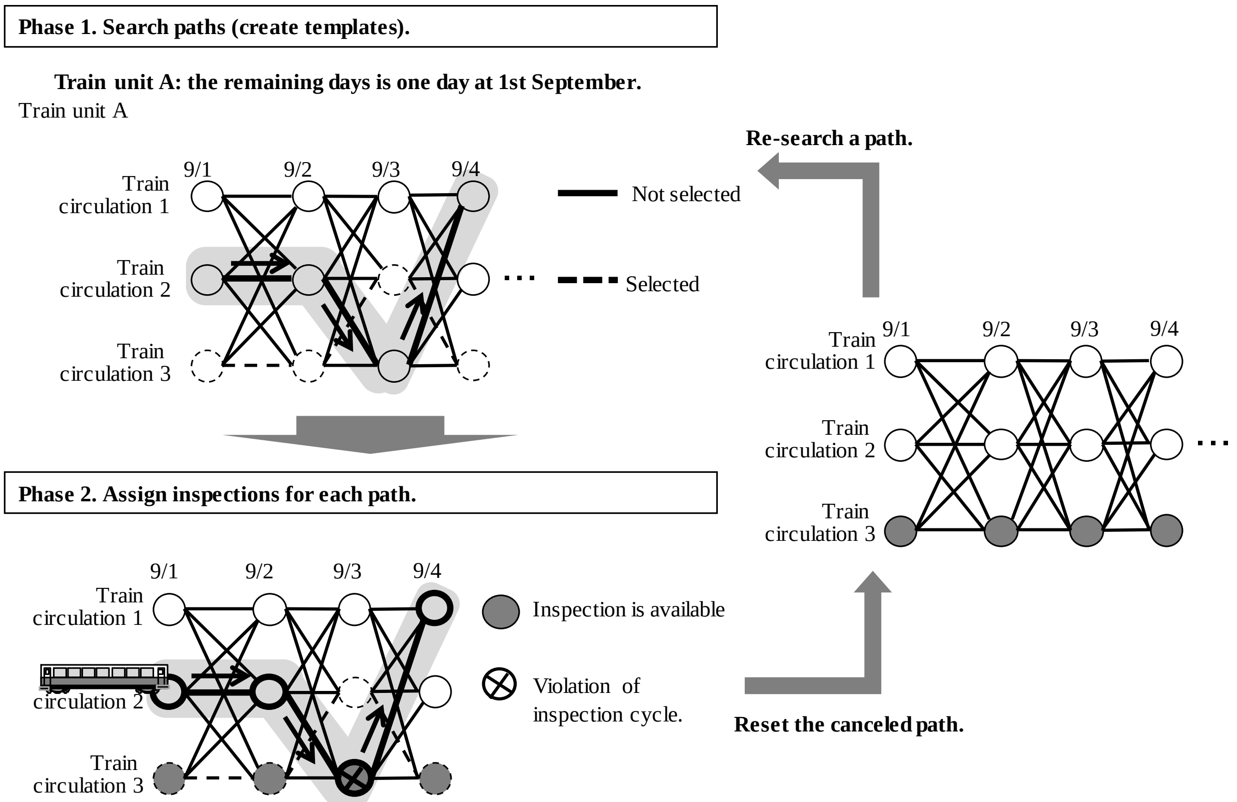

Phase 1. Search paths (create templates).
Train unit A: the remaining days is one day at 1st September.
Train unit A
9/1
9/2
9/3
9/4
Train circulation 1
Train circulation 2
Train circulation 3
Not selected
Selected
Re-search a path.
Phase 2. Assign inspections for each path.
Inspection is available
Violation of inspection cycle.
Reset the canceled path.

