The mathematical approach for train conflict detection and resolution: In case autonomous decentralized traffic control for heavy traffic station

K.M. Kim, S.H Hong, Y.H Kim, B.H Park
Korea Railroad Research Institute, Uiwang-City, Korea
Korea National Railroad College, Uiwang-City, Korea

Abstract

In large station with high density traffic, trains can be hardly controlled by CTC but by station dispatcher because CTC has difficulties in monitoring all states of affairs happening within each station such as departures, arrivals, and shuntings of many trains from different lines and shunting trains between yards and platforms, etc. Therefore the station's dispatcher has to make quick decision about how to reschedule the times and routes for all the trains scheduled within a specific time window. And such decision becomes harder when an unexpected delay occurs because a delay occurring in a train propagates other trains as time goes on. Generally, it is called the conflict detection and resolution to adjust beforehand the distorted schedule due to a delay to original schedule. Our research is different from the state of the arts in that ours determines simultaneously the routes and the times of arrival and departure of trains, although others do only the arrival and departure time of the trains without considering the alternative routes and shunting of the station. This study suggests a mathematical approach for how to detect in advance and resolve efficiently the conflicts occurring within a station and it will be shown how to reduce delay using our approach by means of analysing the schedule of ChyungRyangRi station.

1. Introduction

In large scale station where the train operating pattern is complex and traffic density is high, trains can be hardly controlled by CTC which is used for the local or remote supervision and control of train traffic. When the departure, arrival and shunting of trains from different lines frequently interfere the main line operation, so in many cases, it is very difficult to administer the entire train operation only by the remote control at the central control center. Therefore the station's dispatcher handles local train traffic and has to make quick decision about how to reschedule the time and routes for all the trains scheduled within a specific time window. Moreover, such decision becomes harder when an unexpected delay occurs such as disturbance. We suggest a mathematical model using In-station routing graph about how to detect and resolve efficiently the conflicts occurring. Our approach determines simultaneously the routes and the times of arrival and departure of trains, although other studies do only the arrival and departure time of the trains without considering the alternative routes and shunting of the station. The objective of CDRS is to minimizing sum of weighted delays or maximum delays with satisfying feasibility of operation restrict or dispatcher's requirement.

Fig. 1 Situation of Station Dispatcher
2. Literature review

Generally, the process, the station dispatcher re-schedule the train movement and make an effort to minimize secondary delays, is called CDRS (Conflict Detection and Resolution). Study on an optimum solution to the CDRS started by Higgins et. al.[1] who developed the mixed integer program model rescheduling of trains on a single line track. Mascis et al.[4] suggest the train conflict resolution model based on Job-shop model. This model is applied to the project of MARCO (Multilevel Advanced Railways Conflict Resolution and Operation) in Europe. Kroon et al.[2] and Zwaneveld et al[6] make a route allocation model as the Maximum weight independent set. They do not have the points of view on CDRS. Lamma et al.[3] solve the train conflict resolution problem using Constraint Satisfaction Programming(CSP). CSP is an effective way to find feasible solution. However, the backtracking method on CSP is not optimization approach.

This paper suggests an branch-and-bound approach to optimize simultaneously the routes and the times of arrival and departure of trains, although other studies do only the arrival and departure time of the trains without considering the alternative routes and shunting. So, we develop the In-station routing graph representing compatible routes and setting train arrival or departure time to increase efficiency of CDRS

3. Conflict Detection and Conflict Resolution

3.1 In-Station Routing Graph

Generally, train routing is controlled by interlocking system in station. Interlocking system process traffic control eliminated interference with trains by interlocking logic and route. In this way, our In-station routing graph $G=(V, E_1, E_2)$ method also use interlocking route. Each route (include shunting route) represents as a node($V$). Connect between two nodes with the incompatibility arc($e \in E_1$) in case interlocking routes are simultaneously incompatible and add the movement arc($e \in E_2$) to indicate departure or arrival train state.

