Method for Extracting Knowledge of Train Rescheduling from Data of Operation Records

Shunichi TANAKA

Satoshi KATO

Transport Operation Systems Laboratory, Signalling and Transport Information Technology Division

Takashi SAKAGUCHI

Tomoharu TAKIMOTO

Transport Operation Systems Laboratory, Signalling and Transport Information Technology Division (Former)

When train schedules are disrupted, rescheduling is carried out to recover delays. Although we have developed computational methods for generating train rescheduling plans, incorporating the knowledge of experienced dispatchers into the method remains an unresolved issue. In our previous research, we proposed a method for extracting knowhow as a set of association rules from past dispatcher decision data and to reflect it in train rescheduling algorithms using mathematical optimization. This paper presents a method and verification results, where rules are routinely updated and rule extraction efficiency is improved.

Key words: train rescheduling, mathematical optimization, operational dispatching rule, PrefixSpan method

1. Introduction

When train schedules are disrupted due to accidents, bad weather, or other causes, train schedules are modified to enable operations to recover. Train dispatchers need to have the experience and knowledge to make decisions on how to change train schedules based on their understanding of the disruptions and characteristics of each line. We proposed a method for creating rescheduling solutions according to a single quantitative evaluation index, such as total delay time or minimal passenger inconvenience [1]. However, it is difficult to reflect the knowhow of dispatchers, and policies and characteristics of each railway (hereinafter referred to as “know-how”) with a very simple index like this.

Therefore, we improved our method by extracting this know-how from past operational records, to develop a series of operational dispatching rules, and by creating an operational dispatching plans, based on these rules [2]. However, applying this method to real lines revealed a number of problems. For example, the extraction of synonymous rules and excessive processing time to extract rules. In addition, it is also necessary to have a method which updates the rules as operational records accumulate on a daily basis. In this paper, we report on the solutions to these problems and the results of verification tests in a Japanese urban area.

2. Method and issues and in the previous research method

2.1 Outline of the method in the previous research

Figure 1 shows the overview of the method for creating train rescheduling plans using operational dispatching rules. The method consists of two parts: one part extracts data and accumulates know-how as explicit knowledge in the form of rules, and the other part then reflects the rules as constraints in a train rescheduling algorithm (e.g., an algorithm based on mixed integer programming) for rescheduling plans.

2.1.1 Extracting operational dispatching rules

The first step in extracting rules is to estimate when and how a plan was changed. This is done by comparing the actual operational records with planned operational schedules in chronological order.

Fig. 1 Reflecting knowhow in the train rescheduling plan

After that, the time series of the delay situation and the operational forecast are reproduced and stored as sequential data, at each time when the plan was changed. Then, the rules with a high applicability rate (association rules) are extracted as “operational dispatching rules” from the accumulated sequential data. The details of the rule extraction method are described in Section 2.3.

2.1.2 Train rescheduling with operational dispatching rules

In order to reschedule operational plans, the current operational status is obtained from the operation control system and the future operating situation is predicted. The system then checks to see if any conditions in the predicted operating scenario match the conditions of an operational dispatching rule. If matching rules exist, constraints are generated for the matched events (trains, stations, and arrival/departure events) to satisfy the consequent part. In this way, it is possible to generate a train rescheduling plan that reflects the operational dispatching rules by adding constraint conditions and solving the rescheduling algorithm with a simple indicator such as total delay time.

2.2 Data structure of the operational dispatching rules

As shown in Fig. 2, an operational dispatching rule consists of three parts: a graph pattern with a graph structure, a part with conditions representing the implementation condition, and a part repre-

sending the consequent series of operational arrangements. The composition of each part is explained below.

2.2.1 Graph pattern

The interrelationship of each event targeted by the implementation condition and the train operation arrangement is represented by a directed graph. Nodes represent arrivals and departures (including passing through) of each train at each station, and arcs represent ordinal relations between nodes. Each subgraph, which consists of a certain number of nodes or less, is called a graph pattern. Each operational dispatching rule is specified for one graph pattern.

