Intelligent Train Scheduling on a High-Loaded Railway Network

A. Lova[1], P. Tormos[1], F. Barber[2], L. Ingolotti[2], M.A. Salido[3], and M. Abril[2]

[1] DEIOAC, Universidad Politecnica de Valencia, Spain
{ptormos,alova}@eio.upv.es
[2] DSIC, Universidad Politecnica de Valencia, Spain
{fbarber,lingolotti,mabril}@dsic.upv.es
[3] DCCIA, Universidad de Alicante, Spain
masalido@dccia.ua.es

Abstract. We present an interactive application to assist planners in adding new trains on a complex railway network. It includes many trains with different characteristics, whose timetables cannot be modified because they are already in circulation. The application builds the timetable for new trains linking the available time slots to trains to be scheduled. A very flexible interface allows the user to specify the parameters of the problem. The resulting problem is formulated as a CSP and efficiently solved. The solving method carries out the search assigning values to variables in a given order verifying the satisfaction of constraints where these are involved. When a constraint is not satisfied, a guided backtracking is done. Finally, the resulting timetable is delivered to the user who can interact with it, guaranteeing the traffic constraint satisfaction.

1 Introduction

The problem considered by our application is to obtain a valid and quality scheduling for new trains on a railway network, which may or may not be occupied by other trains. The timetables for the new trains are obtained in a search space that is limited by traffic constraints, user requirements, railway infrastructure and network occupation. The system displays the resulting schedules graphically and provides an interactive interface so that the user can modify them. Planning rail traffic problems are basically optimization problems which are computationally difficult to solve. These problems belong to the NP-hard class of problems. Several models and methods have been analyzed to solve these problems [1], [3], etc. The majority of the papers published in the area of periodic timetabling in the last decade are based on the Periodic Event Scheduling Problem (PESP) introduced by Serafini and Ukovich [8]. Specifically, an efficient model that uses the PESP and the concept of symmetry is proposed in [6]. However, we cannot use the concept of symmetry because: (i) we allow different types of trains; this doesn't guarantee the necessary symmetry to be able to use these models, and (ii) the use of the infrastructure may not be symmetrical. There are works that are related to railway problems such as: the allocation of n trains in a station minimizing the number of used tracks and allowing the departure of trains in the correct order [4], the allocation of new stations through the railway network to increase the number of users [7], etc. There exist tools to solve certain kinds of problems such as the Rescheduling tool [2] or the TUFF scheduler [5]. The Rescheduling tool allows to modify a timetable when trains in a section of track cannot run according to the infrastructure, ensuring that the scheduling rules are not violated. The TUFF scheduler describes a constraint model and solver for scheduling trains on a network of single tracks used in both directions. Computer-aided systems for railways problems have become a very useful tool in assisting planners to obtain an efficient use of railway infrastructures.

2 PROBLEM DESCRIPTION

We propose to add new trains on an heterogeneous high-loaded railway network, minimizing the traversal time of each new train. The timetables for the new trains will be obtained in a search space that is limited by traffic constraints, user requirements, railway infrastructure and network occupation. The problem specification does not demand that all considered trains visit the same sequence of locations. It may there be many types of trains implying different velocities, security margins, commercial stops and journeys. Our method takes into account the following scenario to generate the timetables corresponding to the new trains:

1. two sets of ordered locations $L_D = \{l_k, l_{k+1}, ..., l_{k+m}\}$ and $L_U = \{l_h, l_{h+1}, ..., l_{h+n}\}$, such that $\{\exists i, j \backslash l_i \in L_D \wedge l_{i+1} \in L_D \wedge l_j \in L_U \wedge l_{j+1} \in L_U \wedge l_i = l_{j+1} \wedge l_{i+1} = l_j\}$. A pair of adjacent locations can be joined by a single or double track section. L_D and L_U correspond to the journey of trains going in down and up direction, respectively.
2. a set of trains for each direction. $T_D = \{t_0, t_2, ..., t_d\}$ is the set of trains that visit the locations in L_D in the same order given by this sequence and we said that these trains go in *down direction*. $T_U = \{t_1, t_3, ..., t_u\}$ is the set of trains that visit the locations in L_U in the same order given by this sequence and we said that these trains go in *up direction*. The subscript i in the variable t_i indicates the departure order among the new trains going in the same direction.
3. a journey for each set of trains (T_U and T_D) specifies the traversal time for each section of track in L_D and in L_U ($\mathrm{R}_{i \rightarrow i+1}$), and the minimum stop time (S_i) for commercial purposes in each l_i.

