# FREIGHT LOCOMOTIVE RESCHEDULING AND UNCOVERED TRAIN DETECTION DURING DISRUPTIONS

*Keisuke SATO (Railway Technical Research Institute)*

*Naoto FUKUMURA (Railway Technical Research Institute)*

## ABSTRACT

This paper discusses optimization of freight train locomotive rescheduling under a disrupted situation in the daily operations in Japan. In the light of the current framework of dispatching processes that passenger railway operators modify the entire timetables, the adjusted timetable is distributed to a freight train operator. We solve the locomotive rescheduling problem for the given adjusted timetable in which we change the assignment of the locomotives to the trains as required, considering a periodic inspection of the locomotives. The uncovered train detection problem that selects unassigned trains of less importance is solved if the rescheduling has failed. We formulate the two problems as integer programming problems derived from our network representation of the disrupted situation, and solve the problems by column generation. Our simple speeding-up technique named set-covering relaxation is applied to the rescheduling problem, which has set-partitioning constraints. The column generation subproblem reduced to a shortest path problem with the inspection constraint is solved in polynomial time. Numerical experiments using a real timetable, locomotive scheduling plan and major disruption data in the highest-frequency freight train operation area reveal that satisfactory solutions are obtained within around 30 seconds by a PC even for the cases with 72-hour goal for recovery. The set-covering relaxation speeds up the computation time by a factor of six at a maximum.

*Keywords: Locomotive rescheduling, uncovered train detection, set partitioning, set-covering relaxation, column generation.*

## 1. INTRODUCTION

Mathematical optimization approaches for railway *planning* problems such as timetabling, rolling stock circulation and crew scheduling have been long and widely studied, and have recently brought major success to some train operators (refer to Caprara et al. (2007) and Kroon et al. (2009)). They dramatically reduce the associated operational cost and improve satisfaction of passengers and cargo owners. Meanwhile under a *disrupted* situation in the daily operations where the planned timetables and schedules are not achieved by accidents

or adverse weather conditions, the resolutions of timetable adjustment and rolling stock/crew rescheduling by optimization have not yet been proposed sufficiently as indicated in Jespersen-Groth et al. (2009b) as well as Rezanova and Ryan (2010). One of major factors making the rescheduling optimization difficult is time for train dispatchers in charge to wait for a solution provided by a computer. A solution of acceptable quality must be obtained as quickly as possible to prevent further delays of the trains by done nothing, particularly in areas with high-frequency train operations.

One among the limited literature on optimization approaches for railway disruption management is by Walker et al. (2005), in which timetables and drivers on the Metro line in New Zealand are rescheduled simultaneously. They formulate their problem as a set-partitioning problem with many additional constraints and they aim at minimizing the deviation from the existing timetables and crew schedules. An arbitrary chosen one out of the 36 trains operated in the area is delayed for the numerical experiments and the solutions are obtained in 110 seconds. Huisman (2007) gives a set-covering model for a driver rescheduling problem of passenger railway operator NS of the Netherlands where the timetables are changed due to maintenance work on train tracks. It takes about 15 hours to solve the real size of the problem that has more than 700 driver duties and 7,000 tasks. The computation time is reasonable since the modification of the timetables is anticipated in advance. Quite recently, another crew rescheduling formulation has been proposed by Rezanova and Ryan (2010) for the daily disruption. Their model is based on a set-partitioning problem and it is applied to the data obtained from Danish passenger railway operator DSB S-tog A/S with up to 74 drivers and 91 tasks. The solutions are provided in 26 seconds. Some uncovered trains are detected as their results. Jespersen-Groth (2009a) discusses a rolling stock recovery problem. The problem is aimed at re-routing rolling stocks according to estimated passenger demand, and is formulated as a network flow model. Reportedly, 100 train tasks are recovered in about 70 seconds. In the airline industry, the resource management optimization during disruptions is often discussed, as reviewed in Clausen et al. (2010). In the context of general vehicle routing, Huisman and Wagelmans (2006) and Li et al. (2007) offer vehicle rescheduling algorithms. They do not take into consideration the constraints characteristic to the railway industry such as the periodic vehicle inspections.

In this paper, we discuss optimization of rescheduling of locomotives that haul freight trains in Japan. The Japan Freight Railway Company operates all of approximately 600 freight trains daily except for those in some bay or regional areas. This company is, in almost all of its operational areas, recognized as the second class railway operator, which indicates that its trains run on the roadways owned by the first class railway operators (passenger railway operators) such as the East Japan Railway Company, the Central Japan Railway Company and the West Japan Railway Company. The freight and passenger trains share the same infrastructures and the freight trains run between the passenger ones, which are operated by the first class operators. The frequencies of the passenger trains are very high in the urban areas in Tokyo, Nagoya and Osaka, the three largest cities in Japan. The timetables for the freight and passenger trains are planned and decided once per year after negotiations among the operators involved. The timetables planned as above are sometimes suspended due to accidents or adverse weather conditions in the daily operations. Under such a disrupted situation, train dispatchers serving on the first class railway operator in the area modify the timetables and accordingly the rolling stock schedules in various ways for the

passenger trains to prevent further expansion of the trouble. Some of their efforts are studied in Tomii et al. (2005). At the same time, the dispatchers, having an authority as the first class operator, *adjust* the freight train timetable. In this framework of the current dispatching processes, the options that the freight train operator can take to prevent the disruption from spreading are much restricted; rescheduling its resources (locomotives, freight cars, containers and drivers), requesting for re-adjustment of the freight train timetable, or cancelling the trains. The re-assignment of the resources to the trains is first tried. The freight railway company makes a request to the first class operators for the re-adjustment of the train timetables only if the rescheduling has failed; however, the request is not always approved. Cancelling of the trains is the last resort since it might lead to lose the cargo owners' trust. Thus, the resource rescheduling is the most important part of disruption management for freight trains in Japan. It is, however, very difficult for the freight train operator to decide manually in short time whether the rescheduling is possible in a case of a large disruption. Among the rescheduling problems, the one pertaining to locomotives is crucial. The number of locomotives available is rather limited to compare with freight cars, containers and drivers. Exactly one locomotive must be assigned to each train (unless the operation in tandem is specified) since an additional locomotive may run over a block section or a platform. Note that locomotives in tandem are permissible when the train is a deadhead one without cars. The time that it takes for a locomotive to haul its loads from and to the specified stations ranges from 30 minutes to 20 hours. Furthermore, each locomotive must be inspected for every 72 or 96 hours depending on its type in spite that locomotive depots and workers for the inspections are limited. Hence, two or three days of rescheduling have to be considered, enlarging the problem size to be handled.

Based on the current framework of the freight train dispatching processes in Japan, we study the challenging rescheduling issue concerning locomotives. Given an adjusted timetable and the current schedules and positions of the locomotives, we model the whole issue as two optimization problems. The first one is the *locomotive rescheduling problem*, in which we change the assignment of the locomotives to the trains as needed. It consequently decides whether all of the trains are exactly covered by the locomotives, which is *NP*-complete as we show it later. The second problem is the *uncovered train detection problem* and it is solved only if the rescheduling has failed. The locomotives are assigned to the trains in accordance with the importance of the trains, and the remaining uncovered trains are detected. It will help the freight train operator cancel the trains or make a request to the first class operators for re-adjustment of the timetable. Note here that the evaluation criteria are different between the two problems. If the locomotives can cover all the trains, then the degree of the locomotive rescheduling is preferred to be as small as possible. If there are some trains whose timetable must be re-adjusted or which must be cancelled eventually, they are preferred to be the ones of less importance. We therefore divide the whole issue into the two phases based commonly on our network representation of the disrupted situation. Both of the problems are formulated as integer programming problems. More specifically, the first problem is a set-partitioning problem with side constraints and the second a set-packing problem with the same side constraints (refer to Nemhauser and Wolsey (1988) for set-partitioning, set-packing and set-covering problems). Column generation technique (of which insightful overview is introduced in Desaulniers et al. (2005)) and our approach named set-covering relaxation solve the real size of the problems in real time while achieving good

solution quality. A polynomial-time algorithm for our column generation subproblem reduced to a shortest path problem with the periodic locomotive inspection constraint also contributes to the reduction of computation time.

The rest of the paper is organized as follows. Section 2 gives precise definitions of our freight locomotive rescheduling and uncovered train detection problems on our network representation of the disrupted situation. We also discuss the complexity of the problems. In Section 3, we formulate the rescheduling problem and solve it by column generation combined with our simple speeding-up technique as termed set-covering relaxation. A polynomial-time algorithm for our column generation subproblem is also presented in the section. We subsequently indicate an algorithm for the uncovered train detection problem in Section 4 when the rescheduling problem turns out to be infeasible. Numerical experiments using the real timetable, locomotive scheduling plan and disruption data are explicit in Section 5. Section 6 concludes the paper and we present our view of future work.

# 2. PROBLEM DESCRIPTION

## 2.1. Network Representation

To model the locomotive rescheduling and uncovered train detection problems, we first represent a freight train timetable and locomotive scheduling as a network. Figure 1 is a freight train diagram and schedules of two locomotives planned on the diagram. The locomotive labeled as Loco. 2, for instance, is planned to haul Train 2 from Station C to Station B, and then Train 4 from Station B to Station A. The minimum unit of haulage of a train from and to the specified stations is termed a *task*, and a *sequence of tasks* over several days is planned for each locomotive. The label “Ins.” indicates an *inspection* of the locomotive. An inspection is also included in a sequence of tasks. Assume here that a disruption has occurred and that the departure of Train 2 is delayed. Loco. 2 would miss Train 4 if no action were taken. In this case, we can avoid it if the tasks of Loco. 1 and Loco. 2 are exchanged, i.e., Loco. 1 hauls Train 4 while Loco. 2 does Train 1. This is a simple example and a solution for the locomotive rescheduling problem. We then explain our way of modeling through Figure 1. For the given adjusted timetable, we set rescheduling starting time $h_0$ along with *rescheduling period* $h$, and construct a network shown in Figure 2. The nodes of this network consist of, positions of the locomotives at the starting time $k_1, k_2$, tasks to be covered $i_1, \dots, i_6, i_8$, scheduled or extra inspections $s_1, \dots, s_4$, and the first tasks after the rescheduling period $d_5, d_8$. The directed arcs are drawn between nodes if a locomotive is able to move between them.