2.2.2 Implementation condition (condition part)

When the combination of the attributes and the values of individual nodes and arcs that are elements of a graph pattern, is considered as one condition, the trigger to implement the arrangement is represented by the combination of multiple conditions. The attributes of a node include the line, station, number of lines used, arrival and departure identification, suspension identification, and delay status. The attributes of an arc include the type of arc and the existence of the delay propagation from the origin node to the destination node of the arc. The type of arc distinguishes ordinal relationship between the same train and between different trains.

2.2.3 Series of operational arrangements (consequent part)

The consequent part consists of permutations of the arrangements. The data of each arrangement has the same structure as the implementation conditions and represents the conditions that must be satisfied by the operational forecast after the arrangement. There can be some operation arrangements in which the graph structure itself changes, such as a change in train order or change in rolling stock schedule. In the case of changes in train order or in vehicle operation, the graph pattern represents the relationship of event order before a series of the operation arrangements are implemented, and the rearrangement is represented by the sequential relationship (departure nodes, arc types, arrival nodes) after it.

2.3 Algorithm for extracting operational dispatching rules

2.3.1 Sequential data and association rules

This method focuses on the operational prediction (the actual operation records up to the current time and predicted train operation schedule) that changes each time an arrangement is implemented as shown in Fig. 3. In the graph network representing the operational prediction, suppose that the subgraph x_1 , represented by a small-scale graph pattern, changes its graph structure as $x_2, x_3 \dots$ with the passage of time.

Then Time series data $x = [x_1, x_2, \dots, x_n]$ is created for graph patterns below a certain size at each time, and a data set X is created with each time series data x as an element. The number of series data x in the sequential data set X is called the support of x and is denoted by $\text{sup}(x)$. The ratio, of the sequential data $x = [x_1, x_2, \dots, x_n]$ with the conditional part x_1 and the consequent parts $x_2, \dots, x_n$ together to sequential data in which the conditional part x_1 appears, is called the confidence of the series, and is denoted as $\text{conf}(x)$. The minimum support of a series x is denoted by “minsup,” and the minimum confidence is denoted by “minconf.”

In a sequential data set with the same graph pattern, if the ele-

Fig. 2 Data structure of operational dispatching rules

Fig. 3 Cutting out series data using graph patterns

ments x of the series data set satisfies the conditions of equations (1) and (2), the series $x = [x_1, x_2, \dots, x_n]$, where the conditional part is x_1 and the consequent parts are $x_2, \dots, x_n$, is defined as operational dispatching rule. This is because it can be interpreted as “if x_1 is predicted, then $x_2, \dots, x_n$.”

$$\text{sup}(x) \geq \text{minsup} \quad (1)$$

$$\text{conf}(x) = \frac{\text{sup}(x)}{\text{sup}(x_1)} \geq \text{minconf} \quad (2)$$

Equation (1) expresses that the support t of the series x is larger than or equal to minsup. Equation (2) indicates that the confidence of series x is larger than or equal to the standard value of minimum confidence (minconf). In general, a rule that satisfies these conditions is called an association rule [3].

2.3.2 A method for extracting operational dispatching rules from sequential data sets

The proposed algorithm for extracting operational dispatching rules is as follows.

- (1) For each graph pattern p consisting of a number of nodes less than or equal to a given maximum number N , the sequential data with graph pattern p is searched and the operational dispatching rules with the graph pattern p are extracted from the sequential data by following procedure.
- (2) The PrefixSpan method [4], one of the sequential pattern mining, is used to perform a depth-first search for conditional parts that satisfy the minimum support. In the following, we will refer to the sequential depth-first search for elements to be added to the conditional part from the series data using the PrefixSpan method as “conditional

part extension” and the similar search for the consequent part as “consequent part extension.”