Figure 2 shows the example of network topology to explain the In-station routing graph. There are 18 routes including in-bound, out-bound and shunting. The station has 3 platform ($p_1, p_2, p_3$) and 1 yard ($y_1$). The train 1 be planned to move along the main route $r_2$ and $r_{11}$($a - b - p_2 - e - f$). In addition to, it has the alternate route $r_3$ and $r_{12}$($a - d - p_3 - g - f$). The train 2 allocates inbound route $r_4$ and out-bound route $r_7$($h - e - p_1 - b - c$). The train 3 is a shunting train. So, it has the shunting route, $r_{15}$($p_3 - d - c - b - y_1$). In this case, we make the in-station route graph like Fig. 3.

![Fig. 2 Example of network topology](image-url)
3.2 Conflict Detection

We develop two basic techniques that solve the conflict detection problem in a reasonable time. In first stage, we check the feasibility the train sequence as the sequence conflict detection method. Then, in the second stage, we attempt to calculate the delay of trains using in-station routing graph.

1) Sequence Conflict Detection

When single in/out line case, it is sufficient to fix both the sequence of incoming and outgoing trains. If both sequences are compatible with the number of tracks, an optimal assignment can be calculated in polynomial time. The notation of sequence conflict detection is as follows:

- \( S \) : station
- \( q \) : the number of platforms
- \( n \) : the number of trains
- \( A \) : incoming track segment
- \( B \) : outcoming track segment
- \( \pi_A \) : incoming sequence across the \( A \)
- \( \pi_B \) : incoming sequence across the \( B \)

For general situation, train sequences \( \pi_A : A(1) , A(2) , ..., A(n) \), \( \pi_B : B(1) , B(2) , ..., B(n) \) and platform capacity \( q \), we have to introduce the buffer arcs \( B(t) \rightarrow A(t+q) \) for \( t=1,...,n-q \). The two sequences are incompatible if and only if they create a cycle. There are no feasible solutions satisfying sequence. So, we cannot dispatch trains by First-Come First-Served rule. If this is not the case the earliest starting times for passing through segment \( A \) and \( B \) for each train can be found by longest path calculations. However, this approach will not apply to the multi-in/out line cases.

As an example of the multi-in/out case consider the network in Fig. 4. The train sequences \( \pi_A : 1\rightarrow2\rightarrow3\rightarrow4\rightarrow5, \pi_B : 2\rightarrow3\rightarrow4, \pi_C : 1\rightarrow5 \) and \( q \) is equal to 3. If we find the buffer arc to use general situation approach, \( B(t) \rightarrow A(t+q) \), there are 4 buffer arcs \( B(1) \rightarrow A(4), B(2) \rightarrow A(5), C(1) \rightarrow A(4) \) and \( C(2) \rightarrow A(5) \). The buffer arc \( C(2) \rightarrow A(5) \) makes the cycle but feasible solutions exist trivially. So, we makes the artificial outcoming sequence \( \pi_z \) be satisfied with \( \pi_b \) and \( \pi_c \) such as \( 1\rightarrow2\rightarrow5\rightarrow3\rightarrow4 \). Next,
check the incompatibility for all possible \( \pi_z \) to use general situation approach. This method has a shortcoming that the more problem size is large, the combination of possible \( \pi_z \) increases exponentially. However, the number of trains managed within a specific time window is lower than 10 in our situation.

