Single-line rail rapid transit timetabling under dynamic passenger demand

Eva Barrena a,b,* , David Canca c, Leandro C. Coelho a,d, Gilbert Laporte a,b

a Interuniversity Research Center on Network Enterprise, Logistics and Transportation (CIRRELT), Canada
b HEC Montréal, 3000 chemin de la Côte-Sainte-Catherine, Montréal H3T 2A7, Canada
c School of Engineering, University of Seville, Avenida de los Descubrimientos s/n, 41092 Seville, Spain
d Faculté des sciences de l’administration, Université Laval, 2325 rue de la Terrasse, Québec G1K 7P4, Canada

Abstract
Railway planning is a complex activity which is usually decomposed into several stages, traditionally network design, line design, timetabling, rolling stock, and staffing. In this paper, we study the design and optimization of train timetables for a rail rapid transit (RRT) line adapted to a dynamic demand environment, which focuses on creating convenient timetables for passengers. The objective is to minimize the average passenger waiting time at the stations, thus focusing on passenger welfare. We first propose two mathematical programming formulations which generalize the non-periodic train timetabling problem on a single line under a dynamic demand pattern. We then analyze the properties of the problem before introducing a fast adaptive large neighborhood search (ALNS) metaheuristic in order to solve large instances of the problem within short computation times. The algorithm yields timetables that may not be regular or periodic, but are adjusted to a dynamic demand behavior. Through extensive computational experiments on artificial and real-world based instances, we demonstrate the computational superiority of our ALNS compared with a truncated branch-and-cut algorithm. The average reduction in passenger waiting times is 26%, while the computational time of our metaheuristic is less than 1% of that required by the alternative CPLEX-based algorithm. Out of 120 open instances, we obtain 84 new best known solutions and we reach the optimum on 10 out of 14 instances with known optimal solutions.

1. Introduction
Railway planning is a complex activity which is usually decomposed into several stages, including network design, line design, timetabling, rolling stock, and staffing (Cordeau et al., 1998; Guihaire and Hao, 2008). Traditionally, these problems have been optimized based on the operator’s perspective. In this paper, we provide a new approach, which takes the operator’s restrictions into account, but maximizes the passenger welfare, measured as the average waiting time (AWT) at the stations. The train timetabling problem solved in this paper consists of determining departure and arrival times of each train service to and from each station along a single line rail rapid transit (RRT) line over a long planning horizon characterized by a dynamic passenger arrival pattern.

* Corresponding author at: HEC Montréal, 3000 chemin de la Côte-Sainte-Catherine, Montréal H3T 2A7, Canada. Tel.: +1 514 343 6111x8707; fax: +1 514 343 7121.
E-mail addresses: eva.barrena@cirrelt.ca (E. Barrena), dco@us.es (D. Canca), leandro.coelho@cirrelt.ca (L.C. Coelho), gilbert.laporte@cirrelt.ca (G. Laporte).

http://dx.doi.org/10.1016/j.trb.2014.08.013
0191-2615/© 2014 Elsevier Ltd. All rights reserved.
A service is defined as a trip from an origin to a final destination. Here, a train refers to the service it operates. We consider the case of a two-directional two-track rail rapid transit system, where trains in opposite directions stop at different platforms, which means that departure and arrival times can be designed independently for each direction without interaction at stations (Vuchic, 2007, pp. 304–305). This is a typical design observed in many RRT systems around the world. We analyze the properties of a mathematical model and, on this basis, we develop an efficient heuristic for the timetabling problem, yielding solutions that are adapted to the demand pattern. The proposed heuristic works with small discrete time intervals, as fine as needed. In what follows, we review the relevant literature before positioning our contributions.

When designing railway timetables for passengers it is essential to respect the track capacities, and it is common to optimize an objective function relevant to the operator or to the infrastructure manager, such as minimizing the deviations from an ideal timetable, or minimizing the services' running times (Cacchiani and Toth, 2012). This can be achieved in two ways. The first is to construct a periodic timetable, i.e., a timetable that is constantly repeated, for example with departures at 03, 21 and 46 min every hour. Periodic timetabling models typically follow the Periodic Event Scheduling Problem formulation of Serafini and Ukovich (1989). Periodic timetables have the advantage of being easily memorized by passengers and have proved their ability to deal with large-scale railway networks (Kroon et al., 2009). The alternative, which consists of constructing a non-periodic timetable, is appropriate when demand cannot be assumed to be constant over time or when the service operates in long corridors with a high track density. Several different integer linear programming models have been proposed for the non-periodic timetabling problem. The model of Carey (1994) uses binary variables to describe the precedences between services, and continuous variables to represent departure and arrival times. It minimizes the total service running time. Caprara et al. (2002) have used a time discretization and a representation of the problem on a time–space graph. These authors have applied their model to a single one-way line. Their objective consists of minimizing the deviation with respect to an ideal timetable. Vansteenwegen and Van Oudheusden (2006) have studied the problem of improving passenger service on a small part of the Belgian railway network, taking into account waiting times and delays. They define a set of ideal train buffer times to ensure connections and minimize deviations with respect to them. None of the above contributions considers passenger demand in the design of the timetables. However, this problem has been studied by Ceder (2001, 2009) in a bus timetabling context. This author does not consider a demand curve as a function of time, but estimates the demand from the bus loads and aims to obtain the desired passenger loads instead of minimizing the AWT.

Heuristics have been extensively employed to solve variants of the train timetabling problem, including greedy heuristics with different objectives (Cai et al., 1998; Yuan et al., 2008), mathematical programming-based methods to minimize interchange waiting times (Wong et al., 2008), graph theory-based algorithms (Liebchen and Möhring, 2002), cut-based algorithms (Liebchen, 2005), and sequential decomposition (Caimi et al., 2009). Local search algorithms include a hybrid of tabu search and branch-and-bound (Corman et al., 2010), a combination of insertion, backtracking and dynamic route selection (Burdett and Kozan, 2010b), and a hybridization of simulated annealing with particle swarm optimization (Jamili et al., 2012). Population-based algorithms include ant colony optimization (Chen et al., 2011; Reimann and Leal, 2013) and genetic algorithms (Chung et al., 2009; Nachtigall and Vogel, 1996; Tormos et al., 2008). Constraint satisfaction techniques have been used by Odijk (1996), Abril et al. (2006), and Ingolotti et al. (2006), whereas methods based on linear programming relaxations and duality-based methods have been frequently applied (Brännlund et al., 1998; Cacchiani et al., 2008; Cacchiani et al., 2010a; Cacchiani et al., 2010b; Caprara et al., 2006; Lee and Chen, 2009; Zhou and Zhong, 2007). Other recent heuristics are those of Burdett and Kozan (2010a) and of Furini and Kidd (2013), which are based on a disjunctive graph representation of the timetable and on relaxed dynamic programming, respectively.

Most of these papers optimize an objective function relevant to the service provider. From the infrastructure manager point of view, one commonly used objective is to minimize the deviation from a solution proposed by the operator. This objective is frequently used in the periodic case. From the users' perspective, the objective of minimizing total travel time has been frequently considered. However, these contributions do not explicitly take variable passenger demand into consideration. Passenger demand is commonly assumed to be constant and is represented by a set of origin–destination (OD) matrices corresponding to different hours of a typical day.

With respect to demand behavior, the following contributions have focused on its dynamic structure. Hänseler et al. (2012) investigate ways of computing dynamic OD matrices for train stations. Specifically, these authors have designed an algorithm to predict the OD demand as a function of the train timetabling. Yano and Newman (2001) propose a dynamic programming algorithm for the timetabling of trains used to transport containers arriving dynamically at the origin. Cordone and Redaelli (2011) determine the optimal periodic timetable when the train competes with an alternative mode that offers services between the same origin and destination. They consider a logit modal split model to calculate the demand captured by each transportation mode. Finally, Niu and Zhou (2013) study the train timetabling problem under a dynamic demand scenario, and are interested in how the system behaves under congestion, namely when the demand exceeds the capacity. These authors propose a local search heuristic and a dynamic programming algorithm in order to optimize the timetable for a single station. A genetic algorithm is developed and computation experiments are performed on a real case with 13 stations.

