SINGLE-TRACK TRAIN TIMETABLING WITH GUARANTEED OPTIMALITY: BRANCH-AND-BOUND ALGORITHMS WITH ENHANCED LOWER BOUNDS

Submitted for publication in Transportation Research Part B

Single-track train timetabling with guaranteed optimality: Branch-and-bound algorithms with enhanced lower bounds

Current Email address: xzhou74@asu.com (X. Zhou)

Abstract

A single-track train timetabling problem is studied in order to minimize the total train travel time, subject to a set of operational and safety requirements. This research proposes a generalized resource-constrained project scheduling formulation which considers segment and station headway capacities as limited resources, and presents a branch-and-bound solution procedure to obtain feasible schedules with guaranteed optimality. The search algorithm chronologically adds precedence relation constraints between conflicting trains to eliminate conflicts, and the resulting sub-problems are solved by the longest path algorithm to determine the earliest start times for each train in different segments. This study adapts three approaches to effectively reduce the solution space. First, a Lagrangian relaxation based lower bound rule is used to dualize segment and station entering headway capacity constraints. Second, an exact lower bound rule is used to estimate the least train delay for resolving the remaining crossing conflicts in a partial schedule. Third, a tight upper bound is constructed by a beam search heuristic method. Comprehensive numerical experiments are conducted to illustrate the computational performance of the proposed lower bound rules and heuristic upper bound construction methods.

Keywords
Train timetabling, Lagrangian relaxation, branch and bound, heuristics, railway.

1. Introduction

A train timetable defines the planed arrival and departure times of trains to/from yards, terminals and sidings, and train scheduling plays a vital role in managing and operating complex railroad systems. From a marketing point of view, the level-of-service of train timetables is an important factor that affects travelers' and freight carriers' decisions in choosing desirable transportation modes. From a railroad operations point of view, train timetables serve as essential data input to locomotive and crew scheduling processes, and their temporal and spatial structures directly impact the utilization of scarce resources such as engines, cars, and crews. In order to offer dependable services while maintaining profitability in highly competitive transportation markets, the railroad industry has shown great interest in incorporating innovative train scheduling techniques for both planning and real-time operational applications. For instance, a computer-aided train scheduling system can facilitate planning by systematically determining rail network capacity, identifying chokepoints, as well as evaluating alternate facility maintenance and improvement scenarios. The use of optimization techniques, on the other hand, can help train dispatchers in the rapid construction of cost-efficient schedules in response to random events and demand variability.

A wide range of studies have been devoted to the train scheduling problem in the last three decades, particularly regarding the development of mathematical formulations and solution algorithms. Szpigel (1973) first modeled the train scheduling problem as a mixed integer program and applied Greenberg's branch-and-bound solution framework (1968) (for a general job shop machine scheduling problem) to find timetables that minimize the total transit time, subject to overtaking and crossing headway constraints. Jovanovic and Harker (1991) presented a nonlinear integer programming model that minimizes the deviation between planned schedules and actual schedules, and used a branch-and-bound procedure to generate feasible meet-pass train plans. Carey (1994a, 1994b) and Carey and Lockwood (1995) developed an iterative decomposition approach for solving the train timetabling and pathing problem in a rail network with one-way and two-way tracks; moreover, several node branching, variable fixing and bounding strategies were presented to reduce the search space. Higgins et al. (1995) developed a nonlinear integer programming model for minimizing the total train delay and fuel cost with variable train velocities, and they also presented a branch-and-bound solution algorithm with a lower bound estimator that utilizes a look-ahead rule to calculate the least delay for the remaining conflicts in a partial schedule. As remarked by the authors, the lower bound rule might overestimate the actual remaining delay for some trains theoretically, but only in exceptional cases. Their original efforts in this direction highlight the need for further investigations on strict lower bound rules in an inexplicit enumeration search framework. It should be also noted that several stochastic analytical models (e.g. Petersen, 1982; Chen and Harker, 1990; Higgins and Kozan, 1998) have been developed to estimate the average delay of a train schedule, but these models are not suitable for serving as strict lower bound rules in the deterministic case. Braunlund et al. (1998) proposed an integer programming model to find a profit-maximizing timetable and designed a Lagrangian relaxation method to dualize track capacity constraints. The resulting subproblems were solved by the shortest path algorithm in conjunction with several improved dual updating schemes. Based on a graph-theoretic formulation for the periodic-timetabling problem in a rail line, Caprara et al. (2002) presented a Lagrangian relaxation solution method, in which a dynamic constraint generation (often referred to as relax-and-cut) scheme was used to handle a large number of dualized constraints. In general, without being embedded into the branch-and-bound search scheme, the Lagrangian relaxation approach can hardly guarantee the discovery of optimal solutions, due to the existence of duality gaps and the limitation of primal heuristic search algorithms. Kroon and Peeters (2003) developed a periodic event scheduling model to consider variable trip times for the cyclic railway timetabling problem. Zhou and Zhong (2005) proposed a multi-mode resource-constrained project scheduling formulation to consider acceleration and deceleration time losses in double-track train timetabling applications, and designed several dominance rules to find Pareto optimum schedules in an intercity passenger corridor.

The train scheduling problem is known to be NP-hard (Cai, 1994; Caprara et al. 2002), and optimal solutions are typically unattainable in large-scale and complex instances. To meet the computational requirements in real-world applications, a heuristic approach becomes a necessity for generating feasible and sub-optimal solutions. Specifically, a number of simple priority rules have been widely used because of their low computational complexity and easy implementation. Given a pair of conflicting trains, a simple

priority rule determines which train to schedule next by utilizing underlying objective functions, for instance, the total delay for the minimal travel time criterion (Petersen et al., 1986, Kraay and Harker, 1995), the schedule deviation and tightness for the minimal deviation criterion (Chen and Harker, 1990; Cheng and Chiang, 1998), train priorities (Petersen et al., 1986), and the number of passengers transported (Adenso-Diaz et al., 1999). Along these lines, Zhou et al. (1999) and Dorfman and Medanic (2004) incorporated priority rules into a discrete event simulation framework to solve large-scale real-world train scheduling problems. To further improve solution quality, several recent studies extended simple priority rules to more sophisticated search procedures, such as backtracking search (Adenso-Diaz et al., 1999), look-ahead search (Sahin, 1999), and metaheuristic algorithms (Higgins, 1995; Higgins and Kozan, 1998). Assad (1980), Cordeau et al. (1998), and Newman et al. (2002) provided excellent surveys for the train scheduling problem and discussed its inherent connection with other optimization problems in the field of railroad routing and scheduling.

Although a wide range of sophisticated mathematical formulations and solution algorithms have been proposed to solve the train scheduling problem, the majority of existing methods either require a huge amount of computation time and memory space to find optimal schedules, or produce heuristic results with no guarantee of solution accuracy. In fact, it is essential to recognize the significance of optimal solutions for both planning and operational applications, since an optimal train schedule not only provides a benchmark for systematically evaluating various heuristic algorithms but also generates an exact upper bound on the travel time and operational cost savings attainable with alternative system design scenarios, for instance, by increasing train operating speed and adding new intermediate sidings. To quantify such gains, there is a great need to enhance the solution algorithms so that optimal solutions can be found to more large-scale real-world train scheduling problems. On the other hand, a heuristic solution could be 1%, 5%, or 20% away from the optimum, so simply accepting the current best solution without evaluating its solution quality could lead to an unsatisfactory level-of-service and significant inefficiency in the allocation of limited railroad resources and capacities. In medium-range planning applications, for instance, a passenger train timetable typically is put into effect for a year or several months, so its solution quality could have a dramatic influence on overall profit.

The goal of this research is to enhance the computational efficiency of solution algorithms by taking advantage of problem-specific characteristics in the single-track train scheduling problem. The train timetabling problem is formulated as a generalized resource-constrained project scheduling problem. Based on a schedule generation scheme proposed by De Reyck and Herroelen (1998) for the resource-constrained project scheduling problem with generalized precedence relations, this study presents a branch-and-bound procedure to systematically decompose the original complex problem; the resulting subproblems are solved by the efficient longest path algorithm. By extending the Lagrangian relaxation approach proposed by Braunlund et al. (1998) and Caprara et al. (2002), this research dualizes both segment track capacity and station entering headway capacity constraints to provide a lower bound estimator. Along the direction proposed by Higgins et al. (1995), a strict lower bound rule is derived to estimate the least train delay for the remaining crossing conflicts. Several node selection rules are also incorporated into the branch-and-bound procedure to solve large-scale instances under computational time or space constraints. In addition, a beam search heuristic algorithm is applied to construct a tight upper bound. A detailed assessment of the application of the beam search method in the field of machine scheduling can be found in a study by Sabuncuoglu and Bayiz (1999).

The organization of this paper is as follows. In the next two sections, a branch-and-bound solution procedure based on a job shop scheduling formulation is presented for the single-track train timetabling problem. Section 3 details branching rules and node selection rules, and section 4 sequentially introduces two lower bound rules to reduce the search space. After examining several constructive heuristic methods for providing tight upper bounds in section 5, we evaluate the performance of new exact search algorithms and priority rule-based methods.

2. Mathematical formulations

This study considers a rail system with a single two-way line between stations as illustrated in Fig. 1. The rail line is used by trains running in both directions, and those trains meet and pass at stations. More specifically, the stations along the single-track line include "intermediate" stations (i.e. sidings) and terminals with multiple platforms, and trains can originate and terminate at the stations.

In this study, we formulate the train timetabling problem in planning applications using a job shop structure. Specifically, the optimization problem is concerned with a rail line with a set of single-track segments and a set of trains. Each train is assumed to have a pre-specified departure time and a pre-specified traveling route. Moreover, the free running times at segments are assumed to be constant. The traveling process of an inbound or outbound train along the rail line can be viewed as a procedure for completing a series of tasks (i.e. activities), each task specifying the arrival and departure times of a train on a segment. The subsequent integer programming optimization problem aims to minimize the total travel time, corresponding to a job shop scheduling formulation of minimizing the total completion time under different release times. The notation of parameters and variables are shown in Tables 1 and 2, respectively.