Fig. 3. Overview of the proposed algorithm

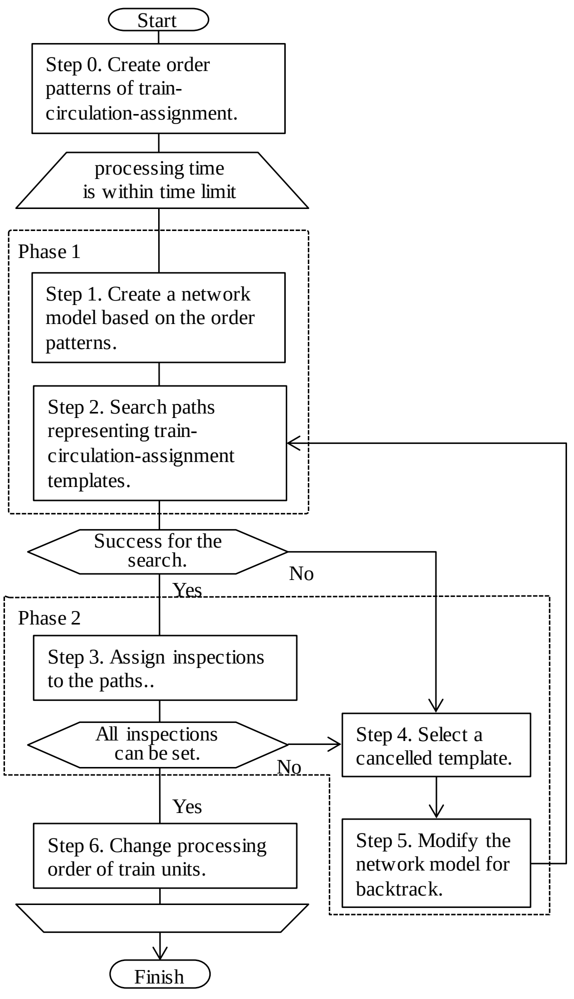

Start
Step 0. Create order patterns of train-circulation-assignment.
processing time is within time limit
Phase 1
Step 1. Create a network model based on the order patterns.
Step 2. Search paths representing train-circulation-assignment templates.
Success for the search.
No
Yes
Phase 2
Step 3. Assign inspections to the paths..
All inspections can be set.
No
Step 4. Select a cancelled template.
Yes
Step 6. Change processing order of train units.
Step 5. Modify the network model for backtrack.
Finish

Fig. 4. Flow of the proposed algorithm

inspection cycle. In addition, exchange is allowed only when consistencies of departure and arrival of train-circulations can be kept. Fig. 5 (b) shows an example of order-patterns. T.C.1 and T.C.3 in BP.1 are respectively exchanged with T.C.5 in BP.2 and T.C.7 in BP.3. The resulted patterns are respectively defined as order-patterns (order-pattern 2-9). By the same way, train-circulations with inspection availabilities included in BP.2 and BP.3 are exchanged with each other.

[Phase1]: Process of Creating Templates of Schedule

(Step1): Create the network model

The network model, which is mentioned in Section III, is created on the basis of train-circulation data, and weights of arcs are set on the basis of the order-patterns generated in step0.

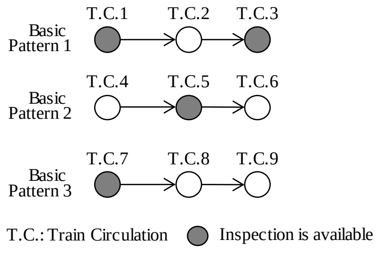
The algorithm sets small weights to the following arcs.

a) Arc that represents the same order of train-circulations with order-patterns. In the case of Fig.5, the arcs connecting T.C.1 and T.C.2, T.C.2 and T.C.3, T.C.4 and T.C.5, T.C.5 and T.C.6 and so on are set small weights.

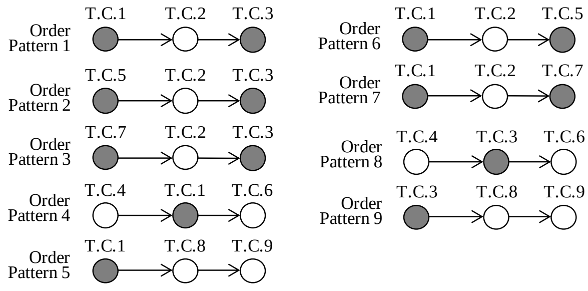
b) Arc between the last node of an order-pattern and the first node of the other order-pattern. In the case of Fig. 5, the arcs connecting T.C.3 and T.C.5, T.C.3 and T.C.7, T.C.4 and T.C.1 and so on are set small weights.