In this paper, we consider a generalization of the non-periodic train timetabling problem under a dynamic demand pattern (Canca et al., 2014), which arises in an RRT context. We first provide two new non-linear formulations to precisely describe this problem in mathematical terms. The objective is to minimize passenger AWT at the stations, and the solutions are train timetables adjusted to a dynamic demand pattern which may neither be regular nor periodic. Barrena et al. (2014) have already proposed equivalent formulations and an exact branch-and-cut algorithm applicable to this model, but it can
only solve relatively small instances to optimality. Here, we take advantage of Riemann’s sums theory to analyze the formulations of the problem and thus extract useful information in order to design a solution algorithm that minimizes passenger AWT. More specifically, we propose an adaptive large neighborhood search (ALNS) metaheuristic capable of solving larger and more realistic instances of the problem. We compare the solutions obtained by our ALNS algorithm with those of a truncated branch-and-cut algorithm (Barrena et al., 2014), both in terms of passenger AWT and of computing time.

The remainder of this paper is organized as follows. In Section 2 we formally describe the problem and present two mathematical formulations for it. In Section 3 we exploit the mathematical properties of the second model to obtain useful information for the design of the solution algorithm. We then describe our ALNS metaheuristic in Section 4. We present the results of extensive computational experiments in Section 5, and conclusions in Section 6.

2. Problem description and mathematical models

We focus on constructing train timetables adapted to a dynamic demand pattern using the same criteria as in Barrena et al. (2014) who represented them in the form of time–space diagrams, as shown in Fig. 1. The horizontal axis represents the planning horizon, and the vertical axis the distance from each station to the first one. Fig. 1(a) illustrates a regular timetable, i.e., the headway between consecutive trains is constant, whereas Fig. 1(b) depicts a non-regular timetable.

Let \( S = \{1, \ldots, n\} \) be the ordered set of stations defining a two-track railway line. The planning horizon is discretized into time intervals of length \( \delta \). Thus, the time instant \( t \in T = \{0, 1, \ldots, p\} \) corresponds to \( \delta \) time units elapsed since the beginning of the planning horizon. The discretization constant \( \delta \) represents the length of the smallest time interval considered in the problem and therefore, from now on we will consider it as the time unit which can be as small as desired. Let \( d_{ij} \) be the passenger demand between stations \( i, j \in S \), \( i < j \) during the interval \( [t-1, t] \). Let \( l_i \) be the length of the segment between stations \( i \) and \( j \), \( h_{\min} \) the minimum headway, i.e., the minimum amount of time required between the departure of two consecutive trains at each station, \( w_{\min} \) and \( w_{\max} \) the minimum and maximum allowed dwell time at stations, and \( s_{\min} \) and \( s_{\max} \) the inverse of the maximum and minimum traveling speed of a train. The inverse of the speed is used to avoid non-linearities in the constraints of the problem. The operator has a set \( M = \{1, \ldots, m\} \) of trains available.

The aim of the problem is to determine train departure times at stations and train speeds on rail segments in order to minimize the passenger AWT at the stations. In what follows, we present the demand pattern and two non-linear formulations for this problem.

2.1. Demand function and passenger average waiting time

In order to determine an optimal timetable under a dynamic demand behavior, we consider the cumulative demand function for each pair of stations. Several authors deal with variable demand, describing it through functions defined from different perspectives. Hurdle (1973) starts from a stepwise cumulative demand function and assumes that the passengers’ arrival pattern can be reasonably approximated by a differentiable function. The author does not require the algebraic expression of this function, but only that it can be drawn. Unlike in Hurdle (1973), our original data are not given in a regular form, the steps height does not correspond to a passenger, but data given at arbitrary points in time define the steps heights, each corresponding to a different number of passengers. We also approximate the cumulative demand by a smooth function which is a sum of sigmoidal functions (Canca et al., 2014).

Daganzo (1997) presents a cumulative plot approach as an efficient way to represent input–output data, but recognizes that in practical applications, data are not always available and one usually has to construct the plot from limited information. This is what happens in our case, where data are irregularly given and we use a sum of sigmoidal functions to approximate the cumulative demand for each origin–destination pair (Canca et al., 2014). As Daganzo (1997) states, Newell (1971) was the author who demonstrated the full potential of cumulative plots as an analysis and thinking tool in connection with his queueing theory work. Starting from a given cumulative demand function, Newell (1971) proposes a formula for the waiting time of passengers, which takes into consideration the time elapsed between the passenger arrival time and the

![Fig. 1. Time–space diagrams of train timetables for a one line corridor. Source: Barrena et al. (2014).](image-url)
departure of the next train after the arrival time of each passenger. This total waiting time is given by the area between the demand curve and a step function.

In order to obtain an easy-to-manage objective function in the optimization problem, we consider an approximation of the area under the cumulative demand curve by calculating the area of the triangles formed by the cumulative demand values between the departure times of two consecutive trains. This is a reasonable approximation, taking into account the fact that the cumulative demand is a monotone increasing function and it is not an exact description of the demand since it is the result of an approximation from demand data given at random points in time. This simplification counts all passengers arriving between the departure of two trains and considers that each passenger arriving in this interval waits on average half of the interval time.

2.2. Formulation 1

The first formulation uses binary variables $y_k$, $k \in \mathcal{M}$, equal to one if and only if train $k$ is used in the solution. Let the launch time of train $k$ at station $i$ be represented by an integer variable $x^k_t$. Let $D_t$ be the cumulative function of the outgoing demand at station $i$, i.e., $D_t = \sum_{i=0}^{t} \sum_{j=1}^{d} d_{ij}$. Note that we combine the demand over different destinations since we measure the passenger welfare in terms of the AWT. We consider a dynamic demand pattern where the original demand functions $d_{ij}$ have certain peaks at rush hours, and the cumulative demand functions $D_t$ can therefore be considered as a sum of sigmoidal functions (Canca et al., 2014), each sigmoidal function representing the outgoing demand from station $i$ to each of the others stations $j \in S$, $j > i$. These variables give rise to a non-linear objective function expressed in terms of the launch times of each pair of consecutive trains. In order to write the following model, we consider two dummy trains, one at the beginning and another at the end of the planning horizon. The problem can then be formulated as follows:

\[
(F1) \quad \text{minimize} \quad \frac{1}{2} \sum_{k \in S} \sum_{i \in \mathcal{M}} (D_i(x^k_i) - D_i(x^k_{i-1}))(x^k_i - x^k_{i-1})
\]

subject to

\[
x^k_0 = 0 \quad i \in S
\]
\[
x^k_{m+1} = p \quad i \in S
\]
\[
p(1 - y_k) \leq x^k_i \leq p \quad i \in S, \quad k \in \mathcal{M}
\]
\[
x^k_i \leq x^k_{i+1} \quad i \in S, \quad k \in \mathcal{M} \setminus \{m\}
\]
\[
x^k_{i+1} = x^k_i + s^k_i l_{i+1} + w^k_{i+1} \quad i \in \mathcal{S} \setminus \{n\}, \quad k \in \mathcal{M}
\]
\[
y_k w_{\text{min}} \leq w^k_i \leq y_k w_{\text{max}} \quad i \in S, \quad k \in \mathcal{M}
\]
\[
y_k s_{\text{min}} \leq s^k_i \leq y_k s_{\text{max}} \quad i \in \mathcal{S} \setminus \{n\}, \quad k \in \mathcal{M}
\]
\[
x^k_{i+1} \geq x^k_i + h_{\text{min}} y_{k+1} \quad i \in S, \quad k \in \mathcal{M} \setminus \{m\}
\]
\[
x^k_i \in \mathbb{N}^+ \quad i \in S, \quad k \in \mathcal{M}
\]
\[
y_k \in \{0, 1\} \quad k \in \mathcal{M}.
\]