Objective function

$$\min Z = \sum_{i=1}^{n} e_{i,\sigma(i,m)} \tag{1}$$

subject to

Departure time constraints:

$s_{i,\sigma(i,1)} \geq r_i, \ \forall i \in I.$ (2)

Free running time constraints:

$e_{i,\sigma(i,k)} = p_{i,\sigma(i,k)} + s_{i,\sigma(i,k)}, \ \forall i \in I, \ k=1,2,\ldots,m.$ (3)

Minimum dwell time constraints:

$s_{i,\sigma(i,k)} \geq e_{i,\sigma(i,k-1)} + d_{i,\sigma(i,k)}, \ \forall i \in I, \ k=2,\ldots,m.$ (4)

Headway constraints at track segments:

$s_{i,j} \geq e_{i',j} + h_j$ or $s_{i',j} \geq e_{i,j} + h_j, \forall i, i' \in I, \ i \neq i', \ j \in J$ (5)

Headway constraints on arrival times at stations:

$e_{i,\sigma(i,k)} \geq e_{i',\sigma(i',k')} + g_u$ or $e_{i',\sigma(i',k')} \geq e_{i,\sigma(i,k)} + g_u, \forall \ i, i' \in I, \ i \neq i', \ \beta(i,k) = \beta(i',k') = u,$ (6)

Maximum dwell time constraints:

$$s_{i,\sigma(i,k)} \leq e_{i,\sigma(i,k-1)} + \overline{d}_{i,\sigma(i,k)} \quad \forall i \in I, \ k=1,2,\ldots,m. \tag{7}$$

Fig. 2 illustrates a train schedule that covers 5 stations and 4 segments. Outbound train 1 meets inbound train 2 at station 3, while train 1 stops at station 1. Constraint (2) ensures that the actual departure time of a train at its starting station is not earlier than its planned departure time. Constraint (3) links the entering and leaving times of a train at a segment. In many real-world rail systems, a train timetable needs to take into account the acceleration and deceleration time losses at stations/sidings. A multi-mode resource-constrained project scheduling approach (Zhou and Zhong, 2005) can be applied to address this issue by considering different running times as multiple modes in which a train travels a segment. For simplicity, the rest of the paper does not explicitly consider acceleration and deceleration time losses. Constraint (4) specifies that the actual dwell time should be no less than the planned stop time, since positive dwell times are typically required for trains to load and unload passengers or freight at stations. Constraint (5) imposes the minimum headway requirement between two consecutive trains running in the same direction or in opposite directions at the same segment. Constraint (6) states that the minimum headway spacing must be satisfied for two consecutive trains to arrive at the same station. In a single-track train scheduling problem, the minimum time headway constraints at stations are mainly imposed for trains running in opposite directions. Typically, $g_u \geq h_j$. For instance, g_u= 3 minutes and h_j= 2 minutes in the case study of this paper. Constraint (7) gives an upper bound on the possible dwell time at a station, that is, the maximum allowable additional delay for a train to yield to the other trains. Another purpose of the maximum dwell time constraint (7) is to significantly reduce the size of the search tree, as indicated by many previous studies

(e.g. Carey and Lockwood, 1995). The user, on the other hand, needs to ensure that the maximum dwell times are long enough to generate a feasible train schedule. Note that, constraint (6) only restricts variable $e_{i,j}$, which is the arrival time of train i at the corresponding upstream station of segment j. That is to day, our model does not impose a restriction between the departure times of two trains at the same station, nor between the arrival time of one train and the departure time of another train at the same station. Please see the Appendix that explains the possible station headways in more general settings.

The "either-or" relation in the segment headway constraint (5) can be expressed as the following two precedence constraints by introducing a binary variable, $y_{i,i',j}$:

$s_{i,j} \geq e_{i',j}+h_j - M \times y_{i,i',j}$, (8)

$s_{i',j} \geq e_{i,j}+h_j - M \times (1-y_{i,i',j})$. (9)

One can also formulate the station arrival headway constraint (6) in an analogous way.

From a perspective of resource-constrained project scheduling, the train scheduling problem essentially considers two types of limited resources related to segments and stations. We need to ensure that the total number of trains being active at a particular time t on each resource (i.e. segment j or station u) is no less than 1.

The headway constraints on segments can be modeled as the following segment track capacity constraints in the resource-constrained project schedule framework:

$\Sigma_i \delta_{i,j,t} \leq 1, \forall\, j \in J, t = 1, 2, \ldots, T.$ (10)

One of the distinguishing features in the train scheduling problem is that the time period for a train to be active at a segment not only covers the time interval during which a train physically occupies the segment track, but also includes the required safety headway. That is, $\delta_{i,j,t}=1$ for $s_{i,j} \leq t < e_{i,j}+h_j$, and 0 otherwise.

Similarly, we can convert the station arrival headway constraints to the following station entering capacity constraints:

$\Sigma_i \varepsilon_{i,u,t} \leq 1, \forall\, u \in U, t = 1, 2, \ldots, T.$ (11)

where $\varepsilon_{i,u,t} =1$ for $e_{i,\sigma(i,k)} \leq t < e_{i,\sigma(i,k)} + g_u$ where $\beta(i,k)= u$ and 0 otherwise.

Essentially, constraints (8) and (9) are used in a standard integer programming formulation to model the "either-or" conditions in constraints (5) and (6), and we further reformulate them as constraints (10) and (11) to construct the Lagrangian lower bound discussed later. It should be also remarked that, in the train scheduling literature, the station capacity is typically related to the number of routes, tracks or platforms in a rail station, but this paper uses the term "capacity" to represent and characterize the limited station arrival headway resources in constraint (6) from the resource-constrained scheduling perspective. That is to say, a station under consideration could have multiple tracks and platforms, but we allow at most one train, at any given time, to enter that station in order to meet the safety requirement.

3. Branch-and-bound solution procedure

3.1 Overview of solution algorithm

To handle the above integer programming model with an exponential number of headway constraints, we present a branch-and-bound search procedure to find optimal schedules through implicit enumeration. In this study, the schedule generation scheme for solving the generalized resource-constrained project scheduling problem is extended and adapted to systematically generate subproblems and chronologically eliminate train conflicts by adding precedence relations. The proposed search procedure assumes that the tasks in an optimum schedule are processed as early as possible according to the prespecified precedence relations, and it can be shown that there exists such a schedule that minimizes the objective function among all feasible schedules (Bartusch et al., 1988). Furthermore, the newly generated subproblems are solved by the longest path algorithm to determine the earliest start times for each train at each segment. In order to effectively reduce the search space, two lower bound rules are embedded into the branch-and-bound search

procedure, namely a Lagrangian relaxation (LR)-based lower bound and a crossing conflict (CC)-based lower bound. Based on the notation shown in Table 3, we further detail the branch-and-bound procedure.

Algorithm 1. Branch-and-bound procedure

Step 1: (Initialization)
Let *ANL* be empty. Apply a heuristic algorithm to generate a good feasible solution and an initial value of *UB*.
Create a new node v=1, and let $P(1)$ contain the departure time constraints, the free running time constraints and the minimum dwell time constraints. Insert this node into the active node list *ANL*.

Step 2: (Stopping criterion)
If all of the active nodes in *ANL* have been visited, or the computational resource bounds in terms of execution time or the number of active nodes in the memory are exceeded, then terminate and output the best solution with an optimality certificate $\frac{UB - LB}{UB}$, where $LB = min_{\ v \in ANL}\ LB(\mathrm{v})$.

Step 3: (Node Selection)
Select an active node v from *ANL* based on a node selection rule (e.g. depth first search and breadth first search).
Step 4: (Branching)
Step 4.1: (Conflict set construction)
Find the earliest conflict time point t in the current partial schedule such that
$\Sigma_i \delta_{i,j,t} > 1$ or $\Sigma_i \varepsilon_{i,u,t} > 1$ (a lexicographic criterion is used to break the ties).
Add all the tasks involved in the conflict into $\Omega(v)$.

Step 4.2: (Precedence constraint generation)
For each task $t(i,j)$ in the set $\Omega(v)$, insert a new node w into *ANL*, copy $P(v)$ to $P(w)$, and add the precedence constraints between task $t(i,j)$ and the other tasks $t(i',j')$ into $P(w)$, that is,
$t(i,j) \prec t(i',j')$ for $\forall i' \neq i$, $t(i',j') \in \Omega(v)$.

Step 4.3: (Schedule generation)
Calculate the earliest start times of tasks that satisfy constraints in the set $P(w)$ by using the longest path algorithm.

Step 5: (Upper bound updating)
If $\Omega(v)$ is empty, then all the tasks have been finished and the corresponding schedule is feasible, so node v should be removed from the active node set. If the objective function value $Z(v)$ of the feasible schedule at node v is smaller than *UB*, then update $UB = Z(v)$.

Step 6: (Fathoming by lower bounds)
For each newly generated node w, compute the lower bound $LB(w)$. If $LB(w) \geq UB(1-\theta)$, this node will be fathomed. Go back to step 2.

3.2 Branching rules and solving subproblems

Before detailing the branching rules used in this paper, we first want to highlight the need to model the train scheduling problem in a generalized resource-constrained project framework. Consider two consecutive tasks $t(i,j)$, $t(i,j+1)$ for an outbound train i in Fig. 3. In a resource-constrained project scheduling model, the duration of task $t(i,j)$ is determined by the starting time and finishing time of the *resource requirement* of the task, corresponding to $[s_{i,j}, e_{i,j}+h_j)$ in a train schedule, as shown in the shaded area. Similarly, the duration of task $t(i,j+1)$ is $[s_{i,j+1}, e_{i,j+1}+h_{j+1})$. The basic resource-constrained project scheduling model assumes that there are zero time lags for finish-to-start precedence relations between predecessor tasks and successor tasks. To be exact, the *resource requirement start time* of the successor

task must be greater than or equal to the *resource requirement finishing time* of the predecessor task. In the context of train scheduling, however, $s_{i,j+1} < e_{i,j}+h_j$, indicating that there is a *minimum time lag* of h_j time units between two consecutive tasks in a train route. In other words, the resource requirements of two tasks overlap with each other by h_j time units. As a result, the train scheduling problem should be modeled as a generalized resource-constrained project problem with minimum time lags, as opposed to a basic resource-constrained project problem.

The train timetable in each node of the search tree is represented by a precedence constraints graph, which is commonly used in the field of machine scheduling. Fig. 4 shows a precedence constraints graph corresponding to the train schedule in Fig. 2. In a precedence graph, there are $n \times m$ vertices corresponding to $n \times m$ tasks $t(i,j)$ associated with n trains, and these tasks are sequentially connected by arcs along the route of a train. In addition, a dummy activity $t(0,0)$ is represented by a source vertex, which has n arcs emanating to the first task of those n trains. In the branching process, the number of branches of a node is equal to the number of tasks in $\Omega(v)$. A branch out of node v corresponds to a task $t(i,j) \in \Omega(v)$ which is the next one to be scheduled. Accordingly, a set of directed arcs are added into the precedence constraints graph to represent precedence relations $t(i,j) \prec t(i',j')$, where task $t(i',j') \in \Omega(v)$. If no conflict can be found in a schedule, then we obtain a leaf node at the bottom of the search tree.

