Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2009, Article ID 728916, 19 pages
doi:10.1155/2009/728916

Research Article

Mathematical Solutions for Solving Periodic Railway Transportation

Miguel A. Salido and Federico Barber

Instituto Universitario de Automática e Informática Industrial, Universidad Politécnica de Valencia, 46022 Valencia, Spain

Correspondence should be addressed to Miguel A. Salido, msalido@dsic.upv.es

Received 28 November 2008; Revised 28 January 2009; Accepted 9 February 2009

Recommended by J. Jiang

Train scheduling has been a significant issue in the railway industry. Over the last few years, numerous approaches and tools have been developed to compute railway scheduling. In this paper, we present a set of heuristics for a constraint-based train scheduling tool, which is a project in collaboration with the National Network of Spanish Railways (RENFE), Spain. We formulate train scheduling as constraint optimization problems. Three heuristics are developed to speed up and direct the search toward suboptimal solutions in periodic train scheduling problems. The feasibility of our problem-oriented heuristics is confirmed with experimentation using real-life data. The results show that these techniques enable MIP solvers such as LINGO and ILOG Concert Technology (CPLEX) to terminate earlier with good solutions.

Copyright © 2009 M. A. Salido and F. Barber. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

1. Introduction

Railway transportation has played a major role in the economic development of the last two centuries. It represented a major improvement in land transport technology and has obviously introduced important changes in the movement of freight and passengers. Over the last few years, railway traffic has increased considerably, which has created the need to optimize the use of railway infrastructures. This is, however, a very difficult task. Thanks to developments in computer science and advances in the fields of optimization and intelligent resource management, railway managers can optimize the use of available infrastructures, obtain more robust timetables [1], and obtain useful conclusions about capacity of their topology [2].

The *MOM* project is a long-term collaboration between the Polytechnic University of Valencia (UPV) and the National Network of Spanish Railways (RENFE). The aim of the project is to offer assistance in the planning of train scheduling, to obtain conclusions about

the maximum capacity of a line, to identify bottlenecks, to determine the consequences of changes, to provide support in the resolution of incidents, to provide alternative planning and real traffic control, and so forth. Besides the mathematical processes, a high level of interaction with railway experts is required to be able to take advantage of their experience.

In this paper, we propose several problem-oriented heuristics for solving periodic train scheduling. The problem formulation is (traditionally) translated into a mathematical model to be solved for optimality by means of mixed integer programming (MIP) techniques. However, hundred of trains, in different directions, along paths of dozens of stations, with constraints about departure and arrival times, generate thousands of inequalities and a high number of variables take only integer values. As is well known, this type of model is far more difficult to solve than linear programming models.

In our framework, the mathematical model is simplified by heuristics in order to be solved efficiently. We present two main classes of heuristics.

(1) Heuristics based on linear programming and local search.

(2) A heuristic based on railway topological characteristics and on constraint satisfaction techniques.

The first heuristic shares the following idea:

(i) to execute the linearized problem,

(ii) to obtain the value of decision variables,

(iii) carry out local search techniques guided by the previous result for assigning values to decision variables such that the entire problem is optimized,

(iv) to execute the simplified linear programming problem.

The second heuristic carries out a study of the railway topological characteristics such as the distance between stations, the number of tracks, and the railway capacity. This heuristic is able to identify the set of stations, where a bottleneck is more probable. Thus, based on the *first-fail* principle, which can be explained as "*To succeed, try first where you are more likely to fail*," the philosophy of our constraint ordering heuristic (COH) [3] is used in this technique. In this way, the problem constraints are classified such that the most restricted constraints are studied first. Then, the problem is partitioned in a set of subproblems such that the solution of each subproblem will generate a traffic pattern (more information can be found in http://www.dsic.upv.es/users/ia/gps/MOM).

2. State of the Art

The train scheduling problem has received considerable attention in the literature: [4] is the first to propose a branch and bound algorithm for train scheduling; [5] defines local search, tabu search, genetic and hybrid heuristics; [6] illustrates a constructive greedy heuristic. Periodic timetables for railway networks is usually modeled by Periodic Event Scheduling Problem (PESP) [7]. It is known that the PESP is NP hard [7]. Approaches to solve PESP instances cover backtracking strategies in a branch-and-bound context [7], genetic algorithms [8], and some classes of cutting planes [9]. The PESP model has also been used by Kroon and Peeters [10], Liebchen [11].

Furthermore, several European companies are also working on similar systems. These systems include complex stations, rescheduling due to incidents [12], and rail network

capacities [13]. These are complex problems for which work in network topology and heuristic-dependent models can offer appropriate solutions.

Many researchers have modeled the problem as a mathematical model. Mathematical programming methodology was first applied to this problem by Amit and Goldfarb [14]. Cordeau et al. [15] presented a survey of relevant optimization models, although the mathematical programming approach is not limited to optimization models. There are studies using heuristic models such as [16]. Optimization models are used in [4, 17–20], and so forth. However, the related literature has experienced a slow growth and, until recently, most contributions were dealing with only simplified models or small instances failing to incorporate the characteristics of real-life applications. Surveys of Assad [21] and Haghani [22] suggest that optimization models for rail transportation were not widely employed in practice, instead simulation models were mostly used [17]. In fact, the large size and the complexity of the problem have hindered the development of optimization models for train scheduling.

Most of these optimization models are solved by using branch and bound techniques. Nevertheless, due to the complexity of the problems, it is necessary the use of heuristics to solve them efficiently. For solving this drawback, we propose some problem-oriented heuristics for solving periodic train scheduling in a single line. These heuristics make the problem easier, and branch and bound algorithms can solve the problem more efficiently.

3. The Mathematical Model

In this section, we provide the terminology used by the railway operators of the National Network of Spanish Railways (RENFE). Furthermore, we present our mathematical model which can be described as a constraint optimization problem. The objective function is to minimize the journey time of all trains. Variables are frequencies, arrival, and departure times of trains at stations and decision variables generated for modeling disjunctive constraints. Constraints are composed by user requirements, traffic rules, and topological constraints. These constraints are composed by the parameters defined by user interfaces and database accesses.

A running map contains information regarding railway topology (stations, tracks, distances between stations, traffic control features, etc.) and the schedules of the trains that use this topology (arrival and departure times of trains at each station, frequency, stops, junctions, crossings, etc.). A sample of a running map is shown in Figure 1, where several train crossings can be observed. On the left side of Figure 1, the names of the stations are presented and the vertical line represents the number of tracks between stations (one-way or two-way). The objective of our system MOM is to obtain a correct and optimized running map taking into account: (i) traffic rules, (ii) user requirements, and (iii) the railway infrastructure topology (parameters of trains to be scheduled).

A railway line is basically composed of stations and one-way or two-way tracks. A dependency can be the following.

Station

Place for trains to park, stop, or pass through. Each station is associated with a unique station identifier. There are two or more tracks in a station where crossings or overtaking can be performed.

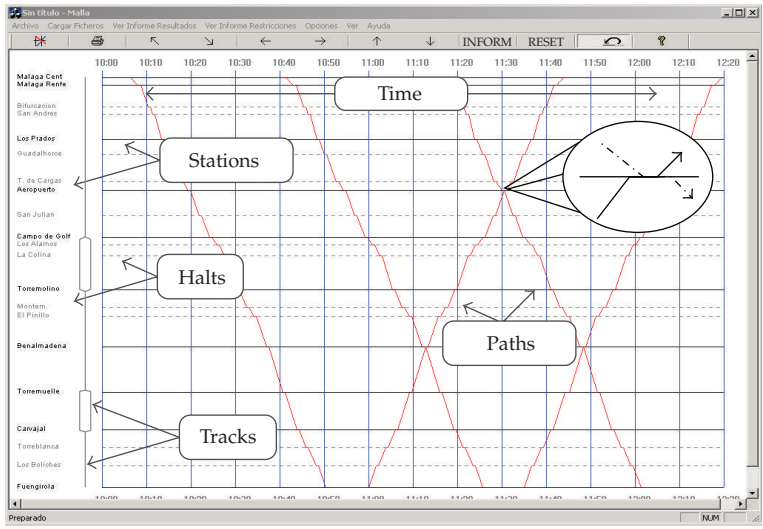


Figure 1: A sample of a running map.