In this formulation, the objective function (1) minimizes the total passenger AWT, which must be divided by the total demand $D_t = \sum_{i \in S} D_i(t)$ in order to obtain the AWT per passenger. Since the total demand is a constant, we do not consider it in the objective function and from now on we will refer to the objective function (1) as the passenger AWT. Constraints (2) and (3) define the dummy train launch times whose role is to ensure that the objective function takes all waiting times and demands into account. Constraints (4) link the variables $x^k_i$ and $y_k$. Specifically, if train $k$ is not used, i.e., $y_k = 0$, then it is launched at the end of the planning horizon, i.e., $x^k_i = p$. On the other hand, if train $k$ is used, i.e., $y_k = 1$, then $x^k_i$ can take any value smaller than $p$, i.e., the train must be launched before the end of the planning horizon. Constraints (5) order the trains in terms of launch times, i.e., together with constraints (4), they ensure that trains with lower index values are launched first. This is a necessary condition for this formulation, but it also breaks equivalent solutions (solutions differing only by train indices and having the same objective function). Constraints (6) ensure that if train $k$ is launched from station $i$ at time $x^k_i$, then this train has to be launched from the next station $i+1$ at time $x^k_{i+1}$, plus the time required to travel from $i$ to $i+1$, plus the dwell time at station $i+1$. Constraints (7) set the bounds for dwell times of launched trains and consider zero dwell times for those that are not launched. Constraints (8) are similar to (7) and bound the inverse of the speed $s^k_i$ of train $k$ leaving station $i$, and set it to zero if train $k$ is unlaunched. Note that constraints (6)–(8) ensure that if train $k$ is not launched, then $x^k_i = p$ for all stations $i$. Constraints (9) play a stronger role than (5) and (6) by ensuring that no two trains are launched from the same station at the same time, and by also considering the minimum headway $h_{\text{min}}$. Integrality and binary conditions on the variables are enforced through constraints (10) and (11). Note that we declare $x^k_i$ as integer variables since they are used in the objective function as the argument of the demand function defined over a discrete set.
2.3. Formulation 2

Our second formulation is similar to the first but disregards speed and waiting times variables. The previous formulation contains a number of equivalent solutions since the same launch time at a station \( i + 1 \) can be obtained by the launch time at the previous station \( i \), and different combinations of arrival and dwell times at station \( i + 1 \). This situation can be avoided by not considering the speeds \( s^i_k \) and dwell times \( w^i_k \) of each train \( k \), but constraining the launch times at stations to lie within the interval in which a service must be launched in order to yield a feasible solution. In this way, one would obtain the same launch times at stations as in F1, avoiding equivalent solutions. This is achieved in our second formulation:

\[
(F2) \quad \text{minimize} \quad \frac{1}{2} \sum_{i \in S} \sum_{k \in \mathcal{M}, j = m + 1} (D_i(x^i_k) - D_i(x^i_{k-1}))(x^i_k - x^i_{k-1})
\]

subject to 2, 3, 4 and 5, 9, 10 and 11, and to

\[
\begin{align*}
\xi^i_k \leq x^i_k + y^i_k(s_{\text{max}}^i l_{i,i+1} + w_{\text{max}}) & \quad \text{i} \in S \setminus \{n\}, \quad k \in \mathcal{M} \\
\xi^i_k + y^i_k(s_{\text{min}}^i l_{i,i+1} + w_{\text{min}}) \leq x^i_{k+1} & \quad \text{i} \in S \setminus \{n\}, \quad k \in \mathcal{M}.
\end{align*}
\]

In this formulation, the objective function (12) is the same as (1). Constraints (13) and (14) enforce the bounds on the launch time from station \( i + 1 \) according to the launch time from station \( i \), and the minimum and maximum speeds and waiting times.

3. Analysis of the mathematical properties of the problem

Formulation 2 is easier to solve than Formulation 1 under a continuous demand pattern, as in Canca et al. (2014). However, the cumulative demand functions \( D_i \) are the sum of sigmoidal functions, which means that the objective function (12) is non-linear and non-convex, and therefore remains beyond the reach of exact algorithms for non-trivial instances.

The train timetabling problem studied by Caprara et al. (2002) is NP-hard, which explains why it cannot be solved exactly for large instances. The proof of NP-hardness is based on a polynomial reduction of the NP-hard Max-Independent Set Problem (MISP) (Garey and Johnson, 1979). In the case of Caprara et al. (2002), a MISP instance is reduced to an initial (ideal) timetable obtaining the Max-Independent set. Our case is different because we do not have an ideal timetable to be fitted as much as possible. Suppose an instance where demand peak is concentrated in a short time period. Then, relaxing the safety headway constraints and allowing identical departure and arrival times for different trains will yield a probably infeasible “ideal” timetable minimizing waiting time where an important number of trains will be incompatible. Then, our problem consists of designing a new timetable, as close as possible to the “ideal” one, but taking into consideration all the constraints previously relaxed. Moreover, a new set of constraints will be imposed, forbidding overtaking at stations. The resulting problem is very similar to the problem described in Caprara et al. (2002). The version of the problem we consider can only be solved exactly for instances with three stations and at most 300 time units in the planning horizon (Barrena et al., 2014). Therefore the use of a heuristic becomes necessary for larger and more realistic instances. In this section we analyze the mathematical properties of the problem, which then provides some directions and motivations for the heuristic to be presented in Section 4. Specifically, in this section we apply Riemann’s theory (Riemann, 1867) to gain insights into the problem. To this end, we first introduce some definitions and properties.

**Definition 3.1.** An \( n \)-order partition of an interval \([a, b]\) is a finite set \( \mathcal{P} = \{x_0, x_1, \ldots, x_n\} \) such that \( a = x_0 < x_1 < x_2 < \ldots < x_n = b \). Intervals \([x_i, x_{i+1}], i = 0, 1, \ldots, n - 1\), are called intervals of the partition \( \mathcal{P} \). If all these intervals have the same length, the partition is called regular. The diameter or mesh size of the partition \( \mathcal{P} \) is defined as \( \delta(\mathcal{P}) = \max\{|x_{i+1} - x_i|, i = 0, 1, \ldots, n - 1\} \). Let \( \mathcal{P} \) and \( \mathcal{P}' \) be two partitions of \([a, b]\). Then \( \mathcal{P}' \) is a refinement of \( \mathcal{P} \) if \( \mathcal{P} \subseteq \mathcal{P}' \).

**Lemma 3.1.** Let \( \mathcal{P} \subseteq \mathcal{P}' \) be two partitions of a closed interval \([a, b]\). It follows directly that \( \delta(\mathcal{P}') \leq \delta(\mathcal{P}) \).

**Definition 3.2.** Let \( f(x) \) be a real function defined on a closed support \([a, b] \subseteq \mathbb{R}\), and let \( \mathcal{P} = \{x_0, x_1, \ldots, x_n\} \) be an \( n \)-order partition of \([a, b]\). The Riemann lower sum of \( f \) for the partition \( \mathcal{P} \) is defined as

\[
s(\mathcal{P}, f, [a, b]) = \sum_{k=1}^{n} m_k(x_k - x_{k-1}),
\]

where \( m_k = \inf \{f(x) : x \in [x_{k-1}, x_k]\} \), \( 0 \leq k \leq n - 1 \), and the Riemann upper sum of \( f \) is defined as

\[
S(\mathcal{P}, f, [a, b]) = \sum_{k=1}^{n} M_k(x_k - x_{k-1}),
\]

where \( M_k = \sup \{f(x) : x \in [x_{k-1}, x_k]\}, 0 \leq k \leq n - 1 \).
Geometrically, \( s(\mathcal{P}, f, [a, b]) \) is the sum of the areas of the rectangles based on the intervals \([x_k, x_{k+1}]\) lying below the graph of \( f \), and \( S(\mathcal{P}, f, [a, b]) \) is the sum of the areas of the rectangles lying above the graph of \( f \).