(Step 2): Create templates of schedule

Templates of schedule are created by searching the shortest paths from each node of the first day of targeted planning horizon considering constraints of connectivity between



(a) Example of basic patterns



(b) Example of order-patterns

Fig. 5 Example of patterns

train-circulations. Each path is individually searched by using the Dijkstra method in sequential order. The Dijkstra method is one of general algorithms for searching the shortest path from a network model. The method makes the searching process fast by memorizing partial solutions and using them to skip the same search processes.

To consider the constraints of coverages of nodes and arcs denoted in Eqs. (2) and (3), the network model is modified every time a path is searched. It means that the nodes included in the selected path are deleted from the network model to avoid being selected by the other train unit.

[Phase 2]: Process of Setting Inspections

(Step 3): Assign inspections

Inspections are assigned to the templates created in phase 1. The algorithm starts checking from the first node of the path searched in step 2 and sets inspections to the nodes with the availability of inspection. To minimize the number of added inspections, the algorithm sets inspections to the nodes that are the last nodes in each inspection cycle. When a violation of inspection cycle remains, the algorithm cancels a solution for other train units and backtracks to address the violation to change the templates created in the previous step (see steps 4 and 5 explained below).

(Step 4): Cancelation

When an inspection-cycle violation remains, the algorithm selects a path for cancelation previously searched as a partial solution in step 2.

(Step 5): Modify the network model

To backtrack beyond the phase, the algorithm modifies the network model and executes the processes from step 2 and 3 again. When a path is selected for cancelation in step 4, the nodes and arcs in the selected path are added to the network model to backtrack. This makes the nodes and arcs selectable for other paths again.

## V. NUMERICAL RESULTS

### A. *Evaluation Criteria*

We compared the proposed algorithm with the commercial Mixed-Integer Programing (MIP) solver Gurobi [18] from the following view points.

(1) Optimality: inspection cost for executing inspections, which is calculated by Eq. (1). To calculate inspection costs, we used 1.0 and 10.0 as cost values for inspection in pre-defined window and not in pre-defined window respectively.

(2) Stability: variance of inspection costs in several trails of the proposed algorithm. The proposed algorithm randomly changes the order of processing train units (see step 6 indicated in the previous section). So, the result of the algorithm could be different on each search. We repeated the algorithm several times, and verified the variance of the inspection costs of the searches.

(3) Scalability: spent time for searching the best solution. In the proposed algorithm, the time is for all processes shown in Fig. 4.

### B. *Data Set of Experiment*

The test data were based on two sets of actual data indicated in Table 1. The parameters of our model introduced in Section III are also indicated in Table 1. In addition, Table 2 shows problem sizes.

We used a Pentium 4 computer (3.2 GHz and 2 GB). The programming language was C++ for the proposed algorithm. Gurobi was processed on a web server with Intel Xeon CPU (3.4GHz, 4 cores, and 20GB).

### C. *Results*

We executed the proposed algorithm ten times using respectively data of Case 1 and Case 2. The limit time of each execution was six hours for Case 1, and 24hours for Case 2.

In the beginning, we indicate one of results of Case 1. After that, the numerical summary of the results are described on the basis of the evaluation criteria mentioned above.

Table 3 shows one of the schedules created by the proposed algorithm. Though some train units were sequentially assigned to some train-circulations with inspection availability, inspections were assigned to only some of them. To minimize the inspection cost, it is desirable that the interval of inspections is close to the inspection cycle. In the result described in Table 3, all intervals were four or five days, which were close to the inspection cycle.

