CIRRELT

Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

A Fast and Efficient Adaptive Large Neighborhood Search Heuristic for the Passenger Train Timetabling Problem with Dynamic Demand

Eva Barrena
David Canca Ortiz
Leandro C. Coelho
Gilbert Laporte

October 2013

CIRRELT-2013-64

Bureaux de Montréal :
Université de Montréal
Pavillon André-Aisenstadt
C.P. 6128, succursale Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
Pavillon Palasis-Prince
2325, de la Terrasse, bureau 2642
Québec (Québec)
Canada G1V 0A6
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

A Fast and Efficient Adaptive Large Neighborhood Search Heuristic for the Passenger Train Timetabling Problem with Dynamic Demand

Eva Barrena[1,2], David Canca Ortiz[3], Leandro C. Coelho[1,4,*], Gilbert Laporte[1,2]

[1] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT)

[2] Department of Management Sciences, HEC Montréal, 3000 Côte-Sainte-Catherine, Montréal, Canada H3T 2A7

[3] School of Engineering, University of Seville, Avenida de los Descubrimientos s/n, 41092,Seville, Spain

[4] Department of Operations and Decision Systems, Université Laval, 2325, de la Terrasse, Québec, Canada G1V 0A6

Abstract. The railway planning process is a complex activity which is usually decomposed into a succession of stages, traditionally network design, line design, timetabling, rolling stock, and personnel planning. In this paper, we study the design and optimization of train timetables adapted to a dynamic demand environment. The objective is to minimize passenger waiting times at the stations. We first describe an integer linear programming formulation which generalizes the non-periodic train timetabling problem under a dynamic demand pattern. We then introduce a fast adaptive large neighborhood search (ALNS) heuristic in order to deal with larger instances and to solve the problem efficiently within short computational times. The algorithm provides timetables that may not be regular or periodic, but adjusted to a dynamic demand behavior. Through extensive computational experiments on artificial and real-world based instances, we demonstrate the computational superiority of ALNS compared with a truncated branch-and-cut algorithm. The average reduction on passenger waiting times is 26%, while the computational time of our ALNS algorithm is less than 1% of that required by the alternative 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.

Keywords. Train timetabling, dynamic demand, regular timetable, exact algorithm, branch-and-cut.

Acknowledgements. This work was partly supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) under grant 39682-10. This support is gratefully acknowledged. We also thank Calcul Québec for providing parallel computing facilities.

---

* Corresponding author: Leandro.Coelho@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2013

1 Introduction

The railway planning process is a complex activity which is usually decomposed into a succession of stages, traditionally network design, line design, timetabling, rolling stock, and personnel planning [19, 23]. The train timetabling problem consists of determining departure and arrival times of each train service to and from each station along a railway line or a network.

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 double direction rapid transit line with two tracks, in which case departure and arrival times can be designed independently for each direction. We develop an efficient heuristic to 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 contribution.

When designing passenger railway timetables it is essential to respect the track capacities, and it is common to optimize an objective function relevant to the operator, such as minimizing the deviations from an ideal timetable, or minimizing the services' running times [6]. This can be achieved in two ways. The first is to construct a periodic schedule, i.e., the schedule that is constantly repeated, for example with departures at 03, 21 and 46 minutes every hour. Periodic timetabling models follows the Periodic Event Scheduling Problem (PESP) formulation of Serafini and Ukovich [35]. Periodic timetables have the advantage of being easily memorized by passengers and have proved their ability to deal with large-scale railway networks [26]. 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 [15] 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. [13] use a time discretization and a representation of the problem on a time-space graph. They 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 [37] have studied the problem of improving the 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.

Heuristics have been extensively employed to solve variants of this problem, including greedy heuristics with different objectives [10, 39], mathematical programming-based methods to minimize interchange waiting times [38], graph theory-based methods [29], cut-based algorithms [28], and sequential decomposition [11]. Local search algorithms include a hybrid of tabu search and branch-and-bound [21], a combination of insertion, backtracking and dynamic route selection [5], and a hybridization of simulated annealing with particle swarm optimization [25]. Population-based algorithms include ant colony optimization [16, 32] and genetic algorithms [17, 30, 36]. Constraint satisfaction techniques have been used by Odijk [31], Abril et al. [1], and Ingolotti et al. [24], whereas methods based on linear programming relaxations and duality-based methods have been frequently applied [3, 7, 8, 9, 14, 20, 27, 40]. Other recent heuristics are those of Burdett and Kozan [4] and of Furini and Kidd [22], 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. 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 considered to be constant and is represented by a set of

meaningful origin-destination matrices corresponding to different hours of a typical day.

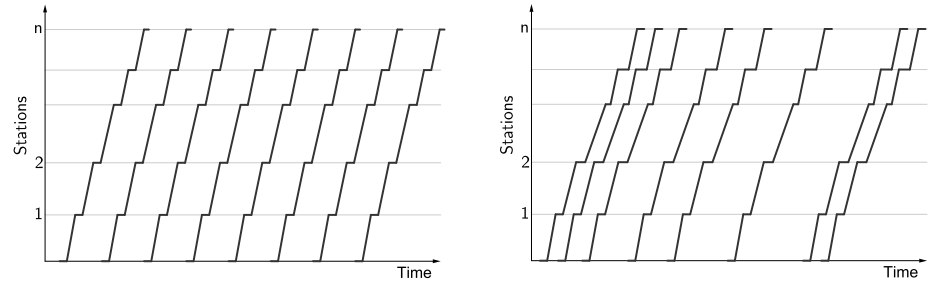
In this paper, we consider a generalization of the non-periodic train timetabling problem under a dynamic demand pattern [12]. We first recall the linear integer formulation introduced by Barrena et al. [2]. This model does not assume any shape for the demand function, but can deal with non-monotonic and even non-convex demand functions. The objective is to minimize passenger average waiting time at stations, and the solutions are train timetables adjusted to a dynamic demand pattern which may neither be regular nor periodic. Barrena et al. [2] have proposed an exact branch-and-cut algorithm applicable to this model, but it can only solve relatively small instances to optimality. As a result, here we propose an adaptive large neighborhood search (ALNS) algorithm capable of solving larger and more realistic instances of the problem. We compare the solutions obtained by our ALNS algorithm with those obtained with the truncated branch-and-cut algorithm of [2], both in terms of average passenger waiting time and of computing time.

This paper is organized as follows. In Section 2 we formally describe the problem and present its mathematical formulation. We then describe our ALNS heuristic in Section 3. We present the results of extensive computational experiments in Section 4, and conclusions in Section 5.