2) Delay Conflict Detection

The dispatching rule is the First Come First Served (FCFS), which is usually adopted in railway practice. Conflict detection in FCFS situation can easily use In-station routing graph. First, make an In-station routing graph representing real-time train schedule within a specific time. Second, check satisfying feasibility of operation restricts or dispatcher’s requirements. Finally, calculate delay expansion on account of movement arc and incompatibility arc. The notation of delay conflict detection is as follows:

- \( T \): the set of delay train
- \( d_t \): conflict delay to train \( t \)
- \( P_y_p \): planned time where the train \( t \) is settable to use route \( p \) (Planned Schedule)
- \( E_y_p \): earliest time where the train \( t \) is available to use route \( p \) (Estimate Schedule)
- \( L_s \): minimum stop time of train \( t \)
- \( \delta_{pq} \): minimum interlocking time where consecutively setting up two routes, \( p \) and \( q \)

<Conflict Detection Algorithm>

Step 1 (Initializing) : Make the In-Station Routing Graph on train schedule with in a specific time window

Step 2 (Estimate Schedule) : \( E_y_p = P_y_p + d_t \) for all \( \forall t \)

Step 3 (Delay Extension lead to E1)

\[
E_y_p = \max\{ E_y_p, \max_{t \in T} (P_y_p + L_s) \}
\]

where \( p \) : in-bound route, \( q \) : out-bound route

Step 4 (Delay Extension lead to E2)

Where there are two incompatible route \( p \) and \( q \) and \( | E_y_p - E_y_q | \leq \delta_{pq} \)

\[
E_y_p = E_y_p + \delta_{pq} \text{ if } E_y_p > E_y_q
\]

\[
E_y_q = E_y_q + \delta_{pq} \text{ otherwise}
\]

Step 5 (Calculate Total Delay) : \( \sum_p \sum_t (E_y_p - P_y_p) \) for all \( \forall p, \forall t \)

Step 6 (Terminating) : Terminate algorithm

3.3 Conflict Resolution

We develop the conflict resolution algorithm formulated as an Integer programming. Our objective is to minimize maximum delay or minimize total weighted day. This model basically consists of independent set constraint on In-station routing graph and setting up train arrival or departure time. Integer programming model commonly has a shortcoming that the more problem size is large, the more problem is difficult to find out the optimal solution. However, trains schedule managed within a specific time window is lower than 10 in a practice field. Our model becomes a small problem having around 100 variables and applies to the branch and bound approach[5].

1) Notation

- \( t \): the Train index
- \( p \): the route index
- \( \text{Inbound Route}(t) \): the available set of in-bound routes of train \( t \)
- \( \text{Outbound Route}(t) \): the available set of out-bound routes of train \( t \)
- \( \text{Route}(t) = \text{Inbound Route}(t) \cup \text{Outbound Route}(t) \)
- \( \text{Incompatible}(p) \): the set of routes which be incompatible with route \( p \)
- \( \text{CommonPf}(p) \): the set of routes which have the platform in common route \( p \)
2) Parameter
- \(d_t\): conflict delay to train \(t\)
- \(P_{yt}\): planned time where the train \(t\) is settable to use route \(p\) (Planned Schedule)
- \(E_{yp}\): earliest time where the train \(t\) is available to use route \(p\) (Estimate Schedule)
- \(\delta_{pq}\): minimum interlocking time where consecutively setting up two routes, \(p\) and \(q\)
- \(W\): non zero large constant
- \(U_{st}\): maximum stop time of train \(t\)
- \(L_{st}\): minimum stop time of train \(t\)

3) Variables
- \(x_{pt}\) = 1 if train \(t\) allocate route \(p\), 0 otherwise
- \(y_{pt}\): the time where the train \(t\) is setting up route \(p\).
- \(z_{pqij}\) = 1 if the time of setting route \(p\) to train \(i\) is faster than route \(q\) to train \(j\)
- 0 otherwise