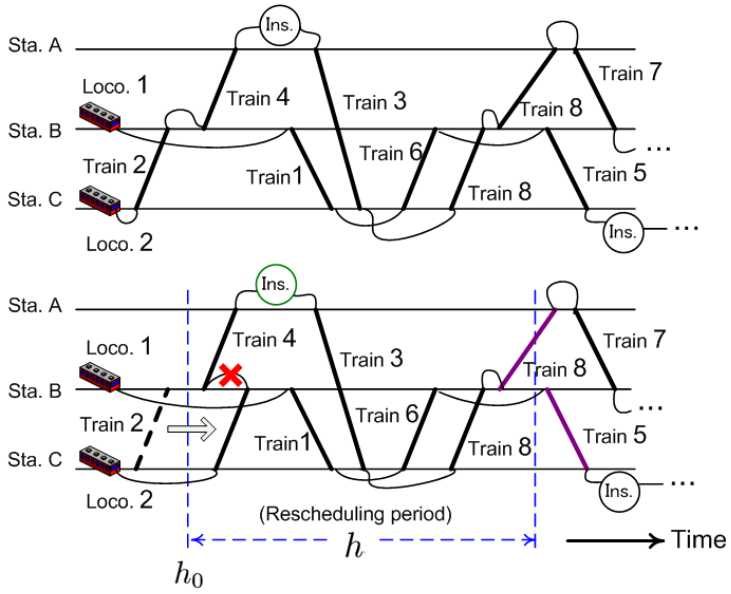


Figure 1 – Planned timetable/locomotive scheduling and adjusted timetable

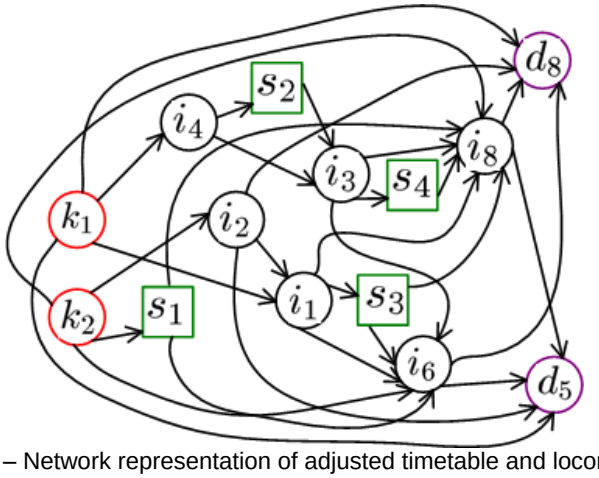


Figure 2 – Network representation of adjusted timetable and locomotives

Formally, we first define node set $V := K \cup I \cup \hat{I} \cup D \cup S$ from a given adjusted timetable, a locomotive scheduling plan, rescheduling starting time $h_0$ and rescheduling period $h$. The set $K$ consists of the locomotives involved. This set also includes reserve locomotives available. Each locomotive $k \in K$ has the information on its previous inspection time $\mathrm{PrevIns\_time}(k)$, time ready for the rescheduling $\mathrm{Ready\_time}(k)$ ($\geq h_0$), and station $\mathrm{Arr\_sta}(k)$ that $k$ is at as of $\mathrm{Ready\_time}(k)$. Inspection interval $R^k$, 72 or 96 hours depending on the type of the locomotive, is also prepared. The locomotive $k$ must be inspected for every $R^k$. We denote by $I$ a set of rescheduling tasks to be covered. A set of deadhead train tasks without cars is denoted by $\hat{I}$. An element $i$ of $I$ or $\hat{I}$ has its departure time $\mathrm{Dep\_time}(i)$ and station $\mathrm{Dep\_sta}(i)$, its arrival time $\mathrm{Arr\_time}(i)$ and station $\mathrm{Arr\_sta}(i)$. Note that $h_0 < \mathrm{Arr\_time}(i) < h_0 + h$ holds, i.e., the arrival time of the rescheduling tasks is neither earlier than the rescheduling starting time nor later than the stopping time of the rescheduling. Since each locomotive has its planned sequence of tasks, its first task after the rescheduling period has expired, or the task whose departure time is later and arrival time is earlier than the stopping time respectively, is definable. For a reserve locomotive, the task is to be in its home depot.

We call the set of such tasks the *convergence tasks* and denote them by $D$. It holds that $|D| = |K|$ by the definition. Each convergence task $d \in D$ has locomotive Due_loco$(d)$ due to be assigned and next inspection time NextIns_time$(d)$ $(\geq h_0 + h)$ which are determined by the planned sequence of tasks, as well as Dep_time$(d)$ and Dep_sta$(d)$. For each node $v \in K \cup I \cup \hat{I}$, an inspection of the locomotives can be carried out after $v$ if a locomotive depot is adjacent to Arr_sta$(v)$. We let $s$ an inspection node after $v$, and then the node has its previous task (or a locomotive) Prev_task$(s)$ which equals to $v$ and station Sta$(s)$. Moreover the node has finishing time of the inspection FinishIns_time$(s)$, which is trivially later than Arr_time$(v)$ (or Ready_time$(v)$). We denote by $S$ the set of the inspection nodes.

Next, we define directed arc set $E$. The first type of arcs is drawn from a locomotive to tasks if pair $(k,\ i) \in K \times (I \cup \hat{I} \cup D)$ satisfies the following condition:

Arr_sta$(k)$ = Dep_sta$(i)$ and Ready_time$(k)$ + Buf_time(Arr_sta$(k)$) ≤ Dep_time$(i)$

where Buf_time$(*)$ is some buffer time needed to assign a locomotive to a train at station $*$. It should be noted that we do not consider shunting problems here since there are many tracks and turnouts in the freight train stations in Japan. An arc is also connected between task node pair $(i_1,\ i_2) \in (I \cup \hat{I}) \times (I \cup \hat{I} \cup D)$ if

Arr_sta$(i_1)$ = Dep_sta$(i_2)$ and Arr_time$(i_1)$ + Buf_time(Arr_sta$(i_1)$) ≤ Dep_time$(i_2)$

holds. There are no outgoing arcs from any member of $D$. For each inspection node $s$, we draw (Prev_task$(s)$, $s$) $(\in (K \cup I \cup \hat{I}) \times S)$ expressing an inspection after the end of its previous task. A task after an inspection is represented by arc $(s,\ i) \in S \times (I \cup \hat{I} \cup D)$ satisfying the condition:

Sta$(s)$ = Dep_sta$(i)$ and FinishIns_time$(s)$ ≤ Dep_time$(i)$.

We now have the node set $V$ and the arc set $E$ forming an acyclic graph, and we add two functions $c$ and $f$ to it, constructing network $N := (V, E, c, f)$. The cost function $c : K \times E \to \mathbb{R}_+$ returns a nonnegative value when a locomotive and an arc are input. For simplicity we substitute $c_e^k$ for $c\,(k, e)$, and it represents the cost for locomotive $k$ to traverse arc $e$. The cost function is used as the criterion for the locomotive rescheduling problem. We set $c_e^k := 0$ for arc $e \in E \cap ((K \cup I \cup \hat{I}) \times (I \cup \hat{I}))$ when its endpoint tasks coincide with two consecutive tasks in the planned sequence of tasks of any locomotive. Some positive cost is set otherwise, since it indicates a change of locomotive assignment. We prepare two cost values $W_1$ and $W_2$, and select either of them according to the workload of the change. For arc $(v, s) \in E \cap ((K \cup I \cup \hat{I}) \times S)$ from a locomotive or a task to an inspection node, its cost is zero if the inspection after the task is included in the planned sequence of tasks of any locomotive. If not, then it means an unscheduled extra inspection and cost $W_3$ is given to the arc. The cost of arc $(s, i) \in E \cap (S \times (I \cup \hat{I} \cup D))$ from an inspection node to a task node is equal to that of (Prev_task$(s)$, $i$). The cost of an arc incoming to the convergence tasks, say $c_{(v,d)}^k$ for $(v, d) \in E \cap (V \times D)$, depends on the locomotive name $k$. The Japanese freight train operator permits an assignment of a locomotive other than the planned one to the task $d$ after a disruption, though the assignment of the planned locomotive is much preferred. Hence we set $c_{(v,d)}^k := 0$ when Due_loco$(d)$ = $k$ and $c_{(v,d)}^k := W_4$ otherwise. There are several types of locomotives running in the same area in Japan, and an additional cost $W_5$ is imposed when the next task of a locomotive is switched to a task that a different type of locomotives plans to haul. When a task should not be covered by the locomotives of a certain type since the type has too small tractive effort or such assignment violates some other operational constraints, we set $c_e^k := \infty$ to prevent it from happening. The demand

function $f : (I \cup D) \to \mathbb{R}_+$ defines the value of importance of each task and it is used as the criterion for the uncovered train detection problem.