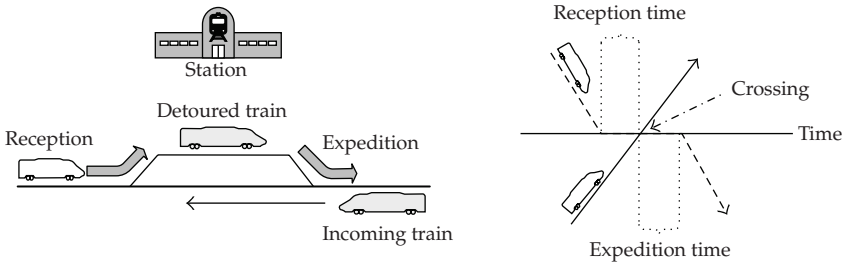


Figure 2: Constraints related to crossing and overtaking in stations.

Halt

Place for trains to stop, pass through, but not park. Each halt is associated with a unique halt identifier.

In Figure 1, horizontal dotted lines represent halts, while continuous lines represent stations. On a rail line, the user needs to schedule the paths of n trains going in one direction and m trains going in the opposite direction. These trains are of a given type, and a scheduling frequency is required.

The type of trains to be scheduled determines the time assigned for travel between two locations on the path. The path selected by the user for a train trip determines which stations are used and the stop time required at each station for commercial purposes. In order to perform crossing in a section with a one-way track, one of the trains should wait in a station. This is called a *technical stop*. One of the trains is detoured from the main track so that the other train can cross or continue (Figure 2).

3.1. Parameters

In the mathematical model the notation mentioned at the end of the paper will be used.

Decision Variables

(i) Variables X's. A variable X_{i-j} determines which train (i or j) arrives earlier to the crossing station. If $X_{i-j} = 0$, then train i arrives at the crossing station first and train j arrives at the same station later:

$$X_{i-j} = \begin{cases} 1, & \text{if train } j \text{ arrives at the crossing station first,} \\ 0, & \text{if train } i \text{ arrives at the crossing station first.} \end{cases} \tag{3.1}$$

(ii) Variables Y's. A variable $Y_{i-j;k-k+1}$ determines the track between station k and station $k+1$ in which train i crosses with train j. If $Y_{i-j;(h-1)-h} = 1$ for $h \leq k$ and $Y_{i-j;p-(p+1)} = 0$ for $p \geq k$ then, the crossing between train i and train j is carried out in station k:

$$Y_{i-j;k-(k+1)} = \begin{cases} 1, & \text{if } Y_{i-j;(h-1)-h} = 1 \text{ for } h \leq k, \\ 0, & \text{if } Y_{i-j;p-(p+1)} = 0 \text{ for } p \geq k. \end{cases} \tag{3.2}$$

3.2. Objective Function and Constraints

The mathematical model is presented in Algorithm 1. Let us suppose a railway line with r stations, n trains running in the down direction, and m trains running in the up direction. We assume that two connected stations have only one line connecting them.

The main complexity of the problem derives in solving the MIP problem due to the decision variables. If we are able to assign values to these decision variables, the linearized problem can be solved more efficiently. Therefore, the main goal of our heuristics is to find values for these decision variables. This assignment will be carried out by means of local search and railway topological knowledge.

The objective function is devoted to travel time. This function reflects mainly the users concerns; however, the railway companies are also interested in saving time due to the more efficient usage of rolling stocks. There are hundreds of studies being undertaken for evaluating savings in travel time (for a review see [23]). The passengers want to reach the destination as soon as possible to carry on their activities.

Thus our objective function:

(1) $\text{Min} \sum_{i=1}^{i=n} (T_i A_r - T_i D_1) + \sum_{j=1}^{i=m} (T_j A_1 - T_j D_r)$

minimizes the travel time of all trains in both directions.

Regarding the constraints, there are three groups of scheduling rules in our railway system: traffic rules, user requirements rules, and topological rules. A valid running map must satisfy and optimize the above rules. These scheduling rules can be modeled using the following constraints:

Traffic rules guarantee crossing, expedition, and reception operations. The main constraints are the following.

(1) $\text{Min}\sum_{i=1}^{i=n}(T_iA_r - T_iD_1) + \sum_{j=1}^{i=m}(T_jA_1 - T_jD_r)$;
Subject To
/frequency constraint $\forall i = 1\cdots n,\ \forall k = 1\cdots r$
(2) $T_{i+1}D_k - T_iD_k = \text{Freq}$;
/Time constrains $\forall i = 1\cdots n,\ \forall k = 1\cdots r$
(3.1) $T_iA_{k+1} - T_iD_k = \text{Time } i_{k-(k+1)}$;
(3.2) $T_jA_k - T_jD_{k+1} = \text{Time } j_{k-(k+1)}$;
/Stations time constrains $\forall i = 1\cdots n,\ \forall k = 1\cdots r$
(4) $T_iD_k - T_iA_k - TSi_k = CSi_k$;
/Constrains to limit journey time $\forall i = 1\cdots n,\ \forall j = 1\cdots m$
(5.1) $T_iA_r - T_iD_1 \le (1 + \delta/100) * \text{Time } i_{1-r}$;
(5.2) $T_jA_1 - T_jD_r \le (1 + \delta/100) * \text{Time } j_{r-1}$;
/Crossing constrains $\forall i = 1\cdots n,\ \forall j = 1\cdots m,\ \forall k = 1\cdots r$
(6.1) $T_jA_k - T_iD_k <= UB * Y_{i-j;k-(k+1)}$;
(6.2) $T_iA_{k+1} - T_jD_{k+1} <= UB * (1 - Y_{i-j;k-(k+1)})$;
/Expedition time constrains $\forall i = 1\cdots n,\ \forall j = 1\cdots m,\ \forall k = 1\cdots r$
(7.1) $T_jA_k - T_iD_k - UB * (X_{i-j} - Y_{i-j;k-(k+1)} + Y_{i-j;(k+1)-(k+2)} - 1) + ET_i <= 0$;
(7.2) $T_iA_k - T_jD_k - UB * (X_{i-j} - Y_{i-j;k-(k+1)} + Y_{i-j;(k+1)-(k+2)} - 2) + ET_j <= 0$;
/Reception time constrains $\forall i = 1\cdots n,\ \forall j = 1\cdots m,\ \forall k = 1\cdots r$
(8.1) $T_iA_k - T_jA_k - UB * (X_{i-j} - Y_{i-j;k-(k+1)} + Y_{i-j;(k+1)-(k+2)} - 1) + RT_i <= 0$;
(8.2) $T_jA_k - T_iA_k - UB * (X_{i-j} - Y_{i-j;k-(k+1)} + Y_{i-j;(k+1)-(k+2)} - 2) + RT_j <= 0$;
/Binary constraints
X_{i-j}; $\forall i = 1\cdots n,\ \forall j = 1\cdots m$
$Y_{i-j;k-(k+1)}$; $\forall i = 1\cdots n,\ \forall j = 1\cdots m,\ \forall k = 1\cdots r$

Algorithm 1: The mathematical model of the railway scheduling problem.

Crossing Constraints (6.1) and (6.2)

Any two trains going in opposite directions must not simultaneously use the same one-way track:

$$(6.1)\ T_jA_k - T_iD_k <= UB * Y_{i-j;k-(k+1)},$$

$$(6.2)\ T_iA_{k+1} - T_jD_{k+1} <= UB * (1 - Y_{i-j;k-(k+1)}).$$

The crossing of two trains can be performed only on two-way tracks and at stations, where one of the two trains has been detored from the main track (Figure 2). Several crossings are shown in Figure 1. Thus, constraints (6.1) and (6.2) that represent the disjunctive constraint which restricts two trains going in opposite directions require the same section of track at the same time.