**Theorem 3.1.** Let \( f : [a, b] \to \mathbb{R} \) be a real function bounded in the variable domain, let \( \mathcal{P} \) and \( \mathcal{P}' \) be partitions of \([a, b]\) such that \( \mathcal{P} \subseteq \mathcal{P}' \). Then

\[
s(\mathcal{P}, f, [a, b]) \leq s(\mathcal{P}', f, [a, b])
\]

and

\[
S(\mathcal{P}', f, [a, b]) \leq S(\mathcal{P}, f, [a, b]).
\]

**Proof.** See Apostol (1957) and Riemann (1867). \( \square \)

### 3.1. Analysis of the problem and of the objective function

We analyze the problem based on our second formulation (Section 2.3), and we observe that at each station \( i \in S \), the departure time variables \( x_i \) of the trains \( k \in \mathcal{M} \) induce an \((m + 1)\)-order partition \( \mathcal{P}_i \) of the time horizon \([0, p]\), that is, \( \mathcal{P}_i = \{0 = x_{i0}, x_{i1}, \ldots, x_{im} = p\} \). The problem consists of finding the partition of the planning horizon at each station such that the objective function (12) is minimized, while ensuring feasibility. We now analyze the objective function by applying Riemann’s theory in order to obtain insightful information that will drive the design of a heuristic algorithm. We first rewrite the objective function (12) as

\[
(OF) \quad \frac{1}{2} \sum_{k \in \mathcal{M} \cup \{0\}} \sum_{k \in \mathcal{M} \cup \{0\}} (D_i(x_k^*) - D_i(x_{k-1}^*)) (x_k^* - x_{k-1}^*) = \frac{1}{2} \sum_{k \in \mathcal{M} \cup \{0\}} \left( \sum_{k \in \mathcal{M} \cup \{0\}} D_i(x_k^*) (x_k^* - x_{k-1}^*) - \sum_{k \in \mathcal{M} \cup \{0\}} D_i(x_{k-1}^*) (x_k^* - x_{k-1}^*) \right).
\]

The function \( D_i(t) \) represents the cumulative demand at each station \( i \in S \) and is therefore monotonically increasing. It follows that for each interval \([x_{i-1}^*, x_i^*] \), the Riemann lower and upper sums are equal to the Riemann left and right sums, respectively, i.e., \( m_k = D_i(x_{i-1}^*) \) and \( M_k = D_i(x_i^*) \). The objective function can then be written as

\[
(OF) = \frac{1}{2} \sum_{i \in S} (S(\mathcal{P}_i, D_i, [0, p]) - s(\mathcal{P}_i, D_i, [0, p])).
\]

This means that for all \( i \in S \), the objective function (15) is minimized when the differences \( S(\mathcal{P}_i, D_i, [0, p]) - s(\mathcal{P}_i, D_i, [0, p]) \) are minimized. The problem then consists of finding the best partitions \( \mathcal{P}_i = \{0 = x_{i0}, x_{i1}, \ldots, x_{im} = p\} \) of the planning horizon at each station \( i \in S \) such that the difference between the upper and lower sums of cumulative demand functions \( D_i(t) \) is minimized, while ensuring feasibility. The order \((m + 1)\) of these partitions is given by the number of available train services.

**Fig. 2** provides a graphical representation of the problem encountered for one station. The cumulative demand functions \( D_i(t) \) are the sum of sigmoidal functions (see Canca et al. (2014)), and the areas with maximum slope are an indicator of the

![Fig. 2. Cumulative demand function at one station \( i \in S \). In light grey, the Riemann lower sums; in dark, the double of the objective function at station \( i \), represented by \( S(\mathcal{P}_i, D_i, [0, p]) - s(\mathcal{P}_i, D_i, [0, p]) \).](image-url)
peak hour demand. The light rectangles represent the Riemann lower sums. Above these, we show in dark the difference between the Riemann upper and lower sums. These dark areas are a visual representation of twice our objective function at each station.

### 3.2. Effect of train insertions and removals

In this section we prove the effect of train insertions and removals in the passenger AWT.

**Theorem 3.2.** Let $P_i$, $i \in S$ be a partition of the time horizon $[0, p]$ at each station $i$, representing a timetable with a set $\mathcal{M}' \subset \mathcal{M}$ of train services. When a train $k \in \mathcal{M}$ is inserted in the timetable, the passenger AWT decreases. In contrast, the passenger AWT increases when a train $k' \in \mathcal{M}'$ is removed from the timetable.

**Proof.** When a train $k \in \mathcal{M}$ is inserted in the timetable, it departs from every station $i \in S$ and therefore, a point $x_i^k$ is added to every partition $P_i$. So, a refined partition $P'_i$ of $P_i$ is obtained at each station and, according to Theorem 3.1, the following inequalities hold:

$$s(P_i, D_i, [0, p]) \leq s(P'_i, D_i, [0, p]) \text{ and } S(P'_i, D_i, [0, p]) \leq S(P_i, D_i, [0, p]).$$

Since $D_i(t) \geq 0$ for all $t \in T$, the Riemann upper and lower sums are also positive, and we can conclude that

$$S(P'_i, D_i, [0, p]) - S(P_i, D_i, [0, p]) \leq S(P'_i, D_i, [0, p]) - S(P_i, D_i, [0, p]), \text{ for all } i \in S.$$ 

This implies that the AWT (Eq. (15)) also satisfies

$$\frac{1}{2} \sum_{i \in S} (S(P'_i, D_i, [0, p]) - s(P'_i, D_i, [0, p])) \leq \frac{1}{2} \sum_{i \in S} (S(P_i, D_i, [0, p]) - s(P_i, D_i, [0, p])).$$

Analogously, when a train is removed, it yields coarser partitions and therefore the objective function value increases.

In fact, when the number of intervals of the partition tends to infinity, the Riemann upper and lower sums become equal and this value is called the Riemann integral (Apostol, 1957). In the train timetabling problem, this means that when the number of trains tends to infinity, the passenger AWT tends to zero, which is obvious.

### 3.3. Selection of the partition or timetable

According to Theorem 3.2, the objective function value decreases with every train insertion so that an infinite number of trains would make the passenger AWT equal to zero. Since there are limitations on the number of trains, which are given by the minimum headway, and by the available number of trains, a good partition should be selected in order to obtain a small AWT.

In order to derive good partitions, we first show that a regular partition, i.e., a regular timetable in the terms of the problem being studied, is not necessarily optimal, especially when the demand is non-regular.

**Counter-example 3.1.** Consider the following cumulative demand function for a problem with only one departure station

$$D(t) = \begin{cases} 
0 & \text{if } t \in [0, 100], \\
-t + 100 & \text{if } t \in (100, 200], \\
100 & \text{if } t \in (200, 900], 
\end{cases}$$

which comes from a non-regular demand function. According to this demand, no passenger arrives during the first interval $[0, 100]$, one passenger per time unit arrives during the interval $(100, 200]$, and no passenger arrives during the interval $(200, 900]$.

Let $P^1 = \{0, 150, 300, 450, 600, 750, 900\}$ be a regular six-order partition of the time horizon $[0, p]$, and $P^2 = \{0, 100, 125, 150, 175, 200, 900\}$ be a non-regular six-order partition of $[0, p]$. The points in the partition indicate the trains departure time and they are represented by vertical dashed lines in Fig. 3. In this figure, the dark areas represent the differences between Riemann upper and lower bounds for partitions $P^1$ and $P^2$, whose values are

$$S(P^1, D, [0, p]) - s(P^1, D, [0, p]) = 15000,$$

and

$$S(P^2, D, [0, p]) - s(P^2, D, [0, p]) = 2500.$$

This means that the objective function value, i.e., the AWT, obtained with the regular partition is six times larger than the one obtained with the selected non-regular one.
So, a regular timetable does not necessarily minimize the difference between the Riemann upper and lower sums, and consequently the passenger AWT. In order to reduce the value of the objective function (12), we attempt to reduce the largest sums. Since \(\frac{b_i - a_i}{C_0} \geq 0\) and \(\frac{D_i(b_i) - D_i(a_i)}{C_0} \geq 0\), their product can be large due to large individual values of each of them. In what follows, we analyze each case separately.

### 3.3.1. Large values of the partition interval lengths \((x_i^k - x_{i-1}^k)\)

According to Definition 3.1, the maximum of the interval lengths at each station is the diameter of the partition \(P_i\). Following Lemma 3.1, refining the partition reduces the diameter, which in turn decreases the value of the objective function. In order to refine the partition, one needs to add a point to it at some time instant in the interval corresponding to the largest diameter over the stations and obtain a partition \(P_0^i\) so that \(\delta(P_0^i) \leq \delta(P_i)\). We will later exploit this observation in Section 4.2.2 when designing the operators of our local search metaheuristic.