- (3) Whenever a conditional part is searched in step (2), the consequent part extension is applied to the searched conditional part (the consequent part is empty at this time). The condition is that the obtained series $x \Rightarrow y$ (denoting the series where x is the conditional part and y is the consequent part) satisfies the criteria of the association rule.
- (4) If one series $x \Rightarrow y$ satisfying the criterion in (3) is obtained, this series is put on hold as a candidate rule and consequent part extension is continued. If an operational dispatching rule with an extended consequent part is output during extension, the rule candidate is discarded because it is redundant. The redundancy of rule B with respect to rule A means that whenever the conditions of Rule A are satisfied, those of Rule B are also satisfied, and those of the consequent of Rule A contains the consequent of Rule B. In the following, we will refer to the operational dispatching rules in this relationship as “redundancy rules.” When no rule is obtained in the extension, the rule candidate is output as an operational dispatching rule.
- (5) If no operational dispatching rule is obtained after the consequent part extension, the conditional part extension in (2) is restarted. If an operational dispatching rule is output as a result of the consequent part extension, the conditional part extension is suspended to avoid redundancy, and the search path is retracted to move another conditional part extension.

2.4 Issues in the previous research

In the previous research, the following issues existed in the step of extracting the operational dispatching rules, so it was insufficient to apply them to practical scale lines.

- Even after implementing the countermeasures for redundant rules with step (4) and (5) in the algorithm described section 2.3, many redundant rules are still generated.
- Even in the case of a relatively small-scale line, it takes a long calculation time to extract the operational dispatching rules.
- There is no update method that allows the extracted operation dispatching rules to be changed by accumulating daily operation results.

3. Method to avoid redundancy rules

In order to make wide use of the operational dispatching rules, it is desirable to eliminate redundant rules as much as possible. In this chapter, we describe a method to avoid redundant rules.

3.1 Method for inclusion relations within the same graph pattern

In step (4) and (5) in the algorithm of Item 2.3.2, the redundancy rules are dealt in the same graph pattern, but it only checks the inclusion relation of the operational dispatching rules that is located in the relationship between ancestors and descendants on the search tree. Figure 4 shows an example of a search tree in the PrefixSpan. When $ab \Rightarrow xy$ is detected as an association rule, $ab \Rightarrow x$ is not output due to the redundancy avoidance in the consequent part extension. The search trees after the conditional part abc are not searched ei-

ther because of the redundancy avoidance in the conditional part extension.

However, in the search tree shown in Fig. 4, $abc \Rightarrow x$ is redundant with respect to $ac \Rightarrow x$, the generation of redundant rules for such a relationship cannot be avoided. In the case of a relationship such as this, it is necessary to check and delete redundant operational dispatching rules after the extraction of operational dispatching rules for each graph is completed.

Fig. 4 Search tree in the PrefixSpan method.

3.2 Measures in inclusion relations between different graph patterns

As a result of the inclusion of nodes that are not involved in the operational dispatching rules in the graph pattern, the operational dispatching rules that have an inclusive relationship or exactly the same operational dispatching rules from different graph patterns can be extracted as shown in Fig. 5.

In such a case, the avoidance method can be taken by identifying the root node, leaf node, and other nodes (intermediate nodes) of the graph for each graph pattern, as shown in Fig. 6. The root node defines the attributes (line, station, etc.) of the operational dispatching rule. While the root node is a necessary node in defining where the rule applies, the intermediate node is considered to be a node used to represent the positional relationship between two distant nodes used in the rule. Therefore, the redundancy rules can be excluded by setting the constraint that all leaf nodes other than the root node must be used in the rule.

3.3 Redundancy rules which cannot be logically determined from graph structure

In the redundancy rules between different graph patterns, there are some rules which are redundant even though they are not included relations in the graph structure. For example, the graph patterns (1) and (2) in Fig. 7 have no inclusion relationship as a graph structure, but the framed subgraph matches both (1) and (2). Therefore, an operational dispatching rule equivalent to the one generated from the graph pattern in (1) can also be generated from the graph pattern in (2). However, it is very difficult to logically find that they represent the same operational dispatching rule from the graph structure and the search algorithm.