2 Problem description and mathematical model

We focus on constructing timetables adapted to a dynamic demand pattern using the same criteria as Barrena et al. [2] who presented train timetables in the form of time-space diagrams, as shown in Figure 1. The horizontal axis represents the planning horizon, and the vertical axis the distance from each station to the first one. Figure 1(a) illustrates a regular timetable, i.e., the headway between consecutive trains is constant, whereas Figure 1(b) depicts a non-regular timetable.

Let $\mathcal{S} = \{1, \ldots, n\}$ be the ordered set of stations defining a two-track railway line. The



(a) Regular timetable (b) Non-regular timetable

Figure 1: Time-space diagrams of train timetables for a one line corridor. Source: Barrena et al. [2]

planning horizon is discretized into time intervals of length δ. Thus, the time instant $t \in \mathcal{T} = \{0, 1, \ldots, p\}$ corresponds to δt time units elapsed since the beginning of the planning horizon. The discretization constant δ 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}^t be the passenger demand between stations $i, j \in \mathcal{S}, i < j$ during the interval $[t-1, t]$. Let l_{ij} 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 aim of the problem is to determine train departure times at stations and train speeds on rail segments in order to minimize the average waiting time of passengers at the stations.

In what follows, we describe one of the formulations proposed by Barrena et al. [2]. This formulation is the one that yielded the best results out of the three formulations presented in that paper. The model makes use of a set $\mathcal{M} = \{1, \ldots, m\}$ of possible trains. It works with binary variables x_{ki}^t equal to one if and only if train k leaves station i at time t, integer variables u_i^t representing the number of passengers boarding the train leaving station i at

time t, and variables f_i^t representing the number of passengers waiting at station i at the end of interval $[t-1,t]$. We assume that $f_i^0 = 0$ since no initial demand is considered. These variables give rise to a linear representation of the objective function with respect to a dynamic demand function. In order to compute the average waiting time (AWT) per passenger, we assume that a passenger arriving within the interval $[t,t+1]$ waits half of this time interval, i.e., $\delta/2$, plus the full δ for each of the remaining time intervals until boarding the next train. The total waiting time is then $\frac{\delta}{2}\sum_{t\in\mathcal{T}}\sum_{i,j\in\mathcal{S},j>i} d_{ij}^t + \delta\sum_{i\in\mathcal{S}\setminus\{n\}}\sum_{t\in\mathcal{T}\setminus\{p\}} f_i^t$. Since the first term is a constant, it is not considered in the objective function. The problem can be formulated as follows:

$$\text{minimize} \quad \delta\sum_{i\in\mathcal{S}\setminus\{n\}}\sum_{t\in\mathcal{T}\setminus\{p\}} f_i^t \tag{1}$$

subject to

$$f_i^t = f_i^{t-1} + \sum_{j\in\mathcal{S},j>i} d_{ij}^t - u_i^t \quad i\in\mathcal{S}\setminus\{n\}, \quad t\in\mathcal{T} \tag{2}$$

$$u_i^t \le \sum_{k\in\mathcal{M}} x_{ki}^t \sum_{t'=0}^{t}\sum_{j\in\mathcal{S},j>i} d_{ij}^{t'} \quad i\in\mathcal{S}\setminus\{n\}, \quad t\in\mathcal{T} \tag{3}$$

$$x_{ki}^t \le \sum_{t'=0}^{t} x_{k-1,i}^{t'} \quad i\in\mathcal{S}, \quad k\in\mathcal{M}\setminus\{1\}, \quad t\in\mathcal{T} \tag{4}$$

$$\sum_{t\in\mathcal{T}}\sum_{k\in\mathcal{M}} x_{k1}^t \le m \tag{5}$$

$$\sum_{t\in\mathcal{T}} x_{ki}^t \le 1 \quad i\in\mathcal{S}, \quad k\in\mathcal{M} \tag{6}$$

$$\sum_{t\in\mathcal{T}} t x_{k+1,i}^t \ge \sum_{t\in\mathcal{T}} t x_{ki}^t + h_{min}\sum_{t\in\mathcal{T}} x_{ki}^t \quad i\in\mathcal{S}, \quad k\in\mathcal{M}\setminus\{m\} \tag{7}$$

$$\sum_{t\in\mathcal{T}} t x_{k,i+1}^t \le \sum_{t\in\mathcal{T}} t x_{ki}^t + s_{max} l_{i,i+1} + w_{max} \quad i\in\mathcal{S}\setminus\{n\}, \quad k\in\mathcal{M} \tag{8}$$

$$\sum_{t\in\mathcal{T}} t x_{ki}^t + s_{min} l_{i,i+1} + w_{min} \le \sum_{t\in\mathcal{T}} t x_{k,i+1}^t \quad i\in\mathcal{S}\setminus\{n\}, \quad k\in\mathcal{M} \tag{9}$$

$$x_{ki}^t \in \{0,1\} \quad i\in\mathcal{S}, \quad k\in\mathcal{M}, \quad t\in\mathcal{T} \tag{10}$$

$$f_i^t, u_i^t \geq 0 \quad i \in \mathcal{S}, \quad t \in \mathcal{T}. \tag{11}$$

The objective function (1) minimizes the total waiting time of passengers. Constraints (2) define the flow of passengers waiting at station i at time t. It is equal to the sum of passengers who were already at the station and arrived in previous periods, plus all passengers who arrived at station i at time t, minus the number of passengers who boarded the train leaving station i at time t. Constraints (3) define upper bounds on the number of passengers boarding a train. In particular, if no train is launched at time t, the flow of passengers boarding the train is zero. If a train is launched, the right-hand side of constraints (3) constitutes an upper bound on the number of passengers boarding the train. From constraints (3) and (11), u_i^t is zero if no train is assigned to leave station i at time t. Constraints (4) order the trains by their indices and are imposed to break the symmetry of this formulation. Constraints (5) limit the number of available trains. Constraints (6) ensure that a train is launched at most once. Constraints (7) establish the relationship between departure times of two consecutive trains at station i, ensuring that the minimum headway h_{min} is respected. Constraints (8) and (9) define the bounds on the departure time from station $i+1$ according to the departure time from station i, and the minimum and maximum speeds and stopping times. Finally, constraints (10) and (11) enforce integrality and non-negativity conditions on the variables. For a detailed analysis of this formulation, the interested reader is referred to Barrena et al. [2].

