10

A Genetic Algorithm for Railway Scheduling Problems

P. Tormos[1], A. Lova[1], F. Barber[2], L. Ingolotti[2], M. Abril[2], and M.A. Salido[2]

[1] DEIOAC, Universidad Politecnica de Valencia, Spain
{ptormos, allova}@eio.upv.es
[2] DSIC, Universidad Politecnica de Valencia, Spain
{fbarber,lingolotti,mabril,msalido}@dsic.upv.es

Summary. This work is focused on the application of evolutionary algorithms to solve very complex real-world problems. For this purpose a Genetic Algorithm is designed to solve the Train Timetabling Problem. Optimizing train timetables on a single line track is known to be NP-hard with respect to the number of conflicts in the schedule. This makes it difficult to obtain good solutions to real life problems in a reasonable computational time and raises the need for good heuristic scheduling techniques. The railway scheduling problem considered in this work implies the optimization of trains on a railway line that is occupied (or not) by other trains with fixed timetables. The timetable for the new trains is obtained with a Genetic Algorithm (GA) that includes a guided process to build the initial population. The proposed GA is tested using real instances obtained from the Spanish Manager of Railway Infrastructure (ADIF). The results of the computational experience, point out that GA is an appropriate method to explore the search space of this complex problems and able to lead to good solutions in a short amount of time.

Key words: Scheduling, Train Timetabling Problem, Genetic Algorithms, Parameterized Regret-Based Biased Random Sampling, Real World Instances.

10.1 Introduction

Genetic Algorithms (GAs) have been successfully applied to combinatorial problems and are able to handle huge search spaces as those arising in real-life scheduling problems. GAs perform a multidirectional stochastic search on the complete search space that is intensified in the most promising areas.

The Train Timetabling Problem (TTP) is a difficult and time-consuming task in the case of real networks. The huge search space to explore when solving real-world instances of TTP makes GAs a suitable approach to efficiently solve it. A feasible train timetable should specify for each train the departure

P. Tormos et al.: *A Genetic Algorithm for Railway Scheduling Problems*, Studies in Computational Intelligence (SCI) **128**, 255–276 (2008)
www.springerlink.com

and arrival times to each dependency of the network in such a way that the line capacity and other operational constraints are taken into account. Traditionally, plans were generated manually and adjusted so as all constraints are met. However, the new framework of strong competition, privatization and deregulation jointly with the increasing of computer speeds are reasons that justify the need of automatic tools able to efficiently generate feasible and optimized timetables.

Assuming TTP is a very complex problem and GA a suitable procedure to cope with it, we have designed a GA for this train scheduling problem. Once the problem has been formally described, a Genetic Algorithm has been designed and validated through its application on a set of real-world problem instances provided by the Spanish Manager of Railway Infrastructure (ADIF). In addition, the heuristic technique described in this work has been embedded in a computer-aided tool that is being successfully used by ADIF.

The chapter is organized as follows. Section 10.2 is devoted to the formal description of the Train Timetabling Problem on a high-loaded railway line. The proposed method, based on a Genetic Algorithm (GA), is described in Section 10.3. Section 10.4 presents some examples and results of the application of the proposed GA to real-world railway timetabling problems. Finally, conclusions and directions for future research are pointed out in Section 10.5.

10.2 The Train Timetabling Problem (TTP)

Given a railway line that may have single as well as double-track sections, the Train Timetabling Problem (TTP) consists in computing timetables for passengers and cargo trains that satisfy the existing constraints and optimize a multicriteria objective function. The railway line may be occupied by other trains whose priority is higher than that of the new ones, and the new trains to be added may belong to different train operators. The locations to be visited by each train may also be different from each other. The timetable given to each new train must be feasible, that is, it must satisfy a given set of constraints. Among the constraints arising in this problem, it is possible the requirement for periodicity of the timetables. Periodicity leads to the classification of TTP as (*i*) Periodic (or cyclic) Train Timetabling and (*ii*) Non Periodic Train Timetabling.

In **Periodic Timetabling** each trip is operated in a periodic way. That is, each period of the timetable is the same. An advantage of a periodic railway system is the fact that such a system's timetable is easy to remember for the passengers. A drawback is that such a system is expensive to operate from the point of view of the use of resources such as rolling stock and crews. The mathematical model called Periodic Event Scheduling Problem (PESP) by Serafini and Ukovich [16] is the most widely used in the literature. In PESP a set of repetitive events is scheduled under periodic time window constraints. Hence, the events are scheduled for one cycle in such a way that the cycle can

be repeated. The PESP model has been used by Nachtigall and Voget [13], Odijk [14], Kroon and Peeters [10], Liebchen [12].

Non Periodic Train Timetabling is especially relevant on heavy-traffic, long-distance corridors where the capacity of the infrastructure is limited due to great traffic densities. This allows the Infrastructure Manager to optimally allocate the train paths requested by the Train Operators and proceed with the overall timetable design process, possibly with final local refinements and minor adjustments made by planner. Many references consider Mixed Integer Problem formulations in which the arrival and departures times are represented by continuous variables and there are binary variables expressing the order of the train departures from each station. The non periodic train timetabling problem has been considered by several authors: Szpigel [18], Javanovic and Harker [8], Cai and Goh [1], Carey and Lockwood [4], Higgins et al. [6], Silva de Oliveira [17], Kwan and Mistry [11], Caprara et al. [3], Ingolotti et al. [7].

The main problem the Spanish Manager of Infrastructure faces is the allocation of the paths requested by transport operators and the process of designing the overall timetable. These timetables are generally non periodic and have to met a wide set of constraints and achieve a multicriteria objective function. A detailed formal description of both the constraints and the objective function is given in the following subsections. First we will introduce the notation that will be used hereafter.

10.2.1 Notation

The notation used to describe the problem is the following.

Parameters:

- T: finite set of trains t considered in the problem. $\mathrm{T} = \{t_1, t_2, ..., t_k\}$
- $\mathrm{T_C} \subset \mathrm{T}$: subset of trains that are in circulation and whose timetables cannot be modified ($\mathrm{T_C}$ can be empty).
- $\mathrm{T_{new}} \subseteq \mathrm{T}$: subset of non-scheduled trains that do not have yet a timetable and that must be added to the railway line with a feasible timetable. Thus $\mathrm{T} = \mathrm{T_C} \cup \mathrm{T_{new}}$ and $\mathrm{T_C} \cap \mathrm{T_{new}} = \emptyset$
- l_i: location (station, halt, junction). The types of locations considered are described as follows:
  - 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.
  - Halt: Place for trains to stop, pass through, but not park. Each halt is associated with a unique halt identifier.
  - Junction: Place where two different tracks fork. There is no stop time.
- N_i: number of tracks in location l_i.

- NP_i: number of tracks with platform (necessary for commercial stops) in location l_i.
- $\mathrm{L} = \{l_0, l_1, ..., l_\mathrm{m}\}$: railway line that is composed by an ordered sequence of locations that may be visited by trains $t \in T$. The contiguous locations l_i and l_{i+1} are linked by a single or double track section.
- $\mathrm{J}_t = \{l_0^t, l_1^t, ..., l_{n_t}^t\}$: journey of train t. It is described by an ordered sequence of locations to be visited by a train t such that $\forall t \in \mathrm{T}, \exists \mathrm{J}_i : \mathrm{J}_t \subseteq \mathrm{L}$. The journey J_t shows the order that is used by train t to visit a given set of locations. Thus, l_i^t and $l_{n_t}^t$ represent the i_{th} and *last* location visited by train t, respectively.
- $\mathrm{T_D}$: set of trains travelling in the *down* direction.
  $t \in \mathrm{T_D} \leftrightarrow (\forall l_i^t : 0 \leq i < n_t, \exists l_j \in \{\mathrm{L} \setminus \{l_m\}\} : l_i^t = l_j \wedge l_{i+1}^t = l_{j+1})$.
- $\mathrm{T_U}$: set of trains travelling in the *up* direction.
  $t \in \mathrm{T_U} \leftrightarrow (\forall l_i^t : 0 \leq i < n_t, \exists l_j \in \{\mathrm{L} \setminus \{l_0\}\} : l_i^t = l_j \wedge l_{i+1}^t = l_{j-1})$. Thus $\mathrm{T} = \mathrm{T_D} \cup \mathrm{T_U}$ and $\mathrm{T_D} \cap \mathrm{T_U} = \emptyset$
- C_i^t minimum time required for train t to perform commercial operations (such as boarding or leaving passengers) at station i (commercial stop).
- $\Delta_{i \to (i+1)}^t$: journey time for train t from location l_i^t to $l_{(i+1)}^t$.
- $\lfloor \mathrm{I}_\mathrm{L}^t, \mathrm{I}_\mathrm{U}^t \rfloor$: interval for departure time of train $t \in \mathrm{T_{new}}$ from the initial station of its journey.
- $\lfloor \mathrm{F}_\mathrm{L}^t, \mathrm{F}_\mathrm{U}^t \rfloor$: interval for arrival time of train $t \in \mathrm{T_{new}}$ to the final station of its journey.

Variables:

- dep_i^t departure time of train $t \in T$ from the location i, where $i \in J_t \setminus \{l_{n_t}^t\}$.
- arr_i^t arrival time of train $t \in T$ to the location i, where $i \in J_t \setminus \{l_0^t\}$.

Planners usually use running maps as graphic tools to help them in the planning process. A running map is a time-space diagram like the one shown in Fig. 10.1 where several train crossings can be observed. The names of the stations are presented on the left side and the vertical line represents the number of tracks between stations (one-way or two-way). Horizontal dotted lines represent halts or junctions, while solid lines represent stations. On a railway network, the planner needs to schedule the paths of n_k trains going in one direction and m_k trains going in the opposite direction for trains of a given type. The trains to schedule can require (or not) a given frequency.

10.2.2 Feasibility of a Solution - Set of Constraints

In order to be feasible, a timetable has to met a set of constraints that can be classified in three main groups depending on whether they are concerning with: (*i*) user requirements (parameters of trains to be scheduled), (*ii*) traffic rules, (*iii*) railway infrastructure topology. The constraints described in this

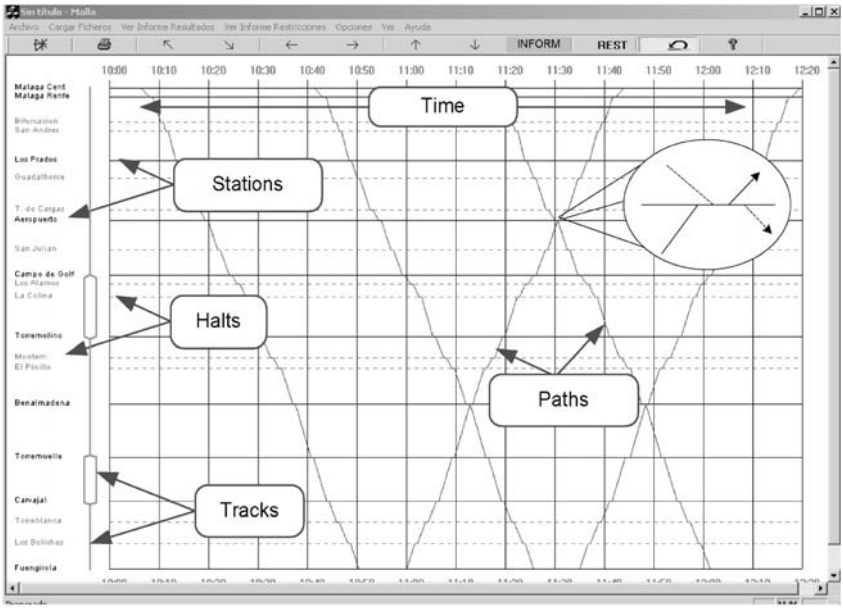


Fig. 10.1: Running Map.

work have been defined together with the Spanish Manager of Railway Infrastructure (ADIF) in such a way that the resulting timetable was feasible and practicable.

User Requirements:

- *Interval for the Initial Departure*: each train $t \in \mathrm{T_{new}}$ should leave its initial station l_0^t at a time dep_0^t such that,

$$\mathrm{I_L^t} \leq dep_0^t \leq \mathrm{I_U^t}. \tag{10.1}$$

- *Interval for the Arrival Time*: each train $t \in \mathrm{T_{new}}$ should arrive to its final station $l_{n_t}^t$ at a time $arr_{n_t}^t$ such that,

$$\mathrm{F_L^t} \leq arr_{n_t}^t \leq \mathrm{F_U^t}. \tag{10.2}$$

- *Maximum Delay*: a maximum delay Λ_t and a minimum journey time M_t are specified for each train $t \in \mathrm{T_{new}}$; thus, the upper bound for the journey time of $t \in \mathrm{T_{new}}$ is given by the following expression:

$$\frac{(arr_{n_t}^t - dep_0^t - \mathrm{M}_t)}{\mathrm{M}_t} \leq \Lambda_t. \tag{10.3}$$

Traffic constraints:

- *Journey Time*: for each train and each track section, a Journey time is given by $\Delta^t_{i\rightarrow(i+1)}$, which represents the time the train t should employ to go from location l^t_i to location l^t_{i+1}. Therefore, the following expression must be fulfilled
$$arr^t_{i+1} = dep^t_i + \Delta^t_{i\rightarrow(i+1)}. \tag{10.4}$$
- *Crossing*: according to the following expression, a single-track section ($i \rightarrow i+1$, *down* direction) cannot be occupied by two trains going in opposite directions ($t \in \mathrm{T_D}$ and $t' \in \mathrm{T_U}$).
$$dep^{t'}_{i+1} > arr^t_{i+1} \vee dep^t_i > arr^{t'}_i. \tag{10.5}$$

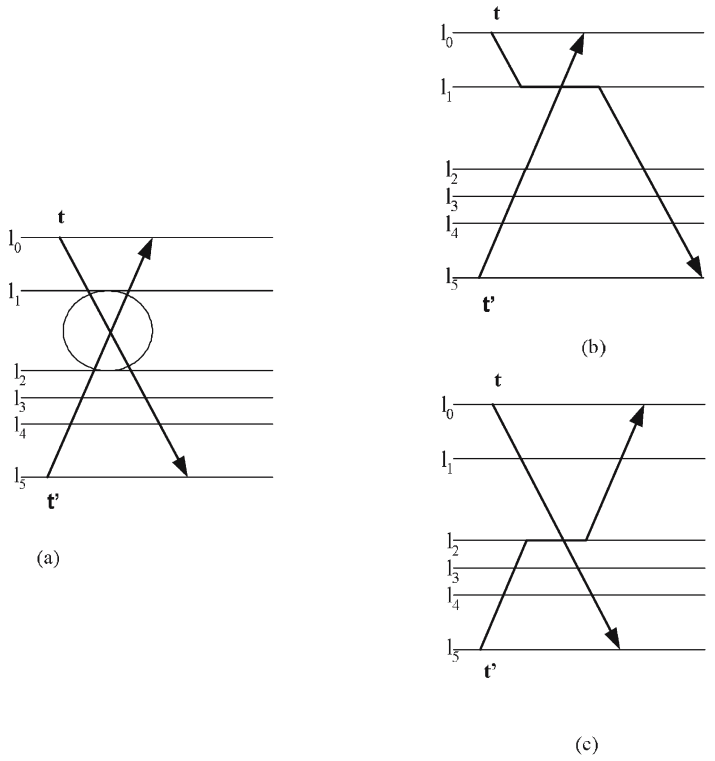


Fig. 10.2: (a) Crossing conflict. (b) Train in Down direction waits. (c) Train in Up direction waits.

- *Commercial Stop*: each train $t \in \mathrm{T_{new}}$ is required to remain in a station l^t_i at least C^t_i time units:

$$dep_i^t \geq arr_i^t + \mathrm{C}_i^t. \tag{10.6}$$

- *Overtaking on the track section*: overtaking must be avoided between any two trains, $\{t, t'\} \subseteq \mathrm{T}$, going in the same direction on any double-track sections, $k \rightarrow (k+1)$, of their journeys:

$$(arr_{k+1}^t > arr_{k+1}^{t'}) \leftrightarrow (dep_k^t > dep_k^{t'}). \tag{10.7}$$

- *Delay for unexpected stop*: when a train t stops in a station j to avoid conflicts with other trains (overtaking/crossing), and no commercial stop was planned ($C_j^t = 0$) in this station, the journey time of train t that corresponds to the previous ($l_{j-1}^t \rightarrow l_j^t$) and next ($l_j^t \rightarrow l_{j+1}^t$) track sections of j must be increased by Γ_t time units. This increase represents the speed reduction of the train due to the braking and speeding up in the station.

$$dep_j^t - arr_j^t > 0 \wedge C_j^t = 0 \rightarrow \Delta_{j-1\rightarrow j} = \Delta_{j-1\rightarrow j} + \Gamma_t \wedge \Delta_{j\rightarrow j+1} = \Delta_{j\rightarrow j+1} + \Gamma_t. \tag{10.8}$$

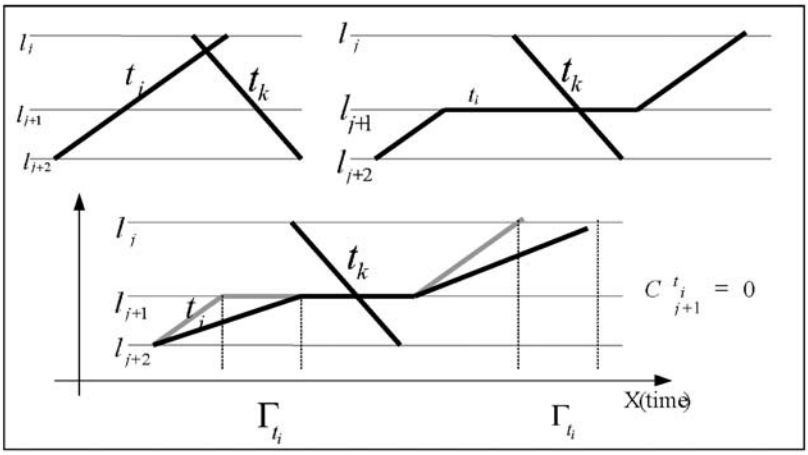


Fig. 10.3: Unexpected Stop.

- *Reception Time*: The difference between the arrival times of any two trains $\{t, t'\} \subseteq \mathrm{T} \wedge \{t, t'\} \subseteq \mathrm{T_{new}}$ in the same station l is defined by the expression below, where R_t is the reception time specified for the train that arrives to l first.

$$arr_l^{t'} \geq arr_l^t \rightarrow arr_l^{t'} - arr_l^t \geq \mathrm{R}_t. \tag{10.9}$$

- *Expedition Time*: The difference between the departure and arrival times of any two trains $\{t, t'\} \subseteq \mathrm{T} \wedge \{t, t'\} \subseteq \mathrm{T_{new}}$ in the same station l is

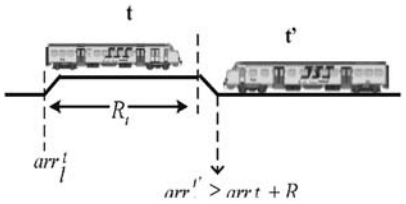


Fig. 10.4: Reception time between train t and train t'.

defined by the expression below, where E_t is the expedition time specified for t.

$$|dep_l^{t'} - arr_l^t| \geq \mathrm{E}_t. \tag{10.10}$$

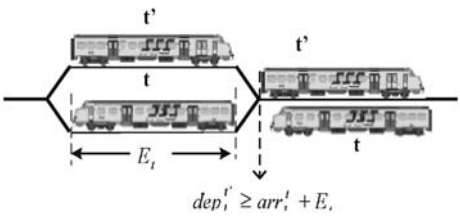


Fig. 10.5: Expedition time between train t and train t'.

- *Simultaneous Departure*: the difference between departure times from the same station of two trains going in opposite directions must be at least S, when both trains stop in the station. This constraint is formulated as:

$$\forall t, t' \in \mathrm{T_{new}} : dep_i^t - arr_i^t > 0 \wedge dep_i^{t'} - arr_i^{t'} > 0 \rightarrow |dep_i^t - dep_i^{t'}| \geq S. \tag{10.11}$$

Infrastructure constraints:

- *Finite Capacity of Stations*: a train $t \in \mathrm{T_{new}}$ could arrive to a location l_i^t if and only if it has at least one available track (with platform, if $C_i^t > 0$). In order to formulate this constraint, consider:

$$\forall x \in \mathrm{T_{new}} : T_x = \{t \in \mathrm{T} : t \neq x, J_t \cap J_x \neq \emptyset\} \text{ and}$$

$$\mathrm{Meet}(x, t, l) = \begin{cases} 1 & \text{if } [arr_l^x, dep_l^x] \cap [arr_l^t, dep_l^t] \neq \emptyset \wedge \mathrm{C}_l^t = 0 \\ 0 & \text{else} \end{cases}$$

$$\mathrm{Meet_P}(x,t,l) = \begin{cases} 1 & \text{if } [arr_l^x, dep_l^x] \cap [arr_l^t, dep_l^t] \neq \emptyset \wedge \mathrm{C}_l^t > 0 \\ 0 & \text{else} \end{cases}$$

Hence, the constraint of finite capacity of stations is formulated as follows:

$$\forall x \in \mathrm{T_{new}}, \forall l \in \mathrm{J}_x : ((\sum_{t \in \mathrm{T}_x} \mathrm{Meet}(x,t,l) + \sum_{t \in \mathrm{T}_x} \mathrm{Meet_P}(x,t,l) < \mathrm{N}_l) \wedge$$

$$(\mathrm{C}_l^x > 0 \rightarrow \sum_{t \in \mathrm{T}_x} \mathrm{Meet_P}(x,t,l) < \mathrm{N_P}_l)). \tag{10.12}$$

- *Closing Time*: Let $[\mathrm{H}_l^1, \mathrm{H}_l^2]$ be the closing time for maintenance operations of station l. The closing time imposes constraints over regular operations -trains can pass but cannot stop in the station (see the next expression). And can even forbid regular operations, trains can neither pass nor stop (i.e.: the number of tracks in the station is decreased to one (see (10.12)).

$$dep_l^t < H_l^1 \vee arr_l^t > H_l^2. \tag{10.13}$$

- *Headway Time*: If two trains, $\{t, t'\} \subseteq \mathrm{T}$, travelling in the same direction leave the same location l_k towards the location l_{k+1}, they are required to have a difference in departure times of at least φ_k^d and a difference in their arrival times of at least φ_k^a. When the blocking type in the track section is *Automatic*, then $\varphi_k^a = \varphi_k^d$. Consider the following expression

$$|dep_k^t - dep_k^{t'}| \geq \varphi_k^d. \tag{10.14}$$

$$|arr_{k+1}^t - arr_{k+1}^{t'}| \geq \varphi_k^a. \tag{10.15}$$

According to the company requirements, the method proposed should obtain the best available solution so that all the above constraints are satisfied. As we previously pointed out, the network could be previously occupied by other trains whose timetable have not been changed. That is to say $\forall t \in \mathrm{T_C}$, the variables arr_i^t and dep_i^t, have been previously instantiated with given values. This means $\forall t \in \mathrm{T_C}, \forall i \in J_t, arr_i^t \in \mathrm{CONSTANT}, dep_i^t \in \mathrm{CONSTANT}$ and the process generates the constraints so that the arrival and departure time of trains in circulation are constants, and it does not generate constraints that only involve variables corresponding to trains in circulation. Next, the process verifies that each new train satisfies each constraint taking into account the remaining new trains as well as all the trains already in circulation. In other words, if a constraint is violated and it relates new trains with trains in circulation, the only timetables that should be modified are those corresponding to new trains.

The set of constraints just described corresponds to Spanish Railway Company requirements and do not match exactly with other published works.

Several criteria can exist to assess the quality of the solution, for example: minimize travel time, minimize the passenger waiting time in the case of changeovers, balance the delay of trains in both directions, etc. The objective function considered in this work is described in the next section.

10.2.3 Optimality of a Solution - Objective function

In order to assess the quality of each solution, we obtain the *optimal solution* (optimal traversal time) for each specific train $t \in \mathrm{T_{new}}$. The optimal solution of train t is computed by the scheduling of the new train t (verifying all problem constraints) on the network being occupied only by trains in circulation ($\mathrm{T_C}$). This optimal solution for train t (Γ^t_{opt}) is the lowest time required by t to complete its journey. The other trains to be scheduled in $\mathrm{T_{new}}$ are ignored.

Once the optimal time for each new train to be scheduled has been computed, the criterion to measure the quality of each solution will be the average delay of new trains with respect to their optimum (δ). That is:

$$\delta_t = \frac{(arr^t_{n_t} - dep^t_0) - \Gamma^t_{\mathrm{opt}}}{\Gamma^t_{\mathrm{opt}}};\ \delta_{\mathrm{U}} = \frac{\sum_{t \in \mathrm{T_U} \cap \mathrm{T_{new}}} \delta_t}{|\mathrm{T_U} \cap \mathrm{T_{new}}|};\ \delta_{\mathrm{D}} = \frac{\sum_{t \in \mathrm{T_D} \cap \mathrm{T_{new}}} \delta_t}{|\mathrm{T_D} \cap \mathrm{T_{new}}|};\ \delta = \frac{\delta_{\mathrm{U}} + \delta_{\mathrm{D}}}{|\mathrm{T_{new}}|}$$

Finally, assuming TTABLE being one problem solution and therefore the timetable for all new trains, the objective function of this problem is formulated as:

$$f\,(TTABLE) = \mathrm{MIN}(\delta) \tag{10.16}$$

If there are no trains in circulation in L ($\mathrm{T} = \mathrm{T_{new}}$), the optimal time of a new train t would be:

$$M_t = \Gamma^t_{\mathrm{opt}} = \sum_{i=0}^{n_t - 1} \Delta^t_{i \rightarrow (i+1)} + \sum_{i=0}^{n_t - 1} C^t_i \tag{10.17}$$

10.3 Solving Process: A Genetic Algorithm Approach

The solving method proposed in this work for the TTP is based on a Job-Shop approach. In short, the General Job-Shop Scheduling Problem arising in many companies implies the execution of a set of orders consisting of jobs, satisfying a set of time and resource constraints. Precedence relationships of jobs of each order means time constraints and resource constraints that occur when one resource can not be simultaneously used by two jobs, that is, a manufacturing machine can not simultaneously perform two jobs. The main objective is to generate a schedule (job timetable) where each job satisfies time and resource constraints with the lowest computational effort and optimizing a measure of performance (usually makespan).

In the case of the TTP, each train can be decomposed in a set of ordered Train-track_section (T-ts) that has to verify a set of time and resource constraints: precedence relations of track sections for each train as time constraints and resource constraints when one-way track section can not be simultaneously occupied by two trains. The objective is to build a schedule (train timetable) where each Train-track_section satisfies time and resource constraints and optimizes a measure of performance with the lowest computational effort.

In this context, if limitation of resources is not taken into account, the problem is reduced to propagating the initial departure time, dep_0^t, from l_0^t until $l_{n_t}^t$ using the journey time $\Delta_{i\to(i+1)}^t$ that corresponds to each train t and each track section $l_i^t \to l_{i+1}^t$. However, in this problem resources have a finite capacity. A single line contains a set of single-track sections that cannot be occupied by more than one train at a given time, just as machines in a job-shop can process only one job at a given time. The TTP is an optimization combinatorial problem whose search space grows exponentially when the number of conflicts increases. This may be due to an increase in the number of trains, or to a decrease in the capacity of stations (number of tracks with platform if commercial stop exists), etc. It is well known that this problem is NP-Hard [2], therefore, heuristic methods are common approaches, able to obtain "good" solutions with a reasonable computational time. These heuristics have to be able to explore the large search spaces arising in this type of problems and to achieve a good solution in a short computational time. Therefore, the GA approach is suitable to cope with the TTP.

Assuming the parallelism just established between both problems, the GA developed to solve the TTP assumes each train as an order t that is split into specific jobs t_{jk} and where each job implies that train t uses one track to go from location j to location k. We represent a job by a pair (t, s_i^t), where t is the train and s_i^t is the i_{th} track section of its journey.

10.3.1 Basic Scheme of the GA

Fig. 10.6 shows the general scheme of a generic genetic algorithm. First, the initial population (P in Fig. 10.6), whose size is POP_SIZE, is generated and evaluated following a scheduling scheme that is described in both Fig. 10.8 and subsection *Initial Population*. The following steps are repeated until the terminating condition *end_cond* (execution time, number of feasible solutions or number of generations), is reached. Some individuals that compose the population P in Fig. 10.6, are modified by applying the procedures `Selection()`, `Crossover()`, and `Mutation()`. Thus, a new population P is obtained in each generation. Each iteration of Fig. 10.6 corresponds to a new generation of individuals.

```
Function Genetic_Algorithm(POP_SIZE, end_cond) As Timetabling

begin
    P=Generate_Initial_Population(POP_SIZE)
    while NOT (end_cond) do
    begin
        P=Selection(P)
        P=Crossover(P)
        P=Mutation(P)
        BEST_L=Evaluate_Population(P)
    end
    return BEST_L
end
```

Fig. 10.6: General Genetic Algorithm.

10.3.2 Definition of Individuals: Solution Encoding

In order to apply a GA to a particular problem, an internal representation for the solution space is needed. The choice of this component is one of the critical aspects for the success/failure of the GA for the problem under study. In the literature, we have found different types of representations for the solution of different scheduling problems. In this work, we have used an activity list as representation of a solution. This solution representation has been widely used in project scheduling. The solution is encoded as a precedence feasible list of pairs (t, s_i^t), that is, if (t, s_x^t) and (t, s_y^t) are the j_{th} and k_{th} gene of a chromosome of the same individual and $x < y$, then $j < k$.

The corresponding train schedule is generated applying a modified version of the Serial Schedule Generation Scheme used in Project Scheduling [9]. In the list of pairs all trains are merged in order to obtain a feasible solution. In our implementation, the new trains are scheduled following the order established by the list. Each individual in the population is represented by an array with as many positions as pairs (t, s_i^t) exist in the railway scheduling problem considered. Fig. 10.7 shows the activity list representation for a problem with N pairs (t, s_i^t).

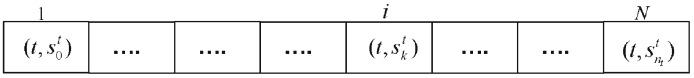


Fig. 10.7: Activity List Representation.

According to this list, (t, s_k^t) is the i_{th} pair to be scheduled. Considering that $s_k^t = l_k^t \rightarrow l_{k+1}^t$, the departure time of train t from l_k^t will be the earliest feasible time from $arr_k^t + C_k^t$. Note that when applying the de-codification process described in subsection 10.3.8, one and only one schedule can be deduced from a given sequence, but different sequences could be transformed into the same schedule.

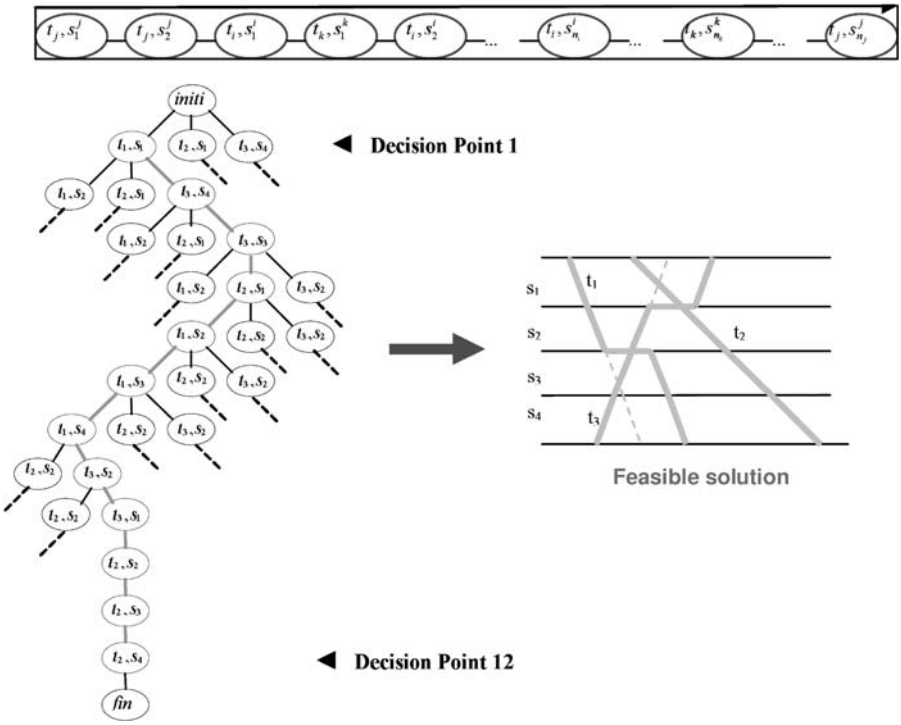


Fig. 10.8: Each activity list represents one path of the search tree.

As Fig. 10.8 shows, an activity list represents one path of the search tree. Different order of the activity list usually will imply a different solution.

10.3.3 Fitness Computation

When applying the GA, we need to define an evaluation function that determines the probability of survival of an individual to the next generation. In this chapter, the average deviation with respect to the optimal solution for the train is returned as the fitness value. In other words, an individual

χ will have the fitness value obtained from the objective function defined in Section 10.2.3.

10.3.4 Initial Population

The GA starts with the generation of an initial population, that is, a set of POP_SIZE feasible solutions. This set of feasible solutions can be obtained with different scheduling techniques. For the design of a GA, the initial population should include a variety of medium to good feasible solutions in order to increase the quality of the best solution in the evolutionary process. In this work, the value of the POP_SIZE is 50. The initial population has been obtained with an iterative heuristic based on random sampling methods and that is repeated POP_SIZE times (Fig. 10.9). N is the total number of track sections for all trains $t \in T_{NS}$. It is shown how is created the activity list by selecting a train (t) and a set of track sections $s \in J_t$ for each iteration, until all the (N) track sections have been scheduled for each new train. A solution is obtained once N iterations have been completed. Each solution gives a scheduling order, $L = (t_x, s_0^{t_x}), ..., (t_z, s_j^{t_z}), ..., (t_y, s_{n_{ty}}^{t_y})$ that represents a new individual of the initial population.

The scheduling method developed implies the search of a path in a tree as the one shown in Fig. 10.8. At each decision point, a train with track-sections unscheduled is selected. This process obtains a feasible timetable with a value of the fitness function.

The main decision in the procedure `Select_Train()` is: which train has to be scheduled at each decision point. Even though a random decision (RANDOM) could be taken selecting the train to schedule randomly, we have applied a *Regret-Based Biased Random Sampling* (RBRS) procedure that makes the selection of a train dependent on its deviation with respect to the optimal solution (optimal running time of the train calculated as indicated in Section 10.2.3). This approach guides the scheduling process in order to obtain better solutions.

The *Parameterized Regret-Based Biased Random Sampling* (RBRS) selects trains of T_{NS} through a random device. The use of a random device can be considered as a mapping $\psi : i \in T_{NS} \rightarrow [0..1]$ where a probability $\psi(i)$ of being selected (being $\sum_{i \in T_{NS}} \psi(i) = 1$) is assigned to each $t \in T_{NS}$. The regret value (ρ_i) for each train $i \in T_{NS}$ compares the priority value of train i $-\nu(i)-$ with the worst priority value $\nu(j)$ of the trains of T_{NS} and is calculated as follows:

$$\rho_i : \max_{j \in T_{NS}} (\nu_j) - (\nu_i). \tag{10.18}$$

Therefore, the parameterized probability mapping $\psi(i)$ is calculated as:

$$\psi(i) : \frac{(\rho_j + \varepsilon)^\alpha}{\sum_{j \in T_{NS}} (\rho_j + \varepsilon)^\alpha}. \tag{10.19}$$

```
Function Generate_Initial_Population(POP_SIZE) As Population

begin
    ref=Get_Low_Bound_Opt_Sol()
    i=0
    P=""
    While (i<POP_SIZE)
    begin
        L= "" //L is a new list of chromosomes
        j=0
        While (j<N)
        begin
            t=Select_Train() //using the RBRS method
            s=Select_Track_Section(t)
            d=Get_Departure(t,s)
            Set_Timetable((t,s),d)
            L=L+(t,s) //(t,s) is inserted in L
            j=j+1
        end
        Set_Fitness_To_Individual(L, ref)
        P=P+L
        i=i+1
    end
    return P
end
```

Fig. 10.9: Procedure to obtain the Initial Population.

This parameterized Regret-Based Biased Random Sampling has been widely and successfully used in project scheduling (Schirmer and Riesenberg [15], Tormos and Lova [19]). The priority value of each train is calculated according to its current delay with respect to the scheduled timetable. Trains with higher delays have more probabilities of being selected. We have implemented the procedure `Generate_Initial_Population()` using the RBRS method to select the next train to be scheduled, with $\alpha = 1$ and $\varepsilon = 0.5$.

10.3.5 Crossover

One of the unique and important aspects of the techniques involving Genetic Algorithms is the important role that recombination (traditionally, in the form of crossover operator) plays. Crossover combines the features of two parent chromosomes to form two offspring that inherit their characteristics. The individuals of the population are mated randomly and each pair undergoes the crossover operation with a probability of P_{cross}, producing two children

by crossover. The parent population is replaced by the offspring population. The crossover is one of the most important genetic operators and must be correctly designed. Crossover must combine solutions to produce new ones. Crossover must preserve and combine "good building blocks" to build better individuals [5]. Given two individuals selected for crossover, a mother **M** and a father **F**, two offspring, a daughter **D** and a son **S** are produced.

We have implemented the well-known one point crossover with $P_{cross} = 0.8$. First we draw a random crossover-point k, with k between 1 and N (number of Train-Track_Section in the problem). The first k positions in **D** are directly taken from **M**, in the same order. The rest of the activities in **D** are taken with their relative order in the father's sequence. In this way, the solution generated, the daughter, is a precedence feasible solution. Obviously, the generation of **S** is similar to the daughter's but **S** inherits the first positions directly from **F**, and the rest of the Train-Track_Section from **M**. The pseudocode for this crossover technique is shown in Fig. 10.10.

```
Function Random_Crossover_Point()
```

```
begin
    //Draw a random integer k, with 1<= k<= N
    //k is the random crossover-point
    //Generation of the daughter
    for i=1 to k do
        Di = Mi
    for i=k+1 to N do
    begin
        I = lowest index  1<= I<= N and Fi not in {D1, ..., D(i-1)}
        Di = Fi
    end
    //Generation of the son
    ........
end
```

Fig. 10.10: Crossover Procedure.

10.3.6 Mutation

Once the crossover operator has been applied and the offspring population has replaced the parent population, the mutation operator is applied to the offspring population. Mutation alters one or more genes (positions) of a selected chromosome (solution) to reintroduce lost genetic material and introduce some extra variability into the population.

The mutation operator that we have implemented works as follows: for each pair (t, s_i^t) in the sequence, a new position is randomly chosen. In order to generate only precedence feasible solutions, this new position must be higher than its predecessor and lower than its successor. The chromosome is inserted in the new position with a probability P_{mut}. In our implementation $P_{mut} = 0.05$.

10.3.7 Selection

Selection is an artificial version of the natural phenomenon called the survival of the fittest. In nature, competition among individuals for scarce resources and for mates results in the fittest individuals dominating over weaker ones. Based on their relative quality or rank, individuals receive a number of copies. A fitter individual receives a higher number of offspring and, therefore, has a higher probability of surviving in the subsequent generation. There are several ways of implementing the selection mechanism.

We have implemented *2-tournament selection*. This selection mechanism implies that two individuals are randomly chosen from the population and compete for survival. The best of them (the one with the best fitness value) will appear in the subsequent population. This procedure is repeated POP_SIZE times, until POP_SIZE individuals are selected to appear in the next population.

10.3.8 Decodification Process

In this subsection, we detail the procedure `Evaluate_Population()` of Fig. 10.6. This procedure receives a population P from which it should obtain POP_SIZE solutions. Each solution will be evaluated according to the objective function that is defined in Section 10.2.3 (`Set_Fitness_Individual()` in Fig. 10.11).

For each pair $p = (t, s_i^t) \in L$, a departure time is computed by means of the function `Get_Departure()`, which returns $d = arr_i^t + C_i^t$ if $i > 0$, otherwise $d = m$ such that m is the initial departure time given by the user.

Considering that s_i^t starts at station l_i^t and ends at station l_{i+1}^t, the procedure `Set_Timetable()` assigns a possible departure and arrival time to train t in each location between l_i^t and l_{i+1}^t, according to the journey time $(\Delta_{i\rightarrow(i+1)}^t)$ defined for this train from l_i^t to l_{i+1}^t. The next step consists in verifying whether all the constraints defined in 10.2.2 are satisfied by the timetable given for t in s_i^t. If any constraint is not satisfied, the departure time in l_i^t is increased until that constraint is satisfied. This increment in the departure time in l_i^t causes a technical stop of the train t at this station. A backtracking may occur if the station is closed for technical operations or if the station does not have enough tracks.

Once a feasible timetable has been found in this track section for train t, the same procedure is repeated with the next chromosome $(t', s_k^{t'})$.

The priority of the trains in each track section, that is, which train should be delayed if a conflict appears, is determined by the order in which each gene is numbered in the activity list. When a conflict occurs between two trains in the same track section, the priority is for the train whose timetable in this track section was assigned first. As different individuals define different priorities among the trains different solutions may be obtained.

```
Procedure Evaluate_Population(P)

begin
    i=0
    While(i<|P|) do
    begin
        L=Get_Individual(i,P)
        k=0
        while(k<|L|)
        begin
            p=Get_Chromosome(L,k)
            d=Get_Departure(p)
            Set_Timetable(p,d)
            k=k+1
        end
        i=i+1
        Set_Fitness_Individual(L)
    end
end
```

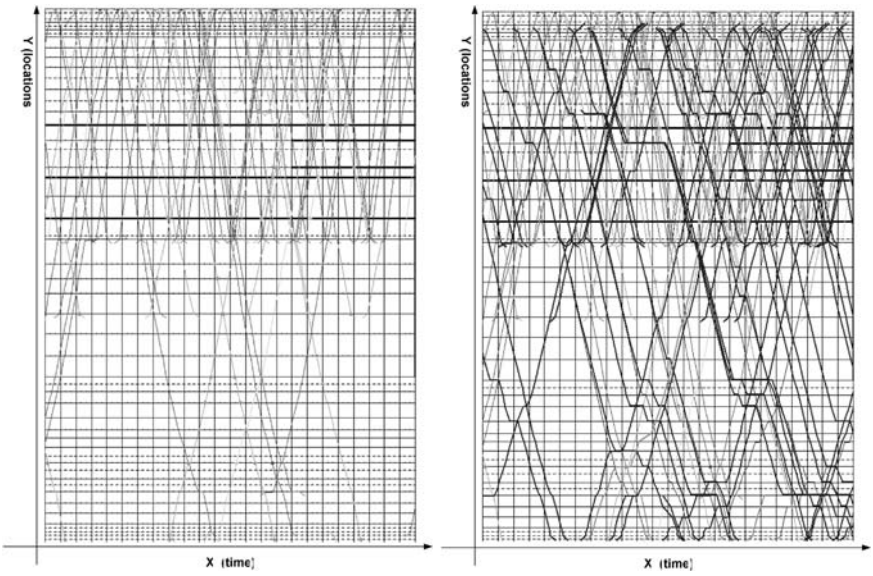
Fig. 10.11: De-codification Process.

The parameter setting of the proposed GA results from previous computational experiments.

10.4 Solving Real Cases with GA

The application of the GA described is illustrated using a real-world problem. The line considered is the railway line between Madrid and Jaen which covers 54 locations (stations, halts, siding, etc); it is 369.4 kilometers long and has 30/23 two/one-way track sections. Each horizontal line in Fig. 10.12 shows the position of each location.

The timetabling shown in Fig. 10.12a corresponds to the traffic in the line Madrid-Jaen and in the time window [12:00, 24:00] there are 103 trains in circulation, before adding the new trains. Each point of the oblique lines corresponds to the position of one train (axis Y) at a given time (axis X).



(a) Trains in Circulation (b) Solution

Fig. 10.12: Graphical Representation of a Problem Example and its Solution.

We have to add 59 new trains to the line in the time window [12:00, 24:00] satisfying the constraints, taking into account the trains in circulation and minimizing the average delay of the new trains. Usually, the planner works only assisted by a graphical tool and schedules trains in a lexicographic order (one train after another). The solution obtained is a feasible solution but not necessarily a good solution due to the multiple combinations that must be considered simultaneously. Fig. 10.12b shows the solution obtained using the GA approach in 300 seconds using a Pentium IV 3.6 Ghz processor. The value of the objective function for this solution is 7.2% (see Section 10.2.3).

10.4.1 Results

The performance of the GA developed has been tested using a set of real-world problems provided by the Spanish Manager of Railway Infrastructure (ADIF). The description of the instances is given in Table 10.1 (columns 2 to 10) by means of: length of the railway line, number of single/double track sections, number of locations and stations, number of trains and track sections (T-ts) corresponding to all these trains, considering trains in circulation and new trains, respectively.

Table 10.1: Real railway problem instances provided by ADIF.

Problems	Infrastructure Description					Trains in Circulation		New Trains	
	Km	1-Way	2-Way	Loc	Stat	Trains	T-ts	Trains	T-ts
1	96	16	0	13	13	47	1397	16	180
2	129	21	0	22	15	27	302	30	296
3	256	38	0	39	28	81	1169	16	159
4	401	37	1	39	24	0	0	35	499

Each problem has been solved using the two constructive methods used to generate the Initial Population that differs in the criterion to select the trains:

- Random selection of each train to be scheduled in each iteration (RANDOM).
- Selection of each train using the Parameterized Regret Biased Based Random Sampling method (RBRS).
- The results obtained by means of these constructive methods are compared against those achieved by the GA with the same computational time.

Table 10.2 summarizes the results for each solving method with respect to the number of solutions generated (# of Solutions) and Average Deviation with respect to the Optimal Solution (ADOS) according to expression (10.16). The tests have been carried out in a Pentium IV 3.6 Ghz processor and the running time was of 300 seconds for all the problems. $\alpha = 1$ and $\epsilon = 0.5$. The different number of solutions generated depending on the method used is because a prune procedure is applied when the RANDOM and RBRS approaches are used. That is, when a partial schedule produces a value of the objective function worse than the best value obtained at the time, then the current iteration is interrupted and the construction of a new one starts. However with the GA approach, the prune is not possible because each iteration must be completed to obtain a fitness value for the solution. This fitness value is necessary to obtain the next generation of individuals.

Table 10.2: Results of the RANDOM, RBRS and the GA scheduling methods.

Problems	RANDOM		RBRS		GA	
	# of Solutions	ADOS	# of Solutions	ADOS	# of Solutions	ADOS
1	267	18	263	15.4	255	15.1
2	611	10.1	608	10.0	313	9.6
3	424	14.7	521	14.1	382	12.4
4	405	19.2	397	17.9	285	16.0

Results of Table 10.2 show that the GA proposed outperforms both RANDOM and RBRS methods for all problem instances considered. These results demonstrate the efficiency of the GA to solve Railway Scheduling problems

against other constructive algorithms and support the idea of developing more sophisticated and powerful GAs to solve complex problems such as Train Timetabling Problem.

10.5 Conclusions

Optimizing a train schedule on a single line track is known to be NP-Hard. This makes it difficult to determine optimum solutions to real life problems in reasonable time and raises the need for good scheduling techniques. The Train Timetabling Problem considered in this work implies the optimization of new trains on a railway line that is occupied (or not) by other trains with fixed timetables. The schedule for the new trains is obtained with a Genetic Algorithm that includes a guided process to build the initial population.

The GA developed has been used to solve real-world instances and its performance has been compared against constructive approaches. The results of the computational experience, point out that GA is an appropriate method to explore the search space of this complex problems and that further research in the design of efficient GA is justified. The GA approach proposed in this work might be improved with the use of local search able to intensify performance around promising regions of local optima. An added value of the proposed GA is that its main concepts are embedded in a computer-aided tool that is being successfully used by the Spanish Manager of Railway Infrastructure.

Acknowledgments

This work has been partially supported by the research projects Future and Emerging Technologies Unit of EC (IST priority - 6th FP), under contract nr. FP6-021235-2 (project ARRIVAL), TIN2004-06354-C02-01 (Min. de Educación y Ciencia, Spain-FEDER), FOM-70022/T05 (Min. de Fomento, Spain) and GV/2007/274 (Generalidad Valenciana). We are grateful to the Spanish Manager of Railway Infrastructure (ADIF) and to Jose Estrada Guijarro for his continuous support throughout this research.

References

1. X. Cai and C. J. Goh. A fast heuristic for the train scheduling problem. *Computers and Operations Research*, 21(5):499–510, 1994.
2. A. Caprara, M. Fischetti, and P. Toth. Modeling and solving the train timetabling problem. *Operations Research*, 50:851–861, 2002.
3. A. Caprara, M. Monaci, P. Toth, and P. Guida. A lagrangian heuristic algorithm for a real -world train timetabling problem. *Discrete Applied Mathematics*, 154:738–753, 2006.

4. M. Carey and D. Lockwood. A model, algorithms and strategy for train pathing. *The Journal of the Operational Research Society*, 46(8):988–1005, 1995.
5. D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Adison Wesley, 1989.
6. A. Higgins, E. Kozan, and L. Ferreira. Heuristic techniques for single line train scheduling. *Journal of Heuristics*, 3(1):43–62, 1997.
7. L. Ingolotti, A. Lova, F. Barber, P. Tormos, M.A. Salido, and M. Abril. New heuristics to solve the csop railway timetabling problem. *Advances in Applied Artificial Intelligence. LNAI, Subseries of Lecture Notes in Computer Science*, 2006.
8. D. Jovanovic and P. T. Harker. Tactical scheduling of rail operations: The scan i system. *Transportation Science*, 25(1):46–64, 1991.
9. J. Kelley. The critical-path method: Resources planning and scheduling. *Industrial Scheduling*, 1963.
10. L. Kroon and L. Peeters. A variable time model for cycling railway timetabling. *Transportation Science*, 37(2):198–212, 2003.
11. R. K. S. Kwan and P. Mistry. A co-evolutionary algorithm for train timetabling. In IEEE Press, editor, *Congress on Evolutionary Computation*, pages 2142–2148, 2003.
12. C. Liebchen. *Periodic Timetable Optimization in Public Transport*. dissertation.de - Verlag im Internet GmbH 2006, 2006.
13. K. Nachtigall and S. Voget. A genetic algorithm approach to periodic railway synchronization. *Computers and Operations Research*, 23:453–463, 1996.
14. M. Odijk. A constraint generation algorithm for the construction of periodic railway timetables. *Transportation Research Part B*, 30(6):455–464, December 1996.
15. A. Schirmer and S Riesenberg. Parameterized heuristics for project scheduling-biased random sampling methods. Technical Report 456, Institute fr Betriebswirtschaftslehre der UNIVERSITT KIEL, 1997.
16. P. Serafini and W. Ukovich. A mathematical model for periodic scheduling problems. *SIAM J. on Discrete Mathematics*, 2:550–581, 1989.
17. E. Silva de Oliveira. *Solving Single-Track Railway Scheduling Problem Using Constraint Programming*. PhD thesis, The University of Leeds, School of Computing, September 2001.
18. B. Szpigel. Optimal train scheduling on a single track railway. In M. In Roos, editor, *Proceedings of IFORS Conference on Operational Research'72*, pages 343–352, 1973.
19. P. Tormos and A. Lova. A competitive heuristic solution technique for resource-constrained project scheduling. *Annals of Operations Research*, 102(1-4):65–81, 2001.