Expedition Time Constraint

At least are required ET_i time units at location k between the arrival and departure times of two trains T_j, T_i going in the opposite direction. Thus, constraints (7.1) and (7.2) that represent the disjunctive constraint which restricts two trains going in opposite directions require a buffer time between the arrival and departure times of two trains:

$$(7.1)\ T_jA_k - T_iD_k - UB * (X_{i-j} - Y_{i-j;k-(k+1)} + Y_{i-j;(k+1)-(k+2)} - 1) + ET_i <= 0,$$

$$(7.2)\ T_iA_k - T_jD_k - UB * (X_{i-j} - Y_{i-j;k-(k+1)} + Y_{i-j;(k+1)-(k+2)} - 2) + ET_j <= 0.$$

Reception Time Constraint

At least are required RT_i time units at location k between the arrival times of two trains T_j, T_i going in the opposite direction. Thus, constraints (8.1) and (8.2) that represent the disjunctive constraint which restricts two trains going in opposite directions require a buffer time between the arrival of two trains:

(8.1) $T_iA_k - T_jA_k - UB * (X_{i-j} - Y_{i-j;k-(k+1)} + Y_{i-j;(k+1)-(k+2)} - 1) + RT_i \leq 0,$

(8.2) $T_jA_k - T_iA_k - UB * (X_{i-j} - Y_{i-j;k-(k+1)} + Y_{i-j;(k+1)-(k+2)} - 2) + RT_j \leq 0.$

User Requirements

The main constraints due to user requirements are:

(i) *type and number of trains* going in each direction to be scheduled,

(ii) *path of trains.* Locations used in each direction,

(iii) *frequency constraint*. The frequency constraint specifies the period (Freq) between departures of two consecutive trains in each direction at the same location. This constraint is very restrictive because, when crossing is performed, trains must wait for a certain time interval at stations. This interval must be propagated to all trains going in the same direction in order to maintain the established scheduling frequency. The user can require a fixed frequency, a frequency within a minimum and maximum interval, or multiple frequencies. Thus, constraint (2) restricts that the time period between departures of two consecutive trains at location k must be equal to Freq:

(2) $T_{i+1}D_k - T_iD_k = \text{Freq},$

topological railway infrastructure and type of trains to be scheduled give rise to other constraints to be taken into account. Some of them are what follows.

(i) Number of *tracks in stations* (to perform technical and/or commercial operations) and the number of tracks between two locations (one-way or two-way). No crossing or overtaking is allowed on a one-way track.

(ii) *Time constraints* determine the necessary time to travel between each two contiguous stations. Thus, constraints (3.1) and (3.2) represent these constraints:

(3.1) $T_iA_{k+1} - T_iD_k = \text{Time } i_{k-(k+1)},$

(3.2) $T_jA_k - T_jD_{k+1} = \text{Time } j_{k-(k+1).}$

(iii) *Station time constraints* restrict the needed time of each train for technical and/or commercial purposes. Thus, constraint (4) determines that commercial stop time is the difference between the departure and the arrival of a train in each station, minus the technical stop time:

(4) $T_iD_k - T_iA_k - TSi_k = CSi_k.$

(iv) *Constraints to limit journey time* restrict allowed time for each train to make the entire travel. Thus, constraints (5.1) and (5.2) determine that the journey time of each train must be lower than the allowed time plus an allowed margin δ:

(5.1) $T_iA_r - T_iD_1 \leq (1 + \delta/100) * \text{Time } i_{1-r}$,

(5.2) $T_jA_1 - T_jD_r \leq (1 + \delta/100) * \text{Time } j_{r-1}$.

In accordance with user requirements, the system should obtain the best solution available so that all the above constraints are satisfied. Several criteria can exist to qualify the optimality of solutions: minimize duration and/or number of technical stops, minimize the total time of train trips (span) of the total schedule, giving priority to certain trains, and so on.

4. Railway Capacity

Railway capacity has been a significant issue in the railway industry. Many approaches and tools have been developed to compute railway capacity. However, capacity is a complex and loosely defined term that has numerous meanings. In general, within a rail concept, capacity can be described as "a measure of the ability to move a specific amount of traffic over a defined rail line with a given set of resources under a specific service plan" [24]. This could mean anything from the number of tons moved, the speed of trains, the on-time-performance, the available track maintenance time, the service reliability, or the maximum number of trains per day that the subdivision can handle.

People might ask how a railway section can be full, if a train passes only once every 10 minutes or so. Most of the time, there is nothing to see [25]. Certainly, there exist many definitions of railway capacity, but there does not yet exist an accepted definition of capacity on railway line. Kreuger classified the capacity into different kinds: theoretical capacity, practical capacity, used capacity, and available capacity.

We formalize railway capacity (theoretical capacity) to design heuristic 2 for solving periodic train scheduling. This technique has been inserted in our system in order to solve this problem and to be able to obtain as good and feasible running map as possible.

4.1. Our Proposal of Railway Capacity

We will adopt the following definition of railway capacity.

Definition 4.1. Railway capacity is the maximum number of trains that can be scheduled in the railway in a fixed period of time.

Based on this definition, railway capacity will be subjected to topological restrictions of the railway (distance between trains, number of lines, speed of trains, etc.). For instance, if all sections have two-way tracks, railway capacity will be four time higher than in sections with only one-way tracks. On the contrary, if there are two very distant cities joined by a one-way track, the railway capacity will be conditioned by this track, because once a train departs from one station to the other station, no train can depart from the second station until the first train arrives. These tracks are bottlenecks, and the railway capacity is conditioned by these tracks. This is the explanation of why there are so few trains visible on the track (the question above).

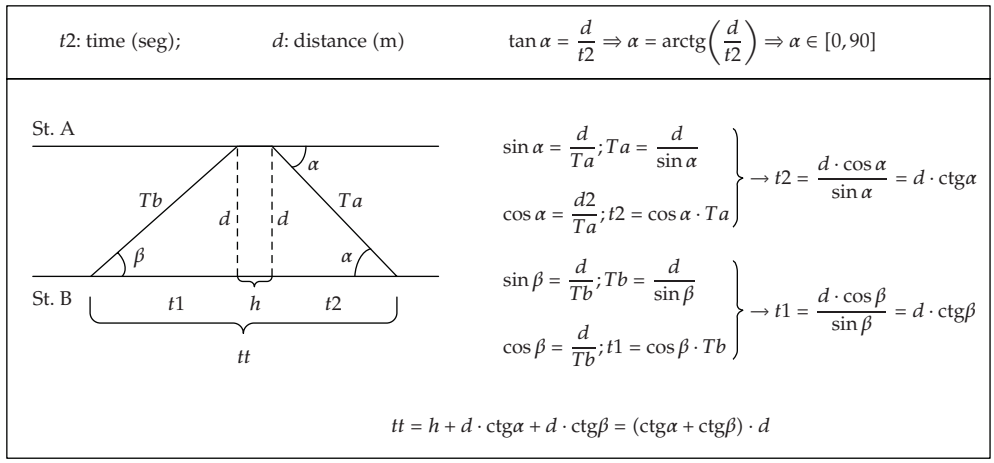


Figure 3: Journey time for a round trip.

In this section, we formalize the distance that trains must maintain to guarantee a feasible schedule. This distance will determine railway capacity. Thus, our aim is to geometrically study each train with tracks between stations.

We assign to each type of train and to each track an angle that represents the speed of this train on this track. As we maintain the distance of each track (d) and the journey time for each train-track ($t2$) in our database, the angle α can be deduced straightforwardly (Figure 3 up).

