

# An algorithm for train-set scheduling based on probabilistic local search

P. Zhao<sup>1</sup>, N. Tomii<sup>2</sup>, N. Fukumura<sup>2</sup>, T. Sakaguchi<sup>2</sup>

<sup>1</sup>*Northern Jiao-Tong University, P.R.China / Railway Technical Research Institute, Japan*

<sup>2</sup>*Railway Technical Research Institute, Japan*

## Abstract

The Train-Set Scheduling (TSS) is one of the most important tasks in railway field. In fact, it is constrained by many maintenance conditions, station capacity and other factors, and is a NP-hard problem. This paper focuses on an algorithm to quickly work out an approximate optimal schedule. The TSS work is divided into two sub-problems: the Train-Set Regular Inspection (TSRI) problem and the Train-Set Connecting (TSC) problem. The TSC is transformed into a Travelling Salesperson Problem (TSP) on a network called a TSS network, where the nodes correspond to trains and the arcs correspond to connections of trains, and a weight expressing the desirability of connection is put on each arc. In our algorithm, first, a regular inspection plan is made and then, a Hamilton tour is found. If the Hamilton tour satisfies the constraints concerning daily inspection, it could represent a feasible train-set schedule. Therefore, when finding a new Hamilton tour based on the local search method, the algorithm considers not only the connection of nodes, but also the inspection regulations. Based on the design, we have developed an approximation algorithm, and through experiments using actual data, we have proved practical train-set schedules can be obtained quickly.

## 1 Introduction

The Train-Set Scheduling (TSS) problem is to make a schedule of train-sets in railway. In a train-set schedule, when each train-set is assigned to a particular train, and when /where train-sets are maintained are prescribed [1]. A train-set schedule is made at the same time when train schedules are renewed, or when

they are changed due to the operation of seasonal trains, etc.

Train-set schedules play an important role in railway in connection with realizability of train schedules. That is, a train schedule prescribes arrival/departure times at stations for each train, and the train-set schedule provides a stable assurance for suitable good condition of train-set for each train.

Although from 70's, interactive systems for making train-set schedule have been developed [2], no attempt to automate practical train set schedules are reported. Hence, train-set schedules are still made manually, which is quite a demanding work requiring a lot of labor, time and experience.

The crew scheduling in railway and aircraft routing are considered to be similar to TSS problem. In practice, the rules of application and maintenance of train-sets are different from those problems. For example, in general, aircraft maintenance is done only at night in airport (or base), but in case of train-set, maintenance is done during daytime. Also, crews start and end their duty at the same location, but in case of train-set scheduling, the starting and the ending locations are usually different. Thus, the algorithms for crew scheduling and/or aircraft routing cannot be used in TSS problem effectively [3][4][5].

The train-set scheduling (TSS) problem is divided into two sub-problems, the train-Set Regular Inspection (TSRI) problem and the Train-Set Connecting (TSC) problem. The TSC problem can be transformed into a kind of the Travelling Salesperson Problem (TSP) on a network called TSS network we introduce in this paper. The TSS network mainly depends on the train timetable that has been inputted. A Hamilton tour in the TSS network represents a train-set schedule under the condition of satisfying the constraints concerning the inspections. After deciding criterion of evaluation for train-set schedule, the TSS is transformed to a problem to find a Hamilton tour, which satisfies all the constraints while minimizing evaluation.

Many different algorithms for TSP have been published, but there is no one always efficient for real problems. Especially, the TSS problem has characteristics such as, complicated constraints and evaluation measures. So, those algorithms cannot be used effectively in real applications. In this paper, an algorithm for TSS problems based on the probabilistic local search is proposed. The algorithm consists of two phases: making a regular inspection plan and finding Hamilton tours. Moreover, the algorithm considers not only the connection of nodes, but also the maintenance constraints when finding Hamilton tours. It has been proved that such algorithm is practically applicable for railway through several experiments using actual data.

This paper describes an outline of TSS problem by using a simple example and then introduces an efficient algorithm based on the probabilistic local search. It also deals with the applicability of the algorithm to actual TSS problems.

## 2 Railway Train-Set Scheduling

### 2.1 An outline of TSS problem

Fig.1 shows a diagram of trains of one day that prescribes stations of

departure/arrival, times of departure/arrival for all trains. A line denotes a train and a number near a line represents the train's number. A train-set schedule is made based on this train diagram.