3.4 Effects of the method of redundancy rules

In order to verify the change in the number of operational dispatching rules extracted with and without the redundancy rule avoidance method, the series data set was generated by reproducing the time series of predicted schedules based on the records during

the operating arrangement. In order to generate the series data set, we used the records of 72 arrangements over two days in about 20 stations including single lines. The arrangement to be covered is the change of station tracks and the change of the order of single line use. The parameters were set to $N = 3, 4$, $\text{minsup} = 2$, and $\text{minconf} = 1.0$. The execution results are shown in Table 1. The reason why the number of operational dispatching rules is larger than the actual number of arrangements is that there are multiple implementation conditions for one operation arrangement series. The results show that most of the rules extracted before the countermeasure are redundant rules, which confirms that the countermeasure for redundant rules is working.

4. Improving the efficiency of the operational dispatching rule extraction algorithm

We confirmed that the algorithm for extracting the operational dispatching rules shown in Chapter 2 required an enormous amount of computation time and was not suitable for large lines or large disturbances. This chapter explains how to use the PrefixSpan method for efficient search in the extraction of operational dispatching rules.

4.1 Search condition for consequent part in the process of conditional part extension

In step (2) in the algorithm of Item 2.3.2, the following theorem can be derived from the confidence condition and the inclusion relation of the condition part (defined as x^*) obtained by the condition part extension of the condition part x .

“When R is defined as the value obtained by dividing the maximum support of the rule candidates of x by the minimum confidence, if the support of x^* is greater than R , then there is no operational dispatching rule with x^* as the conditional part.”

By obtaining R at the time of the consequent part extension of x , it is possible to determine whether or not there is an operational dispatching rule for the searched conditional part x^* by the above theorem at the time of the conditional part extension of x . If it is determined that no rule exists, the search can be accelerated by omitting the consequent part extension.

4.2 Resumption condition of conditional part extension in the process of consequent part extension

In step (4) in the algorithm of Item 2.3.2, the following theorem can be derived from the condition of maximum support of candidate rules stored during the consequent part extension. “If the maximum support of a candidate rule resulting from the extension of the consequent part of condition x is less than the minimum support, then there is no operational dispatching rule in the subsequent condition part extension of x .”

Therefore, if the conditions of the above theorem are satisfied in the consequent part extension, the search area can be narrowed by suspending the conditional part expansion instead of resuming it, which can speed up the process.

4.3 Verifying the effectiveness of algorithm efficiency

The effect of the algorithm efficiency improvement was confirmed for the lines and conditions verified in Section 3.4. The implementation was done in Python on a standard 64-bit Windows PC

with 8.0 GB RAM. The results are shown in Table 2. This result shows that the computation time can be decreased by about 16% to 30% by improving the efficiency of the algorithm. Although it has not been verified, the PrefixSpan method is an algorithm that solves the problem by depth-first search, so it can be expected to be even faster by parallelization [5].

5. Update method of operational dispatching rules

When updating operational dispatching rules, it is still necessary to keep rules [1] which do not pass the confidence level or support level standard value or [2] no longer meet the conditions for either support or confidence, in the event that they can once again be operational dispatching rules with the addition of new cases.

Therefore, rules which can be operational dispatching rules are kept in the database with the status of “regular rule,” “suspended elimination rule,” and “candidate rule.” Also, when there is a redundant rule B for a certain rule A and A is defined as a higher-level rule of B, the existence of the higher-level rule is managed for each rule. Using this information, we have developed a mechanism to delete

Fig. 5 Redundancy rules for different graph patterns

Fig. 6 Root node and leaf nodes

Fig. 7 Event graphs that may indicate the same event

Table 1 Results of verification of effectiveness of redundancy rule reduction

	The number of rules extracted		Decrease rate
	Before counter-measures	After counter-measures	
Maximum number of nodes N=3	6,550	138	97.9%
Maximum number of nodes N=4	954,361	667	99.3%

unnecessary cases while retaining the appropriate ones.

