

Constraint programming for real-time train circulation management problems in railway nodes

J. Rodriguez, L. Kermad

Institut National de Recherche sur les Transports et leur Sécurité,

20, rue Elisée Reclus, F-59650 Villeneuve d'Ascq, France

EMail : joaquin.rodriguez@terre.inrets.fr

Abstract

This paper describes a constraint-based model of the train circulation management problem. The basic concepts of the model are activities, resources, time and allocation constraints. The problem is formulated as a joint scheduling and allocation problem, i.e. there exist degrees of freedom for deciding both which activities to perform and when, and which resources to allocate to these activities. The real life application for the experiments is a railway node near Paris-Nord train station, this node has been chosen because it is a complex junction and the traffic due to North TVG, Eurostar, Thalys and freight trains will notably increase in the years to come.

1. The train circulation management problem

The overall requirement is to allocate routes to the trains arriving at a node. Case of complex node, such as Pierrefitte-Gonesse node, each train will need a sequence of routes to pass through the node. The sequence of routes is planned and if there is no modification of the plan, all trains will pass through the node without delay. Unfortunately, the plan will be modified, and the problem is to minimise the delays caused to the trains.

Two kinds of modifications of the plan are applied : the sequence of the routes of a train is changed, the scheduling of some routes is

598 Computers in Railways

changed. If the first modification is applied, the alternative routes sequence must have the same entrance and exit signals as the ones defined by the route sequence planned. The scheduling of some routes will be changed when a restrictive aspect of a signal is presented to a train.

The manager of the circulation has to decide which modifications to apply to minimise the delays.

In this paper, we will present a computerised model of a railway node which can be used to modify a plan of train circulation. This model can be used to deliver a complete solution, i.e. as an « automatic » mode or it can be used to test a complete or a partial solution, i.e. as a « what if » mode.

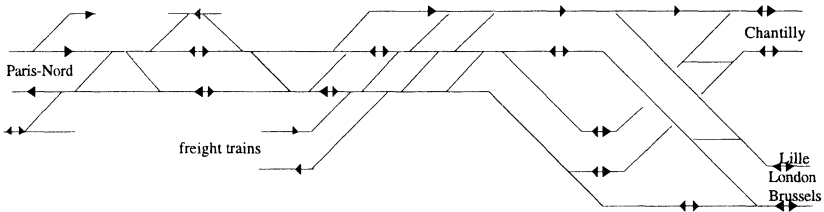


Figure 1 : Pierrefitte-Gonesse railway node.

2. Constraint satisfaction problems

A constraint satisfaction problem (CSP) can be formulated as, Mackworth[8], Montanari[9] : 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 subsets of $D_1 \times \dots \times D_n$, find an assignment of values to the variables while simultaneously satisfying the constraints. It is also possible to search for values that optimise a given criterion.

The problem may be addressed by two categories of backtracking search. The first category is named « retrospective algorithms » includes naive backtrack, backjumping, Gaschnig[3], the other category named « prospective algorithms » includes forward checking, Haralick&Elliot[4], which look ahead to compute some form of local consistency among non instantiated variables. CSP's have 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. Various type of consistency has

been defined, e.g. arc-consistency, Mackworth[8], path-consistency, Montanari[9], k-consistency, Freuder[2], arc-B-consistency, Lhomme[7].

Constraint-programming environments provide a declarative language where the programmer defines the variables and the constraints, e. g. the C++ libraries Ilog Solver, Puget&Albert[11]. The effectiveness of constraint-programming lies in the fact that the programmer can define a declarative model of the problem dissociated from the algorithm to solve it. A large number of problems in many areas such as planning, allocating resources, cutting materials, design of engineering structure,... can be viewed as special cases of CSP. In the next section, we present a model of a routes assignment problem to train circulation on railway node.

3. Constraint-based model of the train scheduling problem

In a general formulation, scheduling is the process of assigning activities to resources in time. No two scheduling problems are alike, a variety of constraints can affect the process of scheduling. Train scheduling can be considered as an instance of a scheduling problem. In this section, we will present a constraint based model of the train scheduling problem, the presentation will use the generic concepts of *activities* and *resources*. We will describe the variables, domains and specific constraints relevant to the problem.

The run of a train in a railway node is a sequence of routes which can be set through interlocking equipment by the operator. The basic idea of the model is to split the choice of the sequence of routes as a sequence of activity assignment choices to resources in order to formulate the problem with time and resource constraints.

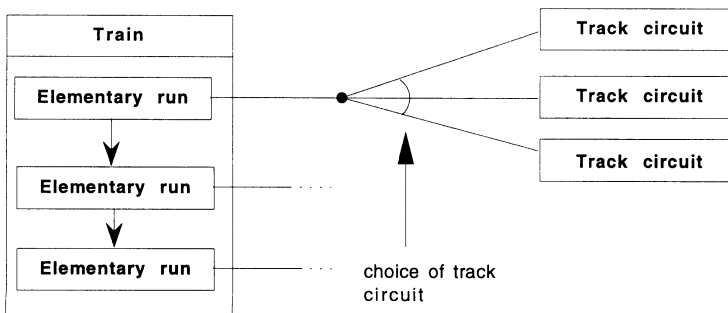


Figure 2 : Train run as track circuits choices.

600 Computers in Railways

The elementary run of a train is the run through a track circuit. We associated an activity to each run through a track circuit. The Figure 2 illustrates our approach, the choice of routes for a train is modelled as a choice of track circuit for each elementary run. The size of the sequence of elementary runs is fixed and is calculated as the maximum number of track circuits within the possible route sequences between the train's entrance and exit points.

3.1. Elementary run

An elementary run has an attribute variable set with an activity. More precisely, with an instance of the class *IntervalActivity* of the *Ilog Scheduler* library, Le Pape[5]. This kind of object is a model of an activity which executes without interruption from its start time to its end time and requires the same resource from the beginning to the end of its execution. Let end_i , $start_i$, $duration_i$ be respectively the starting, end, and duration variables of the activity associated to the elementary run of index i . The domains of these variables are initialised with the overall time interval to schedule. For each activity created for an elementary run, the eq. (1) constraint is posted.

$$end_i = start_i + duration_i \quad (1)$$

Another attribute of an elementary run is the variable of the selected track circuit, this variable will be noted tc_i for the run of index i . For the presentation of the domain and assignment constraint of the variables tc_i , we will consider the simple example of railway infrastructure of Figure 3. If we consider the run from the signal S1 to the signal S4, there are two route sequence choices, the first one is the route sequence [S1-S2, S2-S4] and the second one is [S1-S3, S3-S4]. The maximum size of the track circuit sequences is 5, this gives us the value of the size of the elementary run sequence. In the first line of the Table 1, we reported the sequence of track circuit variables of each elementary run, the next two lines give the assignment constraint of these variables. This constraint is implemented with an instance of a class of *Ilog Solver* which allows the expression of explicit constraint, i.e. given by a list of allowed tuples. The first and second elementary run have no choice for the track circuit variable. For the third one, we can choose the track circuit z3 or z4. The last line of the Table 1 gives the initial domains of all the track circuit variables tc_i . It must be noted that all route sequences cannot have the same number of track circuits, in the example, if the route sequence [S1-S3, S3-S4] is selected, at least one variable tc_i cannot be assigned.

A good design principle is to create a declarative model, i.e. all variables and constraints are created initially and the search code is kept separated from the model. To ensure that our model is declarative, we have created a « fake » track circuit, noted z^* , it is added to the track circuit sequence to get sequences of the same size. Then, all possible solutions can be represented by an assignment for all variables of the model. The fake track circuit z^* is a kind of track circuit, but its behaviour must be different, its main feature is that it must be completely « transparent » for all resource and temporal constraints posted over the elementary run variables. To be transparent mean that if z^* is assigned to a variable tc_i , the constraints will be satisfied and the search can continue as if the elementary run would not be defined. The specified behaviour of the fake track circuit allows it to be included anywhere in the elementary run sequence by shifting one index the next track circuits of the sequence. In the example of Table 1, the value z^* could be set before z_3 then the domains of tc_3 , tc_4 would be respectively $\{z^*, z_4\}$, $\{z_3, z_6\}$. Next, we will describe the resource constraints and temporal constraints which apply to route selection and specify the extensions needed to comply with the specified behaviour of z^* .

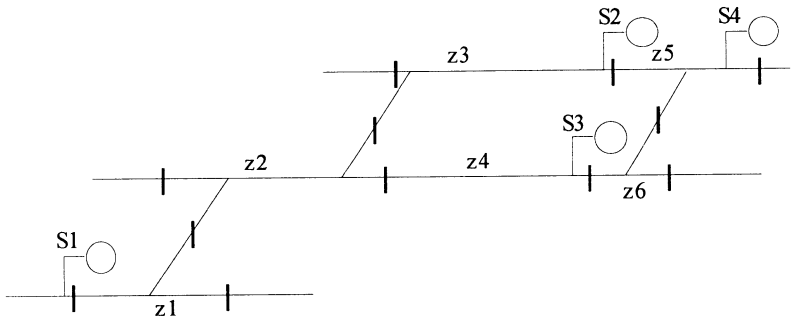


Figure 3 : A railway infrastructure example.

tc_i variables	tc_1	tc_2	tc_3	tc_4	tc_5
S1-S2, S2-S4	z_1	z_2	z_3	z^*	z_5
S1-S3, S3-S4	z_1	z_2	z_4	z_6	z_5
domains of tc_i	$\{z_1\}$	$\{z_2\}$	$\{z_3, z_4\}$	$\{z^*, z_6\}$	$\{z_5\}$

Table 1 : Assignment constraint and domains of track circuit variables.

3.2. Resource constraints

Each track circuit is an element of the infrastructure resources which is modelled by a unary resource. A unary resource is a resource whose capacity is one. To express the resource constraint, we introduce the following notations : a_i is the activity associated to the elementary run of index i , $\text{requires}(\langle \text{activity} \rangle, \langle \text{set of resources} \rangle)$ is a function posting an alternative resource constraint between an activity and a set of resources, $\text{resources}(\langle \text{set of track circuits} \rangle)$ is a function returning the resource set corresponding to a set of track circuits, $\text{dom}(\langle \text{variable} \rangle)$ the domain of a variable. The resource constraint for each elementary run can be stated by,

$$\text{requires}(a_i, \text{resources}(\text{dom}(tc_i))) \quad (2)$$

To comply with the specification of the fake track circuit z^* , we have associated it an infinity capacity resource to satisfy the resource constraint (2) each time z^* is a selected value to a track circuit variable.

The Ilog Scheduler library provides three mechanisms to propagate the resource utilisation constraint to adjust the time bounds of activities, Le Pape[6]. The first mechanism relies on an explicit time-table of the variation of resource utilisation and resource availability over time, for example when $\max(\text{start}_i) < \min(\text{end}_i)$ then activity a_i will use the resource between $\max(\text{start}_i)$ and $\min(\text{end}_i)$. The second mechanism consists of a disjunctive constraint which ensures that the time interval over which two activities requires the same resource cannot overlap. For example, if two activities a_i^k and a_j^l of trains l and m require the same resource, whenever $\min(\text{end}_i^k) > \max(\text{start}_j^l)$ then the constraint $\text{end}_j^l \leq \text{start}_i^k$ is posted. The third mechanism known as « edge finding », Carlier&Pinson[1], considers an arbitrary tuples of activities $\{a_{i_1}^1, \dots, a_{i_n}^n\}$ to prove that some activity $a_{i_j}^j$ must execute first (or last) among $\{a_{i_1}^1, \dots, a_{i_n}^n\}$. These mechanisms by adjusting the time bounds of activities makes it possible to prune many impossible decisions and then improve the efficiency of the search.

3.3. Temporal constraint

The first temporal constraint which can be stated is that the train will arrive at the entrance point of the node after the arrival time planned.

Let *arrivalTime* be the value of the arrival time planned, this constraint can be stated for all activities of a train by,

$$start_i \geq arrivalTime \quad (3)$$

An activity a_i of an elementary run has two successive phases (c.f. Figure 4) :

- run phase : between the time the train commenced to occupy track circuit tc_i and the time the train commenced to occupy tc_{i+1} .
- clearing phase : between the time the train commenced to occupy the track circuit tc_{i+1} and the clearance of the train from the track circuit tc_i .

Let $minRun_i$, $minClearing_i$ the variables for the minimum duration for these two phases. The domain of these variables are initialised with data obtained by simulating the run of trains with a simulator developed by SNCF and called SISYFE, Moulin&Gauyacq[10]. For each sequence of routes, SISYFE provides the different values of $minRun_i$, $minClearing_i$. A table constraint like the one for the assignment constraint (c.f. Table 1) enforces the sequence values of these variables for each route selected.

The constraint between the duration of the activity and the minimum duration of the run, the minimum duration of clearing phase is stated by,

$$duration_i \geq minRun_i + minClearing_i \quad (4)$$

Figure 4 shows a Gantt chart for the five activities of the run [S1-S3, S3-4]. The rectangle activities are coloured black the time during the track circuit is occupied by the train. We assumed that a driver has to stop the train without overrunning the signal with a red aspect, i.e. two trains cannot occupy any track circuit of a block section at the same time. The assumption is translated with a constraint over the start variable of the activities of the track circuit runs belonging to a block, this constraint states that the start of the activities are synchronised. The resulting « over-requirements » of the resources by the activities are shown with a striped rectangle in Figure 4. To express the synchronisation constraint we had to add another variable to the elementary run, this variable named $firstTcBlock_i$ has for each value of tc_i the value of the index of the first track circuit of the block. In the example of Table 1, the domain of $firstTcBlock_5$ would be initialised with {5,4}. The values between tc_i and $firstTcBlock_i$ are maintained by a table constraint and the synchronisation constraint can be stated by,

604 Computers in Railways

$$\min\{j \in \text{firstTcBlock}_i, \min(\text{start}_j)\} \leq \text{start}_i \leq \max\{j \in \text{firstTcBlock}_i, \max(\text{start}_j)\} \quad (5)$$

The time overlapping between track circuit occupancy is expressed by two constraints. The first one is directly derived from the definition of the clearing phase and is stated by,

$$\text{end}_i - \text{start}_{i-1} \geq \text{minClearing}_{i-1} \quad (6)$$

The second one is derived from the definition of the run phase to link two successive end variables and is stated by,

$$\text{end}_i \geq \text{end}_{i-1} - \text{minClearing}_{i-1} + \text{minRun}_i + \text{minClearing}_i \quad (7)$$

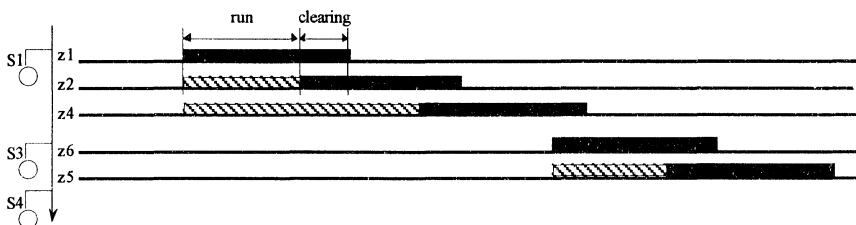


Figure 4 : Gantt chart of the activities of the run [S1-S3, S3-S4].