Fig.2 shows an example of a train-set schedule made from the given train schedule shown in fig.1. A row in fig.2 represents a schedule for one train-set on a day, which is called "a duty." For example, Duty 1 shows that a train-set runs between Station C and Station B as train No.2, then daily inspection is done for the train-set there, then the train-set is driven to station C as train No.1, and again it is driven to station B as train No.8. The train-set driven to station B as train No. 8 in duty1 runs as train No.4 from Station B in duty2 on the following day. On the other hand, the train-set driven to station C as train No. 7 will be operated as train No. 2 on the following day. Therefore, the train diagram is realized by using three train-sets. In this example, the number of the arriving trains at station B is not the same as the number of the leaving ones. This is called timetable unbalance in this paper. A feasible train-set schedule cannot be made if timetable unbalance is occurring in the original timetable. In such cases, some deadhead trains have to be set up. In this example one deadhead between Station C and Station B is set up in duty 3.

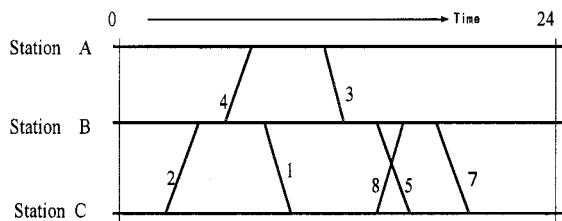


Fig.1 An example of train diagrams

The regular inspection and the daily inspection, which have been imposed by regulation must be considered when making train-set schedules. The daily inspections have to be performed frequently (usually every 72 or 96 hours) involving a visual inspection of all the major systems and it takes about three or four hours. After 30,000km running, the regular inspection is conducted, and it takes about 6 hours or more.

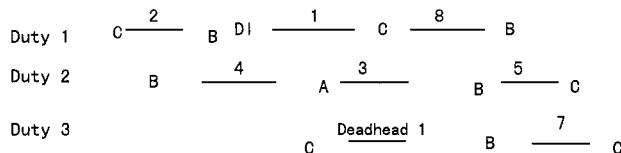


Fig.2 An example of train-set schedules

## 2.2 Constraints of TSS problem

The following conditions have to be taken into a consideration in solving the

TSS problem:

(1) Concerning the train schedule

Trains prescribed in the given train schedule must be assigned to a train-set with particular departure/arrival stations, as well as the departure/arrival times.

(2) Concerning duty

**Station:** the arrival station of the previous train must be the same as the departure station of the next train, when they are connected. When the numbers of trains arriving at a station is not the same as the numbers of trains leaving from the station, a deadhead must be set up so that this constraint is satisfied.

**Turns time:** for turning signal or others, the interval time between the departure time of the previous train and the arrival time of the next train must be larger than the prescribed time (turns time).

**Numbers:** the number of the train-sets used in a train-set schedule cannot be larger than a given number.

**Capacity:** the number of trains staying in a station at the same time cannot be larger than a given number (usually the number of tracks in the station).

(3) Concerning maintenance

**Location:** the daily inspection and the regular inspection can only be done at particular stations (or bases).

**Period:** the daily inspection interval must not exceed the prescribed period, and so does the regular inspection.

**Standard times:** a certain time is needed to perform the daily inspection and the regular inspection.

**Time range:** the daily inspection and the regular inspection must be done during the prescribed time range of a day.

## 2.3 Evaluation Criteria of TSS problem

Many train-set schedules exist which satisfy the constraints described above, while the numbers of train-set used and the deadhead may be different in each schedule. Using as small number of train-sets as possible is expected due to the cost and there are other criteria which mainly derive from cost conditions. We set as the criteria for evaluation of train-set schedules as follows (1) Number of the train-set used; (2) Number of the daily inspection; (3) Number of the regular inspection; (4) Number of the deadhead; (5) Running distance of the deadhead. Of course, these measures should be as small as possible.

## 3 Algorithm based on the Probabilistic Local Search

### 3.1 An outline of TSS problem

If trains are regarded as nodes as show in fig.2, it is indicated that a train-set schedule is actually a Hamilton tour. After structuring a TSS network (details will be introduced in 3.2), if a Hamilton tour satisfies all the constraints described in 2.2, then, the Hamilton tour should be a feasible train-set schedule

that can be used in real applications. Moreover, our objective is to find a Hamilton tour, which has the minimum evaluation measure described in 2.3.

For above purpose, the TSS problem can be considered as a kind of TSP, but differs from general TSP in several aspects. It has characteristics such as, it is a TSP on directed graphs, it has many constraints and complicated evaluation measures.