### 3.3.2. Large values of the cumulative demand \(D_i(x_i^k) - D_i(x_{i-1}^k)\)

According to Theorem 3.1, the finer the partition, the smaller the difference between the Riemann upper and lower sums. This means that refining the partition within the partition interval having the highest increment in demand will reduce the difference in the sums and thus decrease the value of the objective function. This observation is exploited in Section 4.2.2.

### 3.4. Insights and directions

After having analyzed the problem in terms of its mathematical properties, we have gained some insights into how the optimization algorithm should work if passenger welfare is to be maximized. According to Theorem 3.2, the algorithm should insert trains in order to reduce the value of the objective function, i.e., the passenger AWT. As demonstrated in Sections 3.3.1 and 3.3.2, insertions in certain areas of the planning horizon have a much stronger effect in reducing the AWT than in some other areas.

### 4. Adaptive large neighborhood search metaheuristic

What is needed to solve our problem is an algorithm that removes trains from the time–space diagram and inserts new ones. One such algorithm is the adaptive large neighborhood search (ALNS) metaheuristic (Ropke and Pisinger, 2006) which is based on destroy and repair operators, and appears to be an ideal framework for our problem. In our context, destroy operators remove trains, while repair operators insert them back. As per Theorem 3.2, every time an insertion operator is selected, the objective function value is reduced, while it increases whenever a train is removed. This means that if only the improving moves were to be accepted by the heuristic, the removals operators would never be used, and the algorithm would be dysfunctional. This can be remedied by using a probabilistic acceptance criterion such as the one used in simulated annealing since it accepts deteriorating solutions with some probability and allows a better exploration of the search space. The ALNS metaheuristic enables us to simultaneously select good partitions \(P_i\) at each station \(i \in S\) for the demand functions \(D_i\), while ensuring feasibility. In our problem, feasibility is defined by the minimum and maximum train speed and dwell time at stations, which restricts the set of the

---

**Fig. 3.** Graphical representation of \(S(P_i, D, [0, 900]) - s(P_i, D, [0, 900]), i = 1, 2.\)
possible departure times from consecutive stations, as well as by the limitation in the headway between consecutive trains.

We have designed a powerful ALNS metaheuristic in order to maximize the passenger welfare, capable of handling a large number of origin–destination pairs and of simultaneously determining departure times, speeds, stopping times, while minimizing the passenger AWT. The destroy and repair operators are randomly selected at each iteration. Basically, train services are removed and inserted in the initial time space diagram by means of destroy and repair operators. A roulette wheel mechanism controls the choice of the operators, with a probability that depends on their past performance. More concretely, to each operator $i$ are associated a score $p_i$ and a weight $o_i$ whose values depend on the past performance of the operator. Then, given $h$ operators, operator $j$ will be selected with probability $o_j / \sum_{i=1}^{h} o_i$. Initially, all weights are set to one and all scores are set to zero. The search is divided into segments of $\varphi$ iterations each, and the weights are computed by taking into account the performance of the operators during the last segment. At each iteration, the score of the selected operator is increased by $\sigma_1$ if the operator identifies a new best solution, by $\sigma_2$ if it identifies a solution better than the incumbent, and by $\sigma_3$ if the solution is not better but is still accepted. After $\varphi$ iterations, the weights are updated by considering the scores obtained in the last segment as follows: let $o_j$ be the number of times operator $i$ has been used in the last segment $j$. The updated weights are then

$$o_i := \begin{cases} o_i & \text{if } o_i = 0 \\ (1 - \eta) o_i + \eta p_i / o_j & \text{if } o_i \neq 0. \end{cases}$$

where $\eta \in [0, 1]$ is called the reaction factor and controls how quickly the weight adjustment reacts to changes in the operator performance. The scores are reset to zero at the end of each segment.

As in other ALNS implementations (Coelho et al., 2012; Ropke and Pisinger, 2006), we use an acceptance criterion based on simulated annealing. Given a solution $s$, a neighbor solution $s'$ is always accepted if $z(s') < z(s)$, and is accepted with probability $e^{-(z(s') - z(s))/\tau}$ otherwise, where $z(s)$ is the solution cost and $\tau > 0$ is the current temperature. The temperature starts at $\tau_{\text{start}}$ and is decreased by a cooling rate factor $\phi$ at each iteration, where $0 < \phi < 1$.

We now describe the main features of our algorithm.

4.1. Initial solution

The algorithm can be initialized from an empty solution or from an arbitrary solution, for example, a regular solution as in Fig. 1(a). In our implementation, we start with an empty solution. Later, we assess the effect of providing an initial solution on the performance of the algorithm.

4.2. List of operators

When designing the operators, we have paid close attention to the mathematical properties of a solution, as explained in Section 3, and to the need to diversify the search. Given a solution, destroy and repair operators will delete and insert train services in the time–space diagram. Restrictions on the headway, dwell times and speeds are considered by each operator in order to maintain the feasibility of the solution. We now list the destroy and repair operators we have developed. In what follows, $p_1$ is the number of train services removed from the solution and $p_2$ is the number of train services inserted in the solution at each iteration. The parameters $p_1$ and $p_2$ are integers randomly drawn from the interval $[1, \bar{m}_r]$ ($r = 1, 2$), where $\bar{m}_1 = m_1, \bar{m}_2 = m - m_1$, and $m_1$ is the number of train services of the incumbent solution $s$. More precisely, $p_1$ follows a semi-triangular distribution with a negative slope, i.e., $p_1 = \lfloor \bar{m}_r - \sqrt{(1 - u)(\bar{m}_r - 1)^2 + 0.5} \rfloor$, where $u$ is a random value in the interval $[0, 1]$.

4.2.1. Destroy operators

1. Randomly remove $p_1$ train services

This operator randomly selects $p_1$ trains and removes them. It is useful for refining the solution since it does not change the solution much when $p_1$ is small, which happens frequently due to the shape of the probability distribution of $p_1$. However, it still yields a major transformation of the solution when $p_1$ is large.

2. Remove $p_1$ train services from the smallest interval

This operator identifies the two consecutive trains with the smallest interval and removes the earlier one, that is, it removes the train $k - 1$ corresponding to the smallest values of the partition interval lengths $(x_k - x_{k-1})$ (see Section 3.3.1). This procedure is repeated $p_1$ times.

3. Remove $p_1$ train services with the smallest demand

This operator removes the train with the smallest passenger demand in one of its tracks, that is, it removes the train $k$ yielding the smallest $(D_k(x_k') - D_k(x_{k-1}'))$ (see Section 3.3.2). It is applied $p_1$ times.
4.2.2. Repair operators

1. Randomly insert \( \rho_2 \) train services

This operator randomly inserts \( \rho_2 \) trains. Each insertion is achieved by randomly selecting a time instant from the planning horizon at the first station and inserting a train starting at this time instant with random speed and dwell times at the next stations, while ensuring feasibility. Feasibility is ensured by respecting the speed and dwell bounds: starting from station \( i \in S \setminus \{ n \} \) and according to constraints (13) and (14), we calculate the feasible interval, i.e., the interval whose lower bound is given by the maximum speed and minimum dwell time and whose upper bound is given by the minimum speed and maximum dwell time. Any time instant within this interval is a feasible departure time from the next station \( i + 1 \).

2. Insert \( \rho_2 \) train services in the largest interval

This operator inserts a train in the largest inter-departure interval, that is, in the interval corresponding to the largest values of the partition interval lengths \( \left( x^i_k - x^i_{k+1} \right) \) (see Section 3.3.1). Inserting trains in this interval helps reduce the AWT and at the same time avoid large intervals of the planning horizon without train departures. This is important to guarantee service quality since, even if there are not many passengers waiting at a large interval, it is important to provide minimum services for them. Insertions are performed at a random time instant at the first station of this interval, and speed and dwell times at the following stations are randomly selected within their bounds while ensuring feasibility. This process is repeated \( \rho_2 \) times.