We define a regular rule as one that satisfies the confidence and the support and has no higher-level regular rules, a suspended elimination rule as the case in [2], and a candidate rule as the case in [1] for which the confidence and the support are above a criteria value. In this research, in order to prevent the repetition of deletion and revival of operational dispatching rules, the suspended elimination rules are considered as operational dispatching rules that should be extracted. In addition to the above status classification, we also classify the existence or non-existence of higher-level rules to give the rules a status.

If there is a higher-level rule, it is not registered in the database normally because it is a redundant rule, but only when the status of the higher-level rule is 'suspended elimination', it is the lower-level rule registered in the database, and it is indicated as a "with higher-level suspended elimination rule." The reason for registering the lower-level rules at this point is to register the number of cases and counterexamples (cases where the consequent part is not satisfied when the conditional part is satisfied) of the lower-level rules when the upper-level rules are deleted by the passage of a certain period of time from the suspended elimination status, and to generate the lower-level rules seamlessly.

Therefore, if there is a higher level suspended elimination rule, it will be out of the scope of the operational dispatching rule to be extracted. The conditions of each status of the candidate rules and the transition between the statuses are shown in Fig. 8.

In order to respond to changes in the rules themselves over time, the weight of each case is not evaluated uniformly, but the weights of the cases are adjusted so that the older cases are evaluated more lightly and the newer cases more heavily, making it possible to extract rules that match the new environment from the performance data where old and new cases are mixed.

6. A trial of rule extraction on a real line

A simulation of the extraction of operational dispatching rules was conducted on 14 days of actual schedule disruptions in a real urban area.

The extraction of operation control rules was carried out for each day, and the status of the operation control rules was updated after each extraction. The parameters used for the extraction and update of the rules were $N = 4$, $\text{minsup} = 5$, $\text{minconf} = 0.8$, the minimum confidence of the candidate rules was 0.4, and the minimum support of the candidate rules was 1. Since the suspension period for elimination was set to one year for the six-month case, there were no rules that were eliminated after the suspension period expired.

Table 3 shows the number of operational dispatching rules extracted at the end of extraction and update. At the time when case 14

was completed, 65 cases (40 regular rules and 25 suspended elimination rules) were extracted. Within the scope of this trial, there were no rule candidates that corresponded to the rules with the higher level of suspended elimination rules.

Figure 9 shows an example extracted as a regular operational dispatching rule. Of the 18 cases that fit the conditional part of the rule, 15 satisfy the consequent part. In all the target graphs, the nodes represent train arrivals or departures (passing is represented as two nodes with the same time), and the dotted arcs indicate the order in which arrival and departure events occur. The conditions under which the change occurs are indicated by red letters. Therefore, this graph represents a rule with the following conditional and consequent parts.

The conditional part:

- There are two trains, one of which stops at station X and the other passes through station X.
- The passing train passes through station X after the stopping train departs.
- The passing train is delayed in the range of 3 to 15 minutes.

Table 2 Results of verification of effectiveness of algorithm efficiency

	Calculating time for rule extraction[sec]		Decrease rate
	Before applying the method	After applying the method	
Maximum number of nodes N=3	58	41	29%
Maximum number of nodes N=4	17,642	14,808	16%

The consequent part:

- The passing train passes before the arrival of the stopped train at station X.

The computation time required for the extraction varied greatly, ranging from 19 minutes to 20 hours. It took less than four hours except for the case in which the schedule was disrupted all day long due to two fatal accidents. It was found that the time required for extraction was greatly affected not only by the number of operation arrangements and time series data, but also by the shape of the graph pattern to be searched and the nature of the schedule disruption. As for the time required to update the operational dispatching rules, all of them were calculated within 10 minutes, although the time required to update the rules tended to increase as the number of cases increased. It was confirmed that the extraction and update of the rules can be done in a practical time for the case of $N = 4$, because it is sufficient to extract and update the rules every time a series of rescheduled plans have been completed (for example, for each day when a disturbance occurs). In order to extract more complex operational dispatching rules on the scale of the model line, other methods such as parallelization process are necessary, as mentioned earlier.