In this case, the basic order-pattern consisted of from T.C. 1 to T.C.7. The T.C.10 was irregular, and was only set in day 1 and day 2. The order-patterns were underlined using dashed line. For example, train unit 0 was sequentially assigned to train-circulations 1, 2 and 3 as with the order-patterns. While the assignments of train units mostly kept to the order-patterns, there were some disconnections of the order-patterns, such as days 5, 6 and 8. It was probably caused by the buffers included in the order-patterns. In the

Table 1. Data sets and parameters

(a) Data Sets

| Data No. | Size of Depot | Number of Train Units | Planning Horizon [days] | Ratio of Designated Inspections [%] |
|---|---|---|---|---|
| Case1 | small | 10 | 16 | 1.88 |
| Case2 | large | 67 | 28 | 4.53 |

(b) Parameters

| Inspection Cycles [days] | Weights of Arcs | |
|---|---|---|
| | Included in Patterns | not Included in Patterns |
| 5 | 1 | 1000 |

Table 2. The number of variables

| | Variable x | Variable y | Variable z | Variable p | Total |
|---|---|---|---|---|---|
| Case 1 | 480 | 160 | 100 | 1600 | 2340 |
| Case 2 | 15596 | 1876 | 4489 | 125692 | 147653 |

Table 3.The schedule created by the proposed algorithm for Case 1

| | Date | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| Train Unit 0 | 10 | **1** | 2 | 3 | 00 | 00 | **00** | 00 | 00 | 00 | 00 | **00** | 00 | 00 | 00 | 00 |
| Train Unit 1 | 2 | 6 | 7 | **1** | 7 | 6 | 7 | **1** | 7 | 1 | 2 | 3 | **00** | 00 | 00 | 00 |
| Train Unit 2 | 5 | 10 | 00 | **00** | 4 | 5 | 6 | 2 | **1** | 2 | 6 | 7 | **1** | 2 | 3 | 00 |
| Train Unit 3 | 1 | 7 | **1** | 2 | 6 | 2 | **1** | 7 | 6 | 7 | 3 | **4** | 5 | 1 | 2 | 1 |
| Train Unit 4 | 00 | 00 | **00** | 00 | 00 | 00 | 00 | **00** | 00 | 00 | 00 | 00 | **00** | 00 | 00 | 00 |
| Train Unit 5 | 3 | **4** | 5 | 6 | 2 | **1** | 2 | 6 | 2 | 3 | **4** | 5 | 6 | 7 | **1** | 2 |
| Train Unit 6 | 7 | 3 | **4** | 5 | 1 | 7 | 3 | **4** | 5 | 6 | 7 | **1** | 7 | 6 | 7 | 6 |
| Train Unit 7 | **4** | 5 | 6 | 7 | 3 | **00** | 4 | 5 | 3 | **4** | 5 | 6 | 2 | 3 | **4** | 5 |
| Train Unit 8 | 6 | 2 | 3 | **00** | 00 | 4 | 5 | 3 | **00** | 5 | 1 | 2 | 3 | **4** | 5 | 3 |
| Train Unit 9 | 00 | 00 | 00 | **4** | 5 | 3 | 00 | 00 | **00** | 00 | 00 | 00 | **4** | 5 | 6 | 7 |
| Underline: with inspeciton availability, **Bold**: assigned inspection, 00: operation as a spare | | | | | | | | | | | | | | | | |

Table 4. Result in Case 1

| | Inspection Cost | Inspection Cost (Variance) | Number of Inspections in Pre-defined (Average) | Number of Inspections not in Pre-defined (Average) | Rate of Inspections in Pre-defined [%] | Convergence Time [seconds] |
|---|---|---|---|---|---|---|
| Proposed | 32.5 | 0.25 | 32.5 | 0.0 | 100.0 | 3066.0 |
| Gurobi | 50.0 (Interim) | - | - | - | - | ≥ 86400 |

