Robust single-track train dispatching model under a dynamic and stochastic environment: a scenario-based rolling horizon solution approach

Lingyun Meng
Lecturer
School of Traffic and Transportation
Beijing Jiaotong University
1201, SiYuan Building, No.3 ShangYuanCun, HaiDian District, Beijing 100044, China
Tel.: (86)-10-51688547
Email: lymeng@bjtu.edu.cn

Xuesong Zhou
(Corresponding Author)
Assistant Professor
Department of Civil and Environmental Engineering
University of Utah
122 South Central Campus Dr., 210 CME
Salt Lake City, UT 84112-0561, USA
Tel.: (801)-585-6590
Fax: 801-585-5477
Email: zhou@eng.utah.edu

First submission: August 1, 2010
Revised version: February 23, 2011

Submitted for publication in Transportation Research Part B

Abstract

After a major service disruption on a single-track rail line, dispatchers need to generate a series of train meet-pass plans at different decision times of the rescheduling stage. The task is to recover the impacted train schedule from the current and future disturbances and minimize the expected additional delay under different forecasted operational conditions. Based on a stochastic programming with recourse framework, this paper incorporates different probabilistic scenarios in the rolling horizon decision process to recognize (1) the input data uncertainty associated with predicted segment running time and segment recovery times, and (2) the possibilities of rescheduling decisions after receiving status updates. The proposed model periodically optimizes schedules for a relatively long rolling horizon, while selecting and disseminating a robust meet-pass plan for every roll period. A multi-layer branching solution procedure is developed to systematically generate and select meet-pass plans under different stochastic scenarios. Illustrative examples and numerical experiments are used to demonstrate the importance of robust disruption handling under a dynamic and stochastic environment.

Keywords train dispatching, disruption handling, rolling horizon decision making, stochastic optimization

1. Introduction

Capable of transporting a large volume of freight and passenger flows with efficient fuel consumption, railroad plays a vital role in the future sustainable transportation system. To maintain and
further improve their competitive advantages in the rapidly changing multimodal transportation market, railroad industries continuously strive to offer more efficient and reliable services. As tactical plans for the railroad service, train timetables are programmed and updated every year or every season to define routes, orders and schedules of trains running in railroad corridors. In the daily operation stage, train dispatchers need to carry out the tactical plans and recover impacted train schedules from disruptions. For a high-density railroad corridor with limited capacity, the real-time dispatching task becomes extremely complicated, and ineffective disruption handling could significantly downgrade the punctuality and reliability of train services and the overall system performance.

1.1 Literature review on train scheduling

In the area of train scheduling, a wide range of studies are devoted to optimization model formulation and algorithm development. Szpigel (1973) proposed the first mathematical model on the train scheduling problem and applied a branch-and-bound solution method to find timetables that minimize the total transit time, subject to overtaking and crossing headway constraints. Jovanovic (1989) introduced a general formulation of the train dispatching problem as a mixed integer program that minimizes tardiness cost. Jovanovic and Harker (1990) and Jovanovic and Harker (1991) further proposed the SCAN (Schedule ANalysis) system for tactical scheduling of trains and maintenance operations. Carey and Lockwood (1995) presented a mixed integer program and solution algorithms for the train dispatching problem on a double-track rail line with trains operating at different speeds. Carey (1994a) further proposed an extension to more general, more complex rail networks, with choice of lines, station platforms, etc. In a companion paper, Carey (1994b) proposed an extension from one-way to two-way rail lines. A simulation model and an improved analytical line delay model were developed by Hallowell (1993) to analyze an optimizing train dispatching process in scheduled rail operations under uncertainty. Brannlund et al. (1998) introduced a Lagrangian relaxation approach to determine a profit maximizing schedule, in which track capacity constraints are relaxed and profit is measured by estimates of the value of running different types of services at specified times. Oliveira and Smith (2000) and Oliveira (2001) formulated the train scheduling problem for a single track railroad network as a job shop scheduling problem. Recently, Tornquist and Persson (2007) proposed an optimization approach for rescheduling railroad traffic in an N-tracked network. Based on alternative graph reformulation and branch-and-bound solution algorithms, D’Ariano (2008) designed and implemented a decision support system for train dispatchers, namely ROMA, to recover schedules from disturbances automatically. By taking wait-depart decisions and priority decisions into account, Schachtebeck and Schöbel (2010) proposed an integer programming formulation to solve the delay management problem that aims to minimize inconvenience of passengers. By extending optimization models proposed by Zhou and Zhong (2007) and Tornquist and Persson (2007), Castillo (2011) further incorporated user preference into the train timetabling problem.

Given a pair of conflicting trains, simple priority rules have been widely used to determine which train to schedule next by utilizing underlying objective functions, for instance, the total delay for the minimal travel time criterion (Kraay and Harker, 1995), the schedule deviation and tightness for the minimal deviation criterion (Chen and Harker, 1990), and the number of passengers transported (Adenso-Diaz et al., 1999). Furthermore, Dorfman and Medanic (2004) incorporated priority rules into a discrete event simulation framework to solve large-scale real-world train scheduling problems. In a
study by Zhou and Zhong (2007), the solution quality of several priority-based heuristics was evaluated by using optimal schedules obtained by a branch-and-bound method with a set of lower bound rules.

In addition to train dispatching problem under minor disturbances, disruption handling under major service breakdowns is also demanding and challenging. Although railroad accidents/incidents are considered as low-probability distributed failures, in 2007 there were still 13,580 cases in the United States alone (Railroad safety report of US Federal Railroad Administration, 2008). Aiming at major disruptions, Adenso-Diaz et al. (1999) put forward an on-line decision support model which aims to choose the most appropriate solution under certain and known capacity recovery time periods. Chikara et al. (2009) proposed a Petri net based method for optimizing train stop plans under deterministic accident duration. For a given severe disruption with deterministic duration (e.g. track washout for a few hours), Corman (2010) investigated alternative dispatching decisions, i.e. rerouting or cancelling trains, and gave detailed corresponding evaluations by means of a simulation system.

Table 1 compares our proposed approach with the recent related studies in terms of disruption characteristics and solution approaches.

<table>
<thead>
<tr>
<th>Publication</th>
<th>Disruption information</th>
<th>Disruption intensity</th>
<th>Model structure</th>
<th>Objective</th>
<th>Solution algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adenso-Diaz et al. (1999)</td>
<td>C</td>
<td>A</td>
<td>MIP</td>
<td>Maximize the number of passenger transported</td>
<td>H</td>
</tr>
<tr>
<td>Tornquist and Persson (2007)</td>
<td>C</td>
<td>I</td>
<td>MIP</td>
<td>Minimize the total final delay of the traffic or minimize the total delay costs</td>
<td>H</td>
</tr>
<tr>
<td>D’ Ariano et al. (2007)</td>
<td>C</td>
<td>I</td>
<td>MIP</td>
<td>Minimize the maximum secondary delay for all trains at all visited stations</td>
<td>B&amp;B, H</td>
</tr>
<tr>
<td>Corman, et al. (2010)</td>
<td>C</td>
<td>S</td>
<td>S</td>
<td>Multiple criteria such as minimize the total passengers’ delays</td>
<td>AG</td>
</tr>
<tr>
<td>Schachtebeck and Schöbel (2010)</td>
<td>C</td>
<td>I</td>
<td>MIP</td>
<td>Minimize the sum of all delays of all passengers at their final destinations</td>
<td>H, D</td>
</tr>
<tr>
<td>Meng and Zhou (2011)</td>
<td>U</td>
<td>A</td>
<td>MIP, SP</td>
<td>Minimize the expected additional delay under different forecasted operational conditions</td>
<td>B&amp;B, H</td>
</tr>
</tbody>
</table>

Symbol Descriptions:
Disruption information: Certain (C) / Uncertain (U).
Disruption intensity: Minor (I), in minutes / Major (A), blockage of some track around one hour / Severe (S), blockage of some track in hours.
Model structure: Mixed Integer Programming (MIP) / Computer simulation model (S), Stochastic Programming (SP).
Solution algorithm: Branch-and-bound (B&B) / Alternative graphs (AG) / Heuristic (H) / Dynamic programming (D).