7. Conclusion

In order to make the method of extracting the knowhow of dispatchers in train rescheduling applicable to lines with practical complexity, we proposed an advanced algorithm for eliminating re-

Fig. 8 Changes in the status of the rules

dundancy and reducing calculation time, and a method for updating the algorithm on a daily basis. We confirm that the extraction and updating of the operational dispatching rules for use in organizing the operation is completed in a practical time as shown in Chapter 6. By taking these factors into consideration, it is possible to obtain the tacit knowledge of dispatchers as formal knowledge in the lines where daily actual operation data can be obtained.

This research is one part of the efforts toward the automation of creating rescheduling plan. In future work, it is necessary to develop an interface and evaluate its effectiveness, and to establish an overall framework of support for train operations.

References

- [1] Sato, K., Tamura, K., Tomii, N., "A MIP-based timetable re-scheduling formulation and algorithm minimizing further inconvenience to passengers," *Journal of Rail Transport Planning & Management*, Vol. 3, No. 3, pp. 38-53, August 2013.
- [2] Sakaguchi, T., Sato, K., "Application of Rule Mining Technology for Train-order Rescheduling," *Quarterly Report of RTRI*, pp. 172-177, August 2019.
- [3] Liu, B., Hsu, W., Ma, Y., "Integrating classification and associ-

Fig. 9 Example of extracted regular rules.

Table 3 Results of extraction and update of operation dispatching rules

	Number of arrangements	Round number of series data	Round Number of candidate rules	Number of regular rules	Number of suspended elimination rules	Number of Eliminations	Time for rule extraction [min]	Time for rule update [sec]
Day 1	55	8,100	290	0	0	0	47	2
Day 2	12	1,300	430	0	0	0	55	4
Day 3	58	10,000	1,100	0	0	13	108	15
Day 4	22	4,000	1,400	0	0	6	121	20
Day 5	36	3,700	1,500	0	0	24	94	27
Day 6	21	4,900	2,000	0	0	59	167	40
Day 7	53	9,200	2,500	0	0	138	233	62
Day 8	58	8,900	2,900	22	0	154	184	88
Day 9	27	2,900	3,000	22	0	83	19	97
Day 10	55	9,300	3,300	34	6	151	164	125
Day 11	57	9,000	4,500	37	6	182	97	171
Day 12	350	37,000	5,200	23	20	555	1212	317
Day 13	35	5,700	5,300	32	20	188	129	291
Day 14	130	13,000	5,700	40	25	322	177	299

ation rule mining," *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pp. 80-86, August 1998.

- [4] Sharma, P., Balakrishna, G., "PrefixSpan: Mining sequential patterns by prefix-projected pattern," *International Journal of Computer Science & Engineering Survey*, Vol. 2, No. 4, pp. 111-122, April 2011.
- [5] Jian, P., Jiawei, H., Behzad, M., Jianyong, W., et al., "Mining sequential patterns by pattern-growth: the PrefixSpan approach," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 16, No. 11, pp. 1424-1440, October 2004.

Authors

Shunichi TANAKA
Assistant Senior Researcher, Transport Operation Systems Laboratory, Signalling and Transport Information Technology Division
Research Areas: Train Rescheduling, Train Performance Calculation, Simulation Technology

Takashi SAKAGUCHI
Senior Chief Researcher, Head of Transport Operation Systems Laboratory, Signalling and Transport Information Technology Division (Former)
Research Areas: Railroad Operation Scheduling, Combinational Optimization, Operation Management Technologies

Satoshi KATO
Senior Researcher, Transport Operation Systems Laboratory, Signalling and Transport Information Technology Division
Research Areas: Train Rescheduling, Rolling Stock and Crew Scheduling, Mathematical Optimization

Tomoharu TAKIMOTO
Researcher, Transport Operation Systems Laboratory, Signalling and Transport Information Technology Division (Former)
Research Areas: Railroad Safety, Railway Operation