The subproblem at node v is reformulated as an optimization problem that calculates the earliest start time for each task.

$$\min Z = \sum_{i=1}^{n} s_{i,\sigma(i,m)} \tag{12}$$

Subject to

Precedence constraints:

$s_{i',j'} \geq s_{i,j}+C_{i,j,i',j'} \quad \forall$ precedence relation $t(i,j) \prec t(i',j') \in P(v)$. (13)

where $s_{0,0}=0$ and $C_{i,j,i',j'}$ is the arc cost between vertices corresponding to task $t(i,j)$ and task $t(i',j')$.

To obtain the above modified formulation, we replaced each $e_{i,\sigma(i,k)}$ in the original integer programming formulation by $s_{i,\sigma(i,k)}+ p_{i,\sigma(i,k)}$. As free running times are constant, the original objective function in Eq. (1) reduces to the function shown in Eq. (12). Clearly, the departure time constraint can be expressed in terms of start-to-start precedence constraints:

$s_{i,\sigma(i,1)} \geq s_{0,0} + r_i$. (14)

After rearranging terms, the free running time constraint and the dwell time constraint can be combined into:

$s_{i,\sigma(i,k)} \geq s_{i,\sigma(i,k-1)}+ p_{i,\sigma(i,k-1)}+d_{i,\sigma(i,k)}$. (15)

If the minimum headway constraint on segment j specifies a precedence relation: $t(i,j) \prec t(i',j)$, it is equivalent to

$s_{i',j} \geq s_{i,j}+ p_{i,j}+h_j$, (16)

The minimum headway constraint between train i and train j on station u can be converted in the same manner, that is,

$s_{i',\sigma(i',k)} \geq s_{i,\sigma(i,k)} + p_{i,\sigma(i,k)} - p_{i',\sigma(i',k')}+g_u$, (17)

where $\beta(i,k)= \beta(i',k')=u$. It should be noted that, the maximum dwell time constraints are not included in the above precedence graph formulation, but they will be used to check the feasibility of a start time scenario when a schedule is generated.

It is easy to show that the dual problem of the above program is equivalent to the longest path problem (Ahuja et al. 1990), in which the earliest possible start time corresponds to the cost of the longest path from the dummy source vertex to the vertex representing task $t(i,j)$. The label correcting algorithm can be applied here to find the longest path, while its FIFO implementation solves the problem in $O(\overline{m} \times \overline{n})$ time where $\overline{m}$ and $\overline{n}$ are the numbers of arcs and vertices in the precedence constraints graph. For each newly generated node w in the search tree, usually only one or two new precedence constraints are added into the precedence graph of predecessor node v, and the longest path problem at node w only slightly differs from

the one at node *v*. Instead of calculating the longest path at node *w* from the scratch, this research applies a re-optimization technique to solve the resulting arc addition problem.

3.3 Node selection rules

Let us define an active node as a node that has been generated but not yet eliminated or branched from. The node selection rule at step 3 essentially determines how to choose a node to branch from a set of existing active nodes. Typical node selection rules include (1) depth first search (i.e. select an active node that was generated last), (2) breadth first search (i.e. select an active node that was generated first) and (3) least lower bound search (i.e. select a node with the least lower bound). The depth-first search can utilize the computer memory efficiently, as it only keeps active nodes along the path from the root to the leaf node in the search tree, corresponding to an implementation of a first-in-last-out stack structure. The breadth first search is suitable for applying dominance rules between two partial schedules. By selecting an active node with the least lower bound for partitioning, the least lower bound rule aims to minimize the total amount of computation time for finding solutions with quality guarantee. The rational behind this scheme is that any node selection scheme must select a node with the least lower bound in order to increase the global lower bound. Essentially, branching from a node with the least lower bound can gradually decrease the optimality gap (i.e. the difference between the upper bound and lower bound) of the problem.

To find a solution with the best possible optimality guarantee within limited time, we can also apply a stepwise refinement search procedure proposed by Lawler and Wood (1966) as follows: Given a computation time constraint *TC*, this adaptive scheme first tries to use *TC*/2 units of time to find an optimal solution. If an exact solution is intractable, then another *TC*/4 units of time are assigned at the second iteration to search for an approximate solution with approximation degree θ=5%. If it still fails at iteration κ, then $TC/(2\kappa)$ units of time are scheduled to find an approximate solution of θ=5%$\times\ \kappa$. This iterative procedure repeats until the time limit is reached.

4. Bounding rules

For each train to be scheduled, we need to minimize the total travel time, which is the sum of the free running time and additional delay. As the free running time is assumed to be constant, a lower bound estimator just needs to calculate the total additional delay in an incomplete schedule. In this section, we present two different lower bound rules for estimating the minimum additional delay in a partial schedule, and we are particularly interested in trade-offs between the quality of the lower bound and the associated computational costs. This type of trade-offs is extremely important in the context of branch-and-bound search, where the lower bound rule needs to be evaluated for a huge number of nodes in the search tree.

4.1 Lagrangian relaxation lower bound

The following discussion is concerned with three aspects: (1) how to find a valid Lagrangian relaxation to provide strong lower bounds; (2) how to find a set of multipliers to obtain a bound value that is as large as possible; and (3) how to construct efficient algorithms for solving the decomposed problem. The constraints in the train scheduling formulation can be classified into two groups. The first group directly relates to individual trains, such as the departure time constraint, the free running time constraints and the dwell time constraints. The second group, which includes the segment and station capacity constraints, connects different trains, making the timetabling problem difficult to solve. To provide a lower bound on the optimal cost of the original problem, we apply the Lagrangian relaxation technique to relax the complicating segment track and station entering capacity constraints, and accordingly decompose a large-scale train scheduling problem into a series of train-based subproblems that are easier to solve.

By introducing a set of nonnegative Lagrangian multipliers $\pi_{j,t}$ and $\rho_{u,t}$, we incorporate the coupling capacity constraints (10) and (11) in the following objective function as a penalty term:

$$LB(v) = \sum_i e_{i,\sigma(i,m)} + \sum_{j,t}\left[\pi_{j,t}\left(\sum_i \delta_{i,j,t} - 1\right)\right] + \sum_{u,t}\left[\rho_{u,t}\left(\sum_i \varepsilon_{i,u,t} - 1\right)\right], \tag{18}$$

where the multipliers π and ρ can be interpreted as the cost charged for utilizing a segment or a station resource at time t, respectively. Essentially, the major task of the Lagrangian function is to balance the total train delay and the cost for utilizing limited facility resources by choosing appropriate resource prices. To maximize the bound value, we need to solve the following Lagrangian dual problem:

$$\max_{\pi,\rho\geq 0} \quad L = -\sum_{j,t}\pi_{j,t} - \sum_{u,t}\rho_{u,t} + \min_{\{s,e\}}\sum_i L_i \tag{19}$$

where

$$\begin{aligned} L_i = \quad & e_{i,\sigma(i,m)} + \sum_j \sum_{t=s_{i,j}}^{e_{i,j}+h_j-1} \pi_{j,t} + \sum_k \sum_{t=e_{i,\sigma(i,k)}}^{e_{i,\sigma(i,k)}+g_{\beta(i,k)}-1} \rho_{\beta(i,k),t} \\ = \quad & \sum_k p_{i,\sigma(i,k)} + \sum_k (s_{i,\sigma(i,k)} - e_{i,\sigma(i,k-1)}) + \sum_k \sum_{t=s_{i,\sigma(i,k)}}^{e_{i,\sigma(i,k)}+h_{\sigma(i,k)}-1} \pi_{\sigma(i,k),t} + \sum_k \sum_{t=e_{i,\sigma(i,k)}}^{e_{i,\sigma(i,k)}+g_{\beta(i,k)}-1} \rho_{\beta(i,k),t} \\ = \quad & \sum_k \left[p_{i,\sigma(i,k)} + (s_{i,\sigma(i,k)} - e_{i,\sigma(i,k-1)}) + \sum_{t=s_{i,\sigma(i,k)}}^{e_{i,\sigma(i,k)}+h_{\sigma(i,k)}-1} \pi_{\sigma(i,k),t} + \sum_{t=e_{i,\sigma(i,k)}}^{e_{i,\sigma(i,k)}+g_{\beta(i,k)}-1} \rho_{\beta(i,k),t} \right] \end{aligned} \tag{20}$$

Note that, $e_{i,\sigma(i,0)}$ is defined as r_i in the above formulation. Clearly, the original interdependent system is separated into a set of subproblems, and the inner minimization problem is concerned with the sum of L_i over all trains. In a decomposed subproblem L_i, the completion time is expressed as the sum of free running times and dwell times over all segments, and the free running time constraints have been embedded into the objective functions with respect to $\pi_{j,t}$ and $\rho_{u,t}$. For a train i that enters segment $\sigma(i,k)$ at $s_{i,\sigma(i,k)}$ and leaves the segment at $e_{i,\sigma(i,k)}$, Fig. 5 illustrates how this train utilizes the segment capacity and the station entering capacity at its upstream station, while the continuous time axis is discretized into a series of scheduling time intervals.