Considering $\mathrm{t}_y\mathrm{dep\_l}_x$ and $\mathrm{t}_y\mathrm{arriv\_l}_x$ as the departure and arrival times of train t_y from/at location l_x, the problem consists in finding the running map that minimizes the average traversal time, satisfying all the following constraints:

- *Initial departure time.* The first train must leave from the first station of its journey within a given time interval ($[min_D, max_D]$ for trains going in down direction and $[min_U, max_U]$ for trains going in up direction).

$$min_D \leq t_0 dep\_l_k \leq max_D \wedge min_U \leq t_1 dep\_l_h \leq max_U \ . \qquad (1)$$

– *Frequency of Departure.* It specifies the period (F_U/F_D) between departures of two consecutive trains in each direction from the same location,

$$\{\forall t_i, l_j \backslash t_i \in \{T_D - \{t_d\}\} \wedge l_j \in \{L - \{l_{k+m}\}\}\}, t_{i+2}dep\_l_j = t_i dep\_l_j + F_D \ . \quad (2)$$

$$\{\forall t_i, l_j \backslash t_i \in \{T_U - \{t_u\}\} \wedge l_j \in \{L - \{l_{h+n}\}\}\}, t_{i+2}dep\_l_j = t_i dep\_l_j + F_U \ . \quad (3)$$

– *Minimum stops.* A train must stay in a location l_j at least S_j time units,

$$\{\forall t_i, l_j \backslash t_i \in \{T_D \cup T_U\} \wedge l_j \in \{L_D \cup L_U - \{l_k, l_h, l_{k+m}, l_{h+n}\}\}\},$$

$$t_i dep\_l_j \geq t_i arriv\_l_j + S_j \ . \quad (4)$$

– *Exclusiveness.* A single track section must be occupied by only one train at the same time.

$$\{\forall t_j, t_i, l_x, l_y / t_j \in T_D \wedge t_i \in T_U \wedge l_x \in L_D - \{l_{k+m}\} \wedge$$
$$l_y \in L_U - \{l_h\} \wedge l_x = l_{y+1} \wedge l_{x+1} = l_y\},$$

$$t_i dep\_l_y \geq t_j arriv\_l_y \vee t_j dep\_l_x \geq t_i arriv\_l_x \ . \quad (5)$$

– *Reception Time.* At least are required R$_x$ time units at location l$_x$ between the arrival times of two trains going in the opposite direction (Figure 1).

$$\{\forall t_j, t_i, l_x / t_j \in T_D \wedge t_i \in T_U \wedge l_x \in \{L_D - \{l_k\} \cap L_U - \{l_h\}\}\},$$

$$t_j arriv\_l_x \geq t_i arriv\_l_x + R_x \vee t_i arriv\_l_x \geq t_j arriv\_l_x + R_x \ . \quad (6)$$

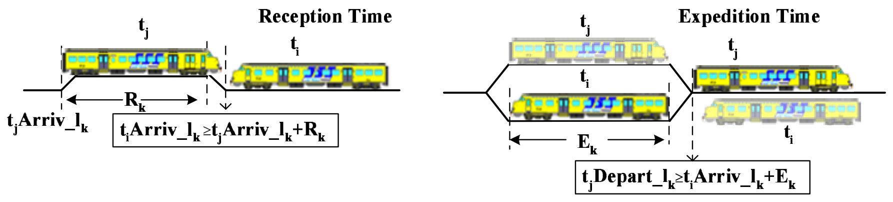


Fig. 1. Reception and Expedition time constraint

– *Expedition Time.* At least are required E_x time units at location l_x between the arrival and departure times of two trains going in the opposite direction (Figure 1).

$$\{\forall t_j, t_i, l_x / t_j \in T_D \wedge t_i \in T_U \wedge l_x \in \{L_D - \{l_{k+m}\} \cap L_U - \{l_{h+n}\}\}\},$$

$$t_j dep\_l_x \geq t_i arriv\_l_x + E_x \vee t_i dep\_l_x \geq t_j arriv\_l_x + E_x \ . \quad (7)$$

– *Precedence Constraint.* Each train employs a given time interval ($R_{x\to x+1}$) to traverse each section of track ($l_x \to l_{x+1}$)in each direction.

$$\{\forall t_i, t_j, l_x, l_y / t_i \in T_D \wedge t_j \in T_U \wedge l_x \in \{L_D - \{l_{k+m}\}\} \wedge l_y \in \{L_U - \{l_{h+n}\}\}\},$$

$$t_i arriv\_l_{x+1} = t_i dep\_l_x + R_{x\to x+1} \wedge \; t_j arriv\_l_{y+1} = t_j dep\_l_y + R_{y\to y+1}. \quad (8)$$

– *Capacity of each station.* The number of trains that may stay simultaneously in a station depends on the number of available tracks in it.
– *Closure times.* Traffic operations and/or passing of trains are not allowed at the closing times of a station.

3 Solving Method: The Sequential Algorithm

In this section we explain the used algorithm to solve the specified problem in Section 2. We have named it *"Sequential"* because of the way that it generates the timetable for each new train (Figure 2). For each iteration, the Sequential

```
01   procedure Sequential_Algorithm(T,C)
02   begin
03      while(Enough_Time())
04         S=Generate_Set_Ref_Station()
05         ref_st=Get_Ref_Station(S)
06         init_dep_down=Get_Init_Dep_Time(min$_D$,max$_D$)
07         init_dep_up=Get_Init_Dep_Time(min$_U$,max$_U$)
08         sched1=Get_Partial_Sched(init_dep_down,l$_k$,ref_st,T$_D$)
09         sched2=Get_Partial_Sched(init_dep_up,l$_h$,ref_st,T$_U$)
10         init_dep_down=t$_0$arriv_l$_{ref\_st}$+S$_{ref\_st}$
11         init_dep_up=t$_1$arriv_l$_{ref\_st}$+S$_{ref\_st}$
12         sched3=Get_Partial_Sched(init_dep_down,ref_st,l$_{k+m}$,T$_D$)
13         sched4=Get_Partial_Sched(init_dep_up,ref_st,l$_{h+n}$,T$_U$)
14         new_sched=sched1+sched2+sched3+sched4
15         if(Is_Better(new_sched,best_sched))
16            best_sched=new_sched
17      end while
18      Show(best_sched)
19   end
```

Fig. 2. Sequential Algorithm

algorithm constitutes a subset of the whole search space where it searches the values for the problem variables that satisfy all the problem constraints (Section 2). The assignment of valid values to the problem variables generates (if there is a feasible solution in the subset) a timetable for each new train (line 08-14 in Figure 2). The elements of the reduced search space depend on three values, which are:

1. *Initial departure time* for the first train going in down direction (init_dep_down in Figure 2). In each iteration is chosen randomly a value that belongs to the

time interval $[min_D, max_D]$ (Constraint 1 in Section 2). This time interval is part of the input parameters I (one of the input parameters of the Sequential Algorithm in Figure 2) given by the final user.

2. *Initial departure time* for the first train going in up direction (init_dep_up in Figure 2). In each iteration is chosen randomly a value that belongs to the time interval $[min_U, max_U]$ (Constraint 1 in Section 2). This time interval is part of the input parameters I (one of the input parameters of the Sequential Algorithm in Figure 2) given by the final user.
3. *Reference Station* (ref_st in Figure 2). When the assigned value to a problem variable, that represents the departure time of a train, violates the Constraint 5 defined in Section 2 (avoid crossing between trains going in opposite direction), the process must decide which of the two trains will have to wait for the section track release. The decision taken by the process will state a priority order between the trains competing by the same resource, the single track section.
When the two trains competing by a single track section are: a train in circulation and a new train, that should be added to the railway network, the priority order always will be the same. The new train will have to wait until the train in circulation releases the single track section.
When the two trains competing by a single track section are new trains, the priority order is decided according to the position of the single track section with respect to one station, which we name the *reference station*. The *reference station* divides the journey of each new train in two parts. The first part: from the initial station of the journey, to the *reference station*. The second part: from the *reference station*, to the last station of the journey. Each train will have priority on the single track sections that belong to the first part of its journey. In Figure 3, S_2 is the reference station and it divides

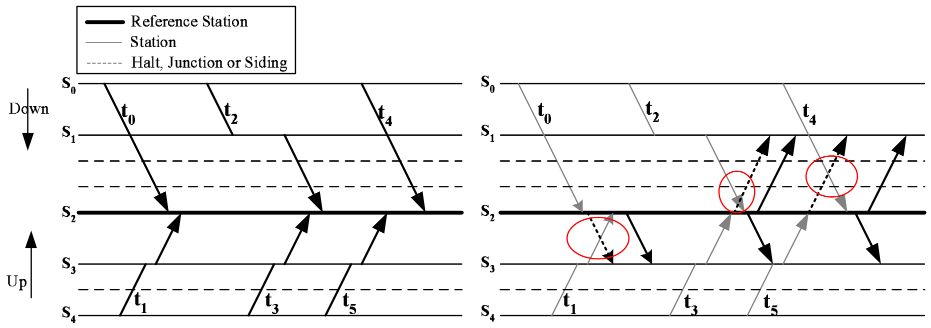


Fig. 3. Priority Order between new trains, defined by a Reference Station

the journey of each new train in two parts. For the trains going in down direction (t_0, t_2 and t_4) the first part of theirs journey is composed of the track sections (S_0-S_1) and (S_1-S_2), the second part is composed of (S_2-S_3)