3 Adaptive large neighborhood search heuristic

The train timetabling problem is NP-hard [13, 19], which explains why it can be solved exactly only for small instances. In Barrena et al. [2], the formulations proposed could only be solved exactly for instances with three stations and at most 300 time units in the planning horizon. Therefore a heuristic becomes necessary for larger and realistic instances. We have designed a powerful ALNS metaheuristic capable of handling a large number of origin-destination pairs and of simultaneously determining departure times,

speeds, stopping times, while minimizing the average passenger waiting times. The ALNS metaheuristic [33] is based on destroy and repair operators randomly selected at each iteration. It has already been adapted to several transportation problems including vehicle routing [33], arc routing [34], and inventory-routing [18]. This heuristic presents itself as a natural framework for the problem at hand, which combines several types of decision variables and constraints. Our ALNS implementation follows the general scheme proposed by Ropke and Pisinger [33]. 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 π_i and a weight ω_i whose values depend on the past performance of the operator. Then, given h operators, operator j will be selected with probability $\omega_j / \sum_{i=1}^{h} \omega_i$. Initially, all weights are set to one and all scores are set to zero. The search is divided into segments of φ 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 σ_1 if the operator finds a new best solution, by σ_2 if it finds a solution better than the incumbent, and by σ_3 if the solution is not better but is still accepted. After φ iterations, the weights are updated by considering the scores obtained in the last segment as follows: let o_{ij} be the number of times operator i has been used in the last segment j. The updated weights are then:

$$\omega_i := \begin{cases} \omega_i & \text{if } o_{ij} = 0 \\ (1-\eta)\omega_i + \eta \pi_i / o_{ij} & \text{if } o_{ij} \neq 0, \end{cases} \tag{12}$$

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 implementations [18, 33], 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 τ_{start} and is decreased by a cooling rate factor ϕ at each iteration, where $0 < \phi < 1$.

We now describe the main features of our algorithm.

3.1 Initial solution

The algorithm can be initialized from an empty solution or from an arbitrary solution, for example, a regular solution as in Figure 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.

3.2 List of operators

Given a solution, destroy and repair operators will delete and insert train services in the time-space diagram. Restrictions about 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, ρ_1 is the number of train services removed from the solution and ρ_2 is the number of trains services inserted in the solution at each iteration. The parameters ρ_1 and ρ_2 are integers randomly drawn from the interval $[1, \overline{m}_r]$ $(r = 1, 2)$, where $\overline{m}_1 = m_s$, $\overline{m}_2 = m - m_s$, and m_s is the number of train services of the incumbent solution s. More precisely, ρ_r follows a semi-triangular distribution with a negative slope, i.e., $\rho_r = \lfloor \overline{m}_r - \sqrt{(1-u)(\overline{m}_r - 1)^2} + 0.5 \rfloor$, where u is a random value in the interval $[0, 1]$.

3.2.1 Destroy operators

1. Randomly remove ρ_1 train services

This operator randomly selects ρ_1 trains and removes them. It is useful for refining the

solution since it does not change the solution much when ρ_1 is small, which happens frequently due to the shape of the probability distribution of ρ. However, it still yields a major transformation of the solution when ρ_1 is large.

2. Remove ρ_1 train services from the smallest interval

This operator identifies the two consecutive trains with the smallest interval, and removes the earlier one. This procedure is repeated ρ_1 times.

3. Remove ρ_1 train services with the smallest demand

This operator removes the train with the smallest passenger demand in one of its tracks, and this is repeated ρ_1 times.

3.2.2 Repair operators

1. Randomly insert ρ_2 train services

This operator randomly inserts ρ_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.

2. Insert ρ_2 train services in the biggest interval

This operator inserts a train in the largest inter-departure interval. 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 selected randomly within their bounds while ensuring feasibility. This process is repeated ρ_2 times.

3. Insert ρ_2 train services near those having the largest demand

This operator inserts a train just before the train with the most loaded track. The insertion is carried out by assigning the departure of the new train from the first station just h_{min} 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

stop time at the same stations. This process is repeated ρ_2 times.

3.3 Parameter settings and pseudocode

In this section we describe the parameter settings we have used. These were set after an early tunning phase. The maximum number of iterations i_{max} is dependent on two parameters, namely the starting temperature τ_{start}, and the cooling rate ϕ. We have decided to set them as follows:

$$\tau_{start} = 60000 \tag{13}$$

$$\phi = (0.01/\tau_{start})^{1/i_{max}} \tag{14}$$

These makes the cooling rate a function of the desired number of iterations, adjusting accordingly the probability that the ALNS mechanism will accept worse solutions. The stopping criterion is reached when the temperature reaches 0.01.

In our implementation, the maximum number of iterations i_{max} was set to 70000, the segment length φ was set to 200 iterations, and the reaction factor η 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. 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 τ_{start} and the process is repeated 20 times.

4 Computational experiments

We now provide some implementation specifications, we describe the instances used to perform the computational experiments, and we present the results of extensive computational experiments. All computations were performed on a grid of Intel Xeon^{TM} processors running at 2.66 GHz with up to 24 GB of RAM installed per node, with the Scientific

Algorithm 1. ALNS heuristic - part 1

Initialize: set all weights equal to 1 and all scores equal to 0.
$s_{best} \leftarrow s \leftarrow initial\ solution$, $\tau \leftarrow \tau_{start}$, $reheatings = 0$.
while ($\tau > 0.01$ and $reheatings < 20$) **do**
    $s' \leftarrow s$.
    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.
    if $z(s') < z(s)$ **then**
        $s \leftarrow s'$;
        if $z(s) < z(s_{best})$ **then**
            $s_{best} \leftarrow s$;
            update the score for the operators used with σ_1;
        else
            update the score for the operators used with σ_2;
        end if
    else
        if s' is accepted by the simulated annealing criterion **then**
            $s \leftarrow s'$;
            update the scores for the operators used with σ_3.
        end if
    end if
    if the iteration count is a multiple of φ **then**
        update the weights of all operators and reset their scores.
    end if

Algorithm 1. ALNS heuristic - part 2 (continued)

```
    τ ← φτ;
    if τ ≤ 0.01 then
        reheatings ← reheatings + 1.
    end if
end while
return s_best;
```

Linux 6.1 operating system. A single thread was used. Our algorithm was coded in C++ and makes use of only one processor and a maximum of only one GB of memory.

4.1 Set of instances

We have used the set of instances initially proposed by Barrena et al. [2] in order to compare the results of our heuristic algorithm 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. 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 minutes;
- discretization constant δ: 1, 2, 4 minutes;
- maximum number of trains m: 5, 10;
- maximum inverse speed of the trains s_{min}: 0.0015 min/m (speed = 40 km/h);
- minimum inverse speed of the trains s_{max}: 0.00075 min/m (speed = 80 km/h);
- minimum headway h_{min}: 12 minutes;
- minimum stopping time at the stations w_{min}: 4 minutes;

- maximum stopping time at the stations w_{max}: 12 minutes.