## 2.2. Feasible Path and Inspection Capacity

We consider a path from a locomotive node to a convergence node on the network. It is the planned or rescheduled sequence of tasks of the locomotive. Let $p$ be such a path and $c_p^k$ be the sum of $c_e^k$ which appears in the path. We call $c_p^k$ the cost of path $p$ of locomotive $k$. Then $p$ with $c_p^k = 0$ is the planned sequence of tasks of $k$ and that with positive cost indicates the workload of rescheduling the locomotive. We also define $f_p$ similarly and call it the importance of path $p$. Recall here, that any locomotive must be inspected for every inspection interval $R^k$. To express a locomotive path in which the periodical inspection is carried out properly, we first prepare function $H^k: (\{k\} \cup S) \to 2^{I \cup \hat{I} \cup D}$ such that, for each locomotive $k \in K$ and inspection $s \in S$,

$$H^k(k) \coloneqq \{\, i \in I \cup \hat{I} \mid \mathrm{PrevIns\_time}(k) + R^k \geq \mathrm{Arr\_time}(i) \,\} \cup \{\, d \in D \mid \mathrm{PrevIns\_time}(k) + R^k \geq \mathrm{NextIns\_time}(d) \,\},$$

$$H^k(s) \coloneqq \{\, i \in I \cup \hat{I} \mid \mathrm{FinishIns\_time}(s) + R^k \geq \mathrm{Arr\_time}(i) \,\} \cup \{\, d \in D \mid \mathrm{FinishIns\_time}(s) + R^k \geq \mathrm{NextIns\_time}(d) \,\}.$$

The set $H^k(k)$ is the tasks to which we can assign the locomotive $k$ without any inspection. The set $H^k(s)$ includes the tasks to which we can assign $k$ after the inspection $s$ is done. Note that $H^k(v_1) \subseteq H^k(v_2)$ is equivalent to $|H^k(v_1)| \leq |H^k(v_2)|$ for any $v_1, v_2 \in \{k\} \cup S$, since the function is based on time. We next take the locomotive and inspection nodes contained in path $p$ and call them $s_0, s_1, \dots, s_n$ in order of their appearance ($s_0 = k$). Let $N_p(s_m,\ s_{m+1}) \subseteq I \cup \hat{I}$ be the tasks between $s_m$ and $s_{m+1}$ in $p$, and $N_p(s_n) \subseteq I \cup \hat{I} \cup D$ the ones after the last inspection $s_n$. We then call that the path $p$ is a *feasible path* if $p$ satisfies the following condition:

$$H^k(s_m) \supseteq N_p(s_m,\ s_{m+1}) \quad \forall m \in \{0, \dots, n-1\},$$
$$H^k(s_n) \supseteq N_p(s_n).$$

The locomotive is properly inspected in a feasible path. We denote by $P^k$ the set of $k$'s feasible paths for each $k \in K$.

Another constraint involving the inspections is the limited number of facilities and workers for the inspections. We let $B$ be the depot set and $T$ be the time span set. The capacity of the inspections for depot $b \in B$ at time span $t \in T$ is expressed as $L_t^b$.

## 2.3. Problem Definitions and Complexity

We are now ready for defining our locomotive rescheduling and uncovered train detection problems. Given the network $N \coloneqq (V, E, c, f)$, the function $H^k$ for each locomotive $k \in K$ and the inspection capacity $L_t^b$ for every $b \in B, t \in T$, we select one path among the feasible paths for each locomotive and exactly cover all the elements in $I \cup D$ by them, i.e., any task node is traversed by only one path. Note that a train whose locomotive operation in tandem is specified is divided into two distinct tasks, and that the elements in $\hat{I}$ need not be covered or can be covered by up to two paths. At the same time, the number of locomotive inspections at $b$ and $t$ included in the selected paths must not exceed $L_t^b$. The locomotive rescheduling

problem decides whether such covering is possible or not, and if possible, finds paths so that the sum of $c_p^k$ is sufficiently small. The uncovered train detection problem is defined as, to find the combination of feasible paths so that the sum of $f_p$ is large enough, under the condition that all the elements in $I \cup D$ are covered by up to one path and the constraints concerning the deadhead tasks and that the inspections are satisfied.

We discuss here the complexity of the decision problem part of our locomotive rescheduling problem. In Maróti and Kroon (2005) which studies routing for train units inspected, one *NP*-complete result is obtained. That is the problem of whether all the nodes of an arbitrary acyclic graph can be exactly covered by disjoint paths or not, where multiple sources (nodes with no in-going arcs) and sinks (nodes with no out-going arcs) of the paths are given and a given one source of them is specified to a certain sink. In our problem, consider a case with one locomotive type, $\hat{I} = S = \emptyset$, every element in $I$ having both in-going and out-going arcs and that $|H^{k_0}(k_0) \cap D| = 1$ holds for some $k_0 \in K$ while $H^k(k) = I \cup D$ for any other $k \in K \setminus \{k_0\}$. In this case, locomotive $k_0$ is specified to the one destination and our problem coincides with the *NP*-complete problem.

# 3. FORMULATION AND ALGORITHM OF RESCHEDULING

## 3.1. Integer Programming Formulation and Overall Algorithm

In this paper, we regard the locomotive rescheduling problem as a variant of a set-partitioning problem and model it as an integer programming problem. For locomotive $k \in K$ and its feasible path $p \in P^k$, let $a_{ip}^k$ be one if task $i \in I \cup \hat{I} \cup D$ is included in $p$, otherwise zero. We similarly define $a_{tp}^{bk}$ for depot $b \in B$ at time span $t \in T$. We introduce a decision variable $x_p^k$ that takes one if $p$ is selected as a rescheduled sequence of tasks of $k$, otherwise zero. We then present the following formulation:

$$\text{minimize} \quad \sum_{k \in K} \sum_{p \in P^k} c_p^k x_p^k \tag{1}$$

$$\text{subject to} \quad \sum_{k \in K} \sum_{p \in P^k} a_{ip}^k x_p^k = 1 \qquad \forall\, i \in I \cup D, \tag{2}$$

$$\sum_{k \in K} \sum_{p \in P^k} a_{\hat{\imath}p}^k x_p^k \leq 2 \qquad \forall\, \hat{\imath} \in \hat{I}, \tag{3}$$

$$\sum_{p \in P^k} x_p^k = 1 \qquad \forall\, k \in K, \tag{4}$$

$$\sum_{k \in K} \sum_{p \in P^k} a_{tp}^{bk} x_p^k \leq L_t^b \qquad \forall\, b \in B \;\; \forall\, t \in T, \tag{5}$$

$$x_p^k \in \{0,1\} \qquad \forall\, k \in K \;\; \forall\, p \in P^k. \tag{6}$$

The objective function expressed as equation (1) seeks to minimize the total sum of the cost of paths selected, i.e., the total workload of the rescheduling. Equation (2) is a set of set-partitioning constraints, which indicates that any task is contained in only one of the paths. A deadhead train task without cars need not be covered or can be covered by up to two paths, and it is expressed as constraint (3). For each locomotive, only one path is selected among its feasible paths according to equation (4). Equation (5) is the inspection capacity constraint for each depot and time span, and the variables take binary values by equation (6).
The optimal solution for this problem is straightforwardly obtained after all the feasible paths $P^k$ are enumerated from the network $N$. The enumeration of all such paths, however, is not reasonable; it would take too much time. In fact, none of them is available before the rescheduling, since the adjusted timetable and consequently the topology of the network cannot be presumed. We therefore adopt column generation technique, which alternately enumerates some of the feasible paths and solves our optimization problem with the integer constraints of the variables relaxed. Additionally we replace the set-partitioning constraints with set-covering constraints temporally in which the covering of the tasks by more than one path is allowed, since the application of column generation to problems that have set-partitioning constraints is known to cause degeneracy, i.e., the improvement of the objective function value is slowed down. Degeneracy is also observed in solving problems with set-covering constraints, particularly when the objective function value is close to optimal. We therefore restore some of the set-partitioning constraints when the objective function value approaches its lower bound, and solve our problem again in a different solution space. We call our relaxation approach set-covering relaxation and show the overall algorithm in Figure 3. The details of the subroutines are displayed in the following subsections.

## 3.2. Initialization and Restricted Master Problem

At Step 1 of the algorithm, we first introduce and initialize iteration counter $\ell \coloneqq 0$ of our column generation as well as lower bound $Z_{LB} \coloneqq 0$ of the objective function value. A subset $P_\ell^k$ of all the feasible paths set $P^k$ is given for each $k$ with $P_0^k \coloneqq \emptyset$. We then replace $P^k$ with $P_\ell^k$ in the locomotive rescheduling problem. Some of the feasible paths (or columns) are generated and added to $P_\ell^k$ through the iterations. Similarly $I_\ell \subseteq I$ and $\hat{I}_\ell \subseteq \hat{I}$ are defined with $I_0 = \hat{I}_0 \coloneqq \emptyset$. The two subsets are related to the set-covering relaxation; we simply permit that the tasks in $(I \cup D) \setminus I_\ell$ are covered by one or more locomotives and that any number of the locomotives are assigned to the deadhead tasks in $\hat{I} \setminus \hat{I}_\ell$. The binary constraint (6) is also replaced with a nonnegative one for column generation. One artificial nonnegative variable $y$ is introduced whose coefficient column consists of the right-hand side of equations (2)-(5) and a huge cost $M$ in the objective function, making the locomotive rescheduling problem always feasible at $x = 0, y = 1$. We give here restoring parameter $G \geq 0$ to enlarge the task sets $I_\ell$ and $\hat{I}_\ell$ forming the original constraints. This parameter is used at Step 4.

