A new Constraint Based Scheduling model for real-time Railway Traffic Management Problem using conditional Time-Intervals

Grégory Marlière [a,1], Sonia Sobieraj Richard[a],
Paola Pellegrini [b], Joaquin Rodriguez [a]
[a] Univ. Lille Nord de France, F-59000 Lille, IFSTTAR, COSYS, ESTAS,
F-59650 Villeneuve d'Ascq, France
[b] Univ. Lille Nord de France, F-59000 Lille, IFSTTAR, COSYS, LEOST,
F-59650 Villeneuve d'Ascq, France
[1] E-mail: gregory.marliere@ifsttar.fr, Phone: +33(0)320438496

Abstract
This paper tackles the real-time Railway Traffic Management Problem (rtRTMP). It is the problem of finding an optimal choice for the train schedules and routes to reduce the delays of trains due to conflicts. We present a new formulation of the rtRTMP. This new formulation is based on a previously proposed one that models railway traffic at a microscopic level with optional activities using a Constraint Based Scheduling (CBS) approach. To ease the modelling of optional activities, a new concept based on a tree data structure and a specific filtering algorithm was extended through the introduction of conditional time-interval variables in Ilog CP-optimizer library. The new formulation of the rtRTMP presented in this paper exploits the conditional time-interval variables. The formulation has been validated with experiments on a large set of instances. The experimental results demonstrate the effectiveness of this new CBS model and show its good performance compared with the state-of-the art RECIFE-MILP algorithm.

Keywords
Real Time Traffic Management, Train Dispatching Problem, Re-routing and re-scheduling trains, Minimize secondary delays, Constraint Propagation

1 Introduction

The design of railway services is a complex process in which the planning of the schedule of trains and the necessary resources can lead to conflicts at the operational level. These conflicts are due to unforeseen perturbation events. The main consequence of these conflicts is the delays suffered by trains and, consequently, the increase of passenger travel time. Delays due to conflicts between two trains are called "secondary" delays. Railway operators try to limit secondary delays inserting time allowances in the timetable design phase. Nevertheless, time allowance is not always sufficient to avoid conflicts or even their propagation to other trains in a snowball (or domino) effect. To limit this propagation, the dispatcher in charge of traffic management can change the dwell times at scheduled stops, the train orders at stations or junctions, or the routes assignment. The problem of finding an optimal choice for the train schedules and routes is defined as the real-time Railway Traffic

Management Problem (rtRTMP) (Pellegrini et al., 2014). A rich literature exists on formulations and methods for solving the rtRTMP, the reader is referred to Lusby et al. (2011), Cacchiani et al. (2014), Fang et al. (2015) for recent literature surveys.

The surveys point out that integer programming (IP) and mixed-integer programming (MIP) models are the most popular approaches along with graph models, while constraint programming (CP) ones are more seldom used. Nevertheless, CP models have some undeniable merits which make them interesting for this problem. In particular, they are able to generate feasible solutions for some hard problems in a short computation time. As an example, to generate the cyclic timetables of the Dutch network (Kroon et al., 2009), the method of the CADANS module to solve the Periodic Event Scheduling Problem (PESP) formulation is based on CP techniques (Schrijver and Steenbeek, 1994). We can also mention that the PESP instances of the whole inter city nertwork of Germany and the south and east subnetworks have been solved with a SAT-solver (Großmann et al., 2012), which uses specific CP techniques for variables with boolean domains.

For a given problem instance, CP models typically have fewer variables and constraints than the other approaches, and therefore requires less memory for the instances formulation. It is also worthwhile mentioning that despite the diversity of models and solutions methods, very few publications compare and analyse their relative performances and advantages.

Since our first proposal of a CP model in (Rodriguez, 2007), new features of CP and Constraint Based Scheduling (CBS) have been developed. CBS extends CP to get stronger propagation algorithms for specific constraints to solve scheduling problems. One feature is the ability to model optional activities along with powerful propagation algorithms (Vilím et al., 2005). In addition, exact algorithms that use hybrid methods (i.e. CP and Linear programming) and provide optimality proofs have been developped (Laborie and Rogerie, 2016). In this research, we aim to deeply investigate some CP and CBS modelling possibilities in the light of the new features developed in the last decade and we initiate a comparison of the achievable performance with the ones of other algorithms.