and (S_3-S_4). For the trains going in up direction (t_1, t_3 and t_5) the first part of theirs journey is composed of (S_4-S_3) and (S_3-S_2), the second part is composed of (S_2-S_1) and (S_1-S_0). In Figure 3 is pointed out by a circle the possible crossing between the trains t_0 and t_1 in case that t_0 left from S_2 as soon as possible. The single track section (S_2-S_3) belongs to the first part of t_1 journey and therefore this train has greater priority than t_0 on this track section. Then, t_0 will have to wait until the single track section (S_2-S_3) is released by the train t_1. The pointed lines represent the position of the train t_0 if it had left from the station S_2 as soon as possible. The continued lines represent the real departure time of the train t_0 after the single track section had been released by t_1. The same reasoning is applied by the other cases pointed out by circles in Figure 3.

The sequential algorithm iterates until the time given by the user has been spent completely or until the user interrupts the execution (line 03 in Figure 2). For each iteration the sequential algorithm compares the obtained scheduling with the best timetable obtained until that time. The scheduling that produces the least average traversal time for each new train is the best scheduling. The function $Is\_Better(new\_timetable, best\_timetable)$ returns TRUE if the new scheduling (*new_sched* in Figure 2), obtained in the current iteration, is better than the best scheduling (*best_sched* in Figure 2), obtained until that time. Finally, the sequential algorithm returns the best scheduling that has been obtained during the running time.

3.1 Description of an Iteration of the Sequential Algorithm

In each iteration, the sequential algorithm generates a complete scheduling for each group of trains (T_D and T_U) in two steps. In Figure 2, *sched1* and *sched2* correspond to the generated scheduling for the first part of the new trains journey going in down and up direction respectively. In Figure 2, *sched3* and *sched4* correspond to the generated scheduling for the second part of the new trains journey going in down and up direction respectively. The complete scheduling is generated in this way in order to state greater priority to each new train on the first part of its journey. In case of existing a crossing possibility in a track section, the greater priority will be given to the new train whose timetable had been assigned previously on that section track. The first and second part of a journey are stated by the chosen reference station at the current iteration. Figure 4 shows how is generated the scheduling for one part of the whole journey. $Verify\_Constraints(st, next\_st, dep\_time, t_i)$ verifies that all problem constraints are satisfied by the train t_i in the track section limited by the stations *st* and *next_st*. This function returns the time that must be added to the departure time of t_i in order to satisfy the violated constraint. If the function returns 0, then none constraint has been violated. Considering the set $L' = \{l_x, l_{x+1}, ..., l_{x+p}\}$ as the ordered set of locations visited by the train t_i, from $l_x = st$ to $l_{x+p} = next\_st$. This function assigns values to the variables $ti\_dep\_l_j$ and $ti\_arriv\_l_h$ such that $x <= j < x + p$ and $x < h <= x + p$ (Figure 5). Figure 6 shows an example

```
procedure Get_Partial_Sched(init_dep,first_st,last_st,T)
begin
   st=first_st
   dep_time=init_dep
   while(st != last_st)
      next_st=Get_Next_St(st)
      i=Get_First_Train(T)
      while($t_i$!=NULL)
         error=Verify_Constraints(st,next_st,dep_time,$t_i$)
         if(error>0)
            if(Is_Required_Frequency())
               i=Get_First_Train(T)
               $t_i$dep_l$_{st}$=$t_i$dep_l$_{st}$+error
            end if
         else
            if(Is_Required_Frequency())
               dep_time=$t_i$dep_l$_{st}$+$F_T$
            else
               dep_time=$t_{i+2}$arriv_l$_{st}$+$S_T$
            end if
            i=i+2
         end if
      end while
      st=next_st
   end while
end
```

Fig. 4. Partial Scheduling

```
int function Verify_Constraints(st,next_st,dep_time,$t_i$)
begin
   k=st
   m=next_st
   trav_time=Get_Traversal_Time(st,next_st)
   $t_i$dep_l$_k$=dep_time
   $t_i$arriv_l$_m$=dep_time+trav_time
   error=Verify_Crossing($t_i$dep_l$_k$,$t_i$arriv_l$_h$)
   if(error=0)
      error=Verify_Overtaking($t_i$dep_l$_k$,$t_i$arriv_l$_h$)
      if(error=0)
         error=Verify_Availability_Tracks($t_i$arriv_l$_h$)
         if(error=0)
            error=Verify_Closure_Time($t_i$arriv_l$_h$)
            if(error=0)
               Set_Timetable(st,next_st)
         end if
      end if
   end if
   return error
end
```

Fig. 5. Constraints Verification and Timetable Assignment

of how is verified the constraint that avoids a crossing between trains going in opposite directions. Consider that train t_2 is the train t_i, the train whose timetable is being created, and t' is the train whose timetable has been created previously and cannot be modified. The initial value assigned as departure time from st=l_x to t_2 causes a crossing of t' with t_2 according to the picture on the left of Figure 6. The function $Verify\_Constraints$ computes the time necessary that must be added to dep_time in order to avoid this crossing and this value is returned by the function. The picture on the right of Figure 6 shows the necessary delay in the departure time of t_2 from S_2.

The precedence constraint (Constraint 8 in Section 2) is used to propagate the values among the variables($trav\_time$ in Figure 5 is the spent time to go from *st* to *next_st*). Minimum stops constraint(Constraint 4) is used to compute the departure time of a train from its arrival time at the same location(S_T in Figure 4). $Get\_Next\_Station(st, T)$ returns the next station to *st* in the journey corresponding to trains belonging to T. $Get\_First\_Train(T)$ returns the first train that starts the journey corresponding to trains belonging to T. $Is\_Required\_Frequency(st)$ returns TRUE if all the trains must keep the same departure frequency in the station *st*.

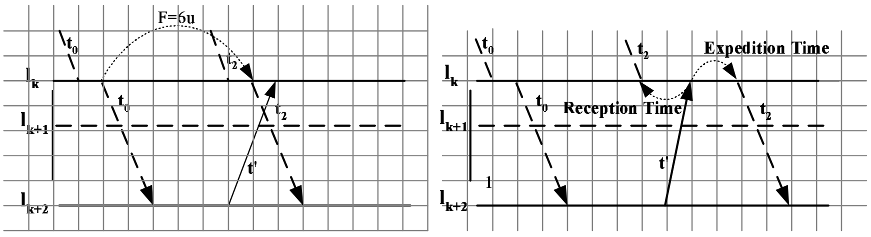


Fig. 6. A crossing detected in a section of track

This solving procedure is the core of a system to assist rail operators in the decision making process. The general architecture of that system is described in the following section.

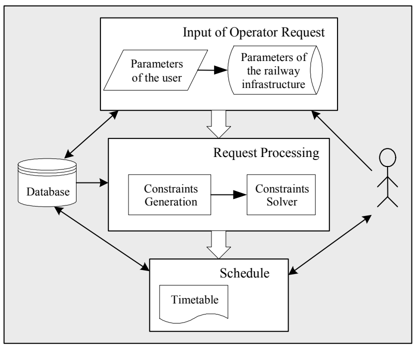
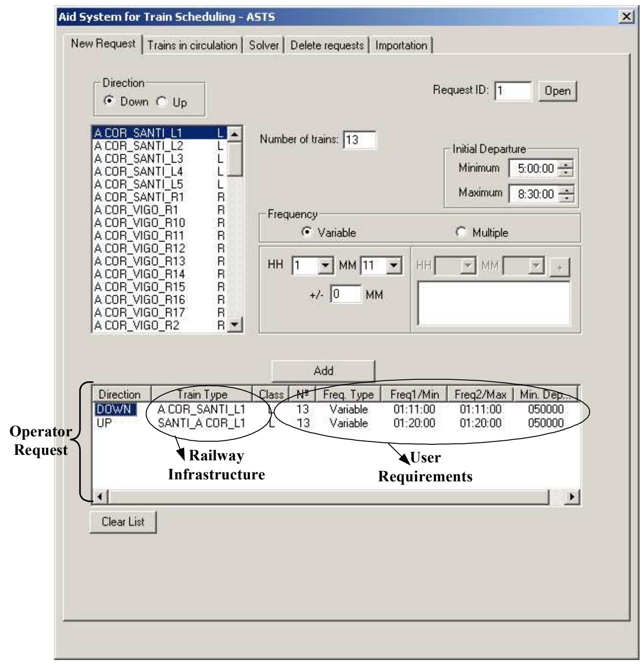


Fig. 7. General System Architecture

4 GENERAL SYSTEM ARCHITECTURE

We have developed an ”Aid System for Train Scheduling” (ASTS) to assist rail operators in the planning and use of railways. ASTS is an interactive application that is implemented in C++ . It connects to a database to store requests and solutions and to keep all information that is related to railway infrastructure, journeys, type of trains, stations, etc. The main modules of our system are described in Figure 7. Following, we explain them in more detail.



Direction: [✓ Down] Up

Request ID: 1

A COR_SANTI_L1	L
A COR_SANTI_L2	L
A COR_SANTI_L3	L
A COR_SANTI_L4	L
A COR_SANTI_L5	L
A COR_SANTI_R1	R
A COR_VIGO_R1	R
A COR_VIGO_R10	R
A COR_VIGO_R11	R
A COR_VIGO_R12	R
A COR_VIGO_R13	R
A COR_VIGO_R14	R
A COR_VIGO_R15	R
A COR_VIGO_R16	R
A COR_VIGO_R17	R
A COR_VIGO_R2	R

Number of trains: 13

Initial Departure
Minimum 5:00:00
Maximum 8:30:00

Frequency: [✓ Variable] Multiple

HH 1 MM 11
+/- 0 MM
HH MM

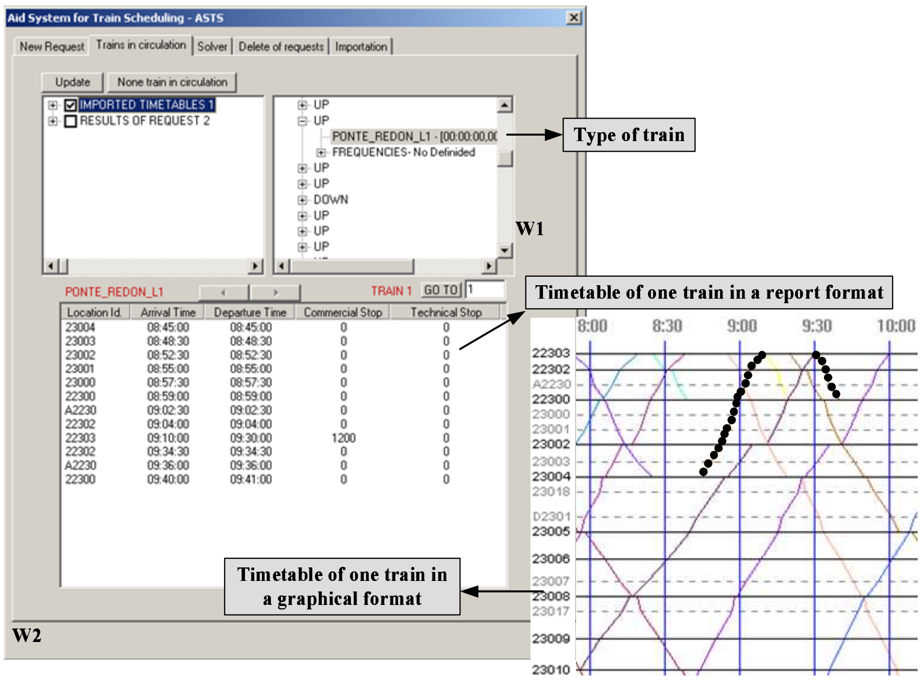
Direction	Train Type	Class	Nº	Freq. Type	Freq1/Min	Freq2/Max	Min. Dep...
DOWN	A COR_SANTI_L1	L	13	Variable	01:11:00	01:11:00	050000
UP	SANTI_A COR_L1	L	13	Variable	01:20:00	01:20:00	050000

Fig. 8. Interface to specify a scheduling request for new trains.

4.1 Input of operator request

The system returns a valid scheduling for each operator requirement. The main input parameters are: network occupation, number and type of trains, interval of allowed frequencies and interval of allowed initial departure times. Figure 8 and Figure 9 show an example where an operator specifies a request to allocate new trains on a railway network. In the example, the operator needs new trains of the type ”A COR_SANTI_L1” for trains going in the down direction and ”SANTI_A COR_L1” for trains going in the up direction. For each type, the operator specifies a frequency (01:11:00 and 01:20:00), a number of trains (13 and 13) and a time interval to start the corresponding journeys ([05:00:00-08:30] and [05:00:00-08:15:00]).

The user uses the interface shown in Figure 9 to specify the network occupation; that is, which trains are in circulation and must be taken into account during the solving process. Each train in circulation may belong to a different type of train. In Figure 9, window W1 shows all the types of trains that are in circulation, and window W2 shows the timetable corresponding to each one of them. ASTS displays the timetable of each train in a report or in a graphical form. An example is given in Figure 9, where the timetable corresponding to "*TRAIN 1*" is displayed both as a report (W2) and in graphical form (line highlighted with black circles). In the graphical form, the horizontal axis represents the time and the vertical axis represents the positions of each train. Thus, each oblique line represents the timetable of one train. The thickest lines represent the timetables for the new trains, and the rest of the lines correspond to timetables of trains in circulation.



Location Id.	Arrival Time	Departure Time	Commercial Stop	Technical Stop
23004	08:45:00	08:45:00	0	0
23003	08:48:30	08:48:30	0	0
23002	08:52:30	08:52:30	0	0
23001	08:55:00	08:55:00	0	0
23000	08:57:30	08:57:30	0	0
22300	08:59:00	08:59:00	0	0
A2230	09:02:30	09:02:30	0	0
22302	09:04:00	09:04:00	0	0
22303	09:10:00	09:30:00	1200	0
22302	09:34:30	09:34:30	0	0
A2230	09:36:00	09:36:00	0	0
22300	09:40:00	09:41:00	0	0

Fig. 9. An interface to specify the network occupation.

4.2 Request Processing

As mentioned in section 3, the timetable of one train depends mainly on three parameters: departure time, frequency between consecutive trains and priority assignment. For instance, Figure 10 shows how the scheduling changes when the departure time is modified. The frequency is kept constant in the two configurations. The complexity of the problem is increased (NP-complete to NP- hard) with the requirement of obtaining a quality timetable. It is not enough for the final user to obtain a correct timetable. This requirement implies finding the best combination of the three cited parameters.

The system performs the search within the time given by the operator. When this time ends, it stores to a database the best scheduling obtained according to the quality criteria given by the operator.

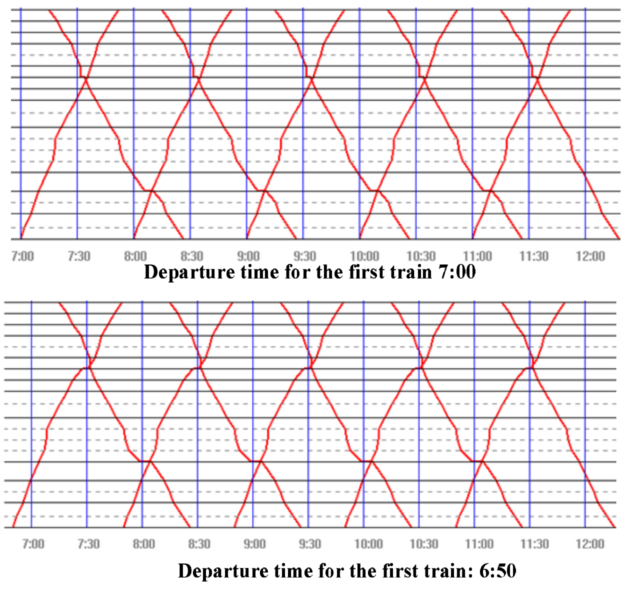
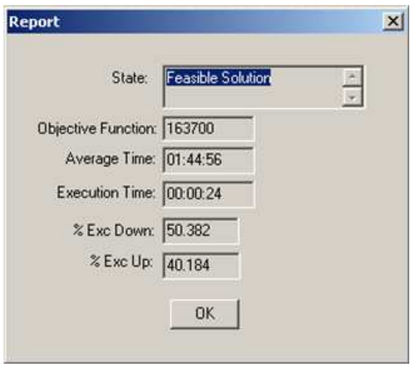


Fig. 10. Three different scheduling for three different departure times for the same group of trains.

4.3 Scheduling delivery

Once the running time given by the operator is completed, the system interrupts its process and stores the best scheduling obtained in this time interval in a database. The operator is sent a report about the results obtained (Figure 11). Later, at any time, the operator can retrieve the resulting timetables from the database in a graphical form (Figure 12). The timetables can be modified manually by the operator. These modifications are checked by the system and only those that do not violate any constraint are allowed. The changes made by the user to timetables can be stored in the database.


Report

State: Feasible Solution

Objective Function: 163700

Average Time: 01:44:56

Execution Time: 00:00:24

% Exc Down: 50.382

% Exc Up: 40.184

Fig. 11. Summary about the resulting schedule.

The resulting scheduling for the new trains is shown in Figure 12. The thickest lines represent the new scheduling. The rest of the lines correspond to timetables of other trains, which in turn, represent the occupation of the railway network, on which the new trains had to be allocated.

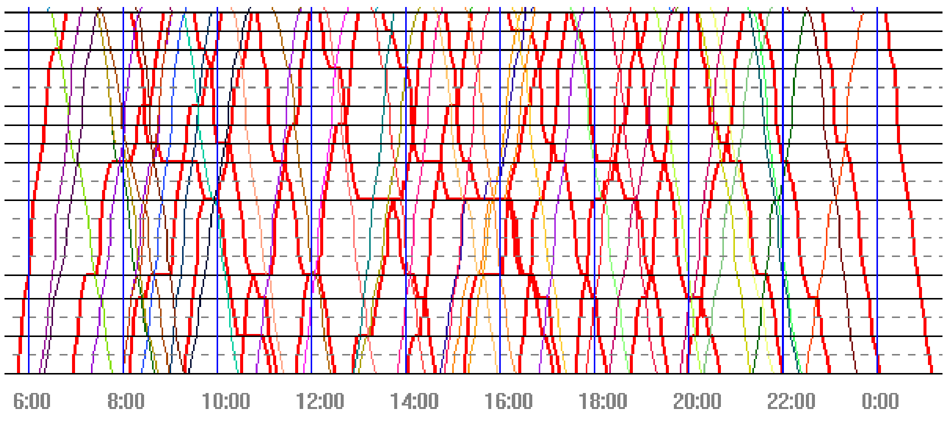


Fig. 12. A graphical example of a resulting schedule.

5 RESULTS

Several parameters may determine the quality of a scheduling for new trains in a railway company. Our system considers as quality criterion:

– Average traversal time for each new train.

In addition to this criterion other quality parameters are provided to the user:

– Number of conflicts managed among new trains as well as between new trains and those in circulation. Solving conflicts implies technical stops and then a less attractive timetable from the customer point of view.
– Divergence, defined as the difference between the average delay (meassured as a percentage) of trains in down direction and the average delay of trains in up direction. Higher divergence could imply worse schedules.

Table 1. Input parameters.

Dir	New Trains	Locat.	Train in Circulation	Freq.	Departure Time Interval
Down	13	20	Yes	[01:10:00-01:20:00]	[05:00:00-08:30:00]
Up	13	20	Yes	[02:00:00-2:20:00]	[05:30:00 08:15:00]

Another factor that affects the quality of the schedules obtained by our system is the running time that the operator gives to solve the problem. Considering the input parameters of Table 1, the system provides three different outputs for an execution time of 5, 10 and 40 seconds respectively (Table 2)

Table 2. Three different outputs.

Execution Time(sec)	No of Conflicts	Divergence	Average Traversal Time
5	Average 3	54%-40%	01:51:00
10	Average 3	53%-38%	01:47:27
40	Average 2	50%-40%	01:45:23

6 CONCLUSIONS

Our system assigns to each train a departure and arrival time for each location of its journey, taking into account the operator requirements and the described traffic constraints. The system accepts a wide range of scenarios on which it is possible allocate new trains correctly and efficiently. One of these scenarios can be a high-loaded railway network with different types of trains. The system responds with different timetable configurations depending on the operator request and the execution time. The operator can make decisions easily and quickly, modifying the input parameters by comparing the different scheduling obtained. This comparison is very complex and very time consuming to do manually in a complex network. As all optimization processes for NP-hard problems, the heuristics can be improved in order to obtain better solutions in less time. Currently, ASTS is in an evaluation phase. We are considering adding other quality criteria in order to increase the scope. We also want to integrate ASTS with other tools proper of the railway company.

References

1. M.R. Bussieck, T. Winter, and U.T. Zimmermann, 'Discrete optimization in public rail transport', *Math. Programming*, **79(1-3)**, 415–444, (1997).
2. C. K. Chiu, C.M. Chou, J.H.M. Lee, H.F. Leung, and Y.W. Leung, 'A constraint based interactive train rescheduling took', *Constraints*, **7**, 167–198, (2002).
3. J.F. Cordeau, P. Toth, and D. Vigo, 'A survey of optimization models for train routing and scheduling', *Transportation Science*, **32(4)**, 380–404, (1998).
4. L. Koci and G. Di Stefano, 'A graph theoretical approach to shunting problems', *Proceedings of ATMOS 2003 Algorithmic Methods and Models for Optimization of Railways, Electronic Notes in Theoretical Computer Science,Elsevier 92*, **1**, (2003).
5. P. Kreuger, J.O. Carlsson, T. Sjoland, and E. Astrom, 'The tuff train scheduler', *German Puebla, editor, Proceedings of the workshop on Tools and Environments for (Constraints) Logic Programming at the International Logic Programming Symposium ILPS'97*, (1997).
6. C. Liebchen, 'Symmetry for periodic railway timetables', *Proceedings of ATMOS 2003 Algorithmic Methods and Models for Optimization of Railways, Electronic Notes in Theoretical Computer Science 92*, **1**, (2003).
7. M.F. Mammana, S. Mecke, and D. Wagner, 'The station location problem on two intersecting lines', *Proceedings of ATMOS 2003 Algorithmic Methods and Models for Optimization of Railways, Electronic Notes in Theoretical Computer Science,Elsevier 92*, **1**, (2003).
8. P. Serafini and W. Ukovich, 'A mathematical model for periodic scheduling problems', *SIAM Journal on Discrete Mathematics*, **2(4)**, 550–581, (1989).