For each train, we need to compute the optimal cost associated with its possible departure times at the origin station and its possible waiting times at the intermediate sidings. The train-based subproblem can be reformulated as a dynamic programming problem with stages corresponding to stations and states corresponding to discretized time points at each station. Given a set of time-dependent resource prices, the dynamic programming algorithm seeks to find the best resource utilization scheme for each train subject to the free running time, departure time and minimum dwell time constraints, which in turn restrict possible state transitions. The dynamic programming formulation can be further represented in a time-space network, which contains two types of arcs: (1) state transition arcs between stations with a cost of the sum of (a) the segment free running time, (b) the minimum dwell time at its upstream station as well as (c) prices for utilizing the segment track capacity and the station entering capacity, and (2) station waiting arcs between adjacent time points with a cost of waiting time at a station. Note that, a waiting arc corresponds to the additional delay besides the minimum dwell time at a station. Fig. 6 illustrates a dwell time pattern with no dwell time being required at station 3 and a minimum dwell time of 1 min at station 4. By considering the minimum dwell time as part of the time resource consumed by a state transition arc, we can ensure that the departure time at the corresponding downstream station satisfies the minimum dwell time constraint. If several trains share the same dwell time pattern, we only need to construct the same time-space network for these trains, since each dwell time pattern corresponds to a set of parallel state transition arcs at each segment and accordingly the same state transition network. Thus, we only need to solve the dynamic programming problem for the same dwell time pattern once. For outbound trains, dynamic programming computation is carried out from the last stage (i.e. station m) backward to the first stage (i.e. the current initial station for train i in a partial schedule). At each stage, we perform a pair-wise comparison backward at each time point between its outgoing state transition arc and its outgoing station waiting arc. In other words, the cost label associated with time t at each station is updated by the minimum of (1) state transition

cost + the cost label of the downstream node of the state transition arc and (2) station waiting cost + the cost label of time t+1 at the same station. The predecessor of each node can be one of two upstream nodes, indicating that either the train should leave at time t now (corresponding to the state transition arc) or wait till time t+1 (corresponding to the station waiting arc). It should be noted that, the latter case does not imply that the train has to leave the station at t+1, since the predecessor of the node associated with t+1 might be the node associated with t+2, meaning that the train will wait for at least 2 minutes after time t. The reader is referred to a paper by Luh et al. (1999) for a more detailed description on solving the general job shop scheduling problem using Lagrangian relaxation and dynamic programming techniques. It is easy to verify that the dynamic programming algorithm for a single train has a complexity of O($T\times m$). At the end of the dynamic programming algorithm, each node in the time-space network has been scanned and associated with a predecessor. In other words, there is an optimal path from each departure time at the initial station to the last station, and then we can backtrack the path in the time-space network based on the departure time of a train from its initial station. A similar procedure can be applied to inbound trains.

Since the dual cost function (19) is not differentiable everywhere, we solve the dual problem by updating $\{\pi_{j,t}, \rho_{u,t}\}$ using the subgradient method, which is intended to iteratively adjust the resource prices by setting

$$\begin{bmatrix}\pi\\ \rho\end{bmatrix}^{q+1} = \begin{bmatrix}\pi\\ \rho\end{bmatrix}^{q} + \alpha^q d^q \tag{21}$$

where superscript q is the iteration index used in the dual updating procedure, and π^q, ρ^q, α^q, d^q denote the segment and station multiplier values, step size and search direction at iteration q, respectively. To overcome "zip-zag" courses in the optimum search process, the search direction is constructed recursively as

$$\begin{cases} d^1 = \Delta^1 \\ d^q = \dfrac{1}{1+\lambda}\Delta^q + \dfrac{\lambda}{1+\lambda} d^{q-1} \qquad \forall q > 1 \end{cases} \tag{22}$$

$$\text{where } \Delta^q = \left[\sum_i \delta_{i,j,t}^{(q)} - 1, \sum_i \varepsilon_{i,u,t}^{(q)} - 1 \right]^T . \tag{23}$$

Our testing experiences indicate that the parameter λ=0.7 is an appropriate value to combine the previous multiplier values with the current subgradient information. The step size parameter is updated as

$$\alpha^q = \mu^q \frac{(\overline{L} - L^q)}{\left\|\Delta^q\right\|}, \tag{24}$$

where $\overline{L}$ is the optimal solution and L^q is the value of L at iteration q. In our implementation, a feasible solution generated from a priority rule-based heuristic method is used to replace the optimal solution $\overline{L}$. Note that, $0<\mu^q<2$ is required to ensure theoretical convergence. Clearly, a small step size μ^q could lead to a slow convergence rate, but a large step size could result in serious oscillations. In our experiment, μ^q is set to $1/(q+1)$, and we stop decreasing μ^q after a certain number of iterations.

It is well known that the applicability and effectiveness of the Lagrangian method rely on having a relatively small number of constraints to dualize. Since there are a total of $(2m-1)\times T$ station and segment resources, the dimension of the capacity constraints in a train scheduling problem is extremely large, leading to a very large $\left\|\Delta^q\right\|$ but a negligible step size in Equation (24). Recognizing that most of the resource constraints are non-binding in an optimal train scheduling solution, a relax-and-cut logic described in Caprara et al. (2002) is adapted here to dynamically relax resource capacity constraints by only dualizing a subset of constraints at every iteration. Specifically, if a resource has not been used within recent several iterations, the algorithm automatically resets the price for the unused resource back to zero. With this dynamic constraint generation scheme, the set of Lagrangian multipliers varies along the iterative process.

In the branch-and-bound process, the previously optimized values of the multipliers at a predecessor node are used as the initial values for multipliers associated with the corresponding successor nodes.

5. Upper bound generation

To find an optimal solution, the branch-and-bound method needs to investigate a huge number of nodes. In contrast, a heuristic method examines only a small number of nodes to construct feasible and satisfactory solutions. In general, a priority rule approach evaluates partial schedules associated with active nodes, and then chooses a subset of branches to search further. Specifically, simple priority rules employ a local evaluation function to select one node at each level of the search tree. For instance, for all the trains involved in a conflict, a minimum-delay rule selects a node that has the least total delay. Although simple priority rules can quickly construct a feasible solution, it is always desirable to use the following improved priority rules to reduce the optimality gap by exploring a larger solution space.

(1) Global priority rule
A global rule attempts to evaluate partial schedules at branches from a broader perspective. A typical implementation is to complete alternative partial schedules for each branch by applying a simple local rule successively until feasible solutions are found, and then using the corresponding objective function value as the priority weight of a branch.

(2) Look-ahead method
Compared to the global priority rule, a look-ahead method considers a shorter evaluation horizon and thus examines fewer nodes. For example, the evaluation horizon can just cover the path trajectories for those trains involved in the current conflict.

(3) Beam search
A beam search algorithm branches nodes level by level, keeps only γ promising nodes at each level, and prunes off the other nodes permanently. Parameter γ is typically referred to as beam width, while either a local rule or global rule can be incorporated to determine promising nodes. As a beam search only investigates limited nodes at each level, it can be viewed as an adaptation of the breadth first search method.

6. Numerical experiments

We want to investigate three important features through a series of numerical experiments: (1) the quality and computational costs of the proposed lower bound rules, (2) the effectiveness of the branch-and-bound search procedure with embedded lower bounds, and (3) the solution quality of the priority rule-based heuristic algorithms. 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 used in 1995, 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 diagram schedules 30 passenger trains and 32 freight trains along this high density rail corridor. Because passenger trains are assumed to hold higher priority than freight trains in our study, we can first schedule a passenger train timetable, and then construct a freight train timetable later. Thus, this study focuses on how to schedule passenger trains to minimize the total travel time. It is desirable to apply and test the proposed algorithm using different real instances, as the computational time of train timetabling algorithms can be affected by station layouts, segment capacities, and the number of possible conflicts. In the following numerical experiments, we focus on the impact of different departure time intervals and numbers of trains to be scheduled, since these two factors mainly determine the structure of a train timetable and the resulting number of possible conflicts. However, the real-world timetable used in this study only offers one instance of departure time pattern with a fixed number of trains. To allow a comprehensive and systematic assessment, we construct random instances to evaluate the performance of the proposed algorithm. The scheduling algorithms are implemented in Visual C++ 6.0 on the Windows XP operating system, and all the experiments are conducted on a PC with a Pentium IV 3.0 GHz processor and 4.0 GB RAM.

6.1 Quality and computational time of lower bound rules

The algorithm parameters for the Lagrangian relaxation-based bound and dynamic programming algorithm are first fine-tuned through five randomly generated train schedules, and then we apply the best settings for the remaining experiments. The quality of a lower bound is measured by a percentage gap between a lower bound estimate and the corresponding optimal value for the *additional train delay* in a partial schedule. It should be noticed that, if the gap is expressed in terms of the *total travel time*, a large constant (i.e. a total free running time) is inserted into both the nominator and the denominator of the relative gap ratio, allowing the results to be much closer to 100%. Using the original train departure times, we first examine the performance of the two lower bound rules as a function of the number of trains. As shown in Fig. 11 and 12, the LR-based rule iteratively improves lower bound estimates, but the additional gap reduction becomes insignificant after 60 iterations. On the other hand, the LR-based rule performs well when handling a small number of trains, but still produces a 50% gap, on average, for large instances. In these complicated cases, many timetables share the same or almost the same dual cost, so the solution degeneracy issue frequently arises, leading to considerable oscillations and inefficiency in the search process. In contrast, the simple crossing conflict-based lower bound rule gives quite consistent estimates with an average gap of 50%-60%, shown in Fig. 12. As expected, when the maximum dwell time for passenger trains is shortened, the strong lower bound formula (i.e. $g+h$ in Eq. (28)) is more likely to be applied, leading to slightly better results. The average departure time interval is another important factor that complicates the search process. Because the actual timetable used in this study only represents one departure time pattern, we generate random instances with different departure time intervals in the following sensitivity analysis. Each data point in Fig. 13 represents the average gap of 10 schedules with 12 trains, in which the departure time intervals follow the third order Erlang distribution (calibrated from the actual timetable). When the departure time interval is large, for example, more than 2 hours, the LR-based lower bound rule outperforms the CC-based rule, even with just 30 iterations. On the other hand, the CC-based lower rule always produces gaps within a range of 50% - 60%, since this ad-hoc rule only considers the existing crossing conflicts in a partial schedule.

6.2 Branch-and-bound algorithm with lower bound rules

Solving large-scale scheduling instances requires an effective and efficient lower bound rule that can be embedded into the branch-and-bound search procedure. Our numerical experiments show that, for a schedule with 20 trains, a single iteration of the LR-based lower bound rule takes an average of 0.141 milliseconds, while the CC-based lower bound rule uses only 0.03 milliseconds to provide an estimate within a 50% optimality gap. As shown in Fig. 11, generating a LR-based lower bound estimate with similar quality typically requires at least 40 iterations, that is, 5.64 milliseconds. Because of the efficient longest path algorithm, generating a schedule at each node (i.e. Step 4.3 in Algorithm 1) only takes about 0.01-0.05 milliseconds. Thus, taking a significant amount of time for constructing a sharp LR-based lower bound at each node might not lead to overall computational savings. As a result, the following experiments only incorporate the simple CC-based rule to solve large size instances. To help the reader understand the complexity of the problem solved, we present a sample train diagram with 26 passenger trains in Fig. 14. Note that, after generating an initial train timetable, the planner can further refine the departure times and stop schedules of passenger trains in order to improve the level-of-service for travelers and provide better time slots for freight trains.