3. Insert \( \rho_2 \) train services near those having the largest demand

This operator inserts a train just before the train with the most loaded track, that is, before the train \( k \) yielding the largest \( \left( D_i(x^i_k) - D_i(x^i_{k+1}) \right) \) (see Section 3.3.2). Inserting trains in these high-demand areas helps reduce the AWT and at the same time implies that train frequency is increased at demand peak hours, which are normally the rush hours, and therefore it is convenient that passengers do not have to wait long during these periods. The insertion is carried out by assigning the departure of the new train from the first station just \( h_{\text{max}} \) time units before the departure of the most loaded train from first station. The new train runs parallel to the most loaded one, i.e., with the same speed and the same dwell time at the same stations. This process is repeated \( \rho_2 \) times.

4.3. Parameter settings and pseudocode

We now describe the parameter settings we have used in our ALNS implementation. These were set after an early tuning phase. The maximum number of iterations \( i_{\text{max}} \) depends on the starting temperature \( \tau_{\text{start}} \) and on the cooling rate \( \phi \). We have set these parameters as follows:

\[
\tau_{\text{start}} = 60,000
\]

\[
\phi = \left( \frac{0.01}{\tau_{\text{start}}} \right)^{1/i_{\text{max}}}.
\]

This makes the cooling rate a function of the desired number of iterations, adjusting accordingly the probability that the ALNS mechanism will accept worsening solutions. The stopping criterion is satisfied when the temperature reaches 0.01. In our implementation, the maximum number of iterations \( i_{\text{max}} \) was set to 70,000, the segment length \( \phi \) was set to 200 iterations, and the reaction factor \( \eta \) was set to 0.7, thus defining the new weights by 70% of the performance on the last segment and 30% of the last weight value. The scores are updated with \( \sigma_1 = 10, \sigma_2 = 5 \) and \( \sigma_3 = 2 \). Algorithm 1 shows the pseudocode of our ALNS implementation. In our experiments, we consider 20 reheatings, that is, after finishing the algorithm, the temperature is set to \( \tau_{\text{start}} \) and the process is repeated 20 times.

4.4. Possible extensions

The aim of the paper is to efficiently solve a specific RRT problem. However, the proposed ALNS can be adapted to handle other features such as overtaking at stations (a common problem in case of metropolitan railways, at least in central stations) by maintaining a list with the train departure order at each station. Several train types can be also managed by considering different speed intervals, as well as a different stopping pattern for each train type. Note that in our formulation, a train can travel on any segment at any speed and can maintain a different dwell time at each station.

An ALNS performs a feasibility test before applying an insertion operator. In this procedure, overtaking at segments is avoided. For specific segments with parallel tracks, this test can be relaxed, allowing the algorithm to manage multi-track segments. In the case of two-directional one-track segments, the feasibility test can be modified by comparing trains in both directions.

5. Computational experiments

We now provide some implementation specifications, we describe the instances used and we present the results of extensive computational experiments. All computations were performed on a grid of Intel Xeon™ processors running at 2.66 GHz.
with up to 24 GB of RAM installed per node, with the Scientific Linux 6.1 operating system and using a single thread. Our algorithm was coded in C++ and uses a single processor and a maximum of only 1 GB of memory.

**Algorithm 1.** ALNS metaheuristic

1: Initialize: set all weights equal to 1 and all scores equal to 0.
2: $s_{\text{best}} \leftarrow s \leftarrow \text{initial solution}, \tau \leftarrow \tau_{\text{start}}, \text{reheatings} \leftarrow 0.$
3: while ($\tau > 0.01$ and reheatings $< 20$) do
4: $s' \leftarrow s.$
5: Select a destroy and a repair operator using the roulette-wheel mechanism based on the current weights. Apply the operators to $s'$ and update the number of times they are used.
6: if $z(s') < z(s)$ then
7: $s \leftarrow s';$
8: if $z(s) < z(s_{\text{best}})$ then
9: $s_{\text{best}} \leftarrow s;$
10: update the score for the operators used with $\sigma_1$;
11: else
12: update the score for the operators used with $\sigma_2$;
13: end if
14: else
15: if $s'$ is accepted by the simulated annealing criterion then
16: $s \leftarrow s';$
17: update the scores for the operators used with $\sigma_3$.
18: end if
19: end if
20: if the iteration count is a multiple of $\varphi$ then
21: update the weights of all operators and reset their scores.
22: end if
23: $\tau \leftarrow \varphi \tau$;
24: if $\tau \leq 0.01$ then
25: reheatings $\leftarrow$ reheatings + 1.
26: end if
27: end while
28: return $s_{\text{best}}$.

5.1. Set of instances

We have used the set of instances initially proposed by Barrena et al. (2014) in order to compare the results of our metaheuristic with those obtained by their branch-and-cut algorithm. A real-world based instance was obtained for the line C5 of the Madrid Metropolitan Railway (Canca et al., 2014; Barrena et al., 2014), which qualifies as an RRT (Vuchic, 2005, page 595). The Spanish railway operator RENFE provided us surveys on the demand data between six stations of line C5: Móstoles-Soto, Móstoles, Las Retamas, Alarcón, San José, and Cuatro Vientos. We have used these survey data to adjust the 15 cumulative demand functions among these stations, and we have used these functions as inputs for our model. The set of artificial instances was generated according to the following parameters:

- number of stations $n$: 3, 6, 10;
- horizon $p$: 200, 400, 600, 800, 1200 min;
- discretization constant $\delta$: 1, 2, 4 min;
- maximum number of trains $m$: 5, 10;
- maximum inverse speed of the trains $s_{\text{min}}$: 0.0015 min/m (speed = 40 km/h);
- minimum inverse speed of the trains $s_{\text{max}}$: 0.00075 min/m (speed = 80 km/h);
- minimum headway $h_{\text{min}}$: 12 min;
- minimum stopping time at the stations $w_{\text{min}}$: 4 min;
- maximum stopping time at the stations $w_{\text{max}}$: 12 min.

These instances will be referred to as $TT-n-p-\delta-m$, e.g., $TT-3-800-2-10$, corresponding to a train timetabling instance with three stations, a planning horizon of 800 min, a discretization constant of two minutes, and a maximum of 10 trains. The C5 instances will refer to the six stations previously mentioned and their corresponding demand functions, varying the rest of parameters. These will be referred to as $C5-p-\delta-m$. The set of instances as well as their solutions are available on [http://www.leandro-coelho.com](http://www.leandro-coelho.com).
Table 1
Summary of the average waiting times and running time for the two algorithms.

<table>
<thead>
<tr>
<th></th>
<th>Regular</th>
<th>Barrena et al. (2014)</th>
<th>ALNS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AWT</td>
<td>Time (s)</td>
<td>AWT</td>
</tr>
<tr>
<td>3 stations</td>
<td>45.19</td>
<td>6656</td>
<td>10.04</td>
</tr>
<tr>
<td>6 stations</td>
<td>60.06</td>
<td>10,803</td>
<td>21.09</td>
</tr>
<tr>
<td>10 stations</td>
<td>60.56</td>
<td>10,803</td>
<td>27.15</td>
</tr>
<tr>
<td>Madrid C5</td>
<td>52.35</td>
<td>10,802</td>
<td>45.71</td>
</tr>
<tr>
<td>Average</td>
<td>54.54</td>
<td>9766</td>
<td>25.99</td>
</tr>
</tbody>
</table>