4) Mathematical Model
\[
\min D = \sum_{p} \sum_{t} w_{pt} (y_{pt} - P_{yt}) \tag{1}
\]
subject to
\[
\sum_{p \in \text{Inbound\_Route}(t)} x_{pt} = 1 \quad \forall t \tag{2}
\]
\[
\sum_{p \in \text{CommonPf}(p)} x_{pt} = x_{pt}, \quad p \in \text{Inbound\_Route}(t), \quad q \in \text{Outbound\_Route}(t) \tag{3}
\]
\[
y_{pt} - y_{qj} \geq \delta_{pq} - W(1 - z_{pqij}), \quad q \in \text{incompatible}(p), \quad \forall p \in \text{Route}(i), \quad \forall q \in \text{Route}(j) \tag{4}
\]
\[
y_{pt} - y_{qj} \geq \delta_{pq} - Wz_{pqij}, \quad q \in \text{incompatible}(p), \quad \forall p \in \text{Route}(i), \quad \forall q \in \text{Route}(j) \tag{5}
\]
\[
y_{pt} \leq W \cdot x_{pt}, \quad p \in \text{Route}(t), \quad \forall t \tag{6}
\]
\[
y_{pt} \geq E_{yp} \cdot x_{pt}, \quad p \in \text{Route}(t), \quad \forall t \tag{7}
\]
\[
L_{st} \leq \sum_{q \in \text{Outbound\_Route}(t)} y_{pt} - \sum_{p \in \text{Inbound\_Route}(t)} y_{pt}, \quad \forall t \tag{8}
\]
\[
D \geq \sum_{p \in \text{Route}(t)} y_{pt} \cdot \sum_{p \in \text{Route}(t)} P_{yp}, \quad \forall t \tag{9}
\]
\[
x_{pt}, z_{pqij}: \text{binary variable}, \quad y_{pt}: \text{integer variable} \tag{10}
\]
Constraint (2-3) ensures that train has just on in-bound route and out-bound route. Constraint (4-5) specifies that the incompatible routes should be keep up the interlocking time, \(\delta_{pq}\). Constraint (6) represent coupling constraint between two variables, \(x_{pt}\) and \(y_{pt}\). Constraint (7) states that the \(y_{pt}\) must be bigger than \(E_{yp}\). Constraint (8) gives an upper and lower bound on the possible dwell time at the station. Constraint (9) needs when the objective function is to minimize maximum delay.

4. Experimental Results
Our experiments are based on the ChyungRyangRi area, a congested area of the Korea railway network including the subway station. The schedule of ChyungRyangRi station has daily 400 schedules excluding shunting schedules, various train patterns stop and go, starting or terminating and train types such as subway, a passenger train and a freight train. We generate hard instance (Table. 1) and confirm that our algorithm outperforms the FCFS when disturbance occurred.(K0522 delay 160sec, A1804 delay 80sec)
### Table. 1 Instance of experiment

In Table 2, we look at the experiment result. We can see that row 2, 3 or 4 summarizes the performance of our solutions with respect to those provided by the current operation, column 2. The total amount of weighted delay is approximately 24% less with respect to FCFS. In addition to, maximum delay time was decreased to 25%. The branch and bound algorithm finds the optimal solutions within average 1.5 seconds.

<table>
<thead>
<tr>
<th>Order of Control</th>
<th>FCFS</th>
<th>Optimization</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Single Route</td>
</tr>
<tr>
<td>Total Delay Time(sec)</td>
<td>1140</td>
<td>1050</td>
</tr>
<tr>
<td>Max. Delay Time(sec)</td>
<td>240</td>
<td>180</td>
</tr>
<tr>
<td>Num of Compatibility</td>
<td>11</td>
<td>9</td>
</tr>
<tr>
<td>Computing Time(sec)</td>
<td>0.17</td>
<td>1.25</td>
</tr>
</tbody>
</table>

**5. Conclusions**

This paper describes an effective optimization algorithm for the CDRS within large station with high density traffic when disturbance happen to train. Experimental results demonstrate that our approach relax schedule confusion, improve reliability and reduce recovery time. Hence, our algorithm may be able to yield significant improvements in the quality of railway business when we compute exactly estimated train arrival and departure time. For these reasons, many researchers recently study autonomous decentralized traffic control. We expects that our conflict detection and resolution models lead to effective traffic control and save traffic resources.

### References