A lot of methods for solving real world TSP have been proposed, such as simulated annealing, genetic algorithms, and neural networks, etc. At present, there is no efficient method for some particular cases. However, although there is also some attempts of using these methods to solve TSS, these methods were not efficient to TSS problem solving. From online test results, it is found that if it only considers the connection of nodes, it is very rare that the Hamilton tour satisfies the constraints concerning the regular inspection and the daily inspection. Based on this fact, we propose an algorithm that can consider the constraints when generating a new Hamilton tour.

In the algorithm, first, a regular inspection plan is tentatively fixed, then the algorithm tries to find Hamilton tours that satisfy daily inspection constraints while minimizing evaluation measures. That is, the algorithm is mainly divided into two phases. Fig. 3 shows the outline of the algorithm.

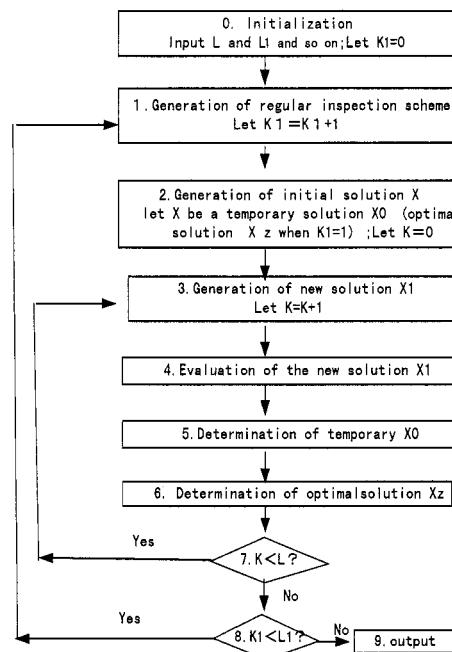


Fig.3 Outline of the algorithm

In Fig.3, L1 is the numbers of the regular inspection plans assumed. L is the number of iteration to find Hamilton tours for one regular inspection plan.

Step2 ~ Step7 are based on the probabilistic local search method, and Step decides whether or not to generate a new regular inspection plan. The initial solution is different in regular inspection plans, so the algorithm is regarded as the multi-start probabilistic local search.

### 3.2 TSS network

#### (1) Basic definitions

A TSS network is described as  $(V, E)$ , where  $V$  is a set of nodes;  $E$  is a set of arcs, a weight is assigned to each arc. The following describes these definitions.

A node corresponds to a train, therefore,  $V = \{v_i | i = 1, \dots, n, n \text{ is numbers of train}\}$ .

An arc,  $e = (v_i, v_j)$ , corresponds to a connection of trains, which means a train-set connection between train  $i$  and train  $j$ .

A weight,  $w_{ij}$  to  $e = (v_i, v_j)$ , is defined as described in formula (1)

$$w_{ij} = \begin{cases} t_{jd} - t_{ia} + p'_{ij}, & (t_{jd} - t_{ia} \geq \alpha * T_{ia \rightarrow jd} + T_{s_{ij}}) \\ 1440 + t_{jd} - t_{ia} + p''_{ij}, & (t_{jd} - t_{ia} < \alpha * T_{ia \rightarrow jd} + T_{s_{ij}}) \end{cases} \quad (1)$$

Where,  $t_{jd}$  : departure time of train  $j$ ;  $t_{ia}$  : arrival time of train  $i$ ;  $T_{ia \rightarrow jd}$  : running time from arrival station of train  $i$  to departure station of train  $j$ ;  $\alpha$ : parameter for deadhead,  $\alpha \geq 1$ ;  $p'_{ij}$ ,  $p''_{ij}$  : additional times for deadhead;  $T_{s_{ij}}$  : turns time at arrival station of train  $i$ .

As observed, the weight is also based on the period as connection time. In making train-set schedules, a long connection time is not desirable; therefore, the arc weight  $w_{ij}$  mainly expresses degree of the expectation of connection between train  $i$  and train  $j$ .

#### (2) Handing of deadhead

If the arrival station of train  $i$  is not the same as the departure station of train  $j$ , the arc  $e = (v_i, v_j)$  does not represent a direct connection; and only means that the connection will need a deadhead. It is not desirable to have so many deadheads because deadheads are non-service to passengers. Therefore, a considerably big weight value is put when the connection needs deadhead.