The following experiments are based on 10 randomly generated scenarios with average departure time intervals of 90 minutes. The maximum train delay time is set to 30 minutes, the search process fathoms a node if the corresponding total stop time for any train is longer than 60 minutes, and the beam search algorithm is used to generate initial feasible solutions. Without integrating any lower bounds, the branch-and-bound search algorithm can only find optimal solutions for instances with at most 13 trains within a 5 minute limit. With the stepwise refinement search procedure by Lawler and Wood (1966) and a 10 min computational time limit, we can solve all 14-train instances within a 5% optimality gap, and all 16-train and 18-train instances within a 10% gap. When the schedule size further increases, the experiments show that the average optimality gaps for train schedules with 20, 22, 24 trains only reduce to 17%, 21% and

30.5%, respectively. In comparison, by embedding the CC-based lower bound, we can optimize a 30-train instance within 4 minutes, indicating that the integration of an efficient lower bound into the branch-and-bound search procedure results in an optimal solution for a realistic large-scale schedule. Table 4 further lists the computational costs in terms of the number of computed nodes and computing time (in seconds) for the branch-and-bound algorithms with the embedded CC-based lower bound rule. Obviously, the inclusion of the lower bound rule dramatically decreases computational costs. It should be remarked that, to handle the cyclic scheduling issue in a 24 hour timetable, this study computes the schedule starting from 12 AM, and it assumes that a train crossing the 12AM time line always yields to any trains that have been scheduled previously after 12AM.

6.3. Effectiveness of priority rule-based heuristic methods

With available optimal solutions, we want to further evaluate the solution quality of heuristic algorithms in terms of total train delays. The quality measure is expressed as the percentage difference between a heuristic solution and the corresponding exact optimal solution. Our results indicate that the minimal-delay rule produces slightly better results than the first-come-first-serve rule, and that their solutions are within 7% of optimality on average. Table 5 shows the performance of improved priority rules as a function of the number of trains. To avoid the complicated cyclic scheduling issue, we do not consider more than 24 trains here. All the improved algorithms embed the minimal delay rule as the local evaluation criterion. In the beam search implementation, only the local evaluation rule is used to select promising nodes, and the beam width, γ, is set to 8 based on calibration results using a separate data set. Compared to the simple priority rule, all the improved methods considerably reduce the relative optimality gap from 6% to less than 5%. The beam search algorithm gives the best results with an average gap of 2.10%, while the global priority rule and look-ahead rule are within a range of 3%-4%. Since all improved priority rules obtain better solutions by examining more nodes in the search tree, it is important to evaluate trade-offs between solution quality and computational effort. As shown in Table 6, the global priority rule needs to examine a relatively large number of nodes. As a simplification of the global priority rule, the look-ahead method can decrease the number of computed nodes by 40% but only leads to a small increase in the relative optimality gap by 12%. Overall, the experimental results suggest that the beam search algorithm offers superior performance with the least computational effort.

7. Conclusions

This study aims to solve the single-track train timetabling problem efficiently, such that the total train travel time is minimized. The train timetabling problem is formulated as a generalized resource-constrained project scheduling problem with segment and station headway capacities as limited resources. A branching scheme is proposed to eliminate train conflicts chronologically by adding precedence relations between trains. Furthermore, the resulting subproblem is reformulated as a longest path problem, which can be solved efficiently by a label correcting algorithm.

By relaxing the coupling track capacity constraints, a Lagrangian relaxation-based lower bound rule is designed to decompose the original complex problem into subproblems for individual trains that can be rapidly solved by a dynamic programming algorithm. An alternative lower bound rule is also developed to estimate the minimum additional train delay required for resolving all the existing crossing conflicts in a partial schedule. As indicated by our results, the Lagrangian relaxation-based lower bound rules can generate tight lower bound estimates for relatively simple instances, but require considerable computational effort when solving large-scale cases. With a simple computation scheme, the proposed ad-hoc crossing conflict-based lower bound rule is able to provide a reasonably good estimate for the train delay and it can considerably improve the branch-and-bound search efficiency. Moreover, improved priority rules significantly reduce the optimality gap by exploring a larger search space, and a beam search algorithm offers superior performance with limited computation costs. In general, the above innovative and integrated branching and bounding schemes are offered as a contribution to the growing body of work on computationally efficient and practically useful algorithms for rail scheduling applications.

Our on-going research focuses on three major directions. First, this current study assumes constant free running times, so a natural extension is to allow variable running times for more realistic applications. Because introducing any new problem dimensions typically increases the computational complexity quite steeply, it is undoubtedly vital to develop efficient and effective approximation and heuristic schemes along this line of research. As indicated in the case study, further successful applications call for exact algorithms that can optimize 24-hour cyclic timetables. It would also be interesting to develop efficient Lagrangian relaxation techniques for train scheduling applications with different criteria, especially in a profit-maximizing framework.

Acknowledgements

The study has benefited from the encouragement of Dr. Hani S. Mahmassani and comments from Dr. Peng Du. The authors would also like to thank Dr. Malachy Carey and two anonymous referees for their constructive suggestions. The authors are responsible for all the results and opinions expressed in this paper.

References

Adenso-Diaz, B., Gonzalez, M. O., Gonzalez-Torre, P., 1999. On-line timetable re-scheduling in regional train services. Transportation Research Part B 33(6), 387–398.

Ahuja, R.K., Magnanti, T.L., Orlin, J.B., 1993. Network flows: theory, algorithms, and applications. Prentice Hall, NJ.

Assad, A., 1980. Models for rail transportation. Transportation Research Part A, 14(3), 205–220.

Bartusch, M., Mohring, R.H., Radermacher, F.J., 1988. Scheduling project networks with resource constraints and time windows. Annals of Operations Research 16(1), 201-240.

Brannlund, U., Lindberg, P.O., Nou, A., Nilsson, J. E., 1998. Railway timetabling using Lagrangian relaxation. Transportation Science 32(4), 358–369.

Cai, X., Goh, C. J., Mees, A. I., 1998. Greedy heuristics for rapid scheduling of trains on a single track. IIE Transactions 30 (5), 481–493.

Caprara, A., Fischetti, M., Toth, P., 2002. Modeling and solving the train timetabling problem. Operations Research 50(5), 851-861.

Carey, M., 1994a. A model and strategy for train pathing with choice of lines, platforms and routes. Transportation Research Part B 28(5), 333–353

Carey, M., 1994b. Extending a train pathing model from one-way to two-way track. Transportation Research Part B 28(5), 395–400.

Carey, M., Lockwood, D., 1995. A model, algorithms and strategy for train pathing. Journal of the Operational Research Society 46(8), 988–1005.

Chen, B., Harker, P. T., 1990. Two moments estimation of the delay on single-track rail lines with scheduled traffic. Transportation Science 24(4), 261-275.

Cordeau, J.-F., Toth, P., Vigo, D., 1998. A survey of optimization models for train routing and scheduling. Transportation Science 32(4), 380–404.

Dorfman, M.J., Medanic, J., 2004. Scheduling trains on a railway network using a discrete event model of railway traffic. Transportation Research Part B 38(1), 81-98.

Greenberg, H.H., 1968. A branch-and-bound solution to the general scheduling problem. Operations Research 16(2), 352– 361.

Higgins, A., Kozan, E., Ferreira, L., 1996. Optimal scheduling of trains on a single line track. Transportation Research Part B 30(2), 147–161.

Higgins, A., Kozan, E., 1998. Modeling train delays in urban networks. Transportation Science 32(4), 346–357.

Jovanovic, D., Harker, P.T., 1991. Tactical scheduling of rail operations: the SCAN I system. Transportation Science 25(1), 46–64.

Kraay D. R., Harker P. T., 1995. Real-time scheduling of freight railroads. Transportation Research Part B 29(3), 213-229.

Kroon, L.G., Peeters, L.W., 2003. A variable trip time model for cyclic railway timetabling. Transportation Science 37(2), 198-212.

Lawler, E.L., Wood, D.E.. 1966. Branch and bound methods: A survey. Operations Research 14(4), 699-719.

Luh, P. B., Chen, D., Thakur, L. S. 1999. An Effective Approach for Job-Shop Scheduling with Uncertain Processing Requirements. IEEE Transactions on Robotics and Automation 15(2), 328-339.

Newman, A.M., Nozick, L., Yano, C.A., 2002. Optimization in the rail industry. Handbook of Applied Optimization. 704-718.

Petersen, E.R., Taylor, A.J., 1982. A structured model for rail line simulation and optimization. Transportation Science 16(2), 192-206.

Petersen, E.R., Taylor, A.J., Martland, C.D., 1986. An introduction to computer aided train dispatching. Journal of Advanced Transportation 20, 63–72.

De Reyck, B., Herroelen, W., 1998. Branch-and-bound algorithm for the resource-constrained project scheduling problem with generalized precedence relations. European Journal of Operational Research 111(1), 152-174.

Sabuncuoglu I., Bayiz M., 1999. Job shop scheduling with beam search. European Journal of Operational Research 118(2), 390-412.

Sahin, I., 1999. Railway traffic control and train scheduling based on inter-train conflict management. Transportation Research Part B 33(7), 511–534.

Sprecher, A., Drexl, A., 1998. Multi-mode resource-constrained project scheduling by a simple, general and powerful sequencing algorithm. European Journal of Operational Research 107(2), 431-450.
Szpigel, B., 1973. Optimal train scheduling on a single track railway. Operations Research'72, North-Holland, Amsterdam, Netherlands, 343–352.
Zhou, L., Hu S., Ma J., Yue, Y., 1998. Network hierarchy parallel algorithm of automatic train scheduling. Proceedings of the Conference on Traffic and Transportation Studies, ICTTS, 358–368.
Zhou, X., Zhong, M., 2005. Bi-criteria train scheduling for high-speed passenger railroad planning applications. European Journal of Operational Research 167(3), 752-771.

Appendix

As shown in Fig. A1, diagrams I, II, III and IV illustrate possible headway constraints between the arrival and departure times of two trains (say trains *a* and *b*) at a station, while these two trains are running at different segments:

(I) between the arrival times of trains *a* and *b* running in opposite directions,
(II) between the departure times of trains *a* and *b* running in opposite directions,
(III) between the departure time of train *a* and the arrival time of train *b* running in the same direction,
(IV) between the arrival time of train *b* and the departure time of train *a* running in the same direction.