To do so, in this paper, we present a new CBS formulation of the rtRTMP that has been validated with experiments on a large set of instances. The performances of the heuristic resolution method for this new CBS formulation has been compared with the one of the state-of-the art RECIFE-MILP heuristic (Pellegrini et al., 2014).

2 CBS formulation

2.1 Scheduling theory

The basic idea of the CBS model of the rtRTMP is that a train passing through a control area is a job. According to scheduling theory, the concept of job is a set of activities linked by a set of temporal constraints. The rtRTMP can be viewed as a joint problem of allocating resources (the infrastructure broken down into track sections) to some activities sequences (the movement of a train).

In a CBS formulation, temporal constraints connect the temporal variables concerning activities (e.g., start, end or duration variables) according to principles which are specific to each application. The resource constraints are linked to the use and sharing of the resources by activities. Resources are divided into consumable or renewable resources, with the latter being either of limited capacity or with limited states. By sharing resources, indirect links between the temporal activity variables are generated by capacity or state resource

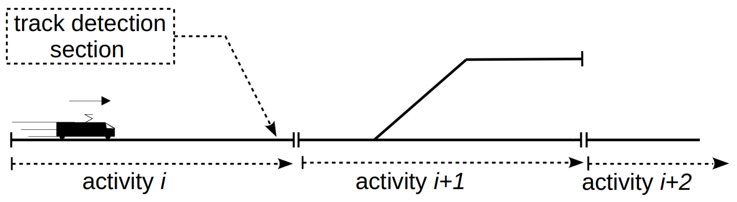


Figure 1: Train movement as a sequence of activities.

constraints.

This modelling approach for train scheduling was first proposed by Spzigel (1973) and then formulates the train scheduling problem on a single track line as a job-shop scheduling problem. Trains are jobs and their traveling through the single track connecting consecutive stations are activities.

In the remaining part of this paper, the formulation of the rtRTMP by Rodriguez (2007) is named RECIFE-CP1 and the new formulation presented in this paper is named RECIFE-CP2. We will refer to RECIFE-CP when we consider common parts to both formulations.

2.2 Microscroscopic model of the rtRTMP

The overall approach named RECIFE-CP is based on a microscroscopic model of the rtRTMP where train movements are controlled with a fixed block signalling system. The first modeling principle characterizing it considers a detailed decomposition of a train journey into a sequence of activities. Each activity is an elementary movement of the train through a track detection section (tds), as illustrated in Figure 1. A track detection section allows the detection of the occupation of a part of the railway infrastructure by a train. Tds's correspond in many railway infrastructures to electric devices named "track circuits" and are part of the block signalling system that ensure the safe movements of trains.

During normal operation, most of the time only one train should be detected by a tds at any point in time. Hence, tds's are modelled as unary resources. A unary resource is a resource allowing only one activity to use it at any point in time. However, an exception occurrs if a train set is splitted to operate two trains or, conversely, two train sets are joined to operate one train. This exception must be taken into account for the tds's corresponding to station platforms where split and join operations are performed.

2.3 Temporal constraints

A second modeling principle of RECIFE-CP consists in the consideration of detailed temporal constraints between activities. They allow modeling some characteristics of the block signalling system[1] such as: the length of trains, the number of signalling aspects, the watching time (e.g. running time of the sight distance), the sectional route release of the interlocking system.

A brief overview of the temporal constraints between activities is illustrated by a time over distance diagram in Figure 2. Along the horizontal axis of this diagram, we have the

[1]Additional characteristics are ommitted , e.g., time for clearing signal or release time, to simplify the presentation