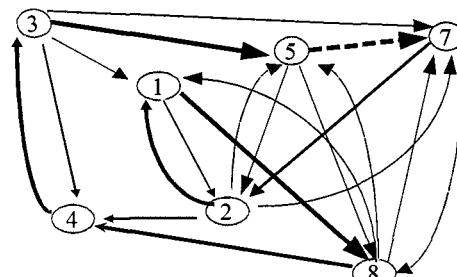


Fig.4 An example of TSS networks

### (3) An example of the TSS network

Fig.4 shows a TSS network based on a train schedule described in fig.1, but it does not give the arcs that need deadhead besides the arc  $e_{5,7}$  to avoid the picture becomes too complicated. The thick line is a Hamilton tour that represents a train-set schedule, and the dotted line means the arc needing deadhead.

### 3.3 Generation of the regular inspection plan

We first calculate the total running distance of trains for the given train schedule. Based on the interval of the regular inspection imposed by regulation, we can calculate the necessary number of the regular inspection which have to be performed in a train set schedule. We denote the number  $m$ , and search nodes which satisfy the time range and the location condition. The set of such nodes are denoted by  $M$ .

$m$  nodes are selected from the set  $M$  randomly, and are marked. The weights of arcs of the marked nodes are amended as described in formula (2). This means that when a node connects to a marked node, the connection time can satisfy the standard time for the regular inspection.

$$w_{ij} = \begin{cases} t_{jd} - t_{ia} + p'_{ij}, & (t_{jd} - t_{ia} \geq \alpha * T_{ia \rightarrow jd} + T_R) \\ 1440 + t_{jd} - t_{ia} + p''_{ij}, & (t_{jd} - t_{ia} < \alpha * T_{ia \rightarrow jd} + T_R) \end{cases} \quad (2)$$

Where,  $T_R$  is the standard time for the regular inspection, others is same as formula (1).

### 3.4 Initial solution

#### 3.4.1 Generation of Hamilton tours

Using the greedy method, an initial Hamilton tour as the initial solution is obtained. Detailed process is as follows:

(1) Search the node  $s$ , which is the earliest departure train from a maintenance station, let it be  $i$ .

(2) Select a node, which will be visited from  $i$  directly based on the following rules.

-Not Visited;

-  $\text{Min } \{w_{ij} \mid e_{ij} \in E\}$

Let the selected node be  $i$ .

(3) Repeat step 2 until all the nodes are visited

Let  $i_0 = s$ , without loss of generality, we denote the initial solution as:

$v_{i_0}, v_{i_1}, v_{i_2}, \dots, v_{i_{n-2}}, v_{i_{n-1}}$ .

#### 3.4.2 Generation of daily inspection plan

In fact,  $v_{i_0}, v_{i_1}, v_{i_2}, \dots, v_{i_{n-2}}, v_{i_{n-1}}$  is a Hamilton tour. If a node  $v_{i_k}$  satisfies the

constraints concerning the time and the location for the daily inspection, and  $w_{i_k i_{k+1}} \geq T_D$  ( $T_D$ : standard time for daily inspection) exists in the Hamilton tour, the daily inspection can be done in node  $v_{i_k}$ , set  $D = \{v_{i_k}\}$ .

Assuming the previous daily inspection is done at node  $v_{i_{d_0}}$ , the next daily inspection can be done at a node where it satisfies  $\max\{\sum_{d=d_0} w_{i_d i_{d+1}} < T_{DP} \mid v_{i_d} \in D\}$  ( $T_{DP}$  is period for the daily inspection).

When  $\{\sum_{d=d_0} w_{i_d i_{d+1}} > T_{DP} \mid \text{no } v_{i_d} \in D\}$  then the daily inspection plan cannot be made, in this case, the number of times of the daily inspection is made big enough for the value meaning of penalty.

### 3.5 Generation of a new solution

#### 3.5.1 Generation of a new Hamilton tour

In order to generate another new solution from the current solution, we have to pay attention to the inspection conditions.

- (1) A node is selected from a current solution, and let the node be node  $B$ . The selection method is as follows:
  - (a) If the current solution satisfies all the constraints of the daily inspection, then a node is selected from all nodes of network randomly.
  - (b) If the current solution does not satisfy the constraints of the daily inspection, there must be a sequence in the solution, whose length is bigger than  $T_{DP}$ , and the daily inspection is not done in it. Therefore, a node is randomly selected from the sequence; which satisfies the constraints concerning the location and the time range for the daily inspection.
- (2) Let the nodes from  $s$  to  $B$  in the current solution be “the visited nodes,” and let others be “the unvisited nodes.”
- (3) A node is selected from the unvisited nodes, and let this node be node  $C$ , and the selection method is as follows:
  - (a) If the current solution satisfies all the constraints concerning the daily inspection, then a node is randomly selected from all unvisited nodes.
  - (b) If the current solution does not satisfy the constraints concerning the daily inspection, select a node from set:  $\{v_j \mid v_j \text{ is not visited and } w_{Bj} \geq T_D\}$ .
- (4) Connect the node  $B$  to node  $C$ .
- (5) Select a node to connect node  $C$  in the same method of the generation of the initial solution, and let this node be node  $i$ .
- (6) Select the next node in the same methods as the way used in generating the initial solution, until all the nodes are visited.