These instances will be referred to as TT-n-p-δ-m, e.g., TT-3-800-2-10, corresponding to a train timetabling instance with three stations, a planning horizon of 800 minutes, discretization constant of two minutes, and a maximum of 10 trains. The set of instances as well as their solutions are available on `http://www.leandro-coelho.com`.

4.2 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 instance TT-3-200-1-5 because it is a small instance for which the optimal solution is known [2]. In Figure 2 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 worse solutions. In our tests, we have executed up to 70000 iterations, but in Figure 2 we only show the results corresponding to the first 30000 iterations since the algorithm has already converged. It can be observed that, as expected, at the beginning of the running process, the algorithm accepts a relatively large number of worse solutions while it becomes more conservative towards the end, and finally converges to the optimal solution. After considering 20 reheatings, the running time for the presented instance TT-3-200-1-5 was equal to 20 seconds. In Figure 2, we show the results of the first run since the algorithm already yields the optimal solution. The running time is then equal to one second.

Regarding the use of an initial solution, we have run all the tests with and without an initial solution and no differences were observed in the results. These findings are typical of other ALNS implementations for different problems and indicates that the algorithm is quite robust since its performance is not affected by the quality of the initial solution.

We present in Table 1 a summary of the AWT per passenger and running time over all

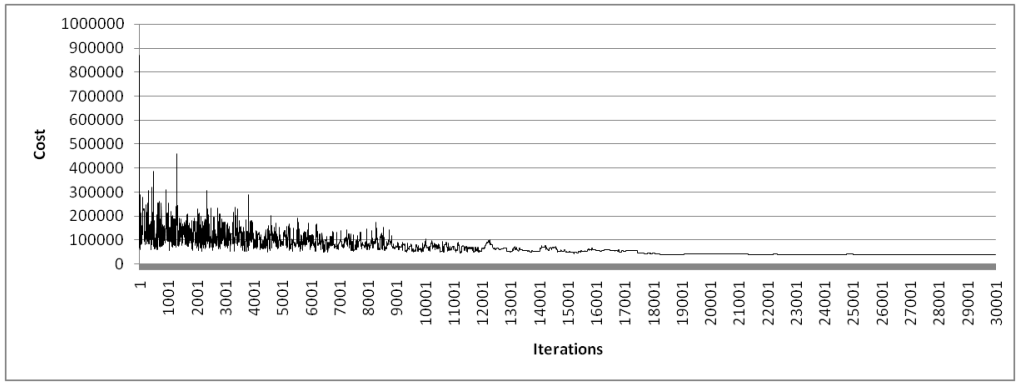


Figure 2: Cost of the solution accepted at the end of each iteration for instance TT-3-200-1-5

instances, along with a comparison with other algorithms. The rows represent the different instance sizes containing three, six and 10 stations, as well as the real-world based case of the line C5 of Madrid Metropolitan Railway. The columns present the AWT when using a regular timetable, the best upper on the AWT obtained by the branch-and-cut algorithm of Barrena et al. [2], which was truncated after three hours of computation, and the solutions of our ALNS algorithm. We note that most instances could not be solved to optimality within the time limit used in [2], and optimal solutions were obtained only for some instances with three stations. It can be observed that the AWT is considerably improved with respect to a regular timetable, and that our ALNS algorithm can significantly improve the previous best known AWT. Our ALNS implementation can solve the problem within much shorter computation times than the truncated branch-and-cut algorithm. On average, the AWT is reduced by 26.58%, and the ALNS heuristic finishes within less than 1% of the running time required by the truncated branch-and-cut algorithm, most of the time converging much earlier.

In order to compare the results of our ALNS algorithm with those obtained by means of branch-and-cut methods [2], we provide in Tables 2, 3, 4 and 5 detailed results for instances with three, six, and 10 stations, and the line C5 of Madrid Metropolitan Railway, respectively. In these tables we show the AWT per passenger for a regular timetable, as

Table 1: Summary of the average waiting times and running time for the two algorithms

	Regular	Barrena et al. [2]		ALNS	
	AWT	AWT	Time (s)	AWT	Time (s)
3 stations	45.19	10.54	6656	**10.04**	**33**
6 stations	60.06	32.09	10803	**21.09**	**85**
10 stations	60.56	49.18	10803	**27.15**	**148**
Madrid C5	52.35	49.78	10802	**45.71**	**73**
Average	54.54	35.40	9766	**25.99**	**84.75**

well as for the timetables obtained in [2] and with our ALNS heuristic. We also show the upper bound (UB) on the objective function, and the running time needed by both algorithms, as well as the lower bound obtained in [2]. In order to compare the two methods, we provide in the last column the *improvement (%)* statistic, which represents the improvement of the upper bound UB_2 obtained with the ALNS over the upper bound UB_1 obtained by [2]. It is computed as *Improvement (%)* $= 100(UB_1 - UB_2)/UB_1$. 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 set to the initial one and the process is repeated for 20 times. We have also stored the solutions after each reheating and 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 heuristic is able to find the optimum in more than 70% of the cases in 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 the instances having larger planning horizons, larger number of trains, and smaller time units, i.e., on the most difficult instances.

In order to derive a graphical insight of the results, we depict in Figures 3 and 4 the improvement of the AWT obtained with ALNS and with the algorithm of [2] over a regular timetable for the cases of six and 10 stations. Once again, it can be observed that the improvement of ALNS over [2] 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 initial solution, except for the smaller instances, where an initial solution provided by a regular timetable enables the ALNS heuristic to reach marginally better final solutions.