In this way, we define the angle α as arc tangent of distance (in meters) divided by time (in seconds). This angle is bounded between 0 and 90 as can be intuitively observed in the running map. Thus an angle $\alpha \rightarrow 0$ means that the train has traveled a very short distance in a long period of time. Alternatively, an angle $\alpha \rightarrow 90$ means that the train has traveled a very long distance in a short period of time.

Once, we have bounded the speed in the interval [0, 90], we will study railway capacity by means of the minimum safety distance for each track. We will focus on periodic trains and one-way tracks. Thus, all trains in each direction are of the same type.

For each track of distance "d" between two contiguous stations and each pair of trains going in opposite directions, the total time (tt) of both journeys (up direction and down direction) is $tt = h + t1 + t2 = h + d \cdot (\cot\alpha + \cot\beta)$, where $h = mrt + met + \varepsilon$ is the sum of the *minimum reception time* (mrt) of the train that arrives plus the *minimum expedition time* (met) of the train that departs and the *security time* (ε). These values ($t1$ and $t2$) are trigonometrically obtained as can be seen in Figure 3.

By applying this formula to each of the k tracks of the line, the railway capacity (denoted by C) for a time period tp is:

$$C = 2 \cdot \left\lfloor \frac{tp}{\text{Max}\left\{h_i + d_i \cdot \left(\cot\alpha_i + \cot\beta_i\right)\right\}_{i=1\cdots k}} \right\rfloor. \tag{4.1}$$

It must be taken into account that the bottleneck of the railway will occur in the station that satisfies $\text{Max}\left\{h_i + d_i \cdot \left(\cot\alpha_i + \cot\beta_i\right)\right\}_{i=1\cdots k}$. Thus, the minimal frequency must be greater than the bottleneck.

Theorem 4.2. *Given a railway R with one-way tracks with capacity C in a period tp, no additional train can be scheduled.*

Proof (proof by contradiction). Assume to the contrary that there is a new train T that can be scheduled in R. Given the capacity C and the period tp, we can obtain the track j with maximum total time. Due to the fact that track j is a one-way track and train T is scheduled on this track, to avoid crossing, this train can only arrive or depart from the critical station (all trains join) in the temporal interval h, that is, between the arrival of the predecessor train and the departure of the successor train. In this case, $tt = h + t1 + t2$, where h is the sum of mrt plus met plus ε, plus the necessary time for train T to *cross the station* (cs). Obviously, $cs > 0$, so that $h = mrt + met + \varepsilon + cs$. However, we define $h = mrt+met+\varepsilon \rightarrow cs = 0$. Number of contradiction. Therefore, any new train can be scheduled in railway R. □

5. Filtering Process: Heuristics

Given the mathematical model presented in Algorithm 1, the problem turns into an MIP problem, in which thousands of inequalities have to be satisfied and a high number of variables only take integer values. As is well known, this type of model is far more difficult to solve than linear programming models.

Here, we present two different types of heuristics that have been inserted in the system with the aim of reducing the number of variables and the complexity of the complete mathematical model.

5.1. Heuristic 1

Given the mathematical model, heuristic 1 works on the decision variables defined in Section 3.

5.1.1. Heuristic 1.0

Heuristic 1.0 is also called *complete*. This heuristic carries out a filtering over the set of constraints from the mathematical model presented in Algorithm 1. Many constraints of types (6), (7), and (8) can be removed according to their departure times and maximum slacks. If a train going in the down direction arrives at the destination before a train going in the up direction departs, then both trains will not cross each other. Thus, a huge number of constraints and decision variables we can eliminate. The original problem maintains $n * m * r$ decision variables (Y's) and $n * m$ decision variables (X's). A railway line with 100 stations and 100 trains going in each direction generates 1.01×10^6 decision variables. This heuristic may significantly reduce the problem size with a reasonable maximum slack ($\alpha \approx 20\%$).

Theorem 5.1. *Heuristic 1.0 is sound and complete.*

Proof. Soundness. Heuristic 1.0 is sound due to the fact that the set of solutions given by Algorithm 1 subsumes the set of solutions obtained by heuristic 1.0. This is because heuristic 1.0 has removed a set of binary variables, and the constraints in which these variables are involved.

```
Heuristic 1.1
/*Limit the stations where two trains can be crossed, that is, the number of decision variables*/
DeterminePossibleCrossing ();
LinearSolution = SolveLinearProblem ();
Crossings = DetectCrossings (LinearSolution);
n = 0;
while (Not Solution)
Solution = SearchCrossingCombination (window,Crossings);
n++;
if (Solution)
FinalSolution = SolveCrossingOrder (Solution);
```

Algorithm 2: Pseudocode of heuristic 1.1.

Completeness. Heuristic 1.0 does not remove any solution. Thus, this heuristic will find the same solution as the one obtained by Algorithm 1. Constraints of type (5) make the set the of removed constraints redundant by heuristic 1.0. By contradiction, we assume that there is a solution that heuristic 1.0 does not find. Without loss of generality, we assume a maximum slack of 20%. We can distinguish two different cases. (1) The lost solution falls into the maximum slack. This is a contradiction because, under this threshold, the restricted problem is the same as Algorithm 1. (2) The lost solution falls outside the maximum slack. This solution is not valid because it does not satisfy constraints (5.1) and (5.2). Therefore, heuristic 1.0 does not lose any solution. □

5.1.2. Heuristic 1.1

Heuristic 1.1 is a metaheuristic based on heuristic 1.0. This heuristic carries out a guided local search over the binary variables. Once many decision variables have been removed by heuristic 1.0, a new filtering process on the reduced problem can eliminate other decision variables by means of a guided local search. Instead of assigning a random station as a crossing station between two opposite trains, heuristic 1.1 performs a linearized execution where the decision variables have been transformed into continues ones. Thus, the crossing between two trains may not be assigned in stations but on a track between two stations. This will be the initial point to start the search to find the station, where the crossing will finally be performed. The search of each crossing between two opposite trains is bounded by $2n + 1$ contiguous tracks. This interval is composed by n tracks located before the obtained crossing and n tracks located after the crossing. In this way, the resultant subproblem can be seen as a combinatorial problem, where all combinations must be performed for guarantee the best possible solution. If the problem has a solution, heuristic 1.1 studies the arrival order to the crossing station such as the objective function is minimized. Otherwise, the interval is increased (n++), and the MIP problem is again solved. This heuristic is useful in anytime environment due to a solution can be found, but heuristic 1.1 tries to find a better solution in the remaining time. To this end, each combination is labeled with the solution obtained and the heuristic searches neighbor combination in order to improve the objective function. Algorithm 2 summarizes the pseudocode of heuristic 1.1.