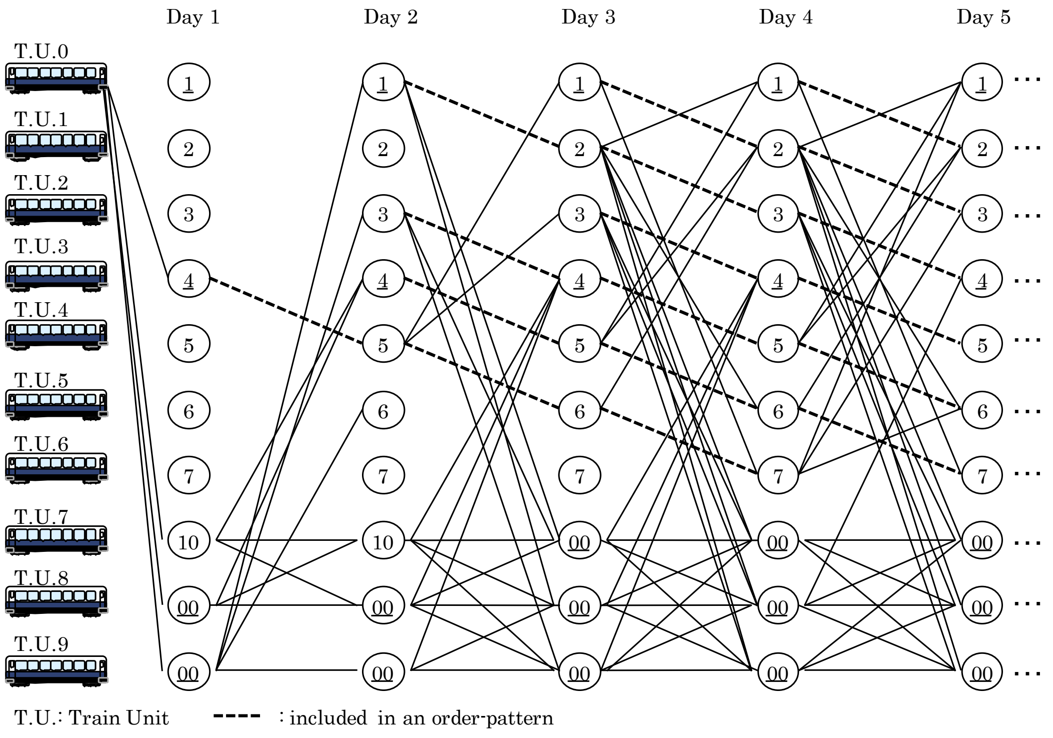


Fig.6 A part of the network model for the schedule shown in Table 3

order-patterns, only T.C.1 and T.C.4 had inspection availabilities. Whereas the intervals between them, which were "from T.C.1 to T.C.4" and "from T.C.4 to T.C.1", were three and four, the inspection cycle was five days. Therefore, if we assigned train units to train-circulations using the order-patterns, the intervals of inspections were three and four days. It were shorter than inspection cycles. To minimize inspection cost, the proposed algorithm created the schedule as partially disconnecting the sequential of the order-patterns.

Figure 6 shows a part of the network model that created for calculating the schedule of Train Unit 0 in the same case of Table 3. The arcs between the train unit and the nodes of the first day were created, only if the arcs kept consistency with the place where the train unit was stored at the day before the first day. Train unit 0 was assigned to T.C.10 and T.C.1 at day 1 and day 2 respectively. Since the arcs from train unit 0 to the nodes of the day 1 were weighted equally, the arcs were selected in order of the identity. In the same way, the arc from the node of day 1 to the node of day 2 was selected. At day 3, the weight of the arc linked from the node of T.C.1 to the node of T.C.2 was smaller than the ones of the other arcs, because the arc was a part of order-pattern. Therefore the arc was selected. In this way, the schedules of the train units were created.

Next, the numerical summary of the results are described.

(1) Optimality: Table 4 shows comparison between the result of the proposed algorithm and the result of Gurobi in Case 1.

The results of the proposed algorithm are average of ten trails. Gurobi could not finish searching an optimal solution within feasible time, which is 24 hours. So, the result of Gurobi described in Table 4 is about an interim solution.

In Table 4, the proposed algorithm searched a better solution, in which the inspection cost was 35.0 percent lower, than Gurobi. In the proposed algorithm, all inspections were scheduled in pre-defined window. It means all nodes assigned inspections have availability of inspection, which is pre-defined (see section III). The detailed result of Gurobi was unknown because the result was interim and the detailed results were not output in the process of calculation.

(2) Stability: In Table 4, the proposed algorithm could get solutions within 0.25 variances in cases 1. Furthermore, Figure 7 shows a convergence process of case 1. Gurobi got an interim solution with inspection cost 50.0 at the beginning of the process. After that, it could not get the better solution. In

Table 5. Result in Case 2

| | Inspection Cost | Inspection Cost (Variance) | Number of Inspections in Pre-defined (Average) | Number of Inspections not in Pre-defined (Average) | Rate of Inspections in Pre-defined [%] | Convergence Time [seconds] |
|---|---|---|---|---|---|---|
| Proposed | 609.1 | 3.08 | 430.3 | 429.1 | 96.0 | 55946.4 |

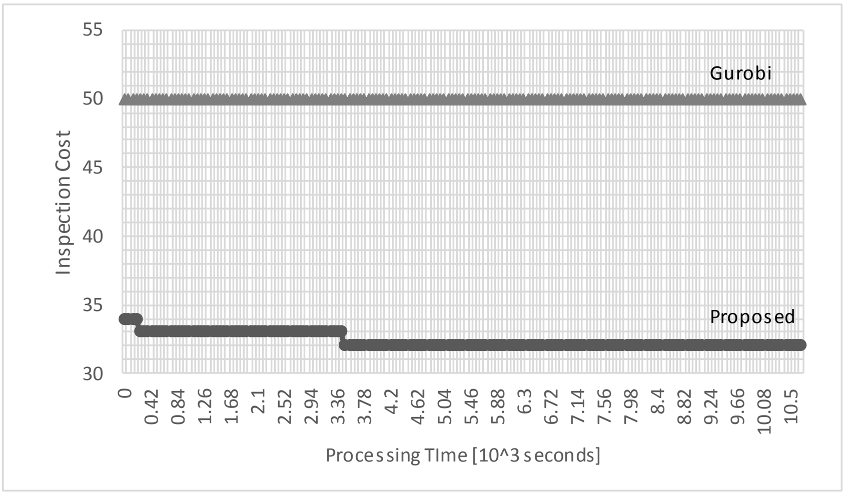


Fig.7 Searching Process (Case1)

the case of the proposed method, it could got a solution near to the best solution, with 106.3 percent of the best solution's cost mentioned above, at the beginning of processing time within one minute. In addition, the method could get the best solution within one hour. These results indicate our method can get a solution within feasible time in small data set.

(3) Scalability: Table 5 shows the result of Case 2. The calculation time was about 15.5 hours. Although the problem size of Case 2 was about 63 times larger than Case 1, the calculation time was only about 18.3 times longer than Case 1. The calculation time is much less than "n log n", n is the total number of decision variables. This indicates that our method can deal with large case within feasible time.

## VI. CONCLUSION

We proposed an algorithm of making a train-unit-assignment schedule using order-patterns. Whereas some constraints of this problem can be represented by a network model, others cannot be represented. If constraints are represented by a network model, there are efficient methods for considering the constraints. Therefore, we divided the problem into two problems on the basis of the type of constraints: train-circulation assignment and inspection assignment. To avoid a too restricted of a limitation of a search space, the proposed algorithm searches for a solution that bridges the two problems by the network model.

Numerical experiments using actual data indicated that the proposed algorithm could search a better solution than commercial solver Gurobi, which uses several algorithms of optimization like branch-and-cut, in feasible processing time. The inspection cost of the proposed algorithm was 35.0 percent lower than the one of Gurobi. In addition, the variance of inspection costs of ten trials was 0.25 and 3.08 in case of small and large data respectively. The result indicates the proposed algorithm is stable though the proposed algorithm randomly searches the best solution. From the viewpoint of processing time, the algorithm converged within one hour and 16 hours in case of small and large data respectively. The calculation time is much less than "n log n", n is the total number of decision variables. These results indicates the proposed algorithm is scalable even if problem size becomes large.

From the above, we confirm that the proposed algorithm is feasible and scalable against actual data. To apply the proposed algorithm to actual use, we need to verify the best limitation of processing time considering the method's stability.

## REFERENCES

[1] Z. Lin, R. S. K. Kwan, "A branch-and-price approach for solving the train unit scheduling problem," Transportation Research Part B: Methodological, Vol. 94, 2016, pp. 97-120.

[2] L. Cadarsoa, A. Marin, "Improving robustness of rolling stock circulations in rapid transit networks," Computers & Operations Research, Vol. 51, 2014, pp. 146-159.

[3] A. Alfieri, R. Groot, L. Kroon, A. Schrijver, "Efficient Circulation of Railway Rolling Stock," Transportation Science, Vol. 40, No. 3, 2006, pp.378-391.

[4] M. Peeters, L. G. Kroon, "Circulation of Railway Rolling Stock: a Branch-and-Price Approach," Computers & Operations Research, Vol. 35, No.2, 2008, pp.538–556.

[5] E. J. W. Abbink, B. W. V. van den Berg, L. G. Kroon, M. Salomon, "Allocation of Railway Rolling Stock for Passenger Trains," Transportation Science, Vol.38, 2004, pp.33–42.

[6] G. L. Giacco, A. D'Ariano, D. Pacciarelli, "Rolling Stock Rostering Optimization Under Maintenance Constraints," Journal of Intelligent Transportation Systems: Technology, Planning, and Operations, Vol. 18, Issue 1, 2013, pp. 95-105.

[7] P. horlacius, J. arsen, M. Laumanns, "An integrated rolling stock planning model for the Copenhagen suburban passenger railway," Journal of Rail Transport 2 Planning & Management, Vol. 5, No. 4, 2015, pp. 240-262.

[8] V. Cacchiani , A. Caprara, P. Toth, "A Lagrangian heuristic for a train-unit assignment problem," Discrete Applied Mathematics, Vol. 161, Issue 12, 2013, pp.1707-1718.

[9] J. Haahra, R. M. Lusby, "Integrating rolling stock scheduling with train unit shunting," European Journal of Operational Research, Vol. 259, Issue 2, 2017, pp. 452-468.

[10] G. Mar´oti, L. G. Kroon, "Maintenance Routing for Train Units: the Transition Model," Transportation Science, Vol. 39, No. 4 , 2005, pp.518–525.

[11] N. Lingaya, J. F. Cordeau, G. Desaulniers, J. Desrosiers, F. Soumis, "Operational car assignment at VIA Rail Canada," Transportation Research Part B Methodological, Vol. 36, No.9, pp.755-778.

[12] J. Andrésa, L. Cadarsob, Á. Marína, "Maintenance Scheduling in Rolling Stock Circulations in Rapid Transit Networks," Transportation Research Procedia, Vol. 10 , 2015, pp.524-533.

[13] Giovanni Luca Giacco, Donato Carillo, Andrea D'Ariano, Dario Pacciarelli, Ángel Marín, "Short-term rolling stock rostering and maintenance scheduling", Transportation Research Procedia, Volume 3, pp.651-659, 2014.

[14] Tatsushi Nishi, Akiyoshi Ohno, Masahiro Inuiguchi, Satoru Takahashi, Kenji Ueda, "A Combined column generation and heuristics for railway short-term rolling stock planning with regular inspection constraints", Computers & Operations Research, Volume 81, pp. 14–25, 2017.

[15] G. L. Giacco. D. Carillo., A. D'Ariano, D. Pacciarelli, A. G. Marin, "Short-term Rail Rolling Stock Rostering and Maintenance Scheduling," Transportation Research Procedia, Vol. 3, 2014, pp. 651-659.

[16] Yung-Cheng (Rex) Lai, Shao-Wei Wang, and Kwei-Long Huang, "Optimized Train-Set Rostering Plan for Taiwan High-Speed Rail", IEEE Transactions on Automation Science and Engineering, Vol.14, Issue 1, pp.286-298, 2017.

[17] T. Tomiyama, T. Sato, K. Okada, T. Wakamiya, "Rescheduling Algorithm Using Operational Patterns for Rolling Stock Operation at Train Depots," Computers in Railways XIII, Vol.127, 2012, pp.555-566.

[18] Gurobi, http://www.gurobi.com/