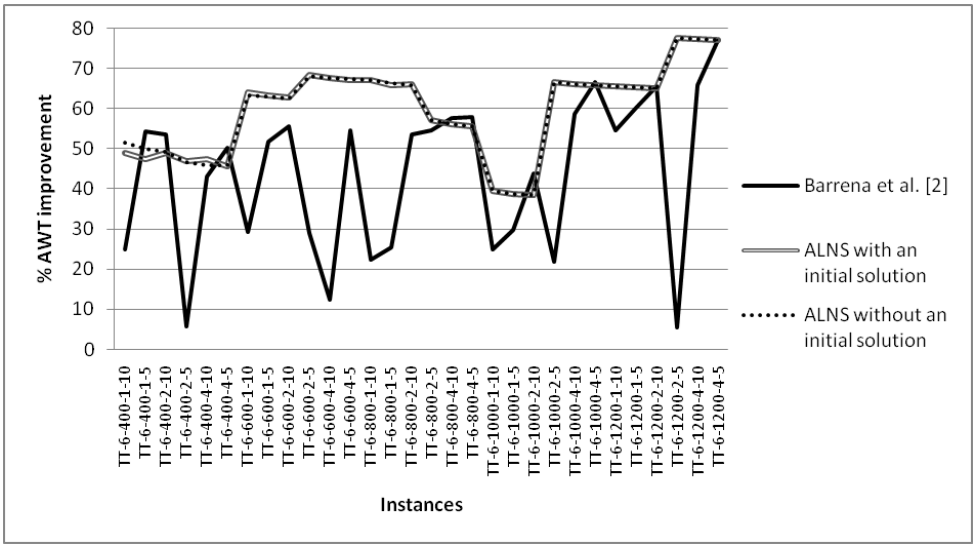


Figure 3: Percentage improvement of the AWT on instances with six stations with respect to a regular timetable

Table 2: Summary of computational results of a regular timetable, of Barrena et al. [2], and of our ALNS on instances with three stations

	Regular	Barrena et al. [2]				ALNS			Barrena et al. [2] vs. ALNS
Instance	AWT	UB_1	LB	Time (s)	AWT	UB_2	Time (s)	AWT	Improvement (%)
TT-3-200-1-5*	19.39	37368	37368	366	6.27	37368	20	6.27	0.00
TT-3-200-2-5*	18.51	35210	35210	28	6.90	35210	10	6.90	0.00
TT-3-200-4-5*	15.81	29428	29428	1	6.93	29428	5	6.93	0.00
TT-3-400-1-5	44.05	102418	52149	10800	9.08	102456	32	9.08	−0.04
TT-3-400-2-5*	43.50	98412	98412	476	9.72	98412	17	9.72	0.00
TT-3-400-4-5*	39.75	87260	87260	35	9.73	87260	8	9.73	0.00
TT-3-600-1-5	44.99	167941	31144	10801	12.34	167175	49	12.29	0.46
TT-3-600-2-5*	43.33	162008	162008	7979	12.90	162008	25	12.90	0.00
TT-3-600-4-5*	39.44	150748	150748	906	13.07	150748	13	13.07	0.00
TT-3-800-1-5	87.36	290061	29381	10802	17.01	290068	60	17.01	0.00
TT-3-800-2-5	89.26	282996	65078	10801	17.60	282996	31	17.60	0.00
TT-3-800-4-5*	97.84	270468	270468	3480	17.85	270468	15	17.85	0.00
TT-3-1000-1-5	112.70	393820	25130	10802	23.10	301576	74	17.69	23.42
TT-3-1000-2-5	111.55	294992	50677	10801	18.30	294992	39	18.30	0.00
TT-3-1000-4-5*	108.02	283516	283516	4627	18.62	283516	20	18.62	0.00
TT-3-200-1-10*	6.48	28961	28961	153	4.86	29032	31	4.87	−0.25
TT-3-200-2-10*	8.17	26366	26366	18	5.42	26502	16	5.44	−0.52
TT-3-200-4-10*	8.17	20744	20744	2	5.48	21332	8	5.58	−2.83
TT-3-400-1-10	22.48	59715	29887	10801	5.29	61717	49	5.47	−3.35
TT-3-400-2-10	19.24	55714	54734	10800	5.94	56338	26	5.99	−1.12
TT-3-400-4-10*	19.24	45592	45592	140	6.04	45592	13	6.04	0.00
TT-3-600-1-10	24.47	99034	17418	10801	7.28	89538	62	6.58	9.59
TT-3-600-2-10	21.75	83512	43803	10801	7.13	84062	32	7.17	−0.66
TT-3-600-4-10*	21.75	70460	70460	8649	7.17	70584	16	7.18	−0.18
TT-3-800-1-10	40.43	167218	19581	10803	9.81	141642	77	8.31	15.30
TT-3-800-2-10	43.65	168932	29814	10801	10.91	135556	42	8.95	19.76
TT-3-800-4-10	43.66	127528	54541	10801	9.47	119832	22	9.02	6.03
TT-3-1000-1-10	55.08	205169	16550	10805	12.03	143477	92	8.42	30.07
TT-3-1000-2-10	52.76	158180	26658	10801	10.28	137028	46	9.04	13.37
TT-3-1000-4-10	52.76	130460	53162	10801	9.65	119736	24	9.02	8.22
Average	45.19	137808	64875	6656	10.54	**129188**	**32.47**	**10.04**	3.91

Table 3: Summary of computational results of a regular timetable, of Barrena et al. [2], and of our ALNS on instances with six stations

	Regular	Barrena et al. [2]				ALNS			Barrena et al. [2] vs. ALNS
Instance	AWT	UB_1	LB	Time (s)	AWT	UB_2	Time (s)	AWT	Improvement (%)
TT-6-400-1-5	38.94	697886	31357	10801	17.70	657026	66	16.67	5.85
TT-6-400-2-5	39.42	618322	89743	10800	16.66	642086	34	17.26	−3.84
TT-6-400-4-5	39.42	575780	235328	10801	16.58	610176	18	17.45	−5.97
TT-6-600-1-5	38.76	1583040	64565	10802	29.12	1272434	109	23.41	19.62
TT-6-600-2-5	39.27	1446760	105349	10800	27.57	1252694	54	24.01	13.41
TT-6-600-4-5	39.29	1091000	260442	10800	22.03	1205644	29	24.13	−10.51
TT-6-800-1-5	91.73	4501100	44831	10803	71.63	1930834	114	30.73	57.10
TT-6-800-2-5	92.21	2331550	52831	10801	38.05	1908152	65	31.32	18.16
TT-6-800-4-5	92.22	1815690	216420	10801	30.83	1853772	31	31.44	−2.10
TT-6-1000-1-5	93.19	2655010	48904	10804	42.25	2009144	151	31.98	24.33
TT-6-1000-2-5	93.73	2281690	63335	10801	37.26	1987872	78	32.59	12.88
TT-6-1000-4-5	93.72	1904200	244483	10801	32.24	1936144	38	32.75	−1.68
TT-6-1200-1-5	145.75	8653510	39394	10806	137.72	2055965	181	32.72	76.24
TT-6-1200-2-5	146.20	3079810	67542	10802	49.94	2035826	84	33.35	33.90
TT-6-1200-4-5	146.14	1973860	133449	10801	33.35	1984188	47	33.51	−0.52
TT-6-400-1-10	18.32	541427	31841	10802	13.73	367925	95	9.33	32.05
TT-6-400-2-10	18.81	299506	54620	10800	8.59	349606	53	9.86	−16.73
TT-6-400-4-10	18.81	264824	112558	10801	8.71	299864	29	9.59	−13.23
TT-6-600-1-10	25.18	1289140	43125	10804	23.72	725086	126	13.34	43.75
TT-6-600-2-10	25.67	738628	85181	10801	14.57	680844	60	13.51	7.82
TT-6-600-4-10	25.68	585832	162597	10800	12.76	652088	33	13.97	−11.31
TT-6-800-1-10	44.29	1964960	18149	10805	31.27	998887	161	15.90	49.17
TT-6-800-2-10	44.77	1292330	59877	10802	21.54	972616	84	16.46	24.74
TT-6-800-4-10	44.78	1123040	159611	10801	19.83	924604	46	16.68	17.67
TT-6-1000-1-10	51.55	2301980	21797	10807	36.64	1022806	196	16.28	55.57
TT-6-1000-2-10	52.07	2803080	54501	10803	45.55	1000818	113	16.90	64.30
TT-6-1000-4-10	52.07	1360370	118285	10802	23.60	954772	54	17.16	29.82
TT-6-1200-1-10	49.60	2420880	27191	10812	38.53	1023010	223	16.28	57.74
TT-6-1200-2-10	50.11	2290650	50684	10803	37.40	1011344	119	17.07	55.85
TT-6-1200-4-10	50.11	1339860	90199	10801	23.28	945752	61	17.02	29.41
Average	60.06	1860857	92940	10802	32.09	**1175733**	**85.07**	**21.09**	22.12