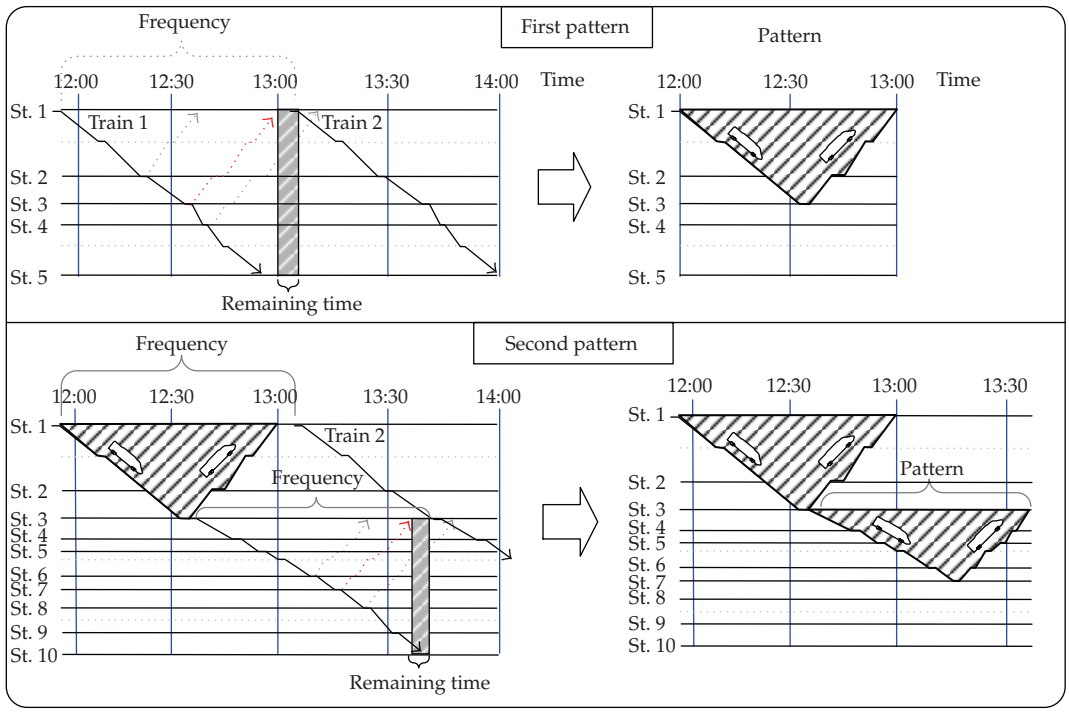


Figure 4: First traffic pattern generation.

5.2. Heuristic 2: Railway Capacity-Based Heuristic

The aim of this heuristic is to solve the railway scheduling problem by means of constraint satisfaction techniques. To this end, the original formulation (MIP problem) is separated into two different subproblems: first, the crossings are solved and then the running maps are calculated. Thus, the problem constraints are classified so that most restricted constraints are studied first [3], based on the principle presented in the introduction. Furthermore, the study of crossings will be partitioned into a set of subproblems so that the solution of each subproblem will generate a traffic pattern. The partition is carried out through the stations that take part in the running map. Each block of the partition is composed by contiguous stations so that each traffic pattern represents the running map corresponding to each block of stations.

The main idea of this heuristic is to generate a traffic pattern based on our definition of railway capacity. Each traffic pattern is generated for each set of stations such that the union of these traffic patterns determines the journey of each train. Figure 4 shows a possible set of stations (block).

The block of stations is selected taking into account the speed of the trains, the distance between stations, and overall the frequency inserted into the problem. Each traffic pattern covers the block of stations necessary for a train (Train 1) to go from the first station of the block to the last station of the block and return from the last station to the first station (round trip). The train making this round trip must arrive to the first station (St. 1) as close as possible but before, the following train departs (Train 2) (Figure 4). Thus, our objective is to minimize the remaining time between the frequency and the round trips. Each possible round trip will

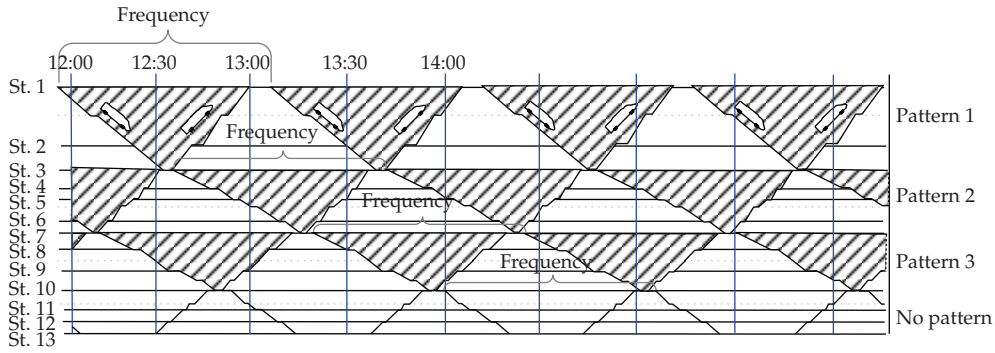


Figure 5: Periodic pattern generation.

involve a different set of constraints. The round trip that minimizes the remaining time will be selected as the *pattern*. This traffic pattern will be composed by a higher number of stations than the rest of feasible round trips.

Once the first traffic pattern has been generated, the block of stations involved in this traffic pattern is temporarily removed in order to study the following pattern with the remaining stations. Figure 4 shows the generation of the second pattern using the same strategy.

Therefore, when the second traffic pattern is generated, heuristic 2 studies the following traffic patterns until there is no station left. In Figure 5, we can observe an example of a running map with three complete traffic patterns and some stations without traffic patterns. However, it is common for there to be some stations left. These stations are not involved in any traffic pattern. We must take into account that the best traffic pattern in a block of stations implies starting the following block of stations in the last station of the previous block. We must check all the traffic patterns together in order to obtain the best journey. Moreover, the first combination of traffic patterns may not be the best solution due to the existence of some combinations of traffic patterns. This combination depends on the number of stations that are not involved in a traffic pattern.

Figure 5 shows an example in which three stations (St. 11,12,13) are not involved in any traffic pattern. Therefore, some combinations are possible, and they are restricted to the set of stations involved in the first traffic pattern. Thus, these three stations can be sorted between the first and the last traffic patterns. In this way, the first traffic pattern may start at the second station and the last traffic pattern may finish at the last station, in the second to the last station or in the third to the last station. However, due to the efficient use of resources, or depending on the importance of the station, it is more appropriate for the first traffic pattern (last traffic pattern) to start (finishes) at the first (last) station. Algorithm 3 summarizes the pseudocode of heuristic 2. There are two modules.

(i) *GenerateTheBestPatterns*. The procedure generates several combinations of traffic patterns, each one starts on a different station. It selects the combination with minor remaining time. The patterns indicate the crossing stations.

(ii) *GenerateTheRunningMap*. It generates a schedule for every possible combination in the arrival order of trains to their crossing stations and selects the schedule that minimize the journey time.

```
Generate-RunningMap (Frequency, Information of Stations, Information of Trains)
   SetOfPatterns = GenerateTheBestPatterns ();
   RunningMap = GenerateTheRunningMap (SetOfPatterns);
GenerateTheBestPatterns ();
   n = CalculateStationsOfBeginning ();
   for i = 1 to n
      RemainingTime = PatternGeneration (i);
      SelectBetterPattern ();
      i = i + 1;
GenerateTheRunningMap (SetOfPatterns)
   for each CrossingStation
      Schedule = CalculateSchedule(i);
      SelectBetterSchedule();
```

Algorithm 3: Pseudocode of heuristic 2.

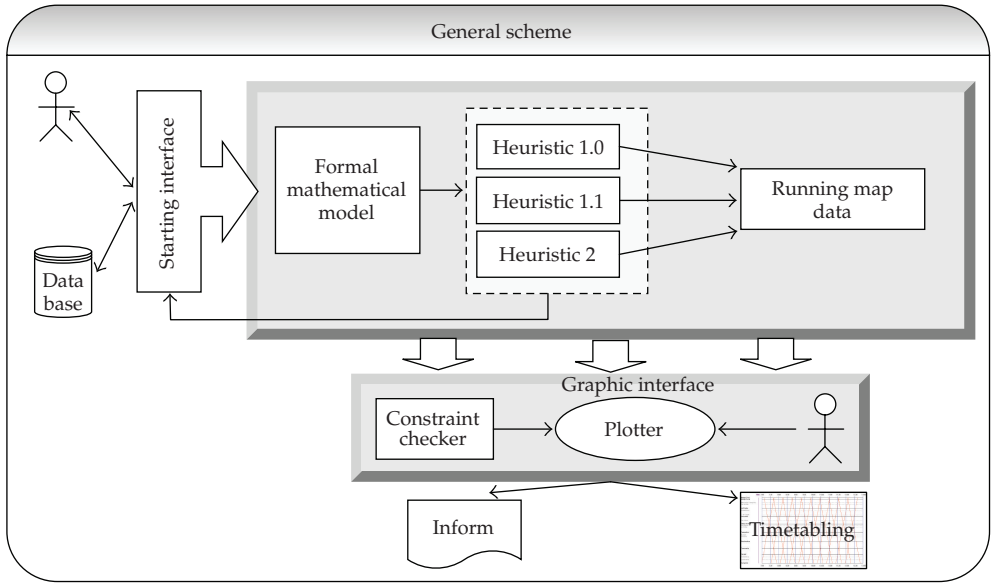


Figure 6: General scheme of our tool.