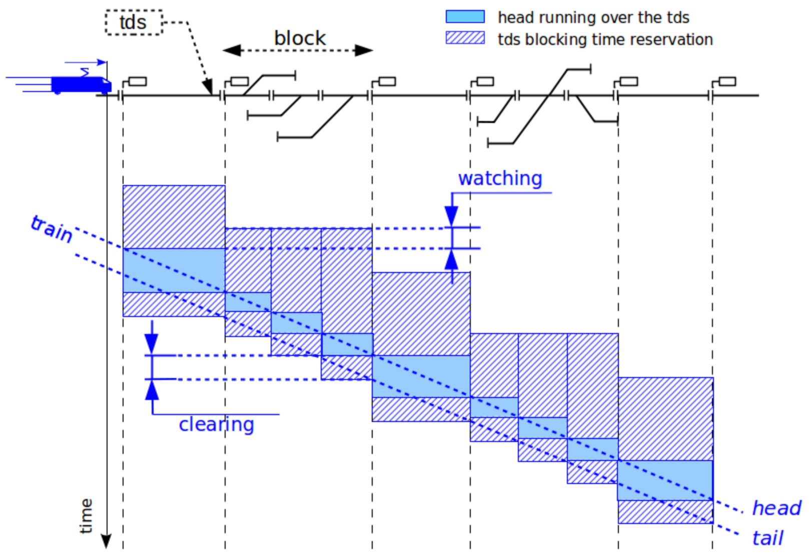


Figure 2: Head running activity and tds blocking time reservation

sequence of tds's that the "blue" train runs through. The line is broken down into blocks that are bounded by signals providing driving information to the train driver. A block can have one or more tds's depending on the configuration of the line. In the diagram, blue dashed lines report the position of the head and the tail of the train.

In RECIFE-CP1, we consider for each tds only one activity. The temporal constraints to maintain safe headway with a preceeding train (i.e., the blocking time theory constraints (Hansen, 2008)) are expressed according the start and end of these activities (Rodriguez, 2007). Instead, in RECIFE-CP2 we define two nested activities for each tds. The first one is the running of the head of the train through the tds. The sequence of the head running activities are shown with filled blue rectangles in Figure 2. Each activity is linked by a "start at end" constraint with the precedent activity. The second activity associated to a tds includes the first one and is extended to contain the reservation time to comply with the blocking time theory constraints. This second type of activities are shown with striped rectangles in Figure 2 for the case of 3-aspect block signalling sytem. All tds's of a block must be reserved when the train reaches the watching distance point of the previous block. The additional detection time due to the length of the train (called clearing time) is shown by the extended striped rectangle. It lasts until the tail of the train is no more detected by the tds. When there are switches within a block, separated tds's for each switch allow the interlocking system to release and set earlier the sequence of incompatible routes and then safely optimise the traffic. The sectional route release of the interlocking system is modelled in RECIFE-CP with separed activities for each tds of a block section (c.f. Figure 2).

Mascis and Pacciarelli (2002) showed that these temporal constraints have the same properties as the ones of a job-shop scheduling problem with blocking and no-wait constraints of the classic scheduling theory.

2.4 Alternative route choices

A third modeling principle of RECIFE-CP consists in considering alternative routes. Therefore, decision variables are defined to select one from the set of alternative routes for each train to avoid conflicts or to reduce secondary delays. The ability to model optional activities (Vilím et al., 2005; Laborie and Rogerie, 2008) has substantially changed the formulation of the model for the re-routing decisions.

In RECIFE-CP1 a train run is modelled with only *one sequence of activities* whatever the chosen route. Hence, each activity does not necessarily correspond to a real train movement through a tds as not all routes have the same tds sequence length. On the other hand, in RECIFE-CP2 a train run is modelled with *as many sequences of activities as route choices*. The sequence of head running activities length is equal to the tds sequence length of the chosen route. It should be noted that each blocking time reservation activity covers a head running activity, thus there is a sequence of blocking time reservation activities "enveloping" the sequence of head running activities. To illustrate both types of model, let us first consider the example of train that has two alternative routes r_1 and r_2 in Figure 3.