### 3.5.2 Generation of a new Hamilton tour

The method is the same as the one described in 3.4.2

### 3.6 Evaluation of solutions

The evaluation measure of a solution is calculated by formula (3)

$$f(x) = \sum_{i=1}^4 p_i t_i \quad (3)$$

where:  $t_1$  is the numbers of train-set used,  $t_2$  is the numbers of the daily inspection planned,  $t_3$  is the numbers of deadhead,  $t_4$  is the running distance of deadhead,  $p_i$  is the weight for  $t_i$ 's respectively.

### 3.7 Determination of a temporary solution

Let  $X_1$  be the new solution, let  $X_0$  be the temporary solution. Comparing evaluation measure of  $X_1$  with  $X_0$ 's

- (1) If the evaluation measure of  $X_1$  is smaller, then let  $X_1$  be  $X_0$ ;
- (2) If the evaluation measure of  $X_1$  is bigger, then let  $X_1$  be  $X_0$  with a small probability.

In order to prevent solution from falling into the local optimum solution, a worse solution is accepted with a certain probability

### 3.8 Determination of an optimum solution

Let  $X_z$  be the optimum solution found so far, and compare the evaluation measure of  $X_1$  with  $X_z$ 's.

- (1) If evaluation of  $X_1$  is smaller, let  $X_1$  be  $X_z$ ;
- (2) If evaluation of  $X_1$  is bigger, do nothing.

## 4 Results of experiments

Experiments were made by using the actual train schedule data of three lines, each of which has its own characteristics. Various elements of the data used for the experiment are shown in table 1. *Numbers of train-set* denotes the numbers of train-set used in the actual train-set schedules. *Average running distance* is the average distance (km) per train. *Unbalance* means that the input train schedule is unbalanced.

The numbers of maintenance base and station are different in three cases. The numbers of trains are also different in three cases. Three cases can be considered as a small scale (Case-1), a middle scale (Case-2) and a large scale (Case-3) respectively.

Table 1 Data used in the experiments

	Number of stations	Numbers of maintenance base	Numbers of train	Input data	Numbers of train-set	Average running instance
Case 1	15	3	138	Unbalance	25	971
Case 2	11	2	283	Unbalance	34	547
Case 3	21	4	432	Unbalance	50	560

We have conducted ten experiments for each case, and proved all the constraints were satisfied every time. It was also confirmed that the numbers of train-sets in the solutions of our algorithm are the same as the numbers of train-sets in the real train-set schedule.

It was also confirmed that the numbers of the daily inspection almost coincides with the actual train-set schedules. Deadheads had to be set up because the input train timetable data were unbalanced, and we have confirmed that in each of the three cases, deadheads were set up properly. The execution times were approximately 12 minutes, 13 minutes, 28 minutes in each case respectively.

## 5 Conclusions

The purpose of this study is to develop an algorithm for making train-set schedules automatically, and such goal has been achieved primarily. First, the TSS problem was divided into two sub-problems, TSRI and TSC, and TSC problem was considered as a kind of TSP. Then we have proposed an algorithm based on the probabilistic local search. By using actual data, it was confirmed that the algorithm produces practical solutions quickly. The experimental results do agree with the actual data very well.

## References

- [1] Norio Tomii, *Invitation to railway system* (in Japanese), Kyoritu publishing company, Tokyo, 2001.
- [2] Norio Tomii: *Railway and computer* (in Japanese), Kyoritu publishing company, Tokyo, 1998.
- [3] Takashi Sakakuchi and Naotugu Nozue: Crew Roster Scheduling Based on Constraint Logit (in Japanese), *RTRI Report*, Vol.10, No.4, pp29-34, 1996
- [4] Ram Gopalan and Kalyan T. Talluri: The Aircraft Maintenance Routing Problem, *Operations Research*, Vol.46, No. 2, pp260-271, 1998
- [5] Cynthia Barnhart and Natashia L. Boland: Flight String Models for Aircraft Fleeting and Routing, *Transportation science*, Vol.32, No.3, pp208-220, 1998.