5.3. General System Architecture

The general outline of our system [26] is presented in Figure 6. It shows several steps, some of which require the direct interaction with the human user to insert requirement parameters, parameterize the constraint solver for optimization, or modify a given schedule. First of all, the user should require the parameters of the railway line and the train type from the central database (Figure 6).

This database stores the set of locations, lines, tracks, trains, and so forth. Normally, this information does not change, but authorized users may desire to change this information. With the data acquired from the database, the system generates the mathematical model.

According to the quality of the required solution and the problem size, the optimization process is carried out in the following ways.

(1) Complete. The process is performed taking into account the entire problem. This decision is carried out when the number of trains and stations is low, or the running time is not a important. In this case, heuristic 1.0 will be selected.

(2) Incremental. The process performs an incremental coordination of trains. It can be useful in anytime systems, where the number of trains and stations is not very high. In this case, heuristics 1.0 and 1.1 are the most appropriate due to the fact that as the number of combinations is checked, the quality of the solution is better.

(3) Topological. The solution is obtained by replicating a pattern found in each block of stations. It is very useful in huge problems with thousand of variables and constraints (hundred of trains and stations) and in real-time systems. Heuristic 2 solves these types of problems efficiently in a short period of time.

However, the system can also automatically recommend or select the appropriate choices depending on different parameters and the complexity of the problem.

If the mathematical model is not feasible, the user must modify the parameters, mainly the most restrictive ones. If the running map is consistent, the graphic interface plots the scheduling. Afterwards, the user can graphically interact with the scheduling to modify the arrival or departure times. Each interaction is automatically checked by the constraint checker in order to guarantee the consistency of changes. The user can finally print out the scheduling, to obtain reports with the arrival and departure times of each train in each location, or graphically observe the complete scheduling topology.

6. Evaluation

The application and performance of this system depends on several factors: railway topology (locations, distances, tracks, etc.), number and type of trains (speeds, starting and stopping times, etc.), frequency ranges, initial departure interval times, and so forth.

In this section, we compare the performance of our heuristics using some well-known branch and bound techniques included in CPLEX and LINGO solvers, because they are the most appropriate tools for solving these types of problems. Thus, each instance was evaluated by a single branch and bound algorithm (B&B), a branch and bound-based algorithm included in CPLEX filtered by heuristic 1.0 and heuristic 1.1, a branch and bound-based algorithm included in LINGO filtered by heuristic 1.1, and our topological algorithm (heuristic 2).

This empirical evaluation was carried out on a real railway infrastructure that joins two important Spanish cities ("*La Coruña*" and "*Vigo*"). The journey between these two cities is currently divided by 40 dependencies between stations (23) and halts (17).

In our empirical evaluation, each set of instances was defined by the 3-tuple $\langle n, s, f \rangle$, where n was the number of trains in each direction, s was the number of stations/halts, and f was the frequency. The problems were generated by modifying these parameters. Thus, each of the tables shown sets two of the parameters and varies the other one in order to evaluate the algorithm performance when this parameter increases. It must be taken into account that runtime of the form "$> xh$." represents that the problem did not finish in x hours, and the best solution found up to date is presented in the journey time column.

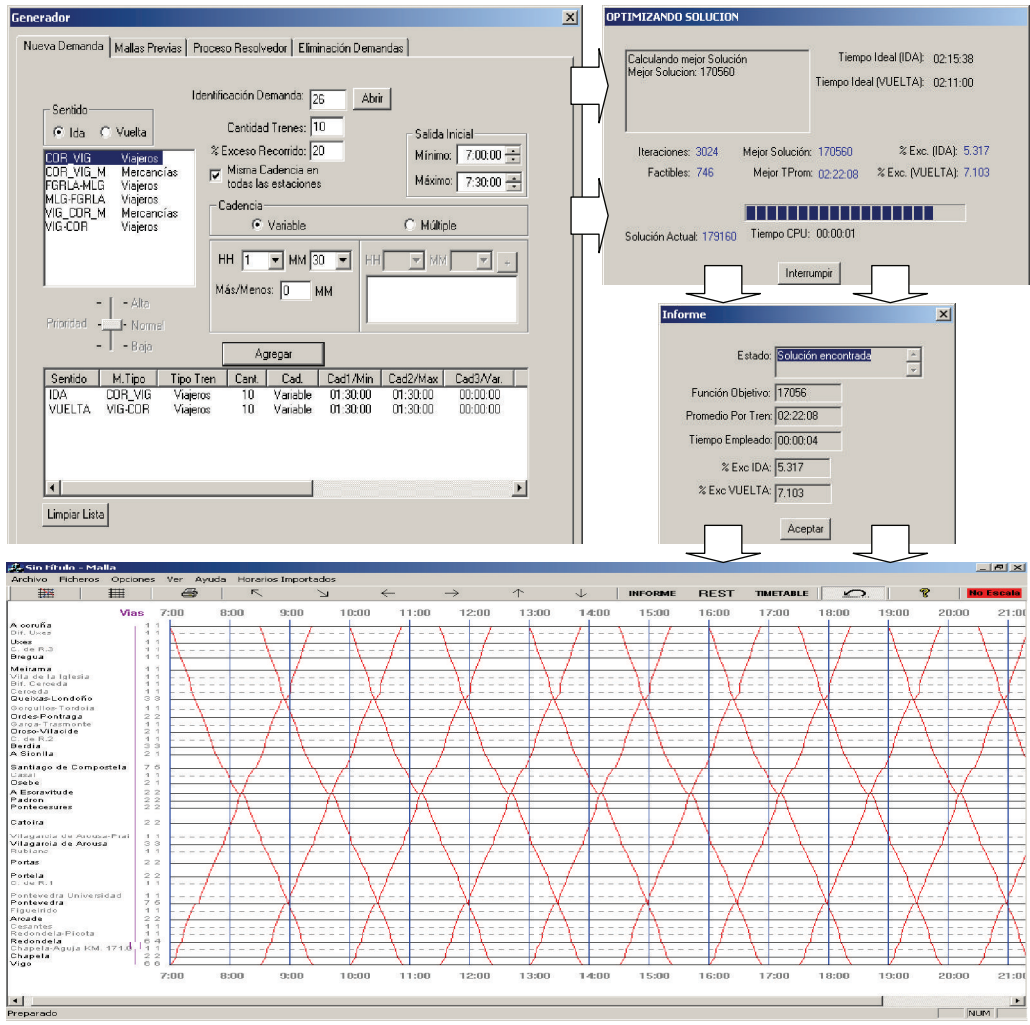


Figure 7: System interface solving and plotting instance $\langle 10, 40, 90 \rangle$.

System interface solving and plotting instance $\langle 10, 40, 90 \rangle$.

In Table 1, we present the runtime and the journey time in problems, where the number of trains was increased from 5 to 50, and the number of stations/halts and the frequency were set at 40 and 90, respectively, $\langle n, 40, 90 \rangle$. The results show that as the number of trains increased, the runtime of B&B, heuristics 1.0 and 1.1, was worse. Heuristic 1.0 obtained the optimal solution for 5,10,15 and 20 trains. However for 50 trains, heuristic 1.0 was aborted in 5 hours, and the best solution was similar than that obtained by heuristic 2. However, heuristic 2 had a lower runtime, due to this, heuristic is independent of the number of trains. Figure 7 shows the system interface executing our heuristic 2 with the instance $\langle 10, 40, 90 \rangle$. The first window shows the user parameters, the second window presents the best solution obtained at that point, the third window presents data about the best solution found, and finally the last window shows the obtained running map.