If there is only one line going into and coming out of a station, then the above four types of minimum headway restrictions should be imposed between trains entering or leaving the station. For a train station that have multiple platforms, if there are several lines running in and out of the station and these lines cross each other, then the minimum headways might be also considered.

For a station with the track configuration shown in Fig. 1, due to safety concerns, the railroad system in China still imposes the minimum headway constraint between the arrival times of two trains (i.e. type I headway) to avoid head-on collisions, especially under one of the following conditions: (1) one of trains is a passenger train, (2) the station is adjacent to a steep (>0.4%) downhill grade, (3) the paths of two trains in the station are not protected by track separation equipments. In addition, the minimum headway restrictions between an arriving train and another departing train (i.e. type III and type IV headways) are also required in China for train stations that use semi-automatic signals.

As an example, Fig. A2 further illustrates the headway constraints for two trains arriving from different directions at station 1. A minimum headway of g_1 minutes is imposed between the arrival times of trains *a* and *b* at station 1. A minimum headway of h_1 minutes is required between the arrival time of train *b* and the departure time of train *a* at station 1. The headway between the arrival time of train *a* and the departure time of train *b* at station 1 should be no less than h_2 minutes. In the rail system in China, g_1 is typically greater than h_1 and h_2.

As our case study focuses on passenger train timetabling for a series of stations shown in Fig. 1, the discussion in our paper only considers the type I headway requirement. It should be remarked that, the other types of station headways can be also modeled in the proposed branch and bound algorithm. That is, based on a generalized project scheduling solution framework, these headway requirements can be converted to start-to-start, start-to-finish, or finish-to-start types of precedence constraints between two operations (with the minimum time lag). The proposed Lagrangian lower bound rule needs to be enhanced to introduce more resource classes that correspond to different types of headways. Since the proposed ad-hoc lower bound focuses on crossing conflicts (type I), it is not suitable for estimating the delay associated with the types II, III and IV headway requirements.

For a general rail line without the above station headway requirements, the station arrival headway constraint in our model can be relaxed. Relaxing this constraint only leads to a slight change in our lower bound estimator. Specifically, for a crossing conflict, our current lower bound is $g+h$, the new lower bound without the arrival time headway constraint should be $h+h=2h$.

List of Figures

List of Tables

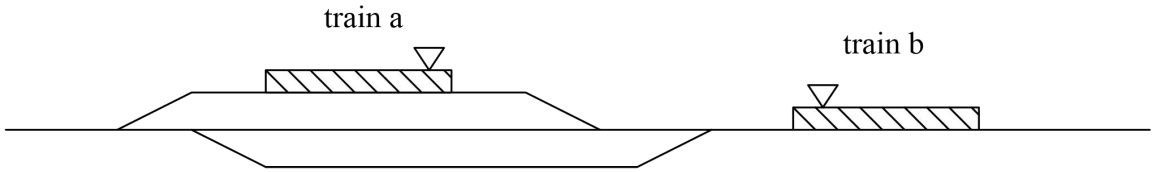


Fig. 1. Two trains meet at a station on a single-track rail line.

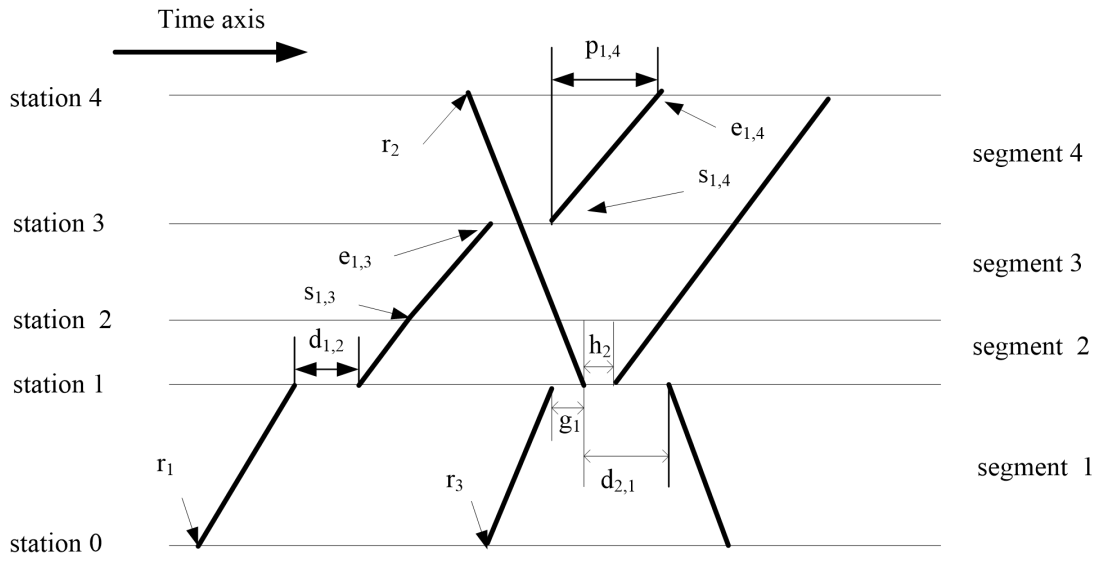


Fig. 2. Illustration of a single-track train schedule.

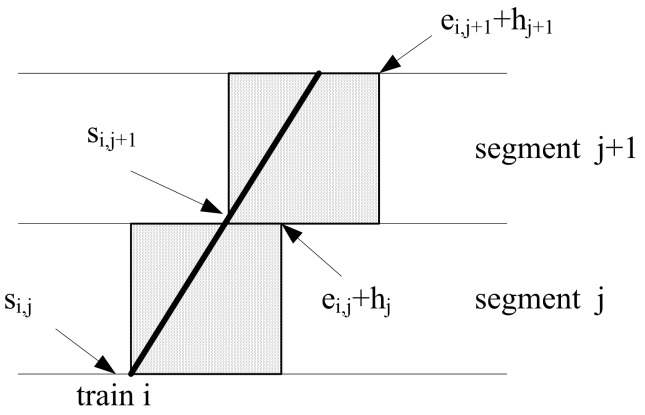


Fig. 3. Illustration of generalized resource constraint scheduling formulation.

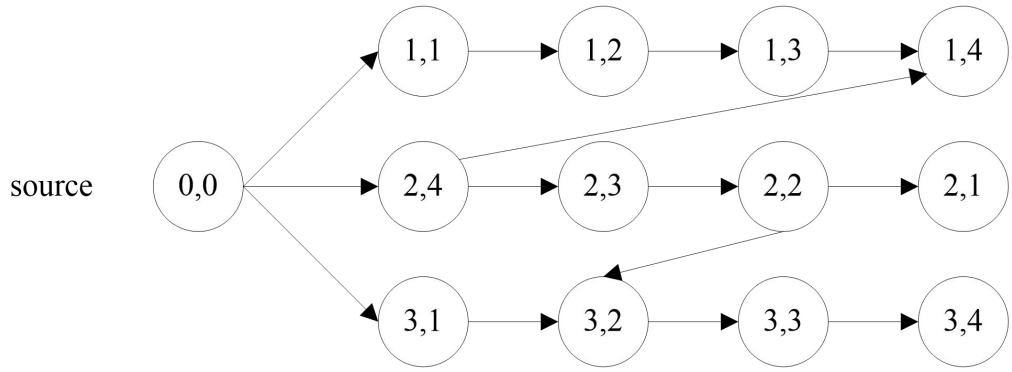


Fig. 4. Precedence constraints graph corresponding to the train schedule in Fig. 2.

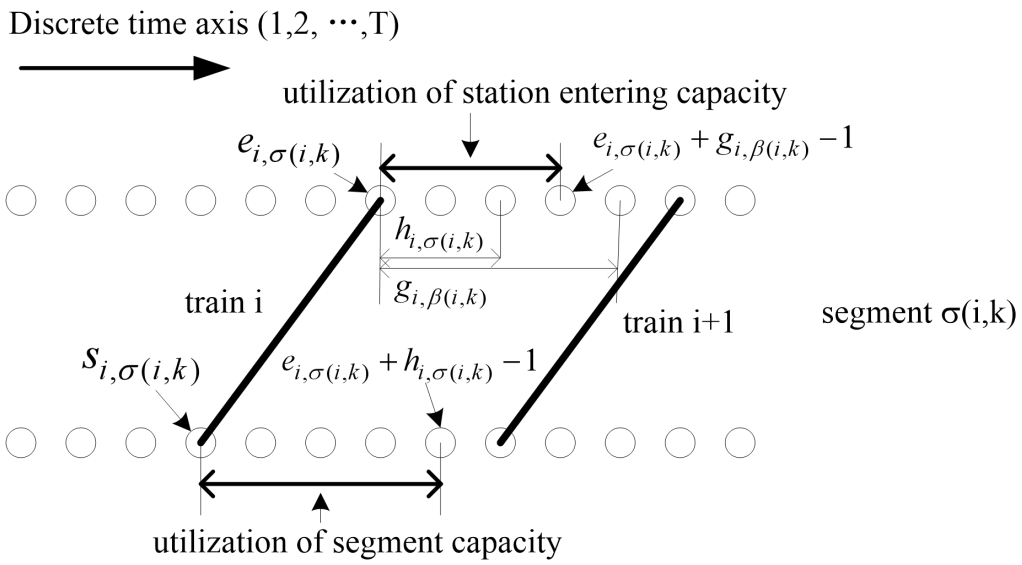


Fig. 5. Illustration of resource utilization by train *i* at segment *σ*(*i*,*k*).