In RECIFE-CP1, a single sequence of six activities is defined, Figure 4 gives the different tds's that can be used by each activity according to the route choices r_1 and r_2. If r_2 is chosen, a tds is assigned to each activity which models the time to block and to run through it. If r_1 is chosen, a "dummy" tds, named tds^*, is added in the sequence of r_1 for an additional "fictive" activity as r_1 have only five tds's. tds^* can be added at any position within the sequence. In the example of Figure 4, it is added in the fourth position, therefore a_4 is the additional fictive activity. In a similar way, more than one tds^* can be added and put at any position within the sequence. The duration of the fictive activities cannot be zero due to the temporal constraints that link the sequence of activities: they are both equal to the clearing time of the previous activity. Therefore, tds^* can be viewed as a tds with zero length. Many activities can require tds^* as fictive elementary runs at any point in time, then the resource tds^* has an infinite capacity to satisfy all these capacity requirements. Adding tds^* with the former properties allows the definition of the same kind of resource and temporal constraints to all the activities of the sequence.

In RECIFE-CP2, we have two sequences of activities for this example. A sequence with five activities for r_1 and a sequence with six activities for r_2. To reduce the number of variables and improve the constraint propagation algorithm, the activities of two routes that have the same tds sequence with same running times are merged. After merging the equivalent activities, we obtain a graph of activities such that a path from the first tds activity to the last tds activity gives a sequence of activities for r_1 and a different sequence activities for r_2. In the example, the activities for the elementary run through tds_1 and tds_7 are merged as they have the same characteristics for r_1 and r_2. Conversely, for the elementary runs through tds_2 and tds_6, two activities are kept separated because the minimal running time for r_1 is different from the one for r_2. If r_1 is chosen, the activities a^{r_2,tds_2}, a^{r_2,tds_3}, a^{r_2,tds_5}, a^{r_2,tds_6} are "non-executed" and all related constraints and variables are useless. Similarly if r_1 is chosen, a^{r_1,tds_2}, a^{r_1,tds_4}, a^{r_1,tds_6} are non-executed. When all route assignments are done, we have to get only one sequence of executed activities for each train coherent with the route choice.

To improve the constraint propagation and hence the solution method, we create in RECIFE-CP2 a hierarchical model with new global constraints on groups of activities. These global constraints allow the encapsulation of a group of activities into one high-level

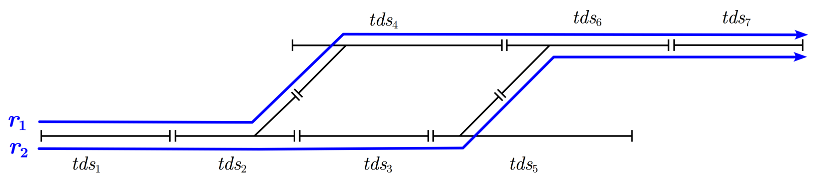


Figure 3: Example of tds route sequences

RECIFE-CP1 model							
sequence of activities		$a_1 \rightarrow$	$a_2 \rightarrow$	$a_3 \rightarrow$	$a_4 \rightarrow$	$a_5 \rightarrow$	a_6
tds sequences	r_1	$(tds_1,$	$tds_2,$	$tds_4,$	$tds^*,$	$tds_6,$	$tds_7)$
	r_2	$(tds_1,$	$tds_2,$	$tds_3,$	$tds_5,$	$tds_6,$	$tds_7)$

RECIFE-CP2 model						
tds sequence	r_1	$(tds_1,$	$tds_2,$	$tds_4,$	$tds_6,$	$tds_7)$
tds sequence	r_2	$(tds_1,$	$tds_2,$	tds_3 $tds_5,$	$tds_6,$	$tds_7)$

graph of head running activities

$$\begin{tikzcd}
 & a^{r_1}_{tds_2} \arrow[r] & a^{r_1}_{tds_4} \arrow[rr] & & a^{r_1}_{tds_6} \arrow[dr] & \\
a^{r_1,r_2}_{tds_1} \arrow[ur] \arrow[dr] & & & & & a^{r_1,r_2}_{tds_7} \\
 & a^{r_2}_{tds_2} \arrow[r] & a^{r_2}_{tds_3} \arrow[r] & a^{r_2}_{tds_5} \arrow[r] & a^{r_2}_{tds_6} \arrow[ur] &
\end{tikzcd}$$

Figure 4: Sequences of activities and tds's for the two RECIFE-CP models

activity. Derived high-level activities can be used with any temporal constraint in the same way as low-level ones.

In the example of Figure 3, we can notice that each activity of the group $G_1 = \{a^{r_1}_{tds_2}, a^{r_2}_{tds_2}\}$ always starts after the end of activity $a^{r_1,r_2}_{tds_1}$. In the same way, activity $a^{r_1,r_2}_{tds_7}$ always starts after the end of each of activity of the group $G_2 = \{a^{r_1}_{tds_6}, a^{r_2}_{tds_6}\}$. Let a_{G_1} (resp. a_{G_2}) the high-level activity linked by a "group constraint" to the group G_1 (resp. G_2), then we can state the precedence constraints $a^{r_1,r_2}_{tds_1} \prec a_{G_1}$ and $a_{G_2} \prec a^{r_1,r_2}_{tds_7}$.

More generally, we define a precedence constraint between a pair of high-level activities $(a_{G_{prec}}, a_{G_{succ}})$ such that each high-level activity is linked by a "group constraint" to a group of activities G_{prec} and G_{succ} respectively: each activity of G_{prec} precedes an activity of G_{succ} and conversely each activity of G_{succ} follows an activity of G_{prec}.

Two group constraints are used: `span`$(a_G, a_1, \ldots, a_n)$ states that activity a_G, if executed, spans over all executed activities of the set $\{a_1, \ldots, a_n\}$; `alternative`$(a_G, a_1, \ldots, a_n)$ states that if activity a_G is executed then exactly only one of activities $\{a_1, \ldots, a_n\}$ is executed and a_G starts and ends together with this chosen one. Activity a_G is non-executed if

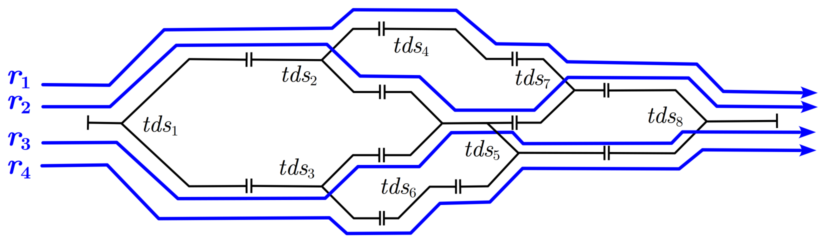


Figure 5: Example of tds sequences to illustrate group constraints

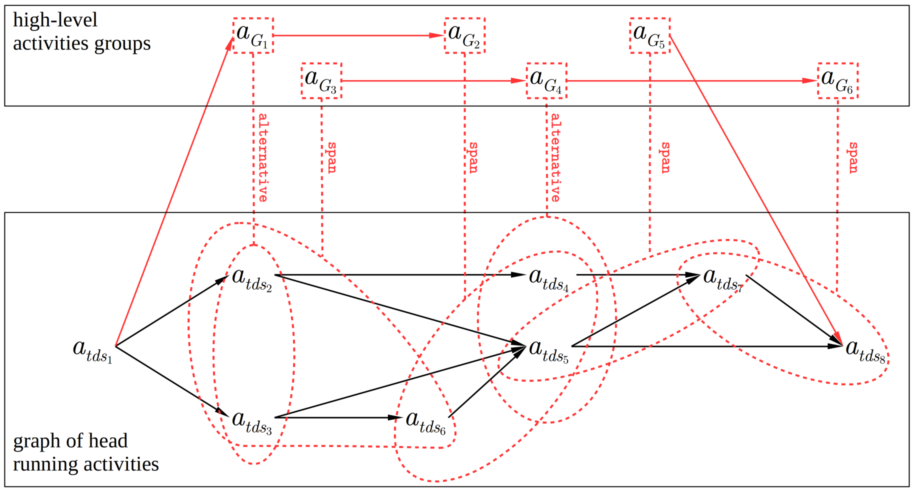


Figure 6: Graph of activities for the example of Figure 5

and only if none of activities $\{a_1, \dots, a_n\}$ is executed (Laborie and Rogerie, 2016).

To illustrate the definition of high-level activities and the group constraints, let us consider another example, depicted in Figure 5. To simplify the presentation, we consider that the elementary runs through a tds have the same characteristics whatever the route considered. Therefore all the activities corresponding to runs through a common tds are merged into one activity which is not indexed by routes. The lower part of the Figure 6 shows the graph of head running activities. Remark that contrary to the example of Figure 3 not all paths of the precedence graph correspond to a tds sequence activities for a route. The activities of each group are shown with red dotted shapes linked with a red dotted line to the corresponding group activity. The links are named with the group constraint used. The first hierarchical activity a_{G_1} is linked by an **`alternative`** constraint to the set of activities $\{a_{tds_2}, a_{tds_3}\}$ to state the precedence constraint $a_{tds_1} \prec a_{G_1}$. The group $G_2 = \{a_{tds_4}, a_{tds_5}, a_{tds_6}\}$ is an example in which the activity a_{G_2} cannot be linked with an **`alternative`** constraint because a_{tds_6} precedes a_{tds_5} thus the activities of G_2 are linked with a `span` constraint to a_{G_2}. This `span` constraint with a_{G_2} allows to state the precedence constraint $a_{G_1} \prec a_{G_2}$. All in all, the group activities of this example add six high-level activities and five precedence constraints (red arrows).

2.5 Conditional time-interval variables

In many works of scheduling theory, the main decisions are assigning resources to activities and scheduling activities. However in industrial applications, it is also necessary to consider the choice of activities that will be executed in the final schedule, for example when there are alternative production processes in response to an order. As in Artificial Intelligence Planning which requires to choose a sequence of actions to achieve a goal, recent developements in scheduling consider problems involving the choice of whether to execute or not some activities. This translates into the introduction of *optional activities*.

Vilím et al. (2005) introduce a tree data structure and a specific constraint propagation algorithm to model optional activities. This was later extended by the introduction of *conditional time-interval variables* in Ilog CP-optimizer library (Laborie and Rogerie, 2008).

A conditional time-interval variable (or time-interval variable for the sake of simplicity), noted a, represents an interval of time of interest in a schedule. In many cases, as in the problem modelled here, a time-interval variable is the time interval in which an activity is executed.

Let $\perp$ a value meaning the interval of time of interest is not present in the solution schedule or an activity is non-executed. The domain of a time-interval variable is a subset of $\{\perp\} \cup \{[s,e)|s,e \in \mathbb{Z}, s \leqslant e\}$. Like any other variable in a constraint satisfaction problem, a time-interval variable is said to be fixed if its domain is reduced to a singleton. Let $\underline{a}$ denotes a fixed time-interval variable, then $\underline{a} = \perp$ means that the activity is non-executed (not present in the solution schedule) or $\underline{a} = [s,e)$ means that the activity is executed (present in the solution schedule). The values s and e are respectively the start and end time of the activity. A time-interval variable is said to be non-executed if it is not considered by any constraint or expression it is involved in, said in a different way, it is as if they were deleted. An execution or presence status noted $pres(\underline{a})$ is equal to 1 if the activity is executed and 0 if it is non-executed.

When a is linked by a precedence constraint to another time-interval variable b and a is non-executed, then the precedence constraint impacts b. More generally all constraint definitions (i.e. propagation algorithms) must specify how they manage non-executed time-interval variables.

The conditional time-interval variables are linked by two kinds of constraints : the logical constraints and the temporal constraints.

The logical constraints link the execution status of the time-interval variables. These constraints are aggregated in a 2-SAT (2-satisfiability) constraint network. For example, the execution status of the time-interval variables for two alternative tds's that correspond to two route choices will be linked by a clause with an $\vee$ operator.

The temporal constraints state the different temporal positions of the start and end events of the time-interval variables, i.e., "start before start" or "start at end". These constraints are aggregated in a Simple Temporal Network (STN) extended to the presence statuses. The temporal constraints are "hybrid" in the sense that they combine the logical aspect of activities (i.e. "executed" or "non-executed") and the temporal aspect (i.e., it represents an activity with a start, end and duration).

Beside the expressiveness of the time-interval variables, the 2-SAT and STN constraint networks ensure a strong constraint propagation and therefore an efficient search for the optimization engine.

3 Formulation

For the formulation, we use a notation close the one introduced by Pellegrini et al. (2014) and then as follows:

T, R, