

Railway infrastructure saturation using Constraint Programming approach

J. Rodriguez¹, X. Delorme^{1,2}, X. Gandibleux²

¹*INRETS (Institut National de Recherche sur les Transports et leur Sécurité), Villeneuve d'Ascq, France.*

²*Université de Valenciennes et du Hainaut-Cambrésis, Valenciennes, France.*

Abstract

Railway infrastructure managers now have to deal with the new operators' requests for capacity. Planning the new construction or reconstruction of infrastructures must be carried out very carefully due to the huge investment needed. This paper studies a Constraint Programming (CP) model of the 'Railway Infrastructure Saturation Problem' (RISP). The model applies to assess the capacity of a junction or a station. Two resolution algorithms have been developed and combined to be applied to a real case study.

1 Introduction

Usually, assessing the capacity of a component of a rail system is carried out by measuring the maximum number of trains that can be operated on it within a certain unit of time. For measuring the capacity of lines, analytical models can be applied. The theoretical expression of the capacity of a railway line in a given direction, noted C , can be defined as :

$$C = \frac{u}{h} \quad (1)$$

where h is the minimum headway time between two successive trains and u is the considered unit of time. The minimum headway time depends on the signaling system installed on the line considered. Expressions which are more accurate can be

used to include more features of the rail system (see [9]). In the case of junctions the analytical models alone can't be applied. In [2] an analytical method is proposed for junctions and stations, but the authors concluded on the need of search tools to find good or optimal solutions of a problem. The measuring of the capacity of junctions is therefore a solution for an optimisation problem [4] called the "Railway Infrastructure Saturation Problem" (RISP). The Railway Infrastructure Saturation Problem of a junction can be stated as follows:

Given the layout of a junction and a set of trains $\mathcal{T}$, how many trains of $\mathcal{T}$ can be routed through the junction within a certain unit of time such, that all safety constraints are satisfied ?

Each train of the set $\mathcal{T}$ has an allowed set of routes and an allowed set of entering times. The set $\mathcal{T}$ can be constructed from the measuring of the capacity of each converging line, these capacities are determined by usual analytical models [9]. In this paper we present how this problem can be tackled within a constraint programming approach.

The paper is organised as follows : the next section introduces the RISP definition and the notations used. The CP model of this problem is presented in section 3. Section 4 details two resolution methods. Section 5 gives the results we obtained. Finally, some discussions and issues are provided in section 6.

2 Problem definition

To define more formally the Railway Infrastructure Saturation Problem, we need to introduce the following notations.

Let $\mathcal{T}$ be the set of trains considered.

Let $\mathcal{R}$ be the set of routes used by trains running on the junction considered.

The function $fr : \mathcal{T} \rightarrow \mathcal{P}(\mathcal{R})$ gives for each train the feasible routes.

The function $fst : \mathcal{T} \rightarrow \mathcal{P}(\mathbb{N})$ gives for each train the feasible start time values.

The function $ra : \mathcal{T} \rightarrow \mathcal{R}$ defines the route assigned to a train.

The function $sta : \mathcal{T} \rightarrow \mathbb{N}$ defines the start time assigned to the train run on the junction.

Let $Inc \subseteq \mathcal{T} \times \mathcal{R} \times \mathbb{N} \times \mathcal{T} \times \mathcal{R} \times \mathbb{N}$ be the relation denoting which timetable assignments are conflicting.

Definition 1 *An instance of a railway saturation problem is a six-tuple $(\mathcal{T}, \mathcal{R}, fr, fst, Inc, u)$, the problem is to find a couple (ra, sta) so that:*

- $\forall t \in \mathcal{T}, ra(t) \in fr(t)$,
- $\forall t \in \mathcal{T}, sta(t) \in fst(t)$,
- $\forall (t, t') \in \mathcal{T} \times \mathcal{T}, (t, ra(t), sta(t), t', ra(t'), sta(t')) \notin Inc$,

and the objective function is to maximize the size of the set $\{t \in \mathcal{T}, sta(t) \leq u\}$.

Later, the railway infrastructure saturation problem is described with a CP formulation. Other formulations have been introduced e.g. as a Set Packing Problem

(SPP) [10]. In [1] we have compared the CP formulation and the SPP formulation. The comparison of different formulations is outside the scope of this paper, we focus on the CP formulation and the improvement of the resolution method.

3 The CP formulation

The aims of the Constraint Programming models were originally to solve feasibility problems : given a set $X_1, \dots, X_n$ of variables, each associated with a domain $D_1, \dots, D_n$ respectively, and a set of constraints $C_1, \dots, C_n$, i.e. a subset of $D_1 \times \dots \times D_n$, the problem is to find an assignment of values to the variables while simultaneously satisfying the constraints. The CP models were extended to solve optimization problems : when a feasible solution is obtained, the value of the objective function is a new upper (resp. lower) bound of a variable, representing the objective function to minimize (resp. to maximize). This restriction is made by posting a new constraint on this variable.

The CP model of the Railway Infrastructure Saturation Problem presented here is borrowed from a CP model of a real-time train scheduling problem [8]. Our CP model focuses on expressing with *explicit terms* the influence of the signalling system on the run of the trains. This feature is important for dealing correctly with infrastructures within heavy traffic conditions like those encountered in junctions or stations.

The run of a train through a junction is a sequence of elementary runs. Each elementary run is the run through a track circuit. An elementary run is considered as an *activity* and each track circuit as the *unary resource* required to process it. Using the notation of section 2, a run of a train $t \in \mathcal{T}$ is a *sequence of n_t activities*. In the CP model, $ra(t)$ is the variable of the route assignement of a train and $fr(t)$ is the domain associated with it. For each activity (i.e. elementary run) of index i of a train t , $tca_t(i)$ denotes the track circuit assignement variable. The domains associated to $tca_t(i)$ are noted $ftc_t(i)$, i.e. the feasible track circuits of elementary run i . These domains are deduced from $fr(t)$. For each train t , a constraint links each variable $ra(t)$ to the variables $tca_t(i)$.

As not all alternative routes can have the same number of track circuits, we have created a fake track circuit to ensure that our model is declarative. The fake track circuit is added to the track circuit sequence to obtain sequences of the same size for all alternative routes. Let $|r|$ be the notation which gives the number of track circuits for a route $r \in \mathcal{R}$. The value of n_t is defined by:

$$n_t = \max_{r \in fr(t)} |r|.$$

After the definition of the number of activities n_t , we present successively the *capacity constraints* of the resources and the *temporal constraints* of the activities.

To define the *capacity constraints* of the unary resources (i.e. track circuits), we have to express that no more than one activity can be processed on a resource at a time. Let $st_t(i)$, $ct_t(i)$, $pt_t(i)$ be respectively the start time, completion time, and processing time variables of the activity associated to the elementary run of index i . The capacity constraint that restricts the use of each track circuit to only

one activity at a time is :

$$\forall t, t' \in \mathcal{T}, \quad \forall i \in [1, n_t], \forall j \in [1, n_{t'}]$$

$$tca_t(i) = tca_{t'}(j) \Rightarrow (ct_t(i) \leq (st_{t'}(j)) \vee ((ct_{t'}(j) \leq st_t(i))) \quad (2)$$

i.e. unless two activities use different ressources, they cannot overlap.

We consider now the definition of the *temporal constraints*. Due to a clearing phase, the time windows of successive activities of a train *overlap* each other, i.e. during that time the train occupies two contiguous track circuits (e.g. see the black rectangles of the Gantt chart in Figure 1). If we consider a block signalling system with 2 aspects, the start of each activity has to be *synchronised* with the start of the activity corresponding to the first track circuit of the current block. For the general case of a block system with n aspects, the synchronisation is established with the entrance in the first track circuit of the $n - 2$ previous block (e.g. see dashed rectangles for $n = 3$ in Figure 1). Let $run_t(i)$, $clr_t(i)$ be the variables for the minimum duration for these two phases, let $ftb_t(i)$ be the variable for the index of the first track circuit of the block. The start time of the running phase of an activity of index i is equal to $ct_t(i - 1) - clr_t(i - 1)$. The temporal constraints

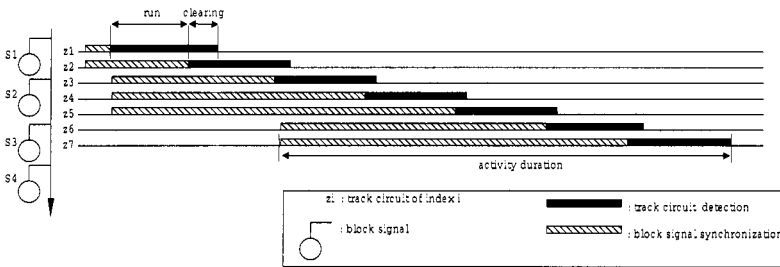


Figure 1: A gantt chart of activities modelling a 3 aspect block signalling system

of a RISP are formulated as follows :

$$pt_t(i) \geq run_t(i) + clr_t(i) \quad (3)$$

$$ct_t(i) - ct_t(i - 1) = run_t(i) + clr_t(i) - clr_t(i - 1) \quad (4)$$

$$\max_{j \in ftb_t(i)} (ct_t(j - 1) - clr_t(j - 1)) \geq st_t(i) \geq \min_{j \in ftb_t(i)} (ct_t(j - 1) - clr_t(j - 1)) \quad (5)$$

Constraint 3 states that the duration of an activity has to cover at least the running and clearing phases of an elementary run. Constraint 4 states the overlapping between the elementary runs. And constraint 5 states the synchronization due to the block system. This last constraint is more complex to implement as $ftb_t(i)$ is actually a variable linked to the variable $fr(t)$.

4 Resolution methods

A CP formulation of a problem may be addressed by two categories of backtracking search. The first category is named “retrospective algorithms”. It includes naive backtrack, backjumping [3]. The other category named “prospective algorithms” includes forward checking [5], which looks ahead to compute some form of local consistency among non instantiated variables. The CP has been extensively studied to develop various consistency algorithms, also named *constraint propagation*. A consistency algorithm makes it possible to reduce the domains of variables by removing values which are inconsistent with the constraints. For example, the Ilog Scheduler library provides three mechanisms to propagate the resource utilisation constraint to adjust the time bounds of activities [6] : timetable, disjunctive constraint and “edge finding”.

In previous experiments of the CP model [7], we used a prospective algorithm to solve real time train scheduling problems. These experiments showed that a complete search for solving instances with more than 12 trains, can not be carried out within reasonable time. As for solving a RISP instance we have to consider much more than 12 trains (e.g. in Table 1 the smaller instance has 125 trains), we gave up the idea of carrying out a complete search. To overcome the difficulty we have developed two approaches. The first one is a *greedy* algorithm [1] which assumed that the route variables $ra(t)$ are set and the unknown ones are the $sta(t)$ variables. The second approach *decomposes* the whole problem into a set of sub-problems whose size allows a search, even incomplete, of alternative routes for trains.

4.1 The greedy algorithm

In this approach, we have restricted, for each train, the set of feasible routes to one route. For each train category running through the junction, we define a “usual” route. We set the route variable $ra(t)$ to the “usual” route. The unknown variables are the $sta(t)$ variables. The greedy algorithm (c.f. Figure 2) uses a constraint propagation algorithm after each decision step on the $sta(t)$ variables. In this algorithm, we used the following notations :

- $propagate(Trains)$: a function which propagates the constraints posted for a set of trains *Trains*.
- $\prec$: an order relation so that two successive trains of a same converging line are separated by one train from all other lines .

The algorithm aims at scheduling all trains *as early as possible*. At each step of the loop, two criteria are used according a lexicographic order for choosing the train to schedule. The first one uses the earliest start time of the trains. If the first criterion is not sufficient to get only one train, the second criterion uses the order relation $\prec$.

```

pendingTrains  $\leftarrow \mathcal{T}$ 
while (pendingTrains  $\neq \emptyset$ ) loop
    candidateTrains  $\leftarrow \{t \in \text{pendingTrains} \text{ with minimum earliest } sta(t)\}$ 
     $t = \min_{\prec}(\text{candidateTrains})$ 
    if ( $sta(t) > u$ ) then exit
     $sta(t) \leftarrow$  earliest  $sta(t)$  value
    pendingTrains  $\leftarrow \text{pendingTrains} \setminus \{t\}$ 
    propagate(pendingTrains)
endWhile

```

Figure 2: The greedy algorithm

4.2 The decomposition approach

In this section, we describe a decomposition approach to solve the RISP instances. We decompose the initial problem into conjunctions of subproblems which are *simple enough to be solved better*. The solution to the original problem is composed by the subproblem solutions. The capacity constraints of the resources are global constraints between subproblems. We can notice that it is these global constraints that ensure the validity of the constructed solution. Subsequently, we describe the decomposition method, then we describe the resolution algorithm to solve the subproblems.

Let P , be the initial problem associated to a set of trains $\mathcal{T}$. The set $\mathcal{T}$ is split into disjoint subsets $\mathcal{T}_i$ such as $|\mathcal{T}_i| = \text{subsetSize}$. $P(\mathcal{T}_i)$ denotes the subproblem of P associated to the subset of trains $\mathcal{T}_i$.

An enumerative algorithm is applied to each subproblem c.f. Figure 3. In this algorithm, we used the following notations :

choice(fr(t)) : This function allows to set a *choice point* i.e. to choose a possible value from the domain $fr(t)$ of $ra(t)$. If the trial value does not lead to a solution (propagation fails), the search state saved at the choice point is restored and the trial value is removed from the domain so that another alternative can be explored. No special criteria is used for selecting the values of $ra(t)$ except when an initial solution is provided (c.f. Section 5.2).

The inner loop of the algorithm searches for each train of $\mathcal{T}_i$, a combination of route/entering time such, that it reduces the time extent of all entering times. If a fail is detected by propagation, it backtracks to the last choice made on step 5. If all route alternatives for all trains of $\mathcal{T}_i$ have failed, no better solution can be found, therefore the outer loop stops. We set a *stopping condition* in order to limit the cpu time spent solving each subproblem. The enumeration is incomplete due to the stopping condition and as no backtrack is done on the start time value selection.

```

1.  $timeExtent \leftarrow \max_{t \in \mathcal{T}_i}(fst(t)) + 1$ 
2. while not stoppingCondition loop
3.   post constraint :  $\max_{t \in \mathcal{T}_i}(fst(t)) < timeExtent$ 

4.   for each  $t \in \mathcal{T}_i$  loop
5.      $ra(t) \leftarrow choice(fr(t))$ 
6.      $sta(t) \leftarrow$  earliest  $sta(t)$  value
7.     if (propagate( $\mathcal{T}_i$ ) fail) then backtrack
8.   endFor
9.    $timeExtent \leftarrow \max_{t \in \mathcal{T}_i}(sta(t))$ 
10. endWhile

```

Figure 3: The algorithm for a subproblem $P(\mathcal{T}_i)$

5 The Computational results

5.1 Infrastructure considered

The CP formulation has been tested in an empirical study with data from the junction of Pierrefitte Gonesse North of Paris (see Figure 4). We noticed three main kinds of trains which travel through this node in both directions :

- TGV between Paris and the High Speed Line (HSL)
- Inter City trains between Paris and Chantilly
- Freight trains between Chantilly and the Grande Ceinture which cut-across the TGV routes

Four relevant scenarios have been tested on this node :

- all kinds of train
- TGV and Inter City trains
- TGV and Freight trains
- Inter City and Freight trains

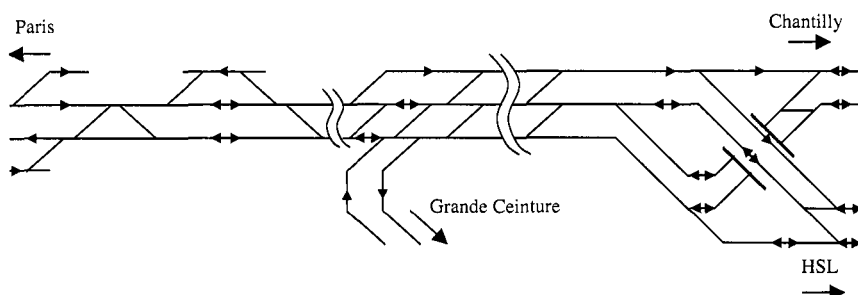


Figure 4: The layout of the Pierrefitte-Gonesse junction

For the CP model, we have generated four instances, one for each scenario. The parameters $\mathcal{T}$ of each instance of the problem is generated from the capacity of the lines considered of each scenario. The capacity of the lines has been evaluated using the expression (1). Table 1 gives the result of this expression and the number of variables and constraints of the CP model instances.

N°	$\mathcal{T}$			Numerical instances	
	TGV	IC	Freight	Variables	Constraints
1	81	76	49	75,307	77,142
2	81	76	0	64,422	66,117
3	81	0	49	44,185	45,543
4	0	76	49	42,099	43,524

Table 1: Instance characteristics for the CP model

5.2 Results

Table 2 provides the results of applying the algorithms described in sections 4.1 and 4.2 to the four problem instances considered. The unit of time considered is $u = 3600$ seconds. We considered for the decomposition algorithm the following parameters values : *subsetSize* = 6 and *stoppingCondition* = (run time < 300 CPU s).

It can be noticed that the greedy algorithm provides good solutions, this shows that the entering time variable has an important impact on performances and that good choices for the “usual routes” have been made. Conversely, the decomposition algorithm has weak performances, it cannot overcome the greedy algorithm on any instance problem. This shows that the decomposition generates good solutions to each subproblem but they induce bad decision search on the global problem. The third column of Table 2 provides better results for using the decomposition

N°	Greedy	Decomp.	Greedy+Decomp.
1	93	80	93
2	97	97	98
3	81	45	101
4	94	67	94

Table 2: Computational results

approach. The idea was to use the results of the greedy algorithm as an initial solution for the decomposition approach, i.e. the *choice()* function (c.f. Figure 3) selects firstly the value of the $ra(t)$ provided by the greedy algorithm. The goal of the decomposition approach was then to improve the solution by searching alternative routes. The results were slightly improved for the scenario 2 but notably improved for scenario 3.

6 Conclusion

We have presented a CP model of the ‘Railway Infrastructure Saturation Problem’. The model applies to assess the capacity of a junction or a station. Two resolution algorithms have been developed :

The first one is a greedy algorithm which assumes that the route variables are set and the unknown ones are the entering time variables.

In the second approach the whole problem is decomposed into a set of sub-problems whose size allows an incomplete search of alternative routes as well as the entering time variables.

The two resolution algorithms have been developed and combined to be applied to a real case study. The first algorithm shows better results than the second one. However, the latter algorithm could improve the result of the former one. This has been done by setting the initial solution of the decomposition algorithm with the greedy algorithm solution.

An interesting future work is to study deeply the performances of different models and resolution methods as initiated in [1], having in mind the goal to experiment potential integrations of different approaches.

Work is also in progress to experiment those methods on a larger-scale infrastructure like the Lille railway node.

References

- [1] X. Delorme, J. Rodriguez, and X. Gandibleux. Heuristics for railway infrastructure saturation. In *Electronic Notes in Theoretical Computer Science 50 NI(2001)* URL: <http://www.elsevier.nl/locate/entcs/volume50.html>, pages 41–55, July 2001.
- [2] L. Florio and L. Mussone. A method of capacity computation for complex railways systems. In *7 th WCTR*, pages 275–291, Sydney-Australia, 1995.
- [3] J. Gaschnig. A general backtracking algorithm that eliminates most redundant tests. In *International Joint Conference on Artificial Intelligence*, pages 457–466, Cambridge MA, 1977.
- [4] P. Hachemane. *Évaluation de la capacité de réseaux ferroviaires*. Thèse 1632, (in French), École Polytechnique Fédérale de Lausanne, 1997.
- [5] R. M. Haralick and G. L. Elliot. Increasing tree search efficiency for constraint satisfaction problems. *Artificial Intelligence*, 14:263–313, 1980.
- [6] C. Le Pape. Three mechanisms for managing resource constraints in a library

for constraint based scheduling. In *INRIA/IEEE Conference on Emerging Technologies and Factory Automation*, Paris-France, 1995.

- [7] J. Rodriguez. Fluidification de noeuds ferroviaires (projet GRRT prospectif). Technical Report RR-00-724-FR, (in French), INRETS, France, december 2000.
- [8] J. Rodriguez and K. Lyes. Constraint programming for real-time train circulation management problem in railway nodes. In *Computers in Railways VI*, pages 597–606, Lisbonne-Portugal, september 1998.
- [9] U.I.C. Fiche 405r. Technical report, UIC, 1978.
- [10] P. J. Zwaneveld, L. G. Kroon, H. E. Romeijn, M. Salomon, S. Dauzère-Pérès, S. P. Van Hoesel, and H. W. Ambergen. Routing trains through railway stations : Model formulation and algorithms. *Transportation Science*, 30(3):181–194, august 1996.