Table 1: Runtime and journey time in problems with different number of trains.

$\langle n, 40, 90\rangle$	B&B	CPLEX (B&B)				LINGO (B&B)		TOPOL.	
		Heuristic 1.0		Heuristic 1.1		Heuristic 1.1		Heuristic 2	
Trains	Run time	Run time	Journey time	Run time	Journey time	Run time	Journey time	Run time	Journey time
5	6″	5″	2:19:48	4″	2:29:33	6″	2:30:54	2″	2:21:54
10	337″	8″	2:20:19	8″	2:26:04	12″	2:31:37	3″	2:22:08
15	601″	13″	2:20:29	12″	2:26:18	19″	2:31:51	3″	2:22:08
20	1065″	16″	2:20:34	16″	2:26:25	25″	2:31:58	3″	2:22:08
50	>5 h.	55″	2:20:43	43″	2:31:09	1098″	2:32:11	7″	2:22:08

Table 2: Runtime and journey time in problems with different number of stations.

$\langle 10, s, 90\rangle$	B&B	CPLEX (B&B)				LINGO (B&B)		TOPOL.	
		Heuristic 1.0		Heuristic 1.1		Heuristic 1.1		Heuristic 2	
Stations	Run time	Run time	Journey time	Run time	Journey time	Run time	Journey time	Run time	Journey time
10	3″	2″	0:25:06	2″	0:25:06	4″	0:25:06	1″	0:25:06
20	303″	3″	1:04:11	5″	1:04:11	8″	1:04:11	2″	1:04:11
30	>1 h.	15″	1:45:38	6″	1:45:08	14″	1:45:38	3″	1:45:08
40	2131″	56″	2:20:10	56″	2:23:36	21″	2:24:36	6″	2:20:22
60	>3 h.	340″	3:33:15	217″	3:39:30	180″	3:40:30	15″	3:30:58

Table 2 shows the runtime and the journey time in problems, where the number of stations was increased from 10 to 60, and the number of trains and the frequency was set at 10 and 90, respectively, $\langle 10, s, 90\rangle$. In this case, only stations were included to analyze the behavior of the techniques. It can be observed that heuristic 2 was better than the others obtaining optimal solutions for 10,20, and 40 stations and better solutions for 30 and 60 stations. Up to 30 stations, heuristic 1.1 has better behavior than heuristic 1.0 (complete heuristic). The B&B technique maintained the worst running time. It is important to note the difference between the instance $\langle 10, 40, 90\rangle$ of the Table 1 and the instance $\langle 10, 40, 90\rangle$ in Table 2. They represent the same instance; however in Table 2, we only used stations (no halts), so the number of possible crossing between trains was much larger. This item reduced the journey time from 2:22:08 to 2:20:22, but the number of combinations increased the running time from 3″ to 6″.

In Table 3, we present the runtime and the journey time in problems, where the frequency was decreased from 140 to 60, and the number of trains and the number of stations were set at 20 and 40, respectively, $\langle 20, 40, f\rangle$. As the frequency decreased, the process solving become harder. The quality of the solutions depends mainly of the line topology. For this reason, heuristic 2 obtained better journey times with frequencies of 100 and 75 minutes, and worse journey time with other frequencies. It can be observed that depending on the problem topology, one heuristic may be better than the others. Therefore, it may be useful for the system to automatically select the appropriate heuristic.

Table 3: Runtime and journey time in problems with different frequencies.

$\langle 20, 40, f \rangle$	B&B	CPLEX (B&B)				LINGO (B&B)		TOPOL.	
		Heuristic 1.0		Heuristic 1.1		Heuristic 1.1		Heuristic 2	
Frequency	Run time	Run time	Journey time	Run time	Journey time	Run time	Journey time	Run time	Journey time
140	15″	17″	2:16:19	15″	2:20:18	24″	2:16:19	4″	2:18:35
120	156″	16″	2:16:17	14″	2:16:17	23″	2:18:47	4″	2:18:55
100	>5 h.	18″	2:22:55	15″	2:23:10	28″	2:22:55	4″	2:19:09
90	1065″	17″	2:20:34	15″	2:26:25	28″	2:31:58	4″	2:22:08
75	>1 h.	>1 h.	2:29:18	>1 h.	—	25″	2:24:16	6″	2:23:30
60	>1 h.	>1 h.	2:21:23	>1 h.	—	>1 h.	—	46″	2:32:11

7. Conclusions

We have reported the design and development of heuristics for solving periodic train scheduling, which is a project in collaboration with the National Network of Spanish Railways (RENFE), Spain. We have formulated the train scheduling as constraint optimization problems. Three heuristics are developed to speed up and direct the search towards suboptimal solutions. The feasibility of our algorithms and heuristics is confirmed with experimentation using real-life data. These heuristics has been inserted into the system to solve periodic timetables more efficiently. This system, at a current stage of integration, supposes the application of methodologies of Artificial Intelligence in a problem of great interest and will assist railway managers in optimizing the use of railway infrastructures and will also help them in the resolution of complex scheduling problems.

Notation

r: Number of stations
n: Number of trains running in the down direction
m: Number of trains running in the up direction
Freq: Frequency between departures of two consecutive trains
T_iA_k: Represents that train i arrives at station k
T_iD_k: Means that train i departs from station k
Time $i_{k-(k+1)}$: The journey time of train i to travel from station k to $k+1$
δ: Is the allowed percentage for a train to arrive at destination
TSi_k: The technical stop times of train i in station k
CSi_k: The commercial stop times of train i in station k
ET_i: The expedition time of train i
RT_i: The reception time of train i
UB: The upper bound of the travel train for this line.

Acknowledgments

This work has been partially supported by the Research Projects TIN2007-29666-E and TIN2007-67943-C02-01 (Min. de Educacion y Ciencia, Spain-FEDER), GV/2007/274 (Generalidad Valenciana), and by the Future and Emerging Technologies Unit of EC (IST priority-6th FP), under Contract no. FP6-021235-2 (project ARRIVAL).

References

[1] M. A. Salido, F. Barber, and L. Ingolotti, "Robustness in railway transportation scheduling," in *Proceedings of the 7th World Congress on Intelligent Control and Automation (WCICA '08)*, pp. 2833–2837, IEEE Press, Chongqing, China, June 2008.

[2] M. Abril, F. Barber, L. Ingolotti, M. A. Salido, P. Tormos, and A. Lova, "An assessment of railway capacity," *Transportation Research Part E*, vol. 44, no. 5, pp. 774–806, 2008.

[3] M. A. Salido and F. Barber, "A constraint ordering heuristic for scheduling problem," in *Proceedings of the 1st Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA '03)*, vol. 2, pp. 476–491, Nottingham, UK, August 2003.

[4] B. Szpigel, "Optimal train scheduling on a single track railway," in *Operational Research '72*, M. Ross, Ed., pp. 343–351, North-Holland, Amsterdam, The Netherlands, 1972.

[5] A. Higgins, E. Kozan, and L. Ferreira, "Heuristic techniques for single line train scheduling," *Journal of Heuristics*, vol. 3, no. 1, pp. 43–62, 1997.

[6] X. Cai and C. J. Goh, "A fast heuristic for the train scheduling problem," *Computers and Operations Research*, vol. 21, no. 5, pp. 499–510, 1994.

[7] P. Serafini and W. Ukovich, "A mathematical model for periodic scheduling problems," *SIAM Journal on Discrete Mathematics*, vol. 2, no. 4, pp. 550–581, 1989.

[8] K. Nachtigall and S. Voget, "A genetic algorithm approach to periodic railway synchronization," *Computers and Operations Research*, vol. 23, no. 5, pp. 453–463, 1996.