Table 4: Summary of computational results of a regular timetable, of Barrena et al. [2], and of our ALNS on instances with 10 stations

	Regular	Barrena et al. [2]				ALNS			Barrena et al. [2] vs. ALNS
Instance	AWT	UB_1	LB	Time (s)	AWT	UB_2	Time (s)	AWT	Improvement (%)
TT-10-400-1-5	43.59	3033290	237065	10801	21.61	3808505	120	27.13	−25.56
TT-10-400-2-5	44.16	2833700	615530	10800	21.17	3778528	62	27.89	−33.34
TT-10-400-4-5	45.69	2531620	1116720	10800	20.01	3583692	36	27.49	−41.56
TT-10-600-1-5	49.64	7519990	767088	10803	42.18	4510632	157	25.30	40.02
TT-10-600-2-5	50.37	6895040	994635	10801	39.62	4480222	86	26.10	35.02
TT-10-600-4-5	50.72	4721520	1299070	10801	28.43	4301624	46	26.08	8.89
TT-10-800-1-5	78.24	16622200	178477	10804	76.40	7828656	214	35.98	52.90
TT-10-800-2-5	78.75	12715800	194557	10801	59.37	7931410	116	37.41	37.63
TT-10-800-4-5	76.97	8954050	404503	10800	43.08	7584952	57	36.80	15.29
TT-10-1000-1-5	100.55	19846900	96539	10806	91.14	8463409	258	38.87	57.36
TT-10-1000-2-5	101.10	20319300	120569	10802	94.18	8387908	138	39.46	58.72
TT-10-1000-4-5	100.51	10387000	381954	10800	49.60	8142144	74	39.31	21.61
TT-10-1200-1-5	136.81	29476600	155406	10808	135.36	8897358	350	40.86	69.82
TT-10-1200-2-5	137.30	29013600	115835	10802	134.05	8891232	170	41.77	69.35
TT-10-1200-4-5	133.25	22723500	214828	10801	106.13	8720024	86	41.96	61.63
TT-10-400-1-10	30.11	3983120	195417	10802	28.37	3022055	169	21.53	24.13
TT-10-400-2-10	33.84	3292210	364183	10800	24.43	3012880	105	22.44	8.48
TT-10-400-4-10	33.75	1285320	534844	10800	11.14	2798428	50	21.91	−117.72
TT-10-600-1-10	28.38	4816760	614681	10805	27.02	2959429	203	16.60	38.56
TT-10-600-2-10	28.86	4314360	750225	10801	25.17	2919640	114	17.35	32.33
TT-10-600-4-10	28.74	2498310	998377	10800	15.99	2750996	60	17.40	−10.11
TT-10-800-1-10	37.08	8006310	28024	10808	36.80	4136948	259	19.02	48.33
TT-10-800-2-10	36.79	7689280	110549	10802	36.30	4040168	155	19.55	47.46
TT-10-800-4-10	36.53	6324230	178630	10801	31.01	3859848	81	19.71	38.97
TT-10-1000-1-10	44.33	9627130	7267.72	10813	44.21	4399797	325	20.20	54.30
TT-10-1000-2-10	45.18	9578170	117209	10803	44.92	4368718	175	21.03	54.39
TT-10-1000-4-10	45.21	8321170	289559	10800	40.14	4173512	93	21.13	49.84
TT-10-1200-1-10	53.80	11534900	14218	10818	52.97	4550840	374	20.90	60.55
TT-10-1200-2-10	53.53	10365300	97221	10805	48.53	4486944	203	21.58	56.71
TT-10-1200-4-10	53.13	9601930	194502	10801	46.00	4300716	94	21.71	55.21
Average	60.56	9961087	379589	10803	49.18	**5169707**	**147.67**	**27.15**	28.97

Table 5: Summary of computational results of a regular timetable, of Barrena et al. [2], and of our ALNS on C5 instances