From Table 1, we can see that most of the previous studies primarily assume certain or deterministic disruption information, with focuses on how to generate feasible schedules efficiently through single-stage optimization formulations and algorithms. Mixed integer programming and computer simulation techniques were typically used to model the train dispatching problem. To handle minor disturbances, the optimization models aim to minimize total delay or generalized costs of all
trains or passengers. Under seriously disturbed conditions, additional performance measures need to be considered to cancel train services or reroute trains. Due to the NP-Hard property of the train dispatching problem, many existing studies used heuristic algorithms to meet the real-time computational efficiency requirement, while branch-and-bound and dynamic programming approaches were used to produce optimal or near-optimal solutions.

Recognizing that limited attention has been devoted to the train dispatching problem in a highly dynamic and uncertain environment, this research aims to investigate an important class of train dispatching problems under random segment running time and capacity breakdown durations, which are typically caused by track washouts, signal failures, mechanical difficulties, and maintenance time periods (Krueger et al., 2000). When an incident occurs on a segment or a station, the primary task of a dispatcher is to regenerate a feasible schedule for trains that need to pass through incident-impacted segments/stations. To do so, the dispatcher first estimates when the capacity can be fully restored (i.e., the duration of capacity breakdown), and accordingly reschedules trains (with different priorities and experienced delay) so that the system-wide performance can be optimized. How to estimate the capacity loss duration is a challenging task in its own right, and the subsequent real-time dispatching task, especially on a congested single-track rail line, demands more solution-searching efforts in a highly dynamic environment. For instance, if the capacity loss lasted for an extended period, a large number of incident-impacted trains would need to be rescheduled as early as possible to form an efficient meeting-pass plan. If the track segment reopens earlier than what the maintenance crew previously predicted, the dispatcher has to quickly mobilize trains waiting at various locations to fully utilize newly recovered capacity.

It should be noted that in the train timetabling stage, a number of studies focus on designing robust timetables that can accommodate uncertainties unfolding in real-time dispatching. Carey and Kwicinski (1994) derived stochastic approximate relationships between scheduled headways and knock-on delays, which are further used to build robust timetables by explicitly considering the expected knock-on effects of headways. Moreover, Carey (1998) developed an optimization model where slack times are allocated to take into account behavioral responses and lateness costs. A two-stage with recourse stochastic programming model and stochastic shortest path techniques are used by Khan and Zhou (2010) to solve the slack time allocation problem under random disturbance. By using max-plus algebra techniques, Goverde (2010) presented a novel approach for timetable stability analysis. In a recent study by Vromans et al. (2006), the relationship between timetable reliability and timetable heterogeneity are examined via a number of simulation experiments. It has been widely recognized that a robust timetable should be able to handle minor perturbations (i.e., few minutes of delays) occurring in real-time, but determining an effective recovery schedule under dramatic non-recurring delays (i.e., major disruptions) is extremely complicated and difficult in its own right, especially for a single-track rail line involving complex meet-pass plans.

1.2 Literature review on disruption handling for road traffic systems and machine scheduling

To better understand the disruption handling and capacity loss duration estimation problem in the broader context of transportation systems, it is necessary to review several related studies in the field of highway traffic engineering. Essentially most traffic incident duration estimation methods view the duration of an incident as a random distribution, and a number of probability density functions have been calibrated in previous studies, for example, a log normal distribution (Golob et al. 1987, Giuliano,
1989, Garib et al., 1997, Sullivan, 1997 and Ozbay and Kachroo, 1999), a log-logistic distribution (Jones et al., 1991), and a Weibull distribution (Nam and Mannering, 2000). To estimate the remaining duration of an incident, one could consider a number of related incident characteristics, such as how long the incident has lasted (known as duration dependence). Nam and Mannering (2000) presented a conditional probability updating model to find the probability distribution of incident duration under certain realized duration and conditions. Ozbay and Kachroo (1999) and Smith and Smith (2001) developed decision tree-based models to predict incident duration based on classified incident type and historical data. Recognizing a variety of uncertainty factors related to traffic incidents in urban transportation networks, Sawaya et al. (2000) proposed a stochastic programming model with recourse to minimize the expected total travel time in a traffic incident management problem. In this study, a priori robust control strategies are proposed to consider multiple future time periods. A similar stochastic programming approach has been also used for a variable message sign (VMS) location model developed by Chiu et al. (2001) and Chiu and Huynh (2007).

Another important field related to train dispatching under capacity breakdowns is stochastic scheduling with machine breakdowns. Stochastic scheduling involving random machine breakdowns has emerged to be an active line of research in the machine scheduling field in the past two decades (Pinedo, 2002). Generally, there are two categories of problems addressed in the literature, depending on the impacts of a machine breakdown to the job being processed. (1) In a preemptive-resume model, if a machine breakdown occurs during the processing of a job, the work done on the job prior to the breakdown is not lost, and the processing of the job can be resumed from where it was interrupted once the machine becomes operable again (Mittenthal and Raghavachari, 1993, Cai and Zhou 1999, 2000). (2) In a preemptive-repeat model, however, if the machine breakdown occurs during the processing of a job, the work done on this job is lost, and the processing task will have to restart for this job after the machine resumes its work (Cai et al., 2003). As a delayed train in an impacted segment needs to move again from the current location, the train dispatching problem under consideration belongs to the first preemptive-resume category.

1.3 Literature review on rolling horizon solution framework and proposed approach

With a special focus on mathematical modeling and operational algorithmic design for schedule recovery strategies, this research considers the robustness of train dispatching under random segment running times and/or a segment capacity breakdown with an uncertain duration. Within a rolling horizon solution framework, we formulate robust “scheduling policies”, corresponding to a series of reactive decisions with uncertain predicted future operational conditions, as a special case of the stochastic scheduling problem with probabilistic data input. Previously, the rolling horizon solution mechanism has been used for real-time traffic signal optimization and dynamic traffic assignment studies. For example, Peeta and Mahmassani (1995) introduced a rolling-horizon solution procedure for solving the real-time dynamic traffic assignment problem for road networks. This operational method has been enhanced by Mahmassani et al. (1998) and Zhou and Mahmassani (2007) using an asynchronous multi-horizon architecture, where multiple real-time traffic estimation, prediction and simulations modules execute in different processors and communicate with each other asynchronously.

This paper aims to offer the following contributions to the growing body of research work on practically-useful and theoretically-rigorous train dispatching models. First, by proposing a stochastic programming paradigm within a rolling horizon framework, we present a mathematical formulation
which models the robust train dispatching performance measures under uncertain traffic conditions. We also develop a multi-layer branching solution procedure for systematically enumerating and evaluating alternative meet-pass plans in order to minimize the expected additional delay under different random scenarios. Through comprehensive numerical experiments, this paper further examines the solution quality and computation time of robust solutions compared to expected-value based solutions, as well as the value of perfect information measure in the context of train dispatching under a stochastic environment.

The remainder of this paper is organized as follows. In section 2, a conceptual illustration is presented for the robust train dispatching problem under capacity breakdowns. By introducing a scenario-based rolling horizon solution approach, section 3 aims to clearly describe an operational decision making process under a highly dynamic and stochastic environment. After presenting a stochastic programming model for each rolling horizon in section 4, section 5 presents a search tree-based solution algorithm to obtain the most robust meet-pass plan. Finally, the proposed model and algorithms are evaluated by numerical experiments in section 6.

2. Conceptual illustration

2.1 Train dispatching with uncertain input data

This paper considers a single-track rail line as illustrated in Fig. 1, with a series of bi-directional track segments and a single type of train. Segments are numbered as 1, 2, …, m and stations are numbered as 0, 1, …, m. Two assumptions are made with regard to the signaling system in this paper: 1) trains can follow each other on a track segment and only one train is permitted on some track segments and 2) trains can only meet and pass at station track sidings.

![Fig. 1 A single-track rail line.](image)

Fig. 2 shows a single-track train planning timetable with 4 trains and 4 stations, where inbound train a meets outbound train b at station 1. For illustrative purposes, the free-flow running time on each segment for both directions is 5 minutes (denoted by \( f \)), the minimum headway between leaving and entering times of two consecutive trains in the same segment is 2 minutes (denoted by \( h \)), and the minimum headway between arrival times of two consecutive trains at the same station is 3 minutes (denoted by \( g \)). Note that, the initial train planning timetable in Fig. 2 satisfies all the above operational and safety constraints.
The stochastic optimization model to be presented can address both random segment running times and uncertain capacity recovery times. Without loss of generality, the following discussion focuses on the capacity breakdown case as a representative major disruption.

Shown in Fig. 3, an incident occurs at minute 10 and completely blocks segment 2. Due to uncertainty in the related track inspection and recovery activities, the capacity loss duration can be viewed as a random variable and characterized by a continuous or discrete probability distribution function. For simplicity, let us consider two random scenarios: (1) scenario $w_1$ with a segment blockage duration of 15 minutes and an occurrence probability of 70% (2) scenario $w_2$ with a duration of 30 minutes and 30% likelihood.

Right after the capacity breakdown starts at minute 10, even without knowing exactly how long the segment blockage will last, the lead dispatcher (i.e. “trick dispatcher”) of the dispatching district, needs to reschedule all affected trains and inform the related locomotive drivers and dispatchers at intermediate stations, with a new train meet-pass plan that covers 10-30 minutes. At minute 30, after knowing the exact duration of the capacity loss from the track maintenance crew, he/she needs to inform the related dispatchers again with a new (i.e. the second) meet-pass plan from minute 30 to 50.

From a stochastic programming point of view, when the lead dispatcher optimizes the first meet-pass plan, he/she needs to consider a relatively long time horizon to comprehensively evaluate alternative schedules, (say from 10 minutes to 70 minutes). With a decision under uncertainty framework, we can decompose the entire optimization horizon into two periods:

1. the first period covers 10 and 30 minutes, and
(2) the second period extends from 30 minutes to 70 minutes.

The first and second periods, respectively, are defined as the roll period, and the look-ahead period defined in a rolling horizon framework as described in section 3. The first-period decision should ensure the feasibility of second-period recourse decisions (subject to capacity breakdown constraints) under all possible conditions, and it should also minimize the expected total deviation or delay of trains within the entire scheduling horizon for different random instances of capacity loss duration. Please note that, we use the term of “period” rather than the commonly used term “stage” in stochastic programming, because “stages” will be used in the proposed rolling horizon framework with a slightly different definition, specifically, “stages” in our definition can be overlapping with each other.

2.2 Robust train meet-pass plan in terms of train precedence relations

Technically, a train meet-pass plan describes how a train yields to another train when they head to the same station, and a related station dispatcher then can adjust the detailed track assignment plan according to the sequence of trains arriving at and departing from the station he/she is managing. In cases of normal operation, the train schedule is represented in train timetable format that shows the exact arrival and departure times of trains at each station. However, due to the uncertainty of random segment running times and incident durations, the desirable train dispatching model should provide a robust train meet-pass sequence (in terms of precedence relations between trains at each station). That is, after the lead dispatcher disseminates the revised train meet-pass plan to the station, if there is further segment running time delays, then arrival and departure times of affected trains will be shifted forward automatically according to the same meet-pass sequence at stations, which still strictly ensures the task dependencies and headway intervals across different segments and stations.

Without using the representation of train meet-pass sequence, the lead dispatcher has to frequently re-adjust, broadcast and coordinate different version(s) of train schedules (in terms of station arrival and departure times) when the new information on the capacity recovery becomes available. Obviously, continuously changing operational plans could make the station dispatchers difficult to respond and simultaneously carry out multiple versions of technical documents.

2.3 Expected performance of train meet-pass plans under stochastic conditions

How to construct the first revised meet-pass plan with uncertain input data is a challenging task in its own right. In this particular example, we assume that the precedence relation between trains $a$ and $b$ in segment 2 is the major decision variable in the first period (for the train meet-pass plan to be disseminated). The second period decision needs to resolve potential conflicts between other trains and trains $a$ and $b$, as well as conflicts between other trains. For simplicity, we let other trains (trains $c$ and $d$) yield to trains $a$ and $b$, if conflicts exist. If there are conflicts between $c$ and $d$, we adopt the schedule that can minimize the total delay for trains $c$ and $d$. The term “total delay” is computed by summing the delay time of each train at the final destination station and the term “conflict” means a safety headway constraint on a segment cannot be satisfied.

Let us consider the scheduling horizon covering 10 to 70 minutes. Under two capacity recovery duration scenarios, Figs. 4 and 5 show two corresponding schedules for the first-period decision where train $a$ takes reopened segment 2 before train $b$. Figs. 6 and 7 correspond schedules where train $b$ uses segment 2 before train $a$. 
Table 2 summarizes the delay of each train and the total delay for two first-period train precedence decisions, namely “train a before b” or “train b before a”, under two different scenarios. In particular, under the first scenario, the “train b before a” decision is better than the “a before b” decision by (30-38=) -8 min, while the “a before b” decision leads to less total delay by (83-87=) -4 min under the second scenario.

The most robust first period decision (i.e. meet-pass plan) in this example is the set of train precedence relations that minimizes the expected total delay over two random scenarios with the 70% vs. 30% occurrence probabilities. Shown in Table 2, decision “b before a” offers the most robust solution in terms of reducing the expected delay from 51.5 min to 47.1 min. Although the relative time saving is only about (51.5-47.1)/51.5= 8.5%, it is theoretically and practically important to find the most robust solution, particularly when the punctuality of train services is directly related to monetary costs and economic dollar values. For example, in Great Britain and the Netherlands, performance contracts between the government and train operating companies specify some minimum required punctuality levels and impose fines for poor performance (European Commission, 2001).

In addition to providing the meet-pass plan in the first period, this solution of stochastic optimization in fact is associated with two branches of remaining schedules in the second period, corresponding to the best and worst case scenarios. Each branch optimizes the total train delay for a given realized duration, subject to the predetermined meet-pass sequence from the first period. Right after the lead dispatcher knows the exact incident duration, one of the solution/schedule branches will be selected and executed as the updated meet-pass starting from minute 30.

More formally, a “solution” consists of a train meet-pass plan for the first period and a set of second-period schedules corresponding to random scenarios. The robustness of a solution is evaluated by the expected performance (expected total delay time) under stochastic conditions.

Fig. 4 second-period schedule for train a first occupying segment 2 under the best scenario, total delay: 38 minutes.
Fig. 5 second-period schedule for \textit{train a first} occupying segment 2 under the worst scenario, total delay: 83 minutes.

Fig. 6 second-period schedule for \textit{train b first} occupying segment 2 under the best scenario, total delay: 30 minutes.

Fig. 7 second-period schedule for \textit{train b first} occupying segment 2 under the worst scenario, total delay: 87 minutes.

Table 2
Delay statistics for two solutions (unit: min).

<table>
<thead>
<tr>
<th>First-period meet-pass plan</th>
<th>train (a) before (b)</th>
<th>train (b) before (a)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Best scenario ((70%))</td>
<td>Worst scenario ((30%))</td>
</tr>
<tr>
<td>(a)</td>
<td>11</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>train b</td>
<td>14</td>
<td>29</td>
</tr>
<tr>
<td>train c</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>train d</td>
<td>6</td>
<td>21</td>
</tr>
<tr>
<td>total delay</td>
<td>38</td>
<td>83</td>
</tr>
<tr>
<td>expected delay across scenarios</td>
<td>38<em>70%+83</em>30% = 51.5</td>
<td>30<em>70%+87</em>30% = 47.1</td>
</tr>
</tbody>
</table>

It is interesting to further investigate how the occurrence probability of random scenarios affects the selection of the final robust solution. Fig. 8 illustrates the expected delay as a function of occurrence probability of the best scenario (i.e. $w_1$). It’s easy to see that if the occurrence probability of the best scenario is smaller than 0.33, the first-period “train a first” decision is more robust, otherwise, the “train b first” should be recommended. Meanwhile, it is easier to select a decision at extreme cases (e.g. at 0.1 and 0.9) where the performance difference of two candidate solutions is dramatic and obvious. For example, the relative time saving is $(42.5-35.7)/42.5=16\%$ when $w_1 = 0.9$.

**Fig. 8** Expected delay as a function of occurrence probability of scenario 1 (best scenario).

### 3. Scenario-based rolling horizon solution framework

#### 3.1 Key definitions: roll period, stage horizon and look-ahead period

This section presents a rolling horizon solution framework for the robust train dispatching problem, by representing uncertain input data through random scenarios. The focus of the following description is on how to use such a solution framework to handle decision dynamics and data stochasticity associated with disruptions.

Given currently available and predicted information on the operational conditions of a rail line (including segment running times and segment blockage durations), in a computer-aided scheduling system, the robust dispatching module optimizes and disseminates the meet-pass plan periodically for every roll period. At the beginning of each roll period, the operational conditions are predicted for a period in the future (i.e. stage horizon), and the predicted information is expressed as a set of probabilistic segment running times and capacity recovery times on impacted segments.
When optimizing a meet-pass plan, the optimization program considers the whole stage horizon, which includes a roll period and a look-ahead period. The look-ahead period is defined as the time period from the end of the roll period to the end of stage horizon.

Fig. 9 illustrates three rolling-horizon stages along the time axis, and \( s = 1, 2, 3 \) represents stage index. Two random scenarios (\( w = 1, 2 \)) are considered at each stage. Each roll period covers 20 minutes, and a stage horizon spans 60 minutes, leading to a look-ahead period of 40 minutes.

Recall the example shown in Fig. 3, starting at 10 minutes, given two random samples of segment running time or capacity breakdown duration \( F(s = 1, w = 1) \) and \( F(s = 1, w = 2) \), stage 1 needs to release one meet-pass plan \( X(s = 1) \) for the first 20 minute interval (10-30 minutes) to the related terminal dispatchers. The optimization problem in this stage aims to minimize the expected delay \( z(s = 1) \) for trains traveling from 10 to 70 minutes with two look-ahead schedules \( Y(s = 1, w = 1) \) and \( Y(s = 1, w = 2) \).

Stage \( s = 2 \) starts at 30 minutes, and the robust train dispatching module will be executed once again with new predicted operational conditions \( F(s = 2, w = 1) \) and \( F(s = 2, w = 2) \), and the trains scheduled at this stage will cover a time period from 30 minutes to 90 minutes.

Considering the required computation time, the train dispatching module in a real-world scheduling system needs to be executed a few minutes before the beginning of each roll period.

![Diagram](image-url)
Compared to the conventional rolling horizon mechanism that only considers deterministic input data (i.e. single scenario), the proposed rolling horizon framework for robust train dispatching has the following unique features.

1. The optimization process for the stage horizon accepts stochastic data $F(s, w)$ represented by different random scenarios $w$, and each scenario is associated with a certain probability of occurrence and a particular instance of random segment times and capacity recovery times.

2. For the current roll period, the final optimization deliverable is a single set of train precedence relation variables, while the look-ahead period involves multiple sets of reactive schedules $Y(s, w)$ with scenario-based train precedence relation variables, each corresponding to an instance of predicted future operational conditions.

3. At stage 1, when a random instance (say from scenario 2, the worst case) is realized, the train dispatcher can select the corresponding look-ahead schedule $Y(s=1, w=2)$ to set up the initial solution for the meet-pass plan $X(s=2)$ at the roll period of stage 2, without requiring a complete re-optimization of all trains involved at stage 2. Shown as the schedule propagation arrow in Fig. 9, this represents the essential characteristics of recourse actions (for different realized scenarios) from a general stochastic programming point of view.

4. Shown as the scenario propagation arrow in Fig. 9, scenarios at stage $s+1$, e.g. ($F(s=2, w=1)$ and $F(s=2, w=2)$) can be viewed as a conditional scenario rooted from stage $s$ ($F(s=1, w=2)$). One can adapt different methods from traffic engineering (e.g. conditional probability updating models) to estimate the remaining duration of an incident, reviewed in section 1.2 of this paper. If a further and unexpected delay (e.g. 50 minutes) is reported by the maintenance crew, then the dispatcher needs to consider additional and new random scenarios at stage 2, and then perform another round of stochastic optimization.

5. Only a single robust solution of train meet-pass plan $X(s)$ at the current roll period is disseminated to the related station dispatchers and locomotive drivers, scenario-based look-ahead schedules are kept as internal data for optimizing and evaluating the expected delay over the entire stage horizon.

### 3.2 Activity sets in roll period and look-ahead period

In this study, we model the train dispatching problem as a special case of resource constrained project scheduling problem, where the project includes a sequence of $m$ activities for each train. In addition, activity $(i, j)$ represents the process of train $i$ traveling in segment $j$, and the renewable resources are entering time and leaving time for each segment. Accordingly, we can decompose all activities involved in the rescheduling problem for the whole stage horizon as two different sets: $A_R$ for train activities involved in the current roll period and $A_L$ for the activities in the look-ahead period. The meet-pass plan to be disseminated specifies the train precedence relationship for all the activities $A_R$ in the roll period. Under each stochastic scenario $w$, the robust optimization program calculates the corresponding look-ahead schedule $Y(s, w)$ for the remaining activities covered by $A_L$. Finally, the dispatcher will select the robust meet-pass plan that minimizes the expected delay over the whole stage horizon under different stochastic operational conditions.

Due to the complexity of train scheduling and the existence of random segment travel times, it might be difficult to construct a rigorous rule that defines the exact set of activities $A_R$. For example,
under one random segment running time scenario, a train activity could be placed before the ending time of a roll period, while, in another scenario with dramatic delays, this activity needs to be scheduled in the look-ahead period.

In this study, we use a pre-processing program to identify a set of train activities in the current roll period, according to the free-flow segment travel times without disturbances and the earliest capacity recovery time. Corresponding to the example in Fig. 3 with the first roll period from 10 to 30 minutes, the activity set \( A_R \) include trains \( a \) and \( b \) in segment 2, and all the other activities are considered in \( A_L \).

When handling multiple trains involved in a roll period, this heuristic rule should be able to cover a sufficient set of (i.e. an enough number of) activities required for carrying out the related meet-pass plan during the current roll period. When random segment running delays occur, some activities in \( A_R(s) \) might not be executed before the end of the current roll period \( s \), i.e. the beginning of the next roll period. In this case, we allow the optimization program to still pick up this activity into set \( A_R(s+1) \) and reschedule it for the next roll period and stage \( s+1 \).

### 4. Stochastic programming models for each rolling horizon

#### 4.1 Notations

Tables 3-4 first list general subscripts and input parameters used in the proposed model. Table 5 describes the decision variables in the proposed optimization model. The unit of all time-related parameters and variables is one minute.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( i )</td>
<td>train index</td>
</tr>
<tr>
<td>( j )</td>
<td>segment/station index</td>
</tr>
<tr>
<td>( J )</td>
<td>set of segments</td>
</tr>
<tr>
<td>( m )</td>
<td>number of segments along a rail line</td>
</tr>
<tr>
<td>( s )</td>
<td>stage index</td>
</tr>
<tr>
<td>( w )</td>
<td>random scenario index, ( w = 1, 2, \ldots, W ). ( W ) is the total number of scenarios</td>
</tr>
<tr>
<td>( h_j )</td>
<td>minimum headway between entering and leaving times of two consecutive trains in segment ( j )</td>
</tr>
<tr>
<td>( g_j )</td>
<td>minimum headway between arrival times of two consecutive trains at station ( j )</td>
</tr>
<tr>
<td>( o_i )</td>
<td>direction indicator for train ( i ), ( o_i = 0 ) for an outbound train and ( o_i = 1 ) for an inbound train</td>
</tr>
<tr>
<td>( \sigma(i, k) )</td>
<td>segment index of train ( i )'s ( k)th traveling segment, ( \sigma_{i,k} = k ) for outbound trains, ( \sigma_{i,k} = m+1-k ) for inbound trains</td>
</tr>
<tr>
<td>( M )</td>
<td>a sufficiently large positive integer</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A_R )</td>
<td>set of train activities in roll period</td>
</tr>
<tr>
<td>( A_L )</td>
<td>set of train activities in look-ahead period</td>
</tr>
<tr>
<td>( AR' )</td>
<td>set of last activity for each train involved in ( A_R )</td>
</tr>
</tbody>
</table>
\( AL' \) = set of last activity for each train involved in \( A_L \)
\( \overline{b}(i, j) \) = planned entering time (beginning time) for train \( i \) in segment \( j \) in the initial planning timetable
\( \overline{e}(i, j) \) = planned leaving time (ending time) for train \( i \) in segment \( j \) in the initial planning timetable
\( \eta_i^- \), \( \eta_i^+ \) = earliness and lateness deviation penalty, respectively, for the final arrival time difference between a schedule and the initial planning timetable
\( \kappa(i) \) = the first starting traveling segment for train \( i \) in the current rolling horizon, \( \kappa(i) > 1 \) for train \( i \) that already left its origin station, \( \kappa(i) = 1 \) train \( i \) that has not departed yet
\( \hat{b}(i) \) = current or predetermined beginning time of train \( i \) on its first starting traveling segment
\( d_{i,j} \) = minimum required station dwell time before train \( i \) entering segment \( j \)
\( j^* \) = segment with capacity breakdown
\( p(w) \) = occurrence probability of scenario \( w \)
\( f_{i,j}(w) \) = running time for train \( i \) on segment \( j \) under scenario \( w \)
\( r(w) \) = capacity breakdown ending time (i.e. recovery time) for segment \( j^* \) under scenario \( w \)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_{i,i',j} )</td>
<td>meet-pass sequence indicator in roll period. ( x_{i,i',j} = 1 ) if train ( i ) is scheduled to use segment ( j ) before train ( i' ), and otherwise ( x_{i,i',j} = 0 )</td>
</tr>
<tr>
<td>( y_{i,i',j}(w) )</td>
<td>meet-pass sequence indicator in look-ahead period under scenario ( w ). ( y_{i,i',j}(w) = 1 ) if train ( i ) is scheduled to use segment ( j ) before train ( i' ), otherwise ( y_{i,i',j}(w) = 0 )</td>
</tr>
<tr>
<td>( b(i, j, w) )</td>
<td>entering time (beginning time) to be scheduled for train ( i ) in segment ( j ) under scenario ( w )</td>
</tr>
<tr>
<td>( e(i, j, w) )</td>
<td>leaving time (ending time) to be scheduled for train ( i ) in segment ( j ) under scenario ( w )</td>
</tr>
</tbody>
</table>

**4.2 Optimization model for entire rolling horizon**

\[
\text{Min } z = \sum_{(i,j) \in A^k \cup AL} \left\{ p(w) \times \left[ \eta_i^+ \times \left( e(i, j, w) - \overline{e}(i, j) \right) + \eta_i^- \times \left( e(i, j, w) - \overline{e}(i, j) \right) \right] \right\}
\]

(1)

Capacity breakdown constraints:

\[
b(i, j^*, w) \geq r(w), \forall (i, j^*) \in A^k \cup A_L, \forall w. \tag{2}
\]

Entering time constraints on the first starting traveling segment:

\[
b(i, \sigma(i, \kappa(i)), w) \geq \hat{b}(i), \forall (i, j) \in A^k \cup A_L, \forall w. \tag{3}
\]

Segment running time constraints:
\[ e(i, j, w) = b(i, j, w) + f_{i,j}(w), \quad \forall (i, j) \in A_R \cup A_L, \forall w. \] (4)

Minimum dwell time constraints:

\[ b(i, \sigma(i, k), w) \geq e(i, \sigma(i, k-1), w) + d_{i,\sigma(i,k)}, \quad \forall (i, \sigma(i, k)) \in A_R \cup A_L, \forall w. \] (5)

Headway constraints for activities in roll period

\[ b(i, j, w) \geq e(i', j, w) + h_j - M \times x_{i,j';j}, \quad \forall (i, j) \in A_R, (i', j) \in A_R, j = 1,..., m, \forall w. \] (6)

\[ b(i', j, w) \geq e(i, j, w) + h_j - M \times (1-x_{i,j';j}), \quad \forall (i, j) \in A_R, (i', j) \in A_R, j = 1,..., m, \forall w. \] (7)

Headway constraints for activities in look-ahead period

\[ b(i, j, w) \geq e(i', j, w) + h_j - M \times y_{i,j';j}(w), \quad \forall (i, j) \in A_L, (i', j) \in A_L, j = 1,..., m, \forall w. \] (8)

\[ b(i', j, w) \geq e(i, j, w) + h_j - M \times (1-y_{i,j';j}(w)), \quad \forall (i, j) \in A_L, (i', j) \in A_L, j = 1,..., m, \forall w. \] (9)

Headway constraints for one activity in roll period and another activity in look-ahead period

\[ b(i, j, w) \geq e(i', j, w) + h_j, \quad \forall (i, j) \in A_L, (i', j) \in A_R, j = 1,..., m, \forall w. \] (10)

The objective function includes the expected schedule deviation penalty for the last activities involved in the roll period and look-ahead period, i.e. \(AR^+\) and \(AL^+\), respectively. One can also construct a nonlinear penalty/cost of delay times (Carey, 1994b).

In the above optimization model for the entire rolling horizon, all the constraints are listed under different random scenario \(w=1, 2,...,W\). Constraint (2) ensures that trains do not use the lost capacity for the blocked segment \(j^*\). For each train in the rolling horizon, constraint (3) sets up the initial boundary conditions starting from its staring segment. Equation (4) links the entering time and the leaving time of a segment through a scenario-dependent segment running time. Constraint (5) connects the train activities of two consecutive segments through the minimum dwell times at a station, while positive dwell times are required for trains to perform loading and unloading services at stations. If both activities are involved in the roll period (belonging to \(A_R\)), constraints (6) and (7) enforce the minimum headway requirement between two consecutive trains running at the same segment, in opposite directions or in the same direction. For each scenario, precedence constraints (8) and (9) specify the same minimum headway requirements for two train activities in the look-ahead period \(A_L\). Furthermore, in the case of one activity in roll period and another activity in look-ahead period, constraint (10) explicitly specifies their relative priorities: activity in \(A_L\) always yields to activity in \(A_R\).

To reduce the presentation complexity, the above model does not formulate two practical considerations: (1) acceleration and deceleration time losses at stations/sidings, and (2) the minimum headway \(g\) between arrival times of two consecutive trains at station \(j\), considered in the previous illustrative example in section 2. On the other hand, the proposed model can be easily extended to
handle those additional requirements, and we refer interested readers to our previous studies (Zhou and Zhong (2005), Zhou and Zhong (2007)) on the related modeling aspects.

### 4.3 Problem Decomposition

As shown in constraints (6) and (7), all scenarios share the same set of the train precedence variables $X$ in the roll period. On the other hand, each scenario has its own precedence indicator vector $Y(w)$, in constraints (8) and (9).

<table>
<thead>
<tr>
<th>Table 6</th>
<th>Vector representation for conceptual models.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>$F(w)$</td>
<td>random input data vector for scenario $w$, consisting of stochastic random segment travel time and capacity breakdown duration $F(w) = [f_{i,j}(w), r(w)]$</td>
</tr>
<tr>
<td>$X$</td>
<td>first-period meet-pass sequence vector, $X = [x_{i,j}(w)]$ in roll period</td>
</tr>
<tr>
<td>$Y(w)$</td>
<td>second-period meet-pass sequence vector under scenario $w$, $Y(w) = [y_{i,j}(w)]$ in look-ahead period</td>
</tr>
</tbody>
</table>

Essentially, the proposed model aims to develop the first-period robust solution $X$ with reactive schedule $Y(w)$, by incorporating a degree of anticipation of uncertainty in $F(w)$ in the optimization model for the entire rolling horizon. Using the vector representation in Table 6, the above model can be described in the following conceptual form:

$$
\text{Min } z = E_w \left[ Q_{AR}(X, w) \right] + E_w \left[ Q_{AL}(X, Y(w), w) \right] 
$$

subject to operational and safety requirement constraints:

$$
X \in \Omega(w), Y(w) \in \Phi(X, w).
$$

$Q(\cdot)$ corresponds to the cost function in equation (1). $Q_{AR}(X, w), Q_{AL}(X, Y(w), w)$ covers the scheduling performance functions for the activity sets $AR'$ and $AL'$, respectively. The reason for having scenario index $w$ in the objective function $Q_{AR}(X, w)$ is because the schedule delays for solution $X$ in the roll period is also directly affected by random segment running times and stochastic capacity loss duration. $\Omega(w)$ is the constraint set for $X$, including constraints (6,7) directly. Because train precedence relations $X$ also defines the entering and leaving times $b(i,j,w)$ and $e(i,j,w)$, the related constraints (2-5) should also be included in $\Omega(w)$. $\Phi(X, w)$ is the constraint set for $Y(w)$, defined by constraints (2-5), and (8-10) for a given $X$ under scenario $w$.

To better understand the proposed model, we compare it with the commonly used two-stage stochastic programming model (Birge and Louveaux(1997)) shown below.

$$
z = C^T X + E_w \left[ Q(X, w) \right] 
$$
s.t. $X \in \Omega$ \hfill (14)

where $Q(X, w) = \min_q q(Y(w), w)$ \hfill (15)

s.t. $Y(w) \in \Phi(X, w)$ \hfill (16)

In the standard form, the first-stage objective function (i.e. $C^T X$) is not affected by the uncertain data input, so it is easy to set up a two-stage structure as the following. The first-stage program includes equations (13, 14), and function $Q(X, w)$ becomes a recourse function, evaluated through equations (15, 16) for each individual scenario. As a result, stage-wise decomposition solution schemes, such as L-Shaped methods, are typically used to solve stochastic programming problems.

Compared to the scenario-independent $C^T X$ in Eq. (13), the first-period objective function in our proposed model $E_w[Q_{AL}(X, w)]$ still needs to be calculated over different random scenarios, and constraint set $\Omega(w)$ is also scenario-dependent.

By adapting the general two-stage stochastic programming structure, we then can transform the original problem in equations (11, 12) to a two-period structure:

**First-period problem** for roll period:

$$\text{Min } z = E_w[Q_{AR}(X, w)] + E_w[Q_{AL}(X, w)]$$ \hfill (17)

s.t. $X \in \Omega(w)$ \hfill (18)

**Second-period problem** for look-ahead period under scenario $w$

$$Q_{AL}(X, w) = \min_{q_{AL}} q_{AL}(Y(w), w)$$ \hfill (19)

s.t. $Y(w) \in \Phi(X, w)$ \hfill (20)

With $Q_{AL}(X, w)$ as a recourse function, the first-period problem only considers variable $X$, while the second-period model uses $X$ as the given input and optimizes $Y(w)$ accordingly. The reformulation needs to solve $W$ second-period problems for different scenarios. For an individual scenario $w$, given $F(w)$ as fixed segment running times and capacity breakdown duration, an instance of the second-period problem is equivalent to a standard train scheduling problem. Due to the non-convexity of the first period problem, we need to resort to an implicit enumeration method to efficiently evaluate the candidate set of feasible solutions $X$. Below is a conceptual description on a sequential solution search progress.
Phase I: Construct a restricted first-period problem

Consider a particular scenario \( w_0 \), and use the realization of uncertain segment running times and capacity breakdown duration to construct constraint set \( \Omega(w_0) \) and objective function \( Q_{AR}(X, w_0) \).

Solve the following restricted problem

\[
\text{Min } q_{AR}(X, w_0) \\
\text{s.t. } X \in \Omega(w_0)
\]

The output is a set of feasible candidate solutions \( X_l \), \( l=1, 2, \ldots, L \), where \( l \) is candidate solution index and \( L \) is the number of promising meet-pass plans.

Phase II: Solve scenario-dependent second-period problems

For each candidate solution \( X_l \) under scenario \( w \), solve the resulting standard train scheduling problem specified by equations (19, 20) using \( F(w) \). The output is \( Q_{AL}(X_l, w) \) for each scenario \( w \).

Phase III: Evaluate expected performance for both periods

For each candidate solution \( X_l \), evaluate the first-period objective function \( E_w[Q_{AR}(X_l, w)] \) under different random scenarios, and then compute the overall objective function

\[
z(X_l) = E_w[Q_{AR}(X_l, w)] + E_w[Q_{AL}(X_l, w)].
\]

Phase IV: Find system-optimal solution for both periods

Find the optimal solution \( X^* = \arg \min_l z(X_l) \), which consists of a (fixed) first-period meet-pass plan \( X_{\lambda} \), \( \lambda \) is the optimal candidate index, and a vector of the second-period optimal solution \( Y_{\lambda}^*(w) \) for each scenario \( w \):

\[
[ X_{\lambda}, Y_{\lambda}^*(w = 1), Y_{\lambda}^*(w = 2), \ldots, Y_{\lambda}^*(w = W) ]
\]

5. Multi-layer branching algorithm

As both periods of the proposed model involve implicit solution enumeration processes to resolve potential conflicts in partial schedules, this section proposes a branch-and-bound algorithm to systematically obtain optimal solutions. Formal descriptions on the use of branch-and-bound methods in solving general stochastic integer programs can be found in papers by Ahmed et al. (2004) and Beraldi and Ruszczyński (2002). In the following discussions, we set \( \eta_i^+ = 1 \) and \( \eta_i^- = 0 \) in the objective function (1) to represent linear penalty for delays.

5.1 Structure of branch-and-bound solution tree

The branch-and-bound algorithm is build upon on a multi-layer solution search tree structure. An
optimal solution is obtained through implicitly enumerating all feasible schedules in the solution tree. Corresponding to the example in Figs. 3-7, Fig. 10 illustrates a search process with three major blocks. The node index in each block is denoted as $n$.

(1) The “candidate solution generation” block contains nodes $c_n$ and enumerates meet-pass precedence decisions in the roll period, and it might have several layers of node branches for conflict resolving.

(2) The “scenario generation” block contains scenario nodes $s_n$ and lists different random capacity breakdown durations, with a single layer of scenario nodes.

(3) The “reactive schedule generation” block contains nodes $r_n$ and covers several layers of conflict-resolving nodes in the look ahead period, for a given realized capacity breakdown duration.

Both “candidate solution generation” and “reactive schedule generation” blocks contain conflict-resolving nodes, which are generated by resolving conflicts between trains. A scenario node carries over the information from a conflict-resolving node in the last layer of the first block period, and further specifies a realized random scenario for schedule generation in the look-ahead period.

In Fig. 10, a conflict between trains $a$ and $b$ in the current schedule first leads to two choices to resolve the conflict: train $a$ first or train $b$ first, corresponding to two conflict-resolving nodes: $c_1$ and $c_2$. Node $c_1$ is further branched into two scenario nodes $s_{11}$ and $s_{12}$, corresponding to the best and worst scenarios in terms of capacity breakdown duration in segment 2. In the third block, reactive schedules are generated through a branch-and-bound algorithm. For instance, nodes $r_1$ and $r_2$ are generated by resolving a conflict between trains $a$ and $d$ in segment 1, and nodes $r_1$ are further branched into $r_3$ and $r_4$ for a conflict between trains $c$ and $d$. Complete schedules are obtained at nodes $r_7$, $r_8$, $r_9$ and $r_{10}$, located at the last layer of the search tree. In particular, with a schedule delay of 38 minutes, node $r_8$ offers the best schedule in the sub-tree rooted from the best-case scenario node $s_{11}$. For the worst-case scenario node $s_{12}$, it has a schedule with the least delay of 83 minutes, and the related conflict-resolving nodes are not plotted in the figure for clarity. By considering the occurrence probability of two scenario nodes $s_{11}$ and $s_{12}$, one can backtrack and calculate the expected delay at node $c_1$ as 51.5 minutes. Similarly, the expected delay of 41.7 minutes at node $c_2$ can be determined through nodes $s_{21}$ and $s_{22}$. The candidate solution “train $b$ first” at node $c_2$ is finally selected as the robust solution for the proposed real-time train dispatching problem with uncertain capacity breakdown duration.

Similar to the well-studied adaptive routing problem for multi-modal transportation applications, the final solution in our problem is an acyclic hyper-path rather than a simple path in the solution tree. A hyper-path in our study has two parts. The first part is a simple path in the “candidate solution generation” block. The second part has a set of scenario-dependent paths for individual realized instances of random scenarios, and each path starts from the last node of the simple path (in the first block) to a node (in the third block) with a complete schedule with the least delay for a given scenario.
Candidate solution generation in roll period
Scenario generation

Reactive schedule generation in look-ahead period

Delay: 50 min
Delay: 38 min

Fig. 10 Structure of solution search tree.

5.2 Multi-layer branching solution procedure

After introducing some additional but necessary notations (Table 7) used in the following node branching process, we will further detail the proposed solution procedure.

Table 7
Notations for node branching process.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>task ( t(i, j) )</td>
<td>an activity of a train arriving at and departing from a segment</td>
</tr>
<tr>
<td>conflicting task set ( \Pi(n) )</td>
<td>a set of tasks involved in the earliest conflict of a conflict resolving node ( n )</td>
</tr>
<tr>
<td>active node ( n )</td>
<td>a node that has been generated but not yet eliminated or branched from</td>
</tr>
<tr>
<td>active node list ( L_C )</td>
<td>a list of active nodes in the roll period</td>
</tr>
<tr>
<td>complete candidate solution node list ( N_C )</td>
<td>a list of conflict-resolving nodes at the last layer of the roll period with complete solutions ready to being branched in the “scenario generation” block</td>
</tr>
</tbody>
</table>

Note that, a capacity breakdown is modeled here as a high-priority task that always exclusively occupies the related segment/station.
Algorithm 1 for Phase I: Candidate solution generation in roll period

Input: Planning timetable, current schedule as the initial solution, starting time of the capacity breakdown $T_0$, ending time of the capacity breakdown for a particular scenario $w_0$, $r(w_0)$ and end time of the roll-period $T_1$.

Output: candidate solution node list $N_C$, which is also the input for Algorithm 2.

Step 1: Initialization

Generate a new node $n_0$ starting with the above given condition. Insert active node $n_0$ into $L_C$.

Step 2: Node Selection and task completion criterion

Select an active node $n$ from $L_C$ to branch from based on a node selection rule (e.g. depth first search and breadth first search). If there is no active node in $L_C$, this algorithm completes. If node $n$ does not have a conflict before $T_1$, move it to candidate solution node list $N_C$.

Step 3: Conflict resolving branching

Step 3.1: (Conflict set construction)

Find the earliest conflict time point $t$ in the current partial schedule at node $n$ and add all the tasks involved in the conflict into $\prod(n)$. Remove node $n$ from $L_C$.

Step 3.2: (Constraint updating and node generation)

For each task $t(i,j)$ in the set $\prod(n)$, insert a new conflict resolving node $n'$ into $L_C$, copy all the tasks from node $n$ into node $n'$, let other trains $i$ involved in $\prod(n)$ yield to train $i$ in order to resolve the conflict at time point $t$, that is, set $x_{i,t;j} = 1$.

Step 3.3: (Schedule generation)

For newly generated nodes, extend the related trains to obtain the updated schedules. Go back to Step 2.

Algorithm 2 for phase II: Reactive scheduling subroutine

Input: Scenario node $s$ with the predetermined meet-pass relation vector $X$ in the roll period, and the realized capacity breakdown ending time and segment running time for each random scenario.

Output: The optimal look-ahead schedule for node $s$ that follows the predetermined meet-pass relation vector $X$.

Algorithm 3 for phases III and IV: Performance calculation and results output

Input: Candidate solution node list $N_C$.

Output: Best solution $X^*$ and corresponding reactive schedule for each scenario.

Step 1: Calculate expected performance of each candidate solution

For each candidate solution node $n$ in $N_C$
Step 1.1: (Scenario node generation and evaluation)

Generate scenario nodes \( s_1, s_2, \ldots, s_W \) for \( W \) random scenarios. Copy all the tasks and precedence constraints (i.e. solution \( X \)) from parent node \( n \) to each scenario node.

For each scenario node, find optimal schedule node \( r_n(w) \) by calling Algorithm 2.

Step 1.2: Evaluate overall expected performance

For candidate solution \( X_i \) at node \( n \), calculate \( E_\epsilon [Q_{\text{at}}(X_i, w)] \), \( E_\epsilon [Q_{\text{at}}(X_i, w)] \) and objective function \( z(X_i) \).

End for

Step 2: Select the best solution

Search all candidate solution nodes in \( N_C \), and select the node with the best overall performance \( Z^* \). Output the corresponding best roll-period schedule \( X^* \) and optimal second-period solution for each scenario.

Algorithm 2 can easily be implemented through a standard branch-and-bound algorithm for train scheduling, which typically includes six steps: 1) Initialization, 2) Stopping criterion, 3) Node selection, 4) Branching, 5) Upper bound generation, and 6) Fathoming by lower bound. We implement the branch-and-bound procedure by following our previous study in Zhou and Zhong (2007). Two methods can be used to generate lower bounds: the Lagrangian relaxation-based lower bound and the crossing conflict-based lower bound. In order to speed up the solution algorithm, we apply the latter method. This lower bound method was shown in Zhou and Zhong (2007) to be able to provide computationally efficient and effective lower bounds compared to the Lagrangian relaxation-based method, especially when dealing with a high traffic density.

To meet the real-time computational requirements for medium or large-sized instances, we then further introduce a priority rule-based heuristic algorithm to reduce the search space and complexity. Train priority is defined as the existing delay at the current time. Trains with larger delay time hold higher priority. When there is a conflict between two train activities which are in roll-period or look-ahead period, the train with higher priority is scheduled to first occupy the segment track. If the two trains have the same priority, we apply a first-come-first-served rule to break the tie. If there is a conflict between two train activities with one in roll-period and the other in look-ahead period, the first train activity in roll-period is assumed to have the priority.
Based on the above scenario-based rolling horizon optimization approach, the corresponding decision support tool and related data flow can be illustrated as in Fig. 11. When an incident occurs, the decision support tool first collects information about the initial train timetable, starting time and place of the incident, as well as real-time infrastructure status and positions of involved trains. Secondly, the tool generates a number of probabilistic incident duration scenarios, and finds the corresponding robust solutions for the entire optimization horizon. The lead dispatcher then broadcasts the optimized meet-pass plan in the roll period to the related station dispatchers and train drivers. When the lead dispatcher is further notified by maintenance crew members about the exact incident duration later, he/she will accordingly select and disseminate the meet-pass plan in the look-ahead period (under the corresponding scenario from the previous stage) to the station dispatchers and train drivers.

6. Numerical experiments

6.1 Initial train planning timetables and stochastic capacity settings

The following case study considers a 138-km single-track rail line that consists of 18 stations from Laizhou to Shaowu in Fujian province, China. Segment free running times and departure times at starting stations follow an actual train diagram, and the average travel time for a passenger train along this rail line is about 170 minutes.

In a 24-hour period, the actual train timetable schedules 26 passenger trains and 32 freight trains along this high-density rail corridor. Without loss of generality, we only consider passenger trains in this paper (Fig. 12), as freight trains (with lower priority) have to yield to passenger trains, and freight trains’ schedule delays are typically associated with relatively lower priorities. The headway times at all stations are 2 minutes, and the minimum dwell time at all stations is set to be 2 minutes for simplicity. All the experiments are performed on an IBM Thinkpad T42 laptop with 1.8 GHz CPU and 1.5 GB memory, and the algorithms are implemented in Visual C++ 6.0 on a Windows XP platform.
To systematically conduct sensitivity analysis for the solution algorithm, two cases are examined in this paper. **Case A** (low density) considers 26 passenger trains from the actual train timetable and a capacity breakdown incident that occurs at 2:40 AM and completely blocks segment 2, depicted in Fig. 12. A rectangle box (without a right vertical line) is used to represent the uncertainty of incident duration. Illustrated in Fig. 13, **Case B** (high density) considers 53 passenger trains, and the corresponding initial train timetable is generated based on a priority rule algorithm. A capacity breakdown incident is assumed to occur at 2:45 AM and completely block segment 5.

The capacity breakdown duration is assumed to follow a Gaussian distribution in our study, with two different types of capacity loss duration: (1) **short duration** with expected duration $\mu = 30$ minutes and standard deviation $\sigma = 10$ minutes; (2) **long duration** with $\mu = 60$ minutes and $\sigma = 20$ minutes. According to Chinese railway practice, the roll-period duration is set to be 60 minutes and the look-ahead period duration is 120 minutes, resulting in a total stage duration of 3 hours. In this paper, the segment running times are assumed to be deterministic for each random scenario, reducing the complexity of generating random scenarios. 200 realized samples of random incident scenarios are used to evaluate the solution quality in this study.

Fig. 12 An actual train timetable and a capacity breakdown incident of **Case A** (low density) with 26 passenger trains. Each vertical thin gridline represents a 10-minute interval, 60-minute intervals are highlighted by vertical solid gridlines, and the y axis shows station numbers.
6.2 Sensitivity analysis of optimization model parameters

Two important optimization model parameters in our proposed framework are the number of candidate solutions (NCS) in the roll period, and the number of random scenarios (NRS) in the look-ahead period. First, we want to determine appropriate settings of these two parameters under the B&B algorithm. The base-line solution uses large numbers of candidate solutions and random scenarios, i.e., NCS=200 and NRS=200. The solution quality is measured in terms of the expected delay of the best candidate solution (i.e. $Z^*$). Fig. 14 depicts the solution quality as a function of NCS ranging from 5 to 200 in Cases A and B under low and high disruption intensity conditions. Obviously, the solution quality improves drastically when NCS increases from 5 to 45. Although there are still some minor improvements when NCS becomes larger than 45 (e.g. 85 for Case A with high disruption intensity), the solution quality improvement becomes marginal with an increase of NCS from 45 to 200. As a result, with the consideration of computational efficiency, NCS=45 is used for the set of experiments in Section 6.3.

Fig. 13 A designed higher-density train timetable and a capacity breakdown incident of Case B (high density) with 53 passenger trains.
Fig. 14 Impacts of number of candidate solutions on solution quality.

Fig. 15 further shows the optimization performance curves as a function of the number of random scenarios (NRS). As expected, a large NRS produces a better result than smaller NRS, but the attainable quality improvement by adopting a large sample size NRS is quite limited. For instance, for Case A (low density) with high disruption intensity, the result of NRS=200 is 18% better than the result of NRS=5. On the other hand, with NRS=20, the result on average is only 10% worse than the result of NRS=200. Considering the overall trade off between computational efficiency and solution quality, the setting of NRS=20 is used for the set of experiments in Section 6.3.
We now try to examine the solution quality and computational performance of the B&B algorithm and priority-rule based heuristic algorithm. With NCS=200 and NRS=200, Fig. 16 illustrates the results and Fig. 17 illustrates the corresponding consumed computation times. It is easy to see that the B&B algorithm produces, on average, 18% better results than the heuristic algorithm, with nearly twice the amount of additional computation time. As expected, Case B with higher traffic density requires more computation time than Case A with lower traffic density, for both B&B algorithm and heuristic algorithm. In our experiments, for Case A, both with low and high disruption intensity, the B&B algorithm can implicitly enumerate all candidate nodes and find optimal solutions within 10 minutes. However, Case B does not find the optimal solutions within a reasonable time limit (e.g. 15 minutes), so we stop the searching process with an optimality gap at 10%.

![Fig. 16 Impacts of solution algorithms on solution quality.](image-url)
6.3 Comparison between robust solutions, EV solutions and value of perfect information

The experiments below aim to examine a theoretically important concept in the field of stochastic programming: expected value-based solution (EV solution). In a real-world train traffic management environment, when facing a capacity breakdown incident, train dispatchers usually first generate a feasible and suboptimal train meet-pass plan using the expected value of capacity loss duration, and further make adjustments when the incident duration information is known for certain. Such a solution is called an EV solution.

Although the capacity loss duration can be viewed as a random distribution, let us consider a hypothetical situation where the track maintenance crew can predict the exact recovery time according to other external factors such as weather conditions and exact breakdown location. In this case, the rail lead dispatcher will be able to make a more informed decision. By considering this ideal situation, we can obtain the best solution under each precisely predicted scenario and then accordingly calculate the expected total train delay for the perfect information case. We then want to further quantify the value of perfect information, i.e., the expected value of perfect information (EVPI), for decisions under uncertain capacity loss duration. More formally, let $V_w(\pi)$ represent the objective function value of the best solution obtained with perfect information (about disruption duration) under scenario $w$, $p(w)$ stand for the probability of scenario $w$ and $V(\pi)$ represent the expected value under all scenarios with perfect information. $V(\pi)$ could be computed by Eq. (23). EVPI is measured by the difference between the objective function value of the best solution obtained by the proposed approach (i.e., $Z^*(\text{RS})$) and $V(\pi)$, shown in Eq. (24).

$$V(\pi) = \sum_w p(w)V_w(\pi)$$

$$\text{EVPI} = Z^*(\text{RS}) - V(\pi)$$
With the fine-tuned algorithm settings in Section 6.2 (i.e., NCS=45 and NRS=20), we further conduct experiments to check the performance of the proposed approach. Using the B&B algorithm, Fig. 18 gives an overall comparison on the expected total train delay of EV solution, robust solution and solution based on perfect information. In terms of expected total train delay time, the robust solutions are better than the EV solutions by 31%, 17%, 14% and 18% respectively in each case. With perfect information, the solution quality of the B&B algorithm can be further improved by an average of 20.75% and 42.5 minutes. Quantifying the value of perfect information could help the railroad company to decide whether it is worth allocating additional resources to improve the accuracy for estimating track repair time durations.

Fig. 18 Comparison of expected delay among EV solution, robust solution and perfect solution (i.e. solution with perfect information).

7. Concluding remarks

This paper is motivated by the need to formulate and solve a (practically challenging and theoretically important) robust single-track train dispatching problem under stochastic segment running times and capacity loss duration. Based on a stochastic programming with recourse model framework, a scenario-based rolling horizon solution approach is used to illustrate the essential characteristics of this problem. Moreover, the key definitions of this approach, mathematical formulations are given in detail after conceptual illustrations. We then develop a multi-phase and multi-layer branching solution procedure which consists of scenario generation block to address the issue of uncertain incident durations. Finally, numerical experiments using a real-world train schedule are conducted to demonstrate the importance of robust train dispatching in a dynamic and stochastic environment.

Continuing advances in real-time train scheduling algorithms essentially depend on modeling and algorithmic advances that recognize the dynamic and stochastic nature of the problem of interest. As train dispatching models have been continuously enhanced in the past few decades to include greater practical and policy realism, the complexity of the corresponding solution-searching algorithms increases. The practically functional rolling horizon solution mechanism in fact complicates the robust schedule optimization aspect compared to the commonly-used assumption of non-overlapping stages in the stochastic programming approach. The work presented in this paper is an example of adapting and extending a well-established stochastic optimization framework to meet the challenges of building
realistic disruption handling tools under dynamic and uncertain data inputs. Our paper illustrates the importance of developing systematic models and solution-finding schemes to handle the increase in computational complexity introduced by virtually any novel problem dimension, such as the robustness measure for train meet-pass plans in our case.

Our future research aims to integrate the procedures described in this paper in real-world on-line train dispatching models, and expand the realm of application of these models. Along the proposed stochastic programming framework, we will address the following three major extensions: (1) dynamic platform and track utilization and train rerouting management under capacity breakdown, which are very important for daily operations in practice; (2) a robust train dispatching model under capacity breakdown on a “double-track” railroad line; and (3) a closed-loop control mechanism by incorporating dispatchers in the solution generation process. Finally, we plan to study robust rerouting scheduling through a rail network, as it may be a more reasonable alternative dispatching strategy for an extremely long capacity breakdown period.

References