Table 2
Summary of computational results of a regular timetable, of Barrena et al. (2014), and of our ALNS on instances with three stations.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Regular</th>
<th>Barrena et al. (2014)</th>
<th>ALNS</th>
<th>ALNS vs. Barrena et al. (2014)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AWT</td>
<td>UB1 AWT</td>
<td>UB2</td>
<td>Time (s) Improvement (%)</td>
</tr>
<tr>
<td>TT-3-200-1-5</td>
<td>19.39</td>
<td>37,368 37,368</td>
<td>37,368</td>
<td>6.27 0.00</td>
</tr>
<tr>
<td>TT-3-200-2-5</td>
<td>18.51</td>
<td>35,210 35,210</td>
<td>35,210</td>
<td>6.90 0.00</td>
</tr>
<tr>
<td>TT-3-200-4-5</td>
<td>15.81</td>
<td>29,428 29,428</td>
<td>29,428</td>
<td>6.93 0.00</td>
</tr>
<tr>
<td>TT-3-300-1-5</td>
<td>44.05</td>
<td>102,418 102,418</td>
<td>102,456</td>
<td>9.08 -0.04</td>
</tr>
<tr>
<td>TT-3-300-2-5</td>
<td>43.50</td>
<td>98,412 98,412</td>
<td>98,412</td>
<td>9.72 0.00</td>
</tr>
<tr>
<td>TT-3-300-4-5</td>
<td>39.75</td>
<td>87,260 87,260</td>
<td>87,260</td>
<td>9.73 0.00</td>
</tr>
<tr>
<td>TT-3-600-1-5</td>
<td>44.09</td>
<td>167,941 167,941</td>
<td>167,175</td>
<td>12.29 0.46</td>
</tr>
<tr>
<td>TT-3-600-2-5</td>
<td>43.33</td>
<td>162,008 162,008</td>
<td>162,008</td>
<td>12.90 0.00</td>
</tr>
<tr>
<td>TT-3-600-4-5</td>
<td>39.44</td>
<td>150,748 150,748</td>
<td>150,748</td>
<td>13.07 0.00</td>
</tr>
<tr>
<td>TT-3-800-1-5</td>
<td>6.48</td>
<td>28,961 28,961</td>
<td>28,961</td>
<td>4.86 0.25</td>
</tr>
<tr>
<td>TT-3-800-2-5</td>
<td>8.17</td>
<td>26,366 26,366</td>
<td>26,366</td>
<td>5.42 0.52</td>
</tr>
<tr>
<td>TT-3-800-4-5</td>
<td>8.17</td>
<td>20,744 20,744</td>
<td>20,744</td>
<td>5.48 0.23</td>
</tr>
<tr>
<td>TT-3-1000-1-5</td>
<td>22.48</td>
<td>59,715 59,715</td>
<td>59,715</td>
<td>5.29 0.00</td>
</tr>
<tr>
<td>TT-3-1000-2-5</td>
<td>19.24</td>
<td>54,734 54,734</td>
<td>54,734</td>
<td>5.94 0.00</td>
</tr>
<tr>
<td>TT-3-1000-4-5</td>
<td>19.24</td>
<td>45,592 45,592</td>
<td>45,592</td>
<td>6.04 0.00</td>
</tr>
<tr>
<td>TT-3-1200-1-5</td>
<td>24.47</td>
<td>99,034 99,034</td>
<td>99,034</td>
<td>7.28 0.00</td>
</tr>
<tr>
<td>TT-3-1200-2-10</td>
<td>21.75</td>
<td>43,803 43,803</td>
<td>43,803</td>
<td>7.13 0.00</td>
</tr>
<tr>
<td>TT-3-1200-4-5</td>
<td>21.75</td>
<td>70,460 70,460</td>
<td>70,460</td>
<td>7.17 0.00</td>
</tr>
<tr>
<td>TT-3-1400-1-5</td>
<td>40.43</td>
<td>167,218 167,218</td>
<td>167,218</td>
<td>8.18 0.00</td>
</tr>
<tr>
<td>TT-3-1400-2-10</td>
<td>43.65</td>
<td>168,932 168,932</td>
<td>168,932</td>
<td>9.81 0.00</td>
</tr>
<tr>
<td>TT-3-1400-4-10</td>
<td>43.66</td>
<td>127,528 127,528</td>
<td>127,528</td>
<td>9.47 0.00</td>
</tr>
<tr>
<td>TT-3-1600-1-5</td>
<td>55.08</td>
<td>205,169 205,169</td>
<td>205,169</td>
<td>12.03 0.00</td>
</tr>
<tr>
<td>TT-3-1600-2-10</td>
<td>52.76</td>
<td>158,180 158,180</td>
<td>158,180</td>
<td>12.28 0.00</td>
</tr>
<tr>
<td>TT-3-1600-4-10</td>
<td>52.76</td>
<td>130,460 130,460</td>
<td>130,460</td>
<td>9.65 0.00</td>
</tr>
<tr>
<td>Average</td>
<td>45.19</td>
<td>137,808 137,808</td>
<td>137,808</td>
<td>10.54 3.91</td>
</tr>
</tbody>
</table>

Fig. 4. Cost of the solution accepted at the end of each iteration for instance TT-3-200-1-5.
5.2. Computational results

We first analyze the performance of Algorithm 1 with respect to its parameters, after having performed various preliminary tests to adjust their settings. The following results are typical of those obtained across all instances. For presentation reasons we have chosen to illustrate them on instance TT-3-200-1-5 because this is a small instance for which the optimal solution is known (Barrena et al., 2014). In Fig. 4 we depict the cost of the incumbent solution at the end of each iteration. This figure illustrates how the algorithm converges and also how it can escape from local optima by accepting worsening

![Fig. 5. Percentage AWT improvement on instances with six stations with respect to a regular timetable.](image)

<table>
<thead>
<tr>
<th>Table 3</th>
<th>Summary of computational results of a regular timetable, of Barrena et al. (2014), and of our ALNS on instances with six stations.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance</td>
<td>Regular Barrena et al. (2014) ALNS ALNS vs. Barrena et al. (2014)</td>
</tr>
<tr>
<td></td>
<td>AWT UB T LB Time (s) AWT UB T Time (s) AWT Improvement (%)</td>
</tr>
<tr>
<td>TT-6-400-1-5</td>
<td>38.94 697,886 31,357 10,801 17.70 657,026 66 16.67 5.85</td>
</tr>
<tr>
<td>TT-6-400-2-5</td>
<td>39.42 618,322 89,743 10,800 16.66 642,086 34 17.26 –3.84</td>
</tr>
<tr>
<td>TT-6-400-4-5</td>
<td>38.36 575,780 235,328 10,801 16.58 610,176 18 17.45 –5.97</td>
</tr>
<tr>
<td>TT-6-600-1-5</td>
<td>38.76 1,583,040 64,565 10,802 29.12 1,272,434 109 23.41 19.62</td>
</tr>
<tr>
<td>TT-6-600-2-5</td>
<td>39.27 1,446,760 105,349 10,801 27.57 1,252,694 34 24.01 13.41</td>
</tr>
<tr>
<td>TT-6-600-4-5</td>
<td>38.96 1,051,580 260,942 10,801 22.03 1,205,644 29 24.13 23.12</td>
</tr>
<tr>
<td>TT-6-800-1-5</td>
<td>39.42 1,583,040 64,565 10,802 29.12 1,272,434 109 23.41 19.62</td>
</tr>
<tr>
<td>TT-6-800-2-5</td>
<td>39.92 1,446,760 105,349 10,801 27.57 1,252,694 34 24.01 13.41</td>
</tr>
<tr>
<td>TT-6-800-4-5</td>
<td>39.27 1,051,580 260,942 10,801 22.03 1,205,644 29 24.13 23.12</td>
</tr>
<tr>
<td>TT-6-1000-1-5</td>
<td>39.92 1,583,040 64,565 10,802 29.12 1,272,434 109 23.41 19.62</td>
</tr>
<tr>
<td>TT-6-1000-2-5</td>
<td>39.92 1,446,760 105,349 10,801 27.57 1,252,694 34 24.01 13.41</td>
</tr>
<tr>
<td>TT-6-1000-4-5</td>
<td>39.27 1,051,580 260,942 10,801 22.03 1,205,644 29 24.13 23.12</td>
</tr>
<tr>
<td>TT-6-1200-1-5</td>
<td>40.42 1,583,040 64,565 10,802 29.12 1,272,434 109 23.41 19.62</td>
</tr>
<tr>
<td>TT-6-1200-2-5</td>
<td>40.92 1,446,760 105,349 10,801 27.57 1,252,694 34 24.01 13.41</td>
</tr>
<tr>
<td>TT-6-1200-4-5</td>
<td>39.92 1,051,580 260,942 10,801 22.03 1,205,644 29 24.13 23.12</td>
</tr>
</tbody>
</table>

Average 60.06 1,860,857 92,940 10,802 32.09 1,175,733 85.07 21.09 22.12
solutions. In our tests, we have executed up to 70,000 iterations, but in Fig. 4 we only show the results corresponding to the first 30,000 iterations since the algorithm has already converged at this point. It can be observed that, as expected, at the beginning of the running process, the algorithm accepts a relatively large number of worsening solutions while it becomes more conservative towards the end, and finally converges to the optimal solution. After executing 20 reheatings, the running time for the presented instance TT-3-200-1-5 was equal to 20 s. In Fig. 4, we show the results of the first run since the algorithm already yields the optimal solution. The running time is then equal to 1 s.

Fig. 6. Percentage AWT improvement on instances with 10 stations with respect to a regular timetable.

Table 4
Summary of computational results of a regular timetable, of Barrena et al. (2014), and of our ALNS on instances with 10 stations.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Regular</th>
<th>Barrena et al. (2014)</th>
<th>ALNS</th>
<th>ALNS vs. Barrena et al. (2014)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AWT</td>
<td>UB</td>
<td>LB</td>
<td>Time (s)</td>
</tr>
<tr>
<td>TT-10-400-1-5</td>
<td>43.59</td>
<td>3,033,290</td>
<td>237,065</td>
<td>10,801</td>
</tr>
<tr>
<td>TT-10-400-2-5</td>
<td>44.16</td>
<td>2,833,700</td>
<td>615,530</td>
<td>10,800</td>
</tr>
<tr>
<td>TT-10-400-4-5</td>
<td>50.37</td>
<td>6,895,040</td>
<td>994,635</td>
<td>10,801</td>
</tr>
<tr>
<td>TT-10-600-1-5</td>
<td>50.72</td>
<td>4,721,520</td>
<td>1,299,070</td>
<td>10,801</td>
</tr>
<tr>
<td>TT-10-800-1-5</td>
<td>78.24</td>
<td>16,622,200</td>
<td>178,477</td>
<td>10,804</td>
</tr>
<tr>
<td>TT-10-800-2-5</td>
<td>78.75</td>
<td>12,715,800</td>
<td>194,557</td>
<td>10,801</td>
</tr>
<tr>
<td>TT-10-800-4-5</td>
<td>80.59</td>
<td>9,615,050</td>
<td>214,828</td>
<td>10,800</td>
</tr>
<tr>
<td>TT-10-1000-2-5</td>
<td>33.84</td>
<td>3,292,210</td>
<td>364,183</td>
<td>10,800</td>
</tr>
<tr>
<td>TT-10-1000-4-5</td>
<td>33.75</td>
<td>1,285,320</td>
<td>534,844</td>
<td>10,800</td>
</tr>
<tr>
<td>TT-10-1200-1-5</td>
<td>28.38</td>
<td>4,816,760</td>
<td>614,681</td>
<td>10,805</td>
</tr>
<tr>
<td>TT-10-1200-2-5</td>
<td>28.86</td>
<td>4,314,360</td>
<td>750,225</td>
<td>10,801</td>
</tr>
<tr>
<td>TT-10-1200-4-5</td>
<td>36.79</td>
<td>7,689,280</td>
<td>110,549</td>
<td>10,802</td>
</tr>
<tr>
<td>TT-10-1200-2-10</td>
<td>33.84</td>
<td>3,292,210</td>
<td>364,183</td>
<td>10,800</td>
</tr>
<tr>
<td>TT-10-1200-4-10</td>
<td>33.75</td>
<td>1,285,320</td>
<td>534,844</td>
<td>10,800</td>
</tr>
<tr>
<td>TT-10-1200-1-10</td>
<td>28.38</td>
<td>4,816,760</td>
<td>614,681</td>
<td>10,805</td>
</tr>
<tr>
<td>TT-10-1200-2-10</td>
<td>28.86</td>
<td>4,314,360</td>
<td>750,225</td>
<td>10,801</td>
</tr>
<tr>
<td>TT-10-1200-4-10</td>
<td>36.79</td>
<td>7,689,280</td>
<td>110,549</td>
<td>10,802</td>
</tr>
<tr>
<td>TT-10-1200-2-10</td>
<td>33.84</td>
<td>3,292,210</td>
<td>364,183</td>
<td>10,800</td>
</tr>
<tr>
<td>TT-10-1200-4-10</td>
<td>33.75</td>
<td>1,285,320</td>
<td>534,844</td>
<td>10,800</td>
</tr>
</tbody>
</table>
In order to compare the two methods, we provide in the last column the running time needed by both algorithms, as well as the lower bound obtained in Barrena et al. (2014). We also show the upper bound (UB) on the objective function, which represents the improvement of the upper bound UB₁ obtained with the ALNS over the upper bound UB₂ obtained by Barrena et al. (2014). It is computed as Improvement (%) = 100(UB₁ - UB₂)/UB₂. Out of 106 open instances without a known optimal solution, we obtain 84 new best known solutions.

The results of Tables 2–5 were obtained with 20 reheatings, that is, after finishing the algorithm, the temperature τ is reset to its initial value and the process is repeated 20 times. We have also stored the solutions after each reheating and we have observed that the same solutions are often obtained after the first reheating. So, we could have avoided the 20 reheatings and reduced the computational time by a factor of 20 without much impact on the quality of the results.
In Table 2, if we disregard the cases where the optimum is reached (marked with *), the resulting average improvement then becomes 7.33%. If we only consider the instances with known optimal solutions, our ALNS metaheuristic is able to find the optimum in more than 70% of the cases within only a fraction of the computational time used by the exact branch-and-cut algorithm. When ALNS does not reach the optimum, the worst optimality gap is only 2.83%. The average results of Tables 2–4 show that, as expected, ALNS is able to obtain better improvements on the larger instances than on the smaller ones. Moreover, we see from Tables 2–5 that ALNS tends to provide better improvements on those instances having a larger planning horizon, a larger number of trains, and smaller time units, i.e., on the most difficult instances.

In order to derive a better insight into our results, we depict in Figs. 5 and 6 the AWT improvement obtained with ALNS and with the algorithm of Barrena et al. (2014) over a regular timetable for the cases of six and 10 stations. Once again, it can be observed that the improvement of ALNS over Barrena et al. (2014) is remarkable, especially on the larger instances. These figures also clearly show that there is nearly no difference between the ALNS results obtained with and without an initial solution, except for the smaller instances, where an initial solution provided by a regular timetable enables the ALNS metaheuristic to reach marginally better final solutions.

6. Conclusions

We have proposed two non-linear programming formulations for a single-line timetabling problem arising in an RRT system under dynamic demand, with the objective of minimizing the AWT per passenger. Through a suitable analysis of the second formulation, we have derived principles that should guide a heuristic in removing trains from a timetable and in introducing new ones. We have concluded that the choice of an ALNS metaheuristic which effectively combines these two operators together with a simulated annealing acceptance criterion should be highly appropriate for our problem. Our ALNS algorithm maximizes passenger welfare, while respecting all constraints and limitations of the problem. To the best of our knowledge, this is the first time that this type of timetabling problem has been efficiently solved under a dynamic demand behavior. In comparison, the exact algorithm developed in Barrena et al. (2014) can only be used to solve very small instances, and yields very large optimality gaps for medium and large instances. By performing extensive computational experiments on real-world based and randomly generated instances, we have compared our ALNS implementation against a state-of-art truncated branch-and-cut algorithm. We have succeeded in reducing the passenger AWT by 26% by using less than 1% of the computation time required by the latter algorithm. Out of the 120 open instances, we were able to improve the best known solution in 84 cases and to reach the optimum for 10 out of 14 instances with known optimal solutions. These clearly confirm the efficiency and superiority of our ALNS metaheuristic.

Acknowledgements

This work was partly supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) under grant 39682-10, and by the Excellence Program of the Andalusian Government under grant P09-TEP-5022. This support is gratefully acknowledged. We also thank the Calcul Québec for providing parallel computing facilities. Thanks are due to the Associate Editor and to the referees for their valuable comments.

References


Cordone, R., Redaeli, F., 2011. Optimizing the demand captured by a railway system with a regular timetable. Transportation Research Part B 45 (2), 430–446.