	Regular	Barrena et al. [2]				ALNS			Barrena et al. [2] vs. ALNS
Instance	AWT	UB_1	LB	Time (s)	AWT	UB_2	Time (s)	AWT	Improvement (%)
$C5$-400-1-5	32.04	351578	44711	10801	28.07	347774	60	27.77	1.08
$C5$-400-2-5	32.27	366436	62644	10801	28.87	365958	32	28.84	0.13
$C5$-400-4-5	32.17	368016	144576	10801	29.33	367672	17	29.31	0.09
$C5$-600-1-5	49.99	713138	30773	10801	45.87	657123	84	42.27	7.85
$C5$-600-2-5	50.26	712808	62346	10801	44.74	685316	43	43.05	3.86
$C5$-600-4-5	49.87	697884	100063	10800	43.94	691816	23	43.58	0.87
$C5$-800-1-5	69.22	1420300	44429	10802	69.22	1167838	105	56.91	17.78
$C5$-800-2-5	69.08	1295520	69211	10801	61.77	1204986	53	57.53	6.99
$C5$-800-4-5	68.03	1297920	130709	10801	61.57	1222216	28	58.09	5.83
$C5$-1000-1-5	84.98	2232260	37783	10804	84.94	1920051	129	73.06	13.99
$C5$-1000-2-5	85.44	2117930	73783	10801	78.77	1985946	66	73.92	6.23
$C5$-1000-4-5	84.63	2069540	132404	10801	76.83	2009336	34	74.66	2.91
$C5$-1200-1-5	109.00	3273840	34870	10806	109.00	2716630	154	90.45	17.02
$C5$-1200-2-5	108.70	3350500	67301	10802	108.70	2796192	80	90.88	16.54
$C5$-1200-4-5	107.73	2961000	127146	10800	96.02	2815152	44	91.39	4.93
$C5$-400-1-10	17.56	205070	21974	10802	16.37	197638	88	15.78	3.62
$C5$-400-2-10	18.96	221142	38235	10801	17.82	200196	47	16.23	9.47
$C5$-400-4-10	18.98	194616	56767	10800	16.45	197020	26	16.63	−1.24
$C5$-600-1-10	25.73	400063	15834	10803	25.73	358608	113	23.07	10.36
$C5$-600-2-10	26.26	396106	38197	10801	25.31	372610	59	23.86	5.93
$C5$-600-4-10	26.30	389040	71623	10801	25.38	368204	32	24.13	5.36
$C5$-800-1-10	35.13	720960	20380	10807	35.13	649808	136	31.67	9.87
$C5$-800-2-10	35.34	732108	32511	10802	35.34	667874	71	32.33	8.77
$C5$-800-4-10	35.44	697328	92559	10801	34.00	667392	38	32.63	4.29
$C5$-1000-1-10	44.44	1167870	18472	10808	44.44	1063456	163	40.47	8.94
$C5$-1000-2-10	44.68	1189540	30750	10802	44.68	1092802	85	41.13	8.13
$C5$-1000-4-10	44.79	1177340	108988	10801	44.57	1089628	45	41.40	7.45
$C5$-1200-1-10	54.50	1637000	11019	10811	54.50	1489795	189	49.60	8.99
$C5$-1200-2-10	54.43	1662340	22438	10803	54.43	1531472	101	50.23	7.87
$C5$-1200-4-10	54.63	1562580	101524	10801	51.62	1527444	54	50.50	2.25
Average	52.35	1186059	61467	10802	49.78	**1080932**	**73.3**	**45.71**	6.87

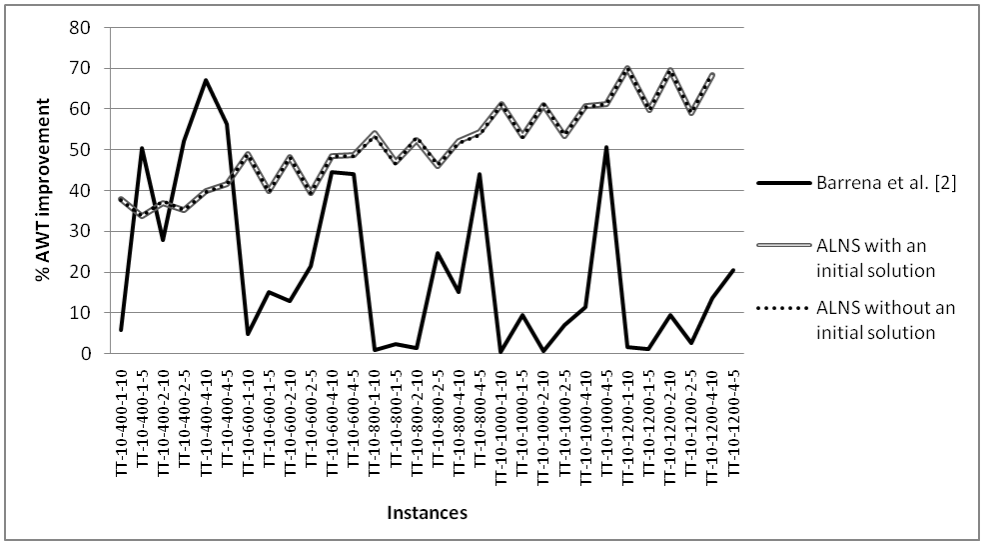


Figure 4: Percentage improvement of the AWT on instances with 10 stations with respect to a regular timetable

5 Conclusions

We have proposed an ALNS algorithm for a train timetabling problem adapted to a dynamic demand pattern, with the objective of minimizing the AWT per passenger. Exact formulations for this problem are available [2] but these can only be used to solve very small instances, and yield very large optimality gaps for medium and large size ones. Through extensive computational experiments on real-world based and randomly generated instances, we have compared our ALNS against a state-of-art truncated branch-and-cut algorithm. We have reduced the average passenger waiting time by 26% by using less than 1% of the computation time required by this 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 superiority of our ALNS heuristic.

References

[1] M. Abril, M. A. Salido, F. Barber, L. Ingolotti, P. Tormos, and A. Lova. Distributed constraint satisfaction problems to model railway scheduling problems. In J. Allan, C. A. Brebbia, A. F. Rumsey, G. Scuitto, S. Sone, and Goodman C. J., editors, *Proceedings of the X International Conference Computers in Railways. Computer System Design and Operation in the Railway and Other Transit Systems*, pages 289–297, Prague, Czech Republic, 10–12, July 2006. WIT Press, Southampton, UK.

[2] E. Barrena, D. Canca, L. C. Coelho, and G. Laporte. Exact formulations and algorithm for the train timetabling problem with dynamic demand. Technical Report CIRRELT-2013-41, Montréal, 2013.

[3] U. Brännlund, P. O. Lindberg, A. Nou, and J. E. Nilsson. Railway timetabling using lagrangian relaxation. *Transportation Science*, 32(4):358–369, 1998.

[4] R. L. Burdett and E. Kozan. A disjunctive graph model and framework for constructing new train schedules. *European Journal of Operational Research*, 200(1): 85–98, 2010.

[5] R. L. Burdett and E. Kozan. A sequencing approach for creating new train timetables. *OR Spectrum*, 32(1):163–193, 2010.

[6] V. Cacchiani and P. Toth. Nominal and robust train timetabling problems. *European Journal of Operational Research*, 219(3):727–737, 2012.

[7] V. Cacchiani, A. Caprara, and P. Toth. A column generation approach to train timetabling on a corridor. *4OR*, 6(2):125–142, 2008.

[8] V. Cacchiani, A. Caprara, and P. Toth. Non-cyclic train timetabling and comparability graphs. *Operations Research Letters*, 38(3):179–184, 2010.

[9] V. Cacchiani, A. Caprara, and P. Toth. Solving a real-world train-unit assignment problem. *Mathematical programming*, 124(1-2):207–231, 2010.

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

[11] G. Caimi, D. Burkolter, T. Herrmann, F. Chudak, and M. Laumanns. Design of a railway scheduling model for dense services. *Networks and Spatial Economics*, 9(1): 25–46, 2009.

[12] D. Canca, E. Barrena, E. Algaba, and A. Zarzo. Design and analysis of demand-adapted railway timetables. *Available online at arXiv:1307.0751644*, 2013.

[13] A. Caprara, M. Fischetti, and P. Toth. Modeling and solving the train timetabling problem. *Operations Research*, 50(5):851–861, 2002.

[14] A. Caprara, M. Monaci, P. Toth, and P. L. Guida. A lagrangian heuristic algorithm for a real-world train timetabling problem. *Discrete Applied Mathematics*, 154(5): 738–753, 2006.

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

[16] D. Chen, M. Lv, and S. Ni. Study on initial schedule optimization model of intercity passenger trains based on ACO algorithm. *International Journal of Advancements in Computing Technology*, 3(4):222–228, 2011.

[17] J.-W. Chung, S.-M. Oh, and I.-C. Choi. A hybrid genetic algorithm for train sequencing in the korean railway. *Omega*, 37(3):555–565, 2009.

[18] L. C. Coelho, J.-F. Cordeau, and G. Laporte. Consistency in multi-vehicle inventory-routing. *Transportation Research Part C: Emerging Technologies*, 24:270–287, 2012.

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

[20] R. Cordone and F. Redaelli. Optimizing the demand captured by a railway system with a regular timetable. *Transportation Research Part B: Methodological*, 45(2): 430–446, 2011.

[21] F. Corman, A. D'Ariano, D. Pacciarelli, and M. Pranzo. A tabu search algorithm for rerouting trains during rail operations. *Transportation Research Part B: Methodological*, 44(1):175–192, 2010.

[22] F. Furini and M. P. Kidd. A fast heuristic approach for train timetabling in a railway node. *Electronic Notes in Discrete Mathematics*, 41:205–212, 2013.

[23] V. Guihaire and J. K. Hao. Transit network design and scheduling: A global review. *Transportation Research Part A: Policy and Practice*, 42(10):1251–1273, 2008.

[24] L. Ingolotti, A. Lova, F. Barber, P. Tormos, M. A. Salido, and M. Abril. New heuristics to solve the CSOP railway timetabling problem. In M. Ali and R. Dapoigny, editors, *Advances in Applied Artificial Intelligence*, volume 4031 of *Lecture Notes in Artificial Intelligence*, pages 400–409. Springer, Berlin Heidelberg, 2006.

[25] A. Jamili, M. A. Shafia, S. J. Sadjadi, and R. Tavakkoli-Moghaddam. Solving a periodic single-track train timetabling problem by an efficient hybrid algorithm. *Engineering Applications of Artificial Intelligence*, 25(4):793–800, 2012.

[26] L. G. Kroon, D. Huisman, E. Abbink, P.-J. Fioole, M. Fischetti, G. Maróti, A. Schrijver, A. Steenbeek, and R. Ybema. The new Dutch timetable: The OR revolution. *Interfaces*, 39(1):6–17, 2009.

[27] Y. Lee and C.-Y. Chen. A heuristic for the train pathing and timetabling problem. *Transportation Research Part B: Methodological*, 43(8):837–851, 2009.

[28] C. Liebchen. A cut-based heuristic to produce almost feasible periodic railway timetables. In S. E. Nikoletseas, editor, *Experimental and Efficient Algorithms. Lecture Notes in Computer Science*, volume 3503, pages 354–366. Springer, Berlin Heidelberg, 2005.

[29] C. Liebchen and R. Möhring. A case study in periodic timetabling. *Electronic Notes in Theoretical Computer Science*, 66(6):1–14, 2002.

[30] K. Nachtigall and S. Voget. A genetic algorithm approach to periodic railway synchronization. *Computers & Operations Research*, 23(5):453–463, 1996.

[31] M. A. Odijk. A constraint generation algorithm for the construction of periodic railway timetables. *Transportation Research Part B: Methodological*, 30(6):455–464, 1996.

[32] M. Reimann and J. E. Leal. Single line train scheduling with ACO. In M. Middendorf and C. Blum, editors, *Evolutionary Computation in Combinatorial Optimization*, volume 7832 of *Lecture Notes in Computer Science*, pages 226–237. Springer, Berlin Heidelberg, 2013.

[33] S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472, 2006.

[34] M. A. Salazar-Aguilar, A. Langevin, and G. Laporte. Synchronized arc routing for snow plowing operations. *Computers & Operations Research*, 39(7):1432–1440, 2012.

[35] P. Serafini and W. Ukovich. A mathematical model for periodic scheduling problems. *SIAM Journal on Discrete Mathematics*, 2(4):550–581, 1989.

[36] P. Tormos, A. Lova, F. Barber, L. Ingolotti, M. Abril, and M. A. Salido. A genetic algorithm for railway scheduling problems. In F. Xhafa and A. Abraham, editors, *Metaheuristics for Scheduling in Industrial and Manufacturing Applications. Studies in Computational Intelligence*, volume 128, pages 255–276. Springer, Berlin Heidelberg, 2008.

[37] P. Vansteenwegen and D. Van Oudheusden. Developing railway timetables which guarantee a better service. *European Journal of Operational Research*, 173(1):337–350, 2006.

[38] R.C.W. Wong, T.W.Y. Yuen, K.W. Fung, and J.M.Y. Leung. Optimizing timetable synchronization for rail mass transit. *Transportation Science*, 42(1):57–69, 2008.

[39] Z. Yuan, A. Fügenschuh, H. Homfeld, P. Balaprakash, T. Stützle, and M. Schoch. Iterated greedy algorithms for a real-world cyclic train scheduling problem. In M. J. Blesa, C. Blum, C. Cotta, A. J. Fernández, J. E. Gallardo, A. Roli, and M. Sampels, editors, *Hybrid Metaheuristics*, volume 5296 of *Lecture Notes in Computer Science*, pages 102–116. Springer, Berlin Heidelberg, 2008.

[40] X. Zhou and M. Zhong. Single-track train timetabling with guaranteed optimality: Branch-and-bound algorithms with enhanced lower bounds. *Transportation Research Part B: Methodological*, 41(3):320–341, 2007.