Let us now consider the previous constraints with the case of selecting the fake track circuit z^* as a value for tc_i . To comply with the constraint (5), the corresponding value of firstTcBlock_i is i . The question is what is the duration of the activity a_i if tc_i is set to z^* ? By definition of the behaviour of the model, the overlapping constraints (6) and (7) must be stated between tc_{i-1} and tc_{i+1} , this is complied with if the duration of activity a_i is equal to the minimum clearing duration of run $i-1$. This is stated by following conditional constraint,

$$(tc_i = z^*) \Rightarrow (\text{duration}_i = \text{minClearing}_{i-1}) \quad (8)$$

To post this constraint, we have to add the domain of minClearing_{i-1} to initial values of the domain of minClearing_i . To complete the requirement, the corresponding value of minRun_i is 0 for z^* .

4. Conclusion

The CSP model has been implemented and successfully validated on a part of the infrastructure of the Pierrefitte-Gonesse node. The constraint-based approach makes it possible to efficiently evaluate the impact of partial solutions of the CSP and then prunes large parts of the tree search. The next step is to use all the data of Pierrefitte-Gonesse node and evaluate the solutions with the simulator SISYFE.

One current modelling assumption is that only the run duration with maximum speed data is used. Current extension of the model to overcome this problem consist of using a resource state to model the aspect sequences completely and link the run durations with the resource states.

The presented model is a part of a project named SAFIR where SNCF and INRETS are involved in the development of a decision support system to traffic control centres. A specific goal of the project is to implement different models of train circulation management problem and to experiment various approaches to solve it.

Acknowledgement

This works was supported by funding from the Direction de la Recherche SNCF. The authors wish to thank SNCF members and particularly Daniel Gauyacq, Véronique Clerc and Nourredine Bencherqui for their helpful suggestions and comments. We also thank N. Rackham and F. Khun for reading this paper.

References

- [1] Carlier J., Pinson E., Adjustment of heads and tails for Job-Shop Problem, *European Journal of Operational Research*, N° 78, pp. 146-161, 1994.
- [2] Freuder, E.C., A sufficient condition for backtrack-free search, *Journal of the ACM*, Vol. 29, N° 1, pp. 24-32, 1982.
- [3] Gaschnig, J., A general backtracking algorithm that eliminates most redundant tests, *International Joint Conference on Artificial Intelligence*, Cambridge MA, p. 457, 1977.



606 Computers in Railways

- [4] Haralick, R.M., Elliot, G.L., Increasing tree search efficiency for constraint satisfaction problems, *Artificial Intelligence*, Vol. 14, pp. 263-313, 1980.
- [5] Le Pape, C., Implementation of resource constraint in Ilog Schedule : A library for the development of constraint-based scheduling systems, *Intelligent Systems Engineering*, Vol. 3, N°2, pp. 55-66, 1994.
- [6] Le Pape, C., Three mechanisms for managing resource constraints in a library for constraint-based scheduling, *INRIA/IEEE Conference on Emerging Technologies and Factory Automation*, Paris, France, 1995.
- [7] Lhomme, O., Consistency Techniques for numeric CSPs, *13th International Joint Conference on Artificial Intelligence*, Chambéry, France, pp. 232-238, 1993.
- [8] Mackworth, A., Consistency in Network of Relations, *Artificial Intelligence*, Vol. 8, pp. 99-118, 1977.
- [9] Montanari, H., Network of constraints : Fundamental properties and applications to pictures processing, *Information Sciences*, Vol. 7, pp. 95-132, 1974.
- [10] Moulin, R., Gauyacq, D., Simulation de systèmes ferroviaires - Un projet décisif, *Revue d'Electricité et d'Electronique*, N° 7, pp. 39-46, 1997.
- [11] Puget, J.F., Albert, P., Solver : Constraint + Objects = Descriptions, Workshop on Object representation, *International Joint Conference on Artificial Intelligence*, Chambéry, France, pp. 93-101, 1993.