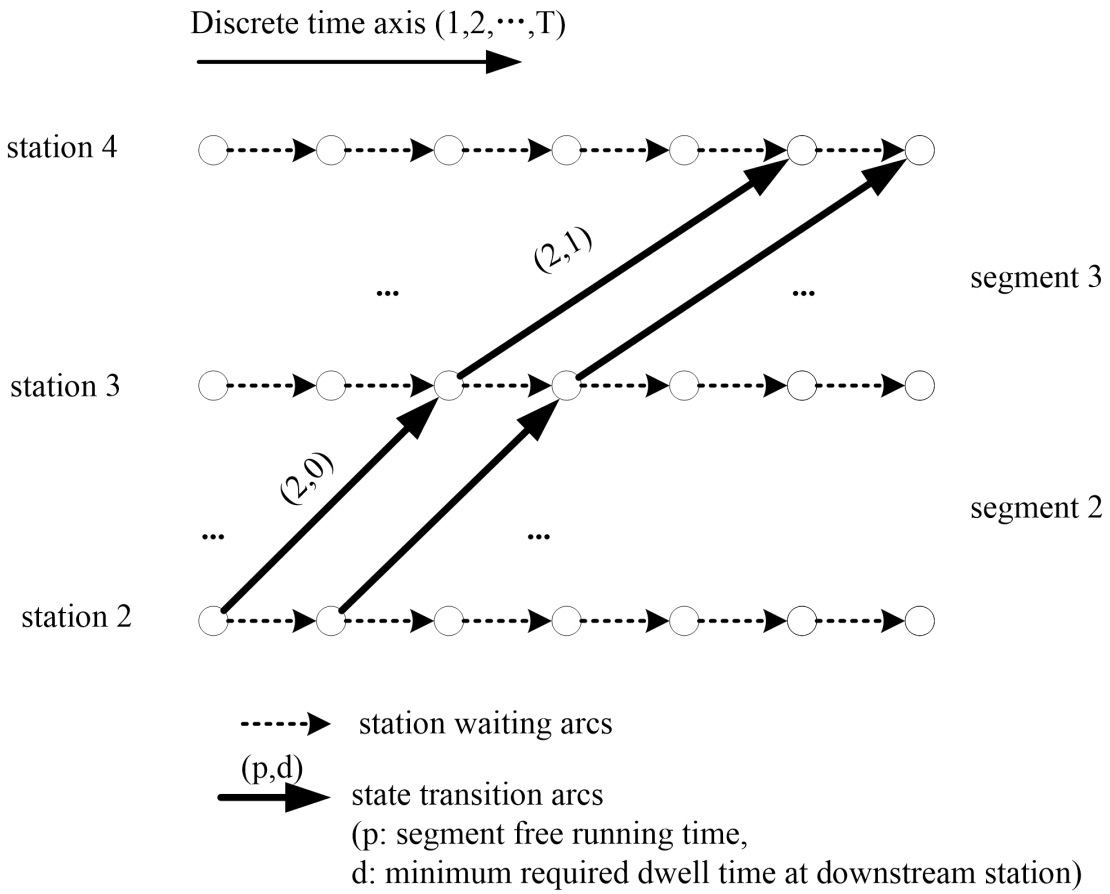
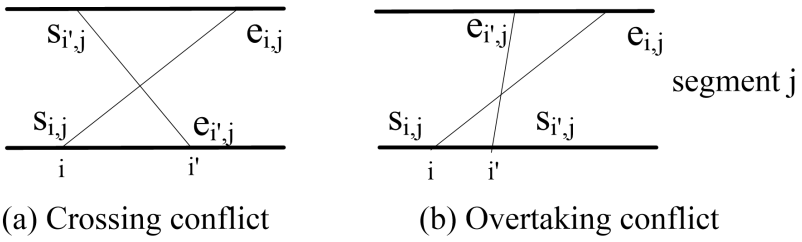


Fig. 6. Illustration of a time-space network used in dynamic programming algorithm.



(a) Crossing conflict (b) Overtaking conflict

Fig. 7. Illustration of a crossing conflict and overtaking conflict.

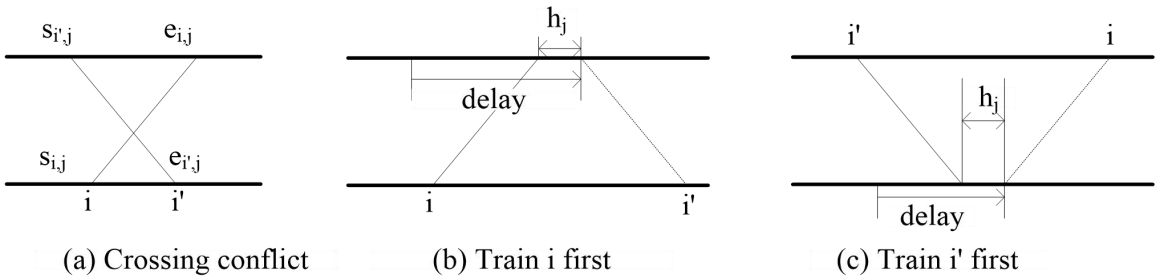


Fig. 8. Illustration of additional delay for resolving a conflict at a segment.

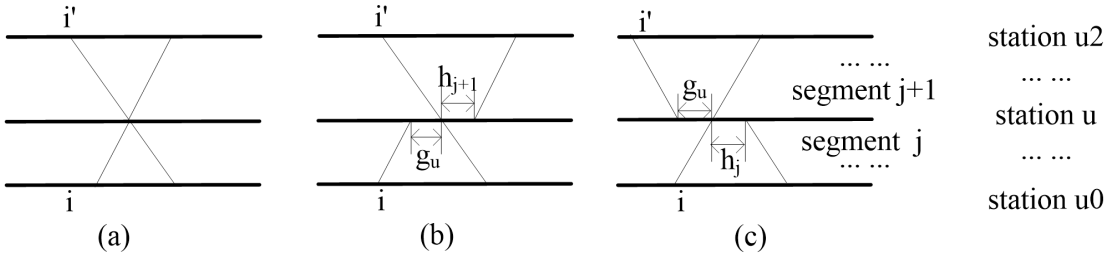


Fig. 9. Illustration of resolving a crossing conflict at an intermediate station.

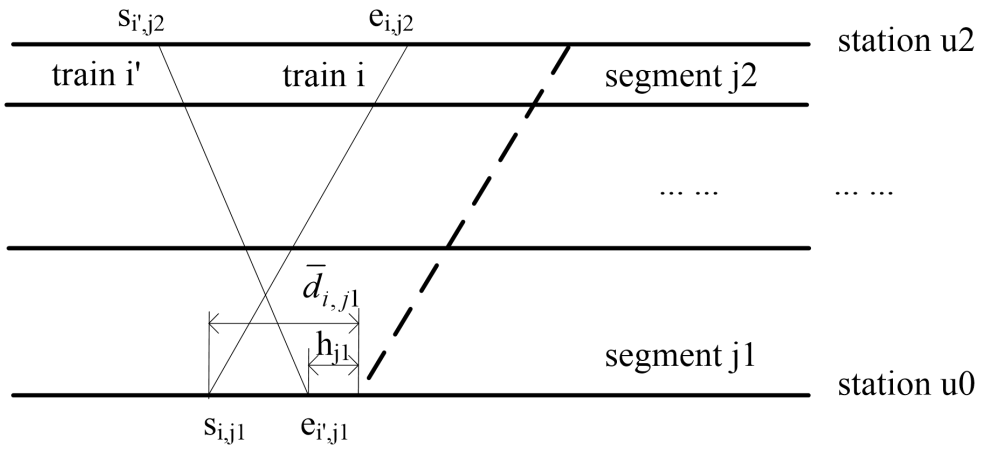


Fig. 10. Feasible region for resolving a crossing conflict at an intermediate station.

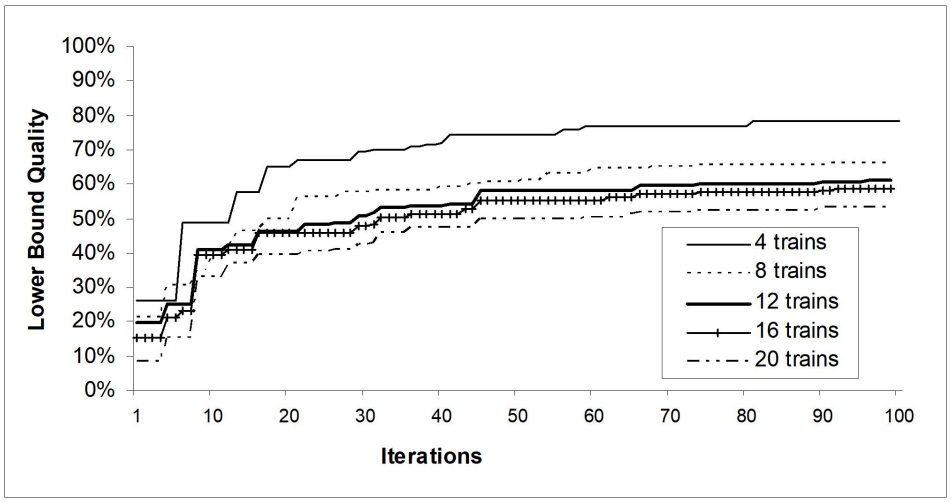


Fig. 11. Estimation quality of the Lagrangian relaxation-based lower bound rule.

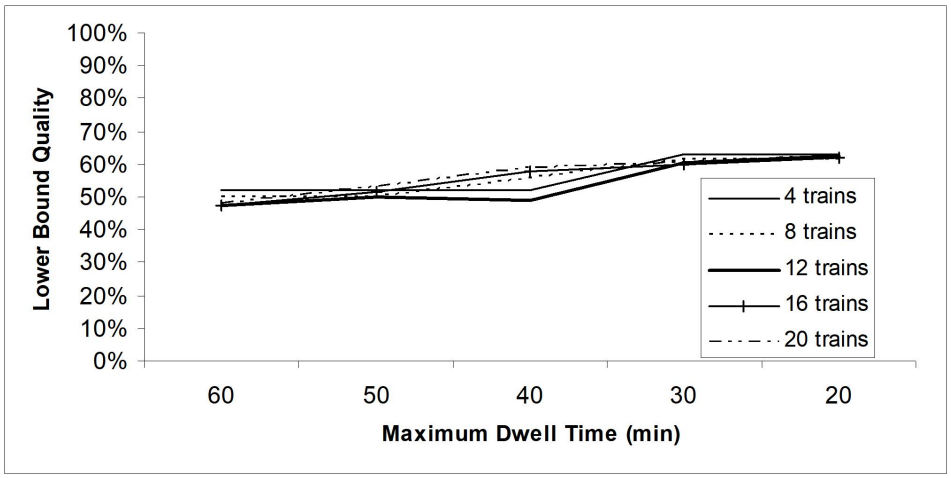


Fig. 12. Estimation quality of the crossing conflict-based lower bound rule.

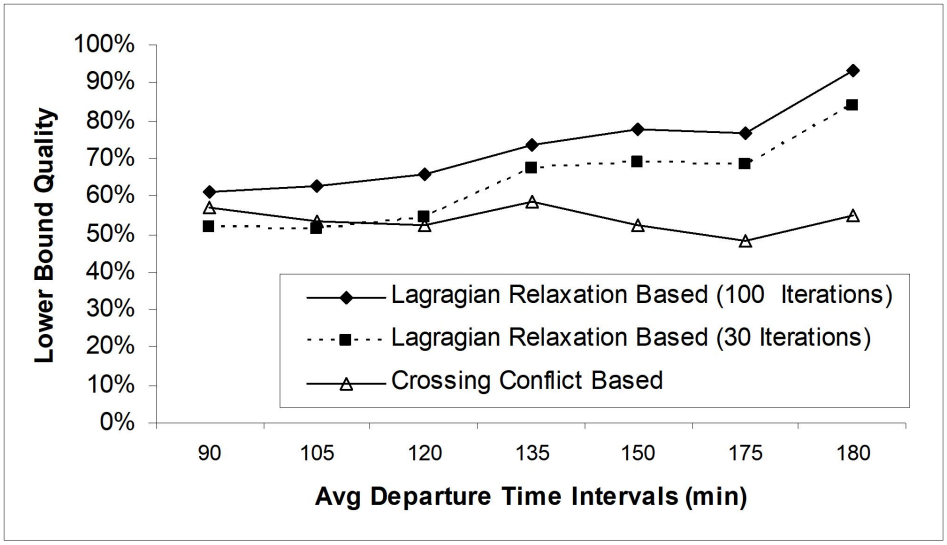


Fig. 13. Estimation quality of lower bound rules for 20 trains as a function of the average departure time interval.

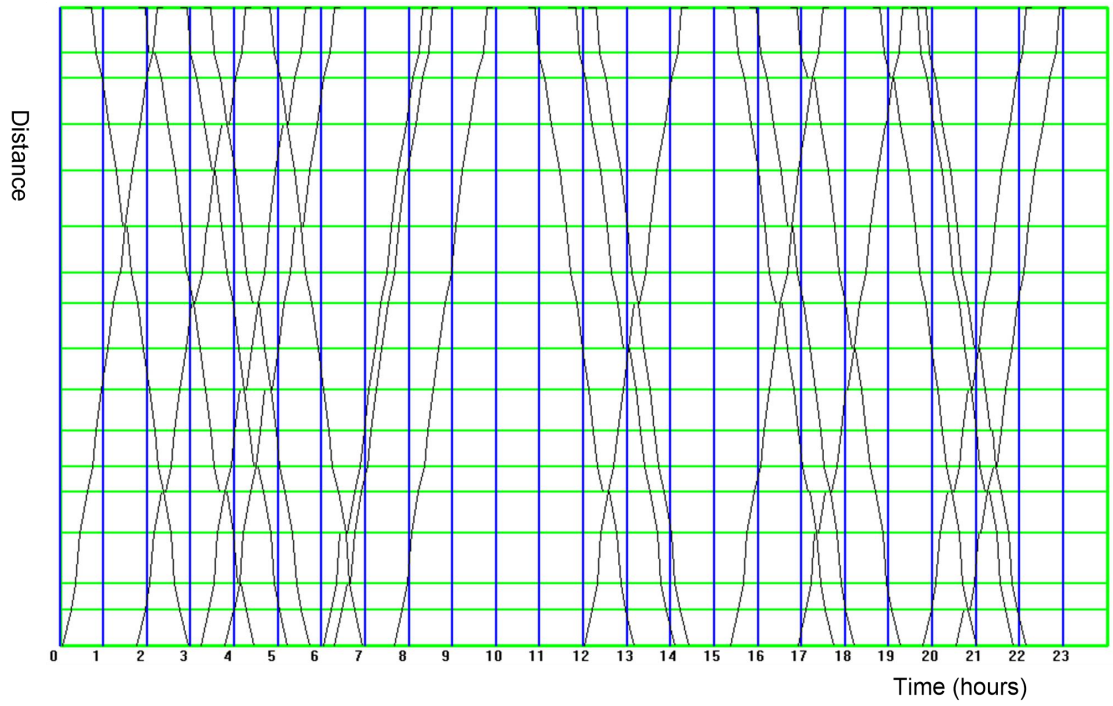


Fig. 14. Sample train diagram with 26 passenger trains.

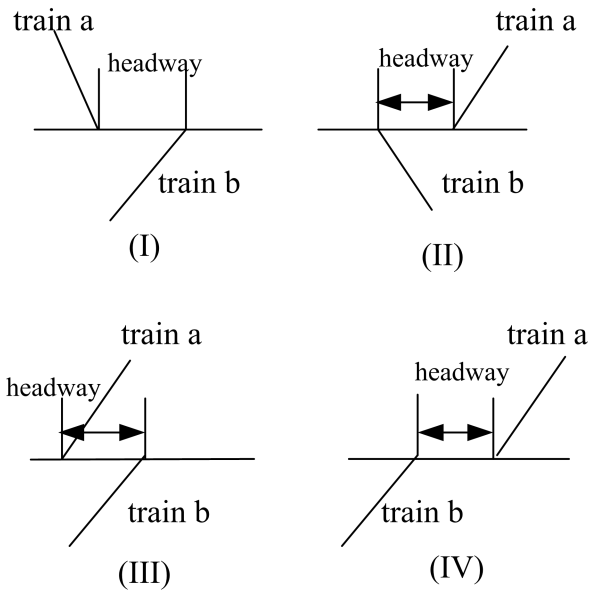


Fig. A1. Possible headway constraints between the arrival and departure times of two trains at a station.

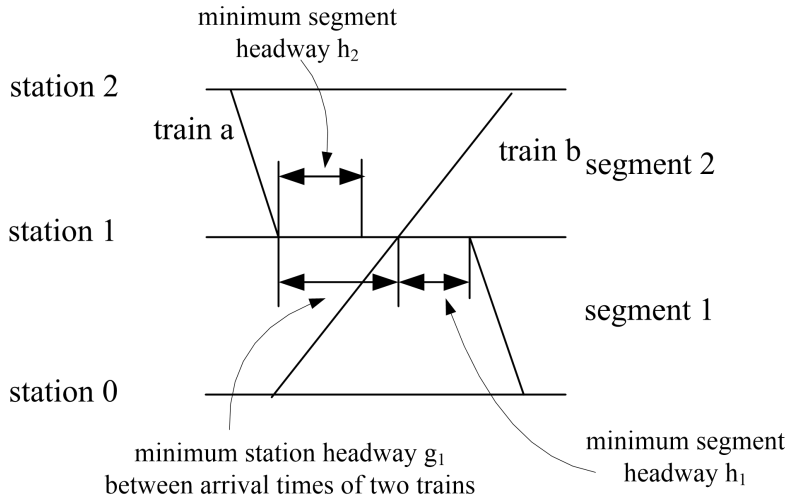


Fig. A2. Headway constraints for two trains arriving from opposite directions at station 1.

Table 1 Subscripts and parameters used in mathematical formulations

Symbol	Definition
i	= train index
j	= segment index
k	= segment sequence number in a train route
u	= station index
I	= set of trains, $\| I \| = n$
J	= set of segments, $\| J \| = m$
U	= set of stations, $\| U \| = m+1$
t	= time index, $t = 1,\ldots, T$, T is the planning time horizon under consideration
$o(i)$	= direction indicator for train i, $o_i=0$ for an inbound train and $o_i=1$ for a outbound train
$\sigma(i,k)$	= segment index of the k^{th} traveling segment in a route for train i, $\sigma(i,k)=k$ for outbound trains, $\sigma(i,k)=m+1-k$ for inbound trains
$\beta(i,k)$	= downstream station number of the k^{th} traveling segment in a given route for train i, $\beta(i,k)=k$ for outbound trains, $\beta(i,k)=m-k$ for inbound trains
r_i	= planned departure time for train i at its first station
$p_{i,j}$	= free running time for train i at segment j
$d_{i,j}$	= minimum required station dwell time before train i entering segment j
$\bar{d}_{i,j}$	= maximum allowed station dwell time before train i entering segment j
h_j	= minimum headway between arrival and departure times of two consecutive trains at segment j
g_u	= minimum headway between arrival times of two consecutive trains at station u
M	= sufficiently large constant

Table 2 Decision variables used in mathematical formulations

Symbol	Definition
$s_{i,j}$	= entering time for train i at segment j, i.e. start time for job i on machine j
$e_{i,j}$	= leaving time for train i at segment j, i.e. end time for job i on machine j
$y_{i,i',j}$	= 1 if train i is scheduled before train i' on segment j, 0 otherwise
$\delta_{i,j,t}$	= 1 if train i occupies segment j at time t, 0 otherwise
$\varepsilon_{i,u,t}$	= 1 if train i occupies station u at time t, 0 otherwise

Table 3 Notations used in the branch-and-bound solution procedure

Symbol	Definition
v,w	= node index
$t(i,j)$	= task index, i.e. activity of train i at segment j
ANL	= active node list
UB	= upper bound of optimal solutions
$LB(v)$	= lower bound of a solution at node v
$\Omega(v)$	= set of trains involved in the earliest conflict at node v
$P(v)$	= set of precedence constraints at node v
θ	= threshold of quality gap

Table 4 Computational performance of the search algorithm with the crossing conflict based lower bound

# of Trains	14	16	18	20	22	24	26	28	30
# of computed nodes	7414	44258	72523	80586	192115	365376	1514743	2732981	4388249
Computation Time (sec)	0.125	1.063	2.594	3.390	7.563	12.984	52.734	109.932	221.459

Table 5 Relative optimality gaps for priority rules

Number of trains	12	14	16	18	20	22	24	Mean
Simple priority rule	4.87%	6.40%	5.47%	7.58%	6.58%	5.08%	7.61%	6.16%
Global priority rule	0.47%	4.46%	3.44%	4.17%	3.11%	3.91%	4.91%	3.50%
Look-ahead method	0.43%	5.18%	3.38%	4.12%	4.35%	4.65%	5.65%	3.90%
Beam search (γ=8)	1.85%	1.26%	1.65%	1.61%	3.52%	2.69%	2.69%	2.10%

Table 6 Number of computed nodes for priority rules

Number of trains	12	14	16	18	20	22	24	Mean
Simple priority rule	22.5	28.8	34.8	41.0	47.1	55.4	55.6	40.74
Global priority rule	238.2	389.2	547.8	990.1	1502.3	1768.4	2422.7	1122.67
Look-ahead method	220.7	358.3	461.2	628.0	756.4	1030.2	1275.8	675.80
Beam search (γ=8)	221.0	234.3	238.7	284.9	331.2	383.8	412.7	300.94