[9] M. A. Odijk, "Construction of periodic timetables, part 1: a cutting plane algorithm," Tech. Rep. 94-61, TU Delft, Delft, The Netherlands, 1994.

[10] L. Kroon and L. Peeters, "A variable trip time model for cyclic railway timetabling," *Transportation Science*, vol. 37, no. 2, pp. 198–212, 2003.

[11] C. Liebchen, *Periodic timetable optimization in public transport*, Ph.D. dissertation, Institut fur Mathematik, Berlin, Germany, 2006.

[12] C. K. Chiu, C. M. Chou, J. H. M. Lee, H. F. Leung, and Y. W. Leung, "A constraint-based interactive train rescheduling tool," *Constraints*, vol. 7, no. 2, pp. 167–198, 2002.

[13] A. H. Kaas, *Methods to calculate capacity of railways*, Ph.D. dissertation, Technical University of Denmark, Lyngby, Denmark, 1998.

[14] I. Amit and D. Goldfarb, "The timetable problem for railways," *Developments in Operations Research*, vol. 2, pp. 379–387, 1971.

[15] J.-F. Cordeau, P. Toth, and D. Vigo, "A survey of optimization models for train routing and scheduling," *Transportation Science*, vol. 32, no. 4, pp. 380–404, 1998.

[16] I. Şahin, "Railway traffic control and train scheduling based on inter-train conflict management," *Transportation Research Part B*, vol. 33, no. 7, pp. 511–534, 1999.

[17] K. Ghoseiri, F. Szidarovszky, and M. J. Asgharpour, "A multi-objective train scheduling model and solution," *Transportation Research Part B*, vol. 38, no. 10, pp. 927–952, 2004.

[18] A. Higgins, E. Kozan, and L. Ferreira, "Optimal scheduling of trains on a single line track," *Transportation Research Part B*, vol. 30, no. 2, pp. 147–161, 1996.

[19] A. Higgins, E. Kozan, and L. Ferreira, "Modelling the number and location of sidings on a single line railway," *Computers and Operations Research*, vol. 24, no. 3, pp. 209–220, 1997.

[20] M. Carey, "A model and strategy for train pathing with choice of lines, platforms, and routes," *Transportation Research Part B*, vol. 28, no. 5, pp. 333–353, 1994.

[21] A. A. Assad, "Models for rail transportation," *Transportation Research Part A*, vol. 14, no. 3, pp. 205–220, 1980.

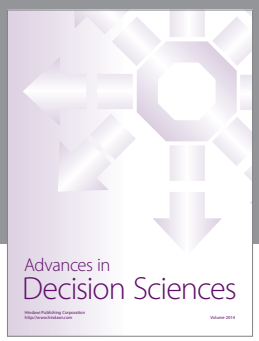
[22] A. E. Haghani, "Rail freight transportation: a review of recent optimization models for train routing and empty car distribution," *Journal of Advanced Transportation*, vol. 21, no. 2, pp. 147–172, 1987.

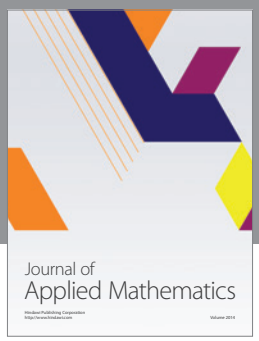
[23] M. Wardman, "The value of travel time a review of british evidence," *Journal of Transport Economics and Policy*, vol. 32, no. 3, pp. 285–316, 1998.

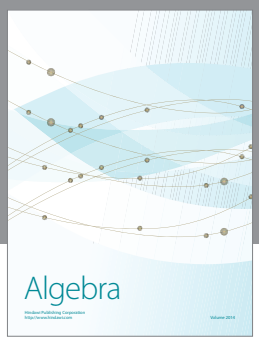
[24] H. Krueger, *Parametric Modelling in Rail Capacity Planning*, Canadian National Railway, Montreal, Canada, 1998.

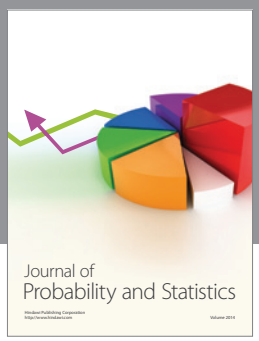
[25] E. Weits, "Railway capacity and timetable complexity," in *Proceedings of the 7th International Workshop on Project Management and Scheduling (PMS '00)*, Osnabrück, Germany, April 2000.

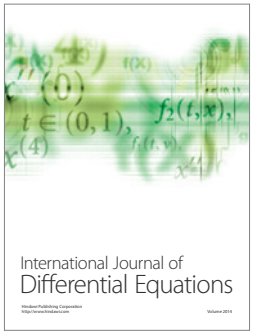
[26] F. Barber, M. A. Salido, L. P. Ingolotti, M. Abril, A. L. Lova, and M. P. Tormos, "An interactive train scheduling tool for solving and plotting running maps," in *Proceedings of the 10th Conference of the Spanish Association for Artificial Intelligence and 5th Conference on Technology Transfer (TTIA '03)*, vol. 3040 of *Lecture Notes in Artificial Intelligence*, pp. 646–655, San Sebastian, Spain, November 2004.

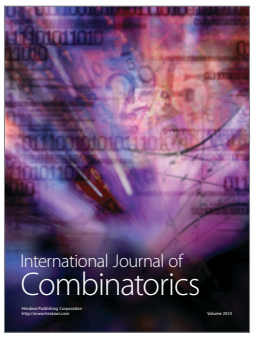


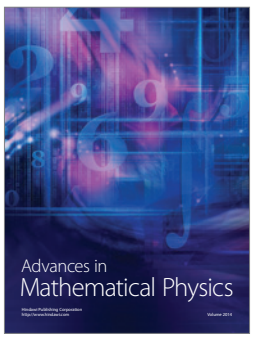


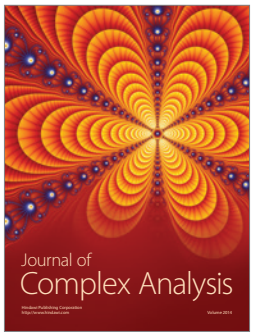


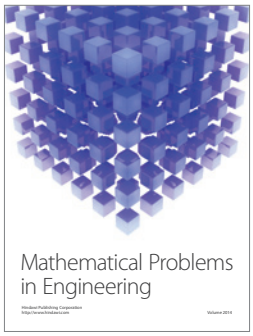


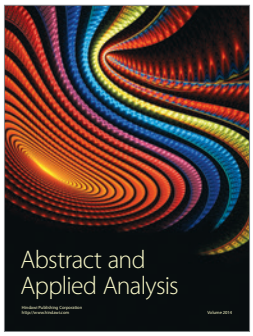


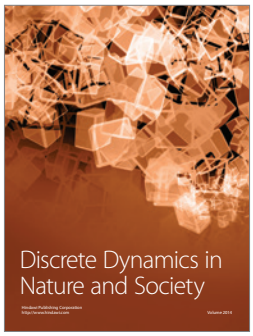


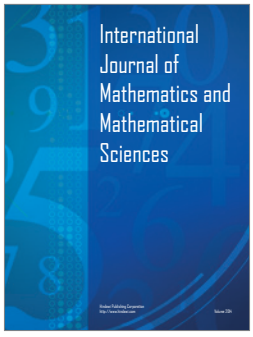


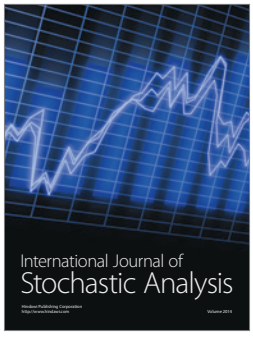












Hindawi
Submit your manuscripts at
http://www.hindawi.com