**Step 1**: *Initialization.*
Set $\ell \coloneqq 0,\ Z_{LB} \coloneqq 0,\ P_\ell^k \coloneqq \emptyset,\ I_\ell = \widehat{I_\ell} \coloneqq \emptyset.$
Replace $P^k$ with $P_\ell^k$.
Introduce dummy variable $y$ with cost $M$.
Apply set-covering relaxation to (2), (3) and LP relaxation to $x$.
Set restoring parameter $G$.
**Step 2**: *Restricted Master Problem* $(RMP_\ell)$.
Solve LP for $(P_\ell^k,\ I_\ell,\ \widehat{I_\ell})$.
Set objective value on $Z_\ell$, dual prices on $\lambda_i,\ \mu^k,\ \nu_t^b$.
**Step 3**: *Column Generation Subproblem* $(SP_\ell)$.
Solve shortest path problem for $(N^\star,\ \lambda_i,\ \mu^k,\ \nu_t^b)$.
Set feasible paths found on $C_\ell^k$, shortest path length on $r_\ell^k$.
**Step 4**: *Lower Bound Update and Restoring Constraints.*
Set $Z_{LB} \coloneqq \max\{\ Z_{LB},\ Z_\ell + \sum_{k\in K} r_\ell^k\ \}$.
If $G \times (Z_\ell - Z_{LB}) \geq -\sum_{k\in K} r_\ell^k$ , then
$I_{\ell+1} \coloneqq I_\ell \cup$ (set of tasks violating (2) at optimal solution for $(RMP_\ell)$).
$\widehat{I_{\ell+1}} \coloneqq \widehat{I_\ell} \cup$ (set of tasks violating (3) at optimal solution for $(RMP_\ell)$).
Else, then $I_{\ell+1} \coloneqq I_\ell,\ \widehat{I_{\ell+1}} \coloneqq \widehat{I_\ell}$.
**Step 5**: *Termination of Column Generation.*
If ( $Z_\ell > Z_{LB}$ or $I_{\ell+1} \neq I_\ell$ or $\widehat{I_{\ell+1}} \neq \widehat{I_\ell}$ ), then
$P_{\ell+1}^k \coloneqq P_\ell^k \cup C_\ell^k,\ \ell \coloneqq \ell + 1$ and go to Step 2.
Else if $Z_\ell = M$, then output "infeasible" and Stop.
**Step 6**: *Integer Solution Search* $(IP)$.
Set $I_\ell \coloneqq I \cup D,\ \hat{I}_\ell \coloneqq \hat{I}$.
Solve IP for $(P_\ell^k,\ I_\ell,\ \widehat{I_\ell})$.
Set objective value on $Z^{IP}$.
If ( $Z^{IP} < M$ ), then output solution and Stop.
**Step 7**: *Feasibility Check* $(BP)$.
Apply branch-and-price.
If integer solution with $Z_\ell < M$ is found, then output it.
Else, then output "infeasible."

Figure 3 – Algorithm for locomotive rescheduling problem

Restricted master problem $(RMP_\ell)$ to the original locomotive rescheduling problem is thus defined for each iteration counter $\ell$ as follows:

$$\text{minimize} \quad \sum_{k\in K} \sum_{p\in P_\ell^k} c_p^k x_p^k + My \tag{7}$$

$$\text{subject to} \quad \sum_{k\in K} \sum_{p\in P_\ell^k} a_{ip}^k x_p^k + y \geq 1 \qquad \forall\, i \in (I \cup D) \setminus I_\ell, \tag{8}$$

$$\sum_{k\in K} \sum_{p\in P_\ell^k} a_{i'p}^k x_p^k + y = 1 \qquad \forall\, i' \in I_\ell, \tag{9}$$

$$\sum_{k\in K} \sum_{p\in P_\ell^k} a_{\hat{\imath}p}^k x_p^k + 2y < \infty \qquad \forall\, \hat{\imath} \in \hat{I} \setminus \hat{I}_\ell, \tag{10}$$

$$\sum_{k\in K} \sum_{p\in P_\ell^k} a_{\hat{\imath}'p}^k x_p^k + 2y \leq 2 \qquad \forall\, \hat{\imath}' \in \hat{I}_\ell, \tag{11}$$

$$\sum_{p \in P_\ell^k} x_p^k + y = 1 \qquad \forall\, k \in K, \tag{12}$$

$$\sum_{k \in K} \sum_{p \in P_\ell^k} a_{tp}^{bk} x_p^k + L_t^b y \leq L_t^b \qquad \forall\, b \in B \;\; \forall\, t \in T, \tag{13}$$

$$y \geq 0, \qquad x_p^k \geq 0 \qquad \forall\, k \in K \;\; \forall\, p \in P^k. \tag{14}$$

The problem is solved by applying a standard linear programming algorithm at Step 2. The optimal objective value is set on $Z_\ell$. At the same time, we obtain the dual prices or the optimal solutions for the dual problem of $(RMP_\ell)$ (refer to Nemhauser and Wolsey (1988) for duality and Lagrangian relaxation related to it). Let $\lambda_i$ the dual price corresponding to task constraint $i$ in equations (8)-(11), $\mu^k$ be that to constraint $k$ in (12) and $\nu_t^b$ be that to constraint $b, t$ in (13).

### 3.3. Column Generation Subproblem

After $(RMP_\ell)$ is solved to optimality for $\ell$, we decide at Step 3 in Figure 3 whether the objective value $Z_\ell$ can further be improved by adding new paths to the partial feasible paths set $P_\ell^k \subseteq P^k$ for each locomotive. We search for such a path by introducing the following column generation subproblem $(SP_\ell)$ which is derived from the dual of $(RMP_\ell)$:

find $\quad k \in K \;\; p \in P^k$

such that $\quad c_p^k - \sum_{i \in I \cup \hat{I} \cup D} a_{ip}^k \lambda_i - \mu^k - \sum_{b \in B} \sum_{t \in T} a_{tp}^{bk} \nu_t^b < 0.$

We reduce the search problem to an unconstrained shortest path problem thought there is a constraint that a path must be chosen from the feasible paths set. We find a path satisfying the above conditions by introducing network $N^\star$ extended from the network $N$ defined in Section 2. The extended network $N^\star$ is constructed as shown below for each locomotive $k$:

- **Step A:** For each task $i \in I \cup \hat{I} \cup D$, prepare index $\mathrm{Ind}(i)$ and set its rank number in ascending order of $\mathrm{Arr\_time}(i)$ ($\mathrm{NextIns\_time}(i)$ for $i \in D$) on $\mathrm{ind}(i)$.
- **Step B:** Make $(|I \cup \hat{I} \cup D| - |H^k(k)| + 1)$ copies of network $N$ and denote them by $N^{|H^k(k)|}, \ldots, N^{|I \cup \hat{I} \cup D|}$, where $N^j := (V^j, E^j, c, f)$.
- **Step C:** For each $N^j$, change destination node $v^j$ of each arc $(s^j, v^j) \in E^j \cap (S^j \times V^j)$ to $v^{|H^k(s^j)|}$.
- **Step D:** For each $N^j$, delete all the incoming arcs of each task node $i^j$ satisfying $\mathrm{ind}(i^j) > j$.

An example of network $N$ is displayed in Figure 4 where $k$ is permissible to haul the three train tasks $i_1, \ldots, i_3$ without any inspection. Assuming that $\mathrm{FinishIns\_time}(s_*)$ of the inspection nodes $s_*$ plus the inspection interval $R^k$ exceeds $\mathrm{NextIns\_time}(d_6)$, the corresponding

extended network $N^\star$ is Figure 5. The locomotive starting from the node $k^3$ can reach the convergence task $d_6^6$ by traversing $s_2^3$ and such a route is a feasible path.
The copied part $N^j$ in the extended network is interpreted as a subnetwork that has first to $j$th tasks in ascending order of the arrival time of all the tasks. In the subnetwork $N^{|H^k(k)|}$, for instance, the copied locomotive $k^{|H^k(k)|}$ of $k$ can reach up to the $|H^k(k)|$th task (if the locomotive can reach the node on $N$). The $|H^k(k)|$ tasks correspond to the trains that $k$ is permissible to haul without any inspection. The locomotive can be assigned to more tasks if it traverses an inspection node on $N^{|H^k(k)|}$.
We assume that, for locomotive $k$, there is a path from $k^{|H^k(k)|} \in K^{|H^k(k)|}$ to an element of $\cup_{j\in\{|H^k(k)|,\ldots,|I\cup\hat{I}\cup D|\}} D^j$ on $N^\star$, i.e., a copied convergence task. We define, such a path in which superscript * of node $v^*$ is deleted, as a $k$-$D$ path. Then the following holds.

**Proposition 1.** For all $k \in K$, the set of $k$-$D$ paths on $N^\star$ is equivalent to $P^k$ on $N$.
*Proof.* Given a $k$-$D$ path $p$ for any $k$, we first show that the path is included in $P^k$. For any node $i \in N_p(s_m,\ s_{m+1})$ between the two inspections $s_m$ and $s_{m+1}$ in $p$, $\mathrm{Ind}(i) \le |H^k(s_m)|$ holds from Step C and D in constructing the extended network $N^\star$. Task $i'$ satisfying $\mathrm{Ind}(i') = |H^k(s_m)|$ is a member of $H^k(s_m)$ (otherwise $H^k(s_m)$ has at most $\mathrm{Ind}(i') - 1$ tasks, a contradiction), therefore $i \in H^k(s_m)$ holds from Step A and the definition of the function $H^k$. The same holds true for $i \in N_p(s_n)$, and $p$ is shown to be a member of $P^k$.
Given the nodes in a feasible path $p \in P^k$, let locomotive $k(=s_0)$ be $k^{|H^k(k)|}$, each task $i \in N_p(s_m,\ s_{m+1})$ be $i^{|H^k(s_m)|}$, inspection $s_{m+1}$ be $s_{m+1}^{|H^k(s_m)|}$ and each $i \in N_p(s_n)$ be $i^{|H^k(s_n)|}$. For any $i \in N_p(s_m,\ s_{m+1})$ it holds that $i \in H^k(s_m)$ by definition, and so $\mathrm{Ind}(i) \le |H^k(s_m)|$ (otherwise $i \notin H^k(s_m)$). The copied node $i^{|H^k(s_m)|}$ has also an index with $\mathrm{Ind}(i^{|H^k(s_m)|}) \le |H^k(s_m)|$, and hence the incoming arc to $i^{|H^k(s_m)|}$ is not deleted at Step D. Meanwhile, $s_m$ and its next node in $p$ is connected by the arc drawn at Step C on the extended network $N^\star$. Therefore, a path exists on $N^\star$ from $k^{|H^k(k)|}$ to $d^{|H^k(s_n)|}$ if we let the endpoint of $p$ be $d$. □

Hence, we can reduce the column generation subproblem $(SP_\ell)$ to a shortest path problem for all $k$ if we impose the values of the dual prices $-\lambda_i, -\mu^k, -\nu_t^b$ on the corresponding nodes. Note that the dual price $-\nu_t^b$ imposed on an inspection node $s$ is selected as follows; $b = \mathrm{Sta}(s)$, and $t$ is the time span which has the minimum value of $-\nu_t^b$ satisfying $t \subseteq [\mathrm{Arr\_time}(\mathrm{Prev\_task}(s)),\ \mathrm{Dep\_time}(w)]$ where $w$ is a node starting from $s$. The number of arcs on the extended network $N^\star$ is usually more than that of nodes, and Dijkstra's algorithm for an acyclic graph solves the problem in $O(|I \cup \hat{I} \cup D||E|)$ for each locomotive, i.e., the maximum number of arcs on $N^\star$. Moreover we can solve it in $O(|K||E|)$ when $h \le R^k$ holds, i.e., the rescheduling period is not longer than the inspection interval of locomotive $k$. That is since $H^k(s) \ge |I \cup \hat{I}|$ holds for any inspection node $s$ and there is no arc from one node on $N^{H^k(k)}$ to another node on $N^j$ for any $j \in \{(|H^k(k)| + 1), \ldots, (|I \cup \hat{I}| - 1)\}$.

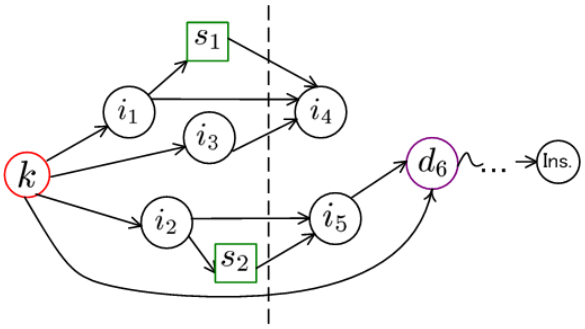


Figure 4 – Initial network $N$

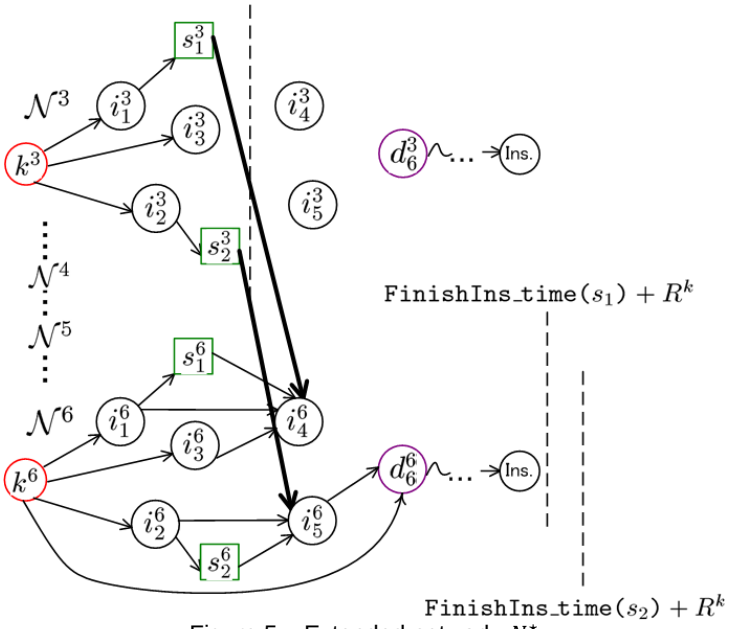


Figure 5 – Extended network $N^\star$

In fact there is no need to construct the extended network $N^\star$. Consider subpath $q$ on the initial network $N$ from locomotive node $k$ to a certain task or inspection node. By introducing label $\mathrm{Reachable}(q) := |H^k(s_m)|$ where $s_m$ is the latest inspection in $q$, we can judge whether locomotive $k$ operated along $q$ is permissible to haul a next task node. We can also reduce the computation time by comparing two subpaths $q_1, q_2$ that have the same endpoint. If $\mathrm{Reachable}(q_1) > \mathrm{Reachable}(q_2)$ holds and the cost of $q_1$ is less than or equal to $q_2$, then $q_2$ can be excluded from the candidate shortest path list. Supposing that there is a subpath, say $q_3$, from the endpoint of $q_2$ to a convergence task, and that the path consisting of $q_2$ and $q_3$ is feasible, then the path consisting of $q_1$ and $q_3$ is trivially feasible and has less cost.
We add to set $C_\ell^k$ feasible paths with a negative cost value obtained by Dijkstra's algorithm. The feasible paths other than the shortest one can also be added which are obtained collaterally by the algorithm. Let $r_\ell^k$ be the shortest path length (let $r_\ell^k := 0$ when $P_\ell^k = \emptyset$). From the complementary slackness condition for linear programming problems, $r_\ell^k$ is always less than or equal to zero.

### 3.4. Lower Bound Update, Restoring Constraints and Termination of CG

The shortest paths of the locomotives found in the preceding subsection is equal to the optimal solution for the Lagrangian relaxation problem of our locomotive rescheduling problem with equations (2), (3), (5) relaxed, if we regard the dual prices $\lambda_i, \nu_t^b$ as Lagrangian multipliers. By the duality theorem (refer to Desaulniers et al. (2005)), at any iteration $\ell$ the optimal value of the Lagrangian relaxation problem is equal to $Z_\ell + \sum_{k\in K} r_\ell^k$, i.e., the objective function value of $(RMP_\ell)$ plus the sum of the shortest path lengths. Since this value can be adopted as the lower bound of the original formulation of our locomotive rescheduling problem, we set $Z_{LB} \coloneqq \max\{Z_{LB},\ Z_\ell + \sum_{k\in K} r_\ell^k\}$ at the beginning of Step 4 in Figure 3.
We refer here to the value of the restoring parameter $G$ and enlarge the task sets $I_\ell$ and $\hat{I}_\ell$ forming the original constraints if the following inequality is satisfied:

$$G \times (Z_\ell - Z_{LB}) \geq -\sum_{k\in K} r_\ell^k.$$

Then we denote the optimal solution for $(RMP_\ell)$ obtained at Step 2 by $\bar{x}, \bar{y}$, and update the task set forming the equality constraint (or the at-most-two constraint) at the next restricted master problem $(RMP_{\ell+1})$ to:

$$I_{\ell+1} \coloneqq I_\ell \cup \{\, i \in I \cup D \mid \sum_{k\in K} \sum_{p\in P_\ell^k} a_{ip}^k\, \overline{x_p^k} + \bar{y} > 1 \,\},$$
$$\widehat{I_{\ell+1}} \coloneqq \hat{I}_\ell \cup \{\, \hat{\imath} \in \hat{I} \mid \sum_{k\in K} \sum_{p\in P_\ell^k} a_{\hat{\imath}p}^k\, \overline{x_p^k} + 2\bar{y} > 2 \,\}.$$

When the inequality involving the parameter $G$ is not satisfied, we let $I_{\ell+1} \coloneqq I_\ell$ and $\widehat{I_{\ell+1}} \coloneqq \hat{I}_\ell$. For $0 \leq G < 1$, the sets $I_\ell$ and $\hat{I}_\ell$ are updated only if $\sum_{k\in K} r_\ell^k = 0$ holds, i.e., the column generation for the restricted master problem with some fixed constraints set converges. For $G = 1$, the constraints are changed when the lower bound $Z_{LB}$ is updated. If we set $G > 1$, then the set $I_\ell$ and $\hat{I}_\ell$ are enlarged when the ratio of the difference of the objective function value and the current lower bound to the negative sum of the shortest path lengths approaches. In the dual problem viewpoint the domain of the variable $\lambda_i \geq 0$ is enlarged to $\lambda_i > -\infty$ by restoring the constraint for $i \in I \cup D$, hence our relaxation approach can be seen as a simple version of the BOXSTEP method (a survey on speeding-up techniques of column generation including the method is displayed in Lübbecke and Desrosiers (2005)).
At Step 5 of our algorithm, we set $P_{\ell+1}^k \coloneqq P_\ell^k, \ell \coloneqq \ell + 1$ and go to Step 2 if $Z_\ell > Z_{LB}$ holds, since the complementary slackness ensures that $r_\ell^k = 0$ is equivalent to $C_\ell^k = \emptyset$. For the cases where the task sets $I_\ell$ and $\hat{I}_\ell$ forming the original constraints are updated at Step 4, we also return to Step 2 since the solution space is changed. Else, then we stop the generation and find an integer solution in the following subsection. However, we stop the algorithm when $Z_\ell = M$ which means $x = 0, y = 1$; it is obvious that there exists no solution for the rescheduling in this case.

### 3.5. Integer Solution Search and Feasibility Check

At the starting point of Step 6 in Figure 3, we have a fractional optimal solution for the locomotive rescheduling problem. We then restore all the set-partitioning constraints by setting $I_\ell \coloneqq I \cup D$, $\hat{I}_\ell \coloneqq \hat{I}$, and find an integer solution by applying branch-and-cut algorithm. We denote by $Z^{IP}$ the objective function value of the integer solution. If $Z^{IP} < M$ holds, then it indicates a not necessarily optimal but feasible solution found. We stop the algorithm there, expecting the solution to be of acceptable quality. Else, then it shows that there may or may not exist a feasible locomotive rescheduling plan since $P_\ell^k \subseteq P^k$, i.e., not all the feasible paths are enumerated at this point. We then apply branch-and-price algorithm $(BP)$ in which the branching rule follows that in Rezanova and Ryan (2010). In the optimal solution for $(RMP_\ell)$, we find locomotive $k$ which fractionally covers task $i$ (or inspection in depot $b$ at time span $t$). We then divide our linear programming problem into two subproblems. One has a constraint such that any feasible path of $k$ must exactly traverse $i$ (or must have an inspection in $b$ at $t$), the other must not. This branching rule is verified by the following proposition via introducing variable vector $z$ with:

$$z_i^k = \sum_{p \in P^k} a_{ip}^k x_p^k \quad \forall k \in K \ \ \forall i \in I \cup \hat{I} \cup D,$$

$$z_t^{bk} = \sum_{p \in P^k} a_{tp}^{bk} x_p^k \quad \forall k \in K \ \ \forall b \in B \ \ \forall t \in T.$$

**Proposition 2**(Rezanova and Ryan (2010))**.** For $I_\ell \coloneqq I \cup D$, $\hat{I}_\ell \coloneqq \hat{I}$, an arbitrary 0-1 assignment to $z$ which does not violate equations (9), (11) (13), and properly adjusted coefficients of $y$ so that $y = 1$ is feasible, there exists an integer solution for $(RMP_\ell)$ when $Z_\ell = Z_{LB}$.

*Proof.* Suppose not. Then we have, by equation (12), fractional optimal solution values $x_{p_1}^k, x_{p_2}^k$ for some $k$ and it holds that the path $p_1$ is equal to the path $p_2$ (the coefficient columns of $x_{p_1}^k$ and $x_{p_2}^k$ have been determined by the value of $z_i^k, z_t^{bk}$). We can set either of them at one and the other at zero. □

If an integer solution is found and the corresponding objective function value is less than $M$, then our problem turns out to be feasible. There is no solution for the rescheduling otherwise, and we subsequently solve the uncovered train detection problem.

## 4. FORMULATION OF UNCOVERED TRAIN DETECTION

We consider the uncovered train detection problem as a variant of a set-packing problem that tries to maximize the sum of the importance of the tasks selected. All the train tasks can be covered by up to one path, and the constraints concerning the deadhead tasks and the inspections must be satisfied. The number of feasible paths selected for a locomotive is zero or one in case there is no feasible path for the locomotive. We give below the integer programming formulation of the problem:

$$\text{maximize} \quad \sum_{k\in K}\sum_{p\in P^k} f_p x_p^k$$

$$\begin{aligned}
\text{subject to} \quad \sum_{k\in K}\sum_{p\in P^k} a_{ip}^k x_p^k &\le 1 && \forall\, i \in I \cup D,\\
\sum_{k\in K}\sum_{p\in P^k} a_{\hat{\imath}p}^k x_p^k &\le 2 && \forall\, \hat{\imath} \in \hat{I},\\
\sum_{p\in P^k} x_p^k &\le 1 && \forall\, k \in K,\\
\sum_{k\in K}\sum_{p\in P^k} a_{tp}^{bk} x_p^k &\le L_t^b && \forall\, b \in B\ \ \forall\, t \in T,\\
x_p^k &\in \{0,1\} && \forall\, k \in K\ \ \forall\, p \in P^k.
\end{aligned}$$

We again relax the integer constraint and apply column generation to the problem. Let $m$ be an iteration counter of the column generation. The feasible paths $P_\ell^k$ enumerated in the locomotive rescheduling problem are added at the first iteration. Any task $i$ with $\sum\sum a_{ip}^k x_p^k = 0$ in the integer solution means an uncovered task and such tasks are output as the result of the algorithm.

# 5. NUMERICAL EXPERIMENTS

## 5.1. Disruption Cases and Computational Environment

This section presents computational results obtained by applying our algorithms to data on the railway lines shown in Figure 6. The lines include the highest-frequency freight train operation area in Japan, where more than 250 freight trains are operated daily. The distance between Kuroiso Station and Shimonoseki Freight Station is about 1,300 km and it takes almost one day for DC electric locomotives to haul the trains between the stations.

We apply delay and cancelation information reported on the website by Japan Freight Railway Company (2006) to the timetable and the DC locomotive schedule plan for year 2006 (obtained from Railway Freight Association (2006)). We set $|K| = 144$ and assume that there is no reserve locomotive available. The locomotives are divided into seven types with the inspection interval being 72 or 96 hours. We choose five major cases among the real disruption logs in from July to November and their summary is shown in Table I. Three to 24 (17% of all) locomotives will miss their next trains to haul without rescheduling; indicating the lower bound of the number of locomotives whose schedules need to be changed. 36 to 72 hours are set as the rescheduling period since there is a task whose running time is over 20 hours. Table II shows the average size of the problems for each rescheduling period, in which the density is defined by $|E| \,/\, \binom{|V|}{2} \times 100$.

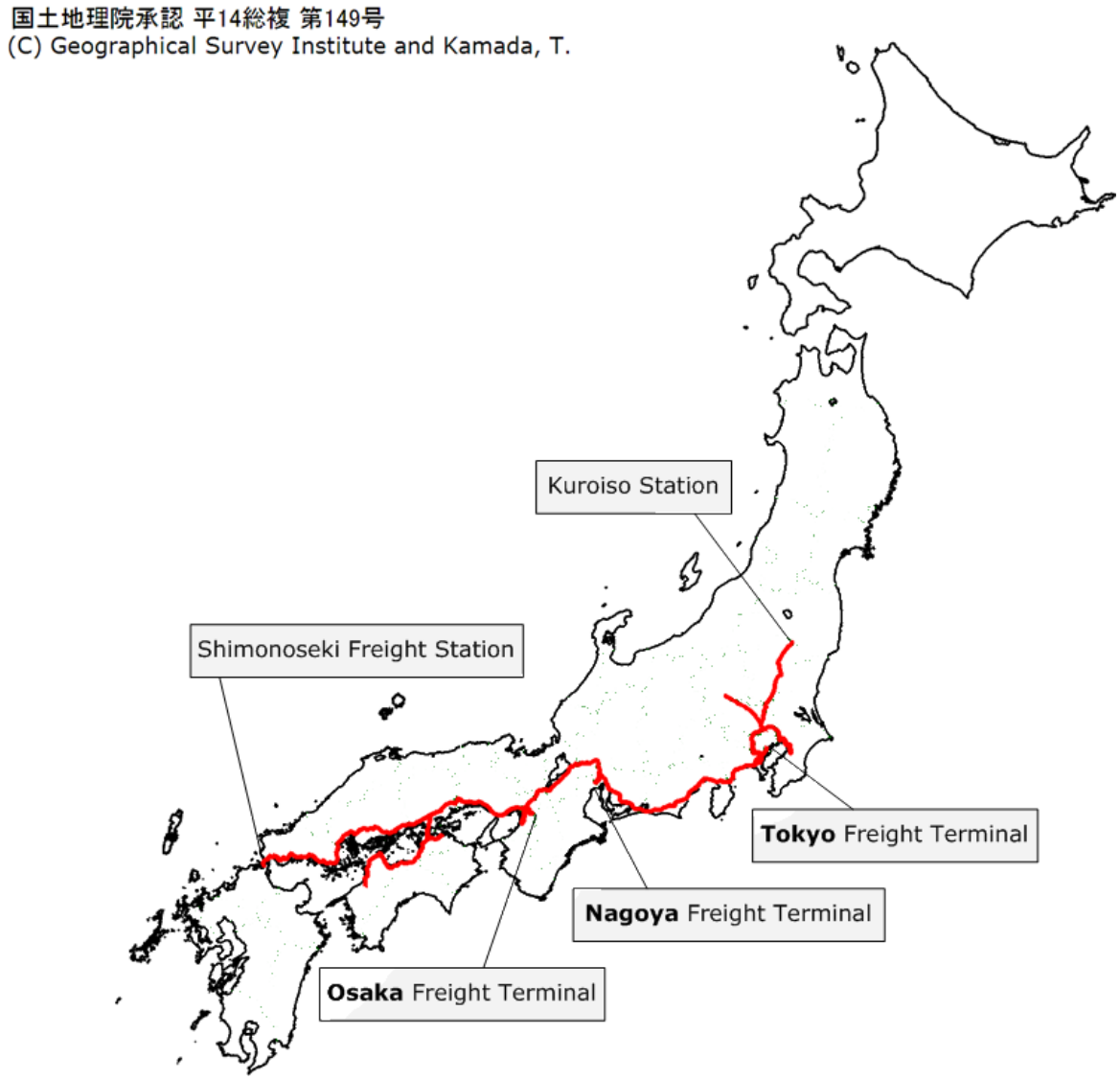


Figure 6 – Major freight railway lines and stations in Japan (Kamada (2009) and GSI)

Table I – Disruption cases

| No. | # Cancelled trains | # Delayed trains | Average delay time (h) | # Locos to miss next trains |
|---|---|---|---|---|
| 1 | 0 | 12 | 3.1 | 3 |
| 2 | 0 | 16 | 2.2 | 4 |
| 3 | 0 | 18 | 3.1 | 5 |
| 4 | 0 | 15 | 4.4 | 10 |
| 5 | 2 | 50 | 3.1 | 24 |

Table II – Instance size (average of 5 cases)

| Rescheduling period (h) | $\lvert I \cup D \rvert$ | $\lvert V \rvert$ | $\lvert E \rvert$ | Density (%) |
|---|---|---|---|---|
| 36 | 602.2 | 1,119.8 | 17,866.6 | 2.86 |
| 48 | 776.0 | 1,404.8 | 29,259.6 | 2.97 |
| 60 | 958.2 | 1,696.8 | 43,027.8 | 2.99 |
| 72 | 1,132.0 | 1,981.8 | 59,887.4 | 3.05 |

The time span for the inspections is set to twelve hours and the capacity for each existing depot and each time span is one plus the number of scheduled inspections in the depot and at the time span. The upper limit for the number of feasible paths $|C_\ell^k|$ added to $P_\ell^k$ for each $k$ is, one for $\ell = 0$, 66 for $\ell = 1$ and five otherwise. We set the restoring parameter $G$ to 3.0. The cost values are, $W_1 = 100, W_2 = 160, W_3 = 180, W_4 = 300, W_5 = 400$ and $M = 30000$, based on the opinions by the experienced workers who were in charge of the locomotive rescheduling. The importance of each task is given similarly. Preliminary experiments show that this combination of parameter values causes much computation time than others.
The programs are implemented in Java SE 6, calling the Java API of IBM ILOG CPLEX 12.1 for solving $(RMP_\ell)$, $(IP)$ and $(BP)$. An interior point method is applied to $(RMP_\ell)$ according to Desaulniers et al. (2005), in which they point out that less iteration is expected for the convergence of column generation if we take rather an analytic center than an extreme point of the optimal face as the solution for $(RMP_\ell)$. All the experiments are carried out on the 32-bit Windows PC having Core i7 CPU with 3.2 GHz and 3 GB RAM. The four CPU cores are used by CPLEX and the shortest path problem on four locomotives is concurrently solved in $(SP_\ell)$.

## 5.2. Locomotive Rescheduling Results

Table III shows the results of our algorithm applied to the locomotive rescheduling problem. The algorithm is run ten times for each disruption case and rescheduling period. The number of locomotives whose sequence of tasks is changed, the quality of the solutions, the number of column generation iterations, the sum of feasible paths enumerated and the real time for the computation are presented in the table. All the results shown in Table III are the average values of the ten trials except for the maximum computation time among the trials. We first note that several different solutions are obtained from trial to trial for the disruption cases No. 4 and 5. For the cases and the rescheduling periods where the same solutions are obtained, the number of iterations and subsequently the number of enumerated feasible paths differ from trial to trial. That derives from the facts that, the feasible paths are enumerated by $(SP_\ell)$ in a different order for each trial due to the multi-threading, the dual optimal solution is not unique for $(RMP_{\ell+1})$, and these cause the linear programming solver to return different values of $\lambda_i,\ \mu_k,\ \nu_t^b$.
The number of locomotives whose sequence of tasks modified is around twice as many as the locomotives whose schedules need to be changed for the cases No. 1, 4 and 5. Generally, dispatchers in charge of locomotive rescheduling try to exchange the planned sequences of a disrupted locomotive with an undisrupted one, and our results seem to be comparable to them. Our solution for the case No. 1 is manually assessed and the rescheduling of the six locomotives is favorably evaluated by experienced workers of dispatching processes. Although that does not hold for the cases No. 2 and 3, the gap defined by $(Z^{IP} - Z_{LB}) \,/\, Z_{LB} \times 100$ indicates that the solutions very close to optimal are obtained. In many cases, the optimal integer solution is obtained at the termination of Step 5 in our algorithm as it is also seen in Rezanova and Ryan (2010), where they conclude that it comes from the special structure of the constraint matrix.
It should also be noted that the algorithm stops at Step 5 for all the infeasible cases. The reason for the rescheduling to become feasible when we set the rescheduling period longer

is that some of the tasks cannot be covered unless the planned sequences of the locomotives of different types are exchanged. These sequences, however, have to be exchanged again prior to the stopping time of the rescheduling. The Japanese freight train operator imposes an operational constraint that, to a convergence task of a certain locomotive in its planned sequence of tasks, only locomotives with the same type as the locomotive may be assigned. There is no time and place for the re-exchange in the short rescheduling period while there is a chance of it after the 36 hours.

Table III – Rescheduling results and computation time (average of 10 trials)

| No. | Rescheduling period (h) | # Locos whose tasks changed | $Z^{IP}$ | Gap (%) | $\ell$ | $\sum \mid P_\ell^k \mid$ | Time (s) | $(RMP_\ell)$ | $(SP_\ell)$ | $(IP)$ | Max. time (s) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 36 | 6.0 | 1,200.0 | 0.00 | 6.0 | 8,921.3 | 0.9 | 0.4 | 0.5 | 0.0 | 1.0 |
| 1 | 48 | 6.0 | 1,200.0 | 0.00 | 7.8 | 9,546.6 | 1.6 | 0.6 | 0.9 | 0.0 | 1.9 |
| 1 | 60 | 6.0 | 1,200.0 | 0.00 | 7.1 | 9,630.1 | 2.2 | 0.6 | 1.5 | 0.0 | 2.4 |
| 1 | 72 | 6.0 | 1,200.0 | 0.00 | 7.6 | 9,735.6 | 3.7 | 0.9 | 2.9 | 0.0 | 4.3 |
| 2 | 36 | 16.0 | 5,335.0 | 0.00 | 22.5 | 10,340.0 | 3.8 | 2.6 | 1.2 | 0.0 | 4.2 |
| 2 | 48 | 14.0 | 5,235.0 | 0.00 | 27.6 | 11,638.6 | 7.2 | 4.1 | 3.1 | 0.0 | 8.6 |
| 2 | 60 | 14.0 | 5,120.0 | 1.23 | 27.3 | 12,494.1 | 12.3 | 5.0 | 6.2 | 1.2 | 17.0 |
| 2 | 72 | 14.0 | 4,840.0 | 0.46 | 23.5 | 13,597.5 | 15.3 | 4.8 | 9.2 | 1.3 | 18.6 |
| 3 | 36 | infeasible | - | - | 22.3 | 10,795.2 | 3.9 | 2.6 | 1.3 | 0.0 | 4.9 |
| 3 | 48 | 18.0 | 8,430.0 | 0.00 | 23.8 | 12,561.6 | 6.7 | 3.7 | 3.1 | 0.0 | 7.7 |
| 3 | 60 | 16.0 | 7,850.0 | 0.00 | 26.2 | 14,850.6 | 12.1 | 5.4 | 6.7 | 0.0 | 13.4 |
| 3 | 72 | 17.0 | 7,710.0 | 0.00 | 35.2 | 17,867.0 | 25.9 | 11.0 | 14.9 | 0.0 | 32.4 |
| 4 | 36 | infeasible | - | - | 11.7 | 9,572.7 | 1.6 | 1.0 | 0.6 | 0.0 | 1.9 |
| 4 | 48 | 20.4 | 5,560.0 | 0.00 | 18.6 | 10,898.3 | 3.7 | 1.8 | 1.8 | 0.0 | 5.8 |
| 4 | 60 | 20.8 | 5,560.0 | 0.00 | 29.0 | 14,185.5 | 10.9 | 5.4 | 5.5 | 0.0 | 13.2 |
| 4 | 72 | 20.4 | 5,240.0 | 0.00 | 37.7 | 17,784.2 | 23.9 | 10.9 | 13.0 | 0.0 | 27.8 |
| 5 | 36 | infeasible | - | - | 3.0 | 8,728.0 | 0.5 | 0.1 | 0.4 | 0.0 | 0.5 |
| 5 | 48 | 45.0 | 16,160.0 | 0.25 | 25.2 | 12,678.5 | 7.4 | 3.8 | 2.4 | 1.2 | 9.0 |
| 5 | 60 | 43.5 | 15,632.0 | 1.77 | 28.2 | 16,073.3 | 14.4 | 6.8 | 5.2 | 2.3 | 16.5 |
| 5 | 72 | 44.4 | 14,362.0 | 0.52 | 39.7 | 18,787.6 | 28.8 | 13.1 | 13.0 | 2.7 | 32.3 |

Table IV – Rescheduling solution summary of case No. 3

| No. | Rescheduling period (h) | # Extra inspections | # Locos whose original task not assigned at end of rescheduling period | # Locos assigned to tasks of different loco type |
|---|---|---|---|---|
| 3 | 48 | 6 | 10 | 6 |
| 3 | 60 | 7 | 8 | 5 |
| 3 | 72 | 7 | 4 | 7 |

Table V – Computation time without set-covering relaxation (average of 10 trials)

| No. | Rescheduling period (h) | $\ell$ | $\sum \mid P_\ell^k \mid$ | Time (s) |
|---|---|---|---|---|
| 4 | 36 | 9.7 | 10,182.5 | 1.8 |
| 4 | 48 | 26.2 | 16,761.0 | 9.2 |
| 4 | 60 | 49.4 | 30,262.0 | 42.3 |
| 4 | 72 | 82.0 | 48,383.0 | 161.7 |
| 5 | 36 | 3.0 | 8,728.0 | 0.5 |
| 5 | 48 | 27.5 | 16,263.9 | 10.5 |
| 5 | 60 | 45.6 | 25,857.5 | 33.8 |
| 5 | 72 | 68.5 | 35,357.2 | 84.2 |

The number of locomotives whose sequence of tasks modified does not decrease for the case No. 3, 4 and 5 though we take the rescheduling period longer, while the objective function value does. This outcome is possible since we do not directly minimize the number but do the sum of rescheduling penalties divided into the five types. Table IV presents the number of unscheduled extra inspections (the value times the cost $W_3$ is included in $Z^{IP}$ as a part of the total cost) in the rescheduling solution for the case No. 3. The number of locomotives assigned to a different convergence task from the planned locomotive (the value times $W_4$ is in $Z^{IP}$) and the number of locomotives whose rescheduled sequence of tasks includes a task originally hauled by a different locomotive type (the value is relevant to the cost $W_5$) are also displayed in the table. The value concerning the penalty $W_4$ gets smaller for a longer rescheduling period as there are more opportunities for a locomotive to be assigned to its planned convergence task. By comparing the 72-hour result with the 60-hour one, we see that four more locomotives are assigned to their own convergence tasks by involving one additional locomotive rescheduled.

All the solutions are obtained in ten seconds with the rescheduling period being 48 hours and within around 30 seconds for 72 hours, which is acceptable enough for the freight train operator. The operator will be able to even output and compare different solutions by adjusting the cost values within the admissible time for the locomotive rescheduling. The short computation time also enables the operator ready for the re-adjustment of the timetable arising from further disruptions. It should be noted that the short computation time is not achieved without our set-covering relaxation; Table V presents the number of column generation iterations and the computation time for the cases No. 4 and 5 where $I_0 := I \cup D$, $\widehat{I_\ell} := \hat{I}$, i.e., the instances are solved without set-covering relaxation. The values of $Z_\ell$ decrease slowly, and it causes the enumeration of many feasible paths that are not selected in the solutions. Our relaxation approach speeds up the computation time by a factor of six at a maximum.

## 5.3. Uncovered Train Detection Results

For the cases where the rescheduling is identified as infeasible, we have carried out the uncovered train detection. Table VI shows the number of tasks uncovered and the computation time. It is observed that there exist feasible paths to all the locomotives for every case. Bigger values of $f_*$ is set to the convergence tasks since the haulage of the trains after the rescheduling period has expired is more important. We then have results such that some unimportant, short-distance trains in the rescheduling period are uncovered. Exactly optimal or almost optimal solutions are obtained in terms of the objective function. It takes more computation time to solve the problem than to do the rescheduling problem in spite that there are approximately 10,000 feasible paths enumerated in advance. 27% to 77% of overall time is spent for finding the integer solutions. This observation indicates that the rescheduling algorithm is superior to the uncovered train detection algorithm for the feasibility decision of locomotive re-assignment.

Table VI – Uncovered train detection results (average of 10 trials)

| No. | Rescheduling period (h) | # tasks uncovered | Gap (%) | $m$ | $\sum \mid P_m^k \mid$ | Time (s) | $(RMP_m)$ | $(SP_m)$ | $(IP)$ |
|---|---|---|---|---|---|---|---|---|---|
| 3 | 36 | 3.0 | 0.01 | 3.6 | 411.9 | 7.0 | 1.1 | 0.5 | 5.4 |
| 4 | 36 | 1.0 | 0.00 | 5.5 | 705.3 | 3.0 | 1.2 | 0.5 | 1.3 |
| 5 | 36 | 12.3 | 0.01 | 8.3 | 2,044.4 | 4.7 | 2.8 | 0.6 | 1.3 |

# 6. CONCLUSIONS

This paper has presented a study on the optimization of freight train locomotive rescheduling under a disrupted situation in the daily operations in Japan. In the light of the current framework of the dispatching processes that the passenger railway operators have the authority to modify the whole timetables, the adjusted timetable is given for the freight train operator. We solve the locomotive rescheduling problem for the given adjusted timetable in which we change the assignment of the locomotives to the trains as needed, taking into account the periodic inspection of the locomotives as well as the inspection capacity. It is simultaneously decided whether all of the trains can be exactly covered by the available locomotives, which is *NP*-complete. The uncovered train detection problem that selects unassigned trains of less importance is solved only if the rescheduling has failed. Based on our network representation of the disrupted situation, we formulate the two problems as the integer programming problems and solve them by column generation. We apply the set-covering relaxation to the locomotive rescheduling problem, which has set-partitioning constraints. The column generation subproblem that has the inspection constraint is solved in polynomial time. Numerical experiments have been carried out by using the real timetable of long-distance trains, the locomotive scheduling plan and major disruption data including the case where at least 17% of the locomotives must have been rescheduled. The results indicate that our algorithms provide the solutions within approximately 30 seconds for the cases with 72-hour goal for recovery involving more than 1,100 train tasks. The set-covering relaxation speeds up the computation time by a factor of six at a maximum. The quality of the solutions is also satisfactory since the objective function values are within 1.8% of optimality and the number of locomotives rescheduled seems to be comparable to the number of those manually rescheduled by the dispatchers. For the small disruption case, our solution is assessed by the experienced workers of the actual dispatching processes and it is favorably evaluated.

We are now developing a computer-aided rescheduling system for practical use with the algorithms presented in the paper installed. The system also includes the freight driver rescheduling algorithm by Sato and Fukumura (2010). Meanwhile, the timetable modification algorithm by Tomii et al. (2005) handles the timetable of freight trains in addition to those of the passenger trains. Message passing between their algorithm and ours is left for future study.

## ACKNOWLEDGMENT

The authors thank anonymous reviewers for their valuable comments. The data for the numerical experiments are partially provided by courtesy of Mr. Eiki Shigeta and Mr. Hiroshi Kawakami of the Japan Freight Railway Company. The authors also thank them for assessing our rescheduling solution.

## REFERENCES

Caprara, A., Kroon, L., Monaci, M., Peeters, M. and Toth, P. (2007). Passenger railway optimization. In: Handbooks in Operations Research and Management Science: Transportation (Barnhart, C. and Laporte, G., Eds.), Vol. 14, pp. 129–187. Elsevier, Amsterdam.

Clausen, J., Larsen, A., Larsen, J., and Rezanova, N. J. (2010). Disruption management in the airline industry - concepts, models and methods. Computers and Operations Research, 37, 809–821.

Desaulniers, G., Desrosiers, J. and Solomon, M. M., Eds. (2005). Column Generation. Springer, New York.

Huisman, D. (2007). A column generation approach for the rail crew re-scheduling problem. European Journal of Operational Research, 180, 163–173.

Huisman, D. and Wagelmans, A. P. M. (2006). A solution approach for dynamic vehicle and crew scheduling. European Journal of Operational Research, 172, 453–471.

Japan Freight Railway Company (2006). Operational information. http://www.jrfreight.co.jp/i_daiya/index.html (in Japanese).

Jespersen-Groth, J. (2009a). Decision Support for the Rolling Stock Dispatcher. Doctoral Thesis, Informatics and Mathematical Modelling, Technical University of Denmark.

Jespersen-Groth, J., Potthoff, D., Clausen, J., Huisman, D., Kroon, L., Maróti, G. and Nielsen, M. N. (2009b). Disruption management in passenger railway transportation. In: Robust and Online Large-Scale Optimization (Ahuja, R. K., Möhring, R. H. and Zaroliagis, C. D., Eds.), Lecture Notes in Computer Science, 5868, 399–421.

Kamada, T. (2009). Hakuchizu KenMap Ver. 8.4. http://www5b.biglobe.ne.jp/~t-kamada/CBuilder/kenmap.htm (in Japanese).

Kroon, L., Huisman, D., Abbink, E., Fioole, P.-J., Fischetti, M., Maróti, G., Schrijver, A., Steenbeek, A. and Ybema, R. (2009). The new Dutch timetable: the OR revolution. Interfaces, 39, 6–17.

Li, J.-Q., Mirchandani, P.B. and Borenstein, D. (2007). The vehicle rescheduling problem: model and algorithms. Networks, 50, 211–229.

Lübbecke, M. E. and Desrosiers, J. (2005). Selected topics in column generation. Operations Research, 53, 1007–1023.

Maróti, G. and Kroon, L. (2005). Maintenance routing for train units: the transition model. Transportation Science, 39, 518–525.

Nemhauser, G. L. and Wolsey, L. A. (1988). Integer and Combinatorial Optimization. John Wiley & Sons, New York.

Railway Freight Association (2006). The 2006 Japan Freight Railway Company Timetable. Railway Freight Association, Tokyo (in Japanese).

Rezanova, N. J. and Ryan, D. M. (2010). The train driver recovery problem - a set partitioning based model and solution method. Computers and Operations Research, 37, 845–856.

Sato, K. and Fukumura, N. (2010). An algorithm for freight train driver rescheduling in disruption situations. Quarterly Report of RTRI, 51, 72–76.

Tomii, N., Tashiro, Y., Tanabe, N., Hirai, C. and Muraki, K. (2005). Train rescheduling algorithm which minimizes passengers' dissatisfaction. In: Innovations in Applied Artificial Intelligence (Ali, M., Esposito, F., Eds.), Lecture Notes in Artificial Intelligence, 3533, 829–838.

Walker, C. G., Snowdon, J. N. and Ryan, D. M. (2005). Simultaneous disruption recovery of a train timetable and crew roster in real time. Computers and Operations Research, 32, 2077–2094.

Keisuke SATO* and Naoto FUKUMURA.
Transport Information Technology Division, Railway Technical Research Institute,
2-8-38 Hikari-cho, Kokubunji-shi, Tokyo 185-8540, Japan.
Tel.: +81-(0)42-573-7311, Fax.: +81-(0)42-573-7305.
E-mail: keisato@rtri.or.jp (Keisuke SATO).
(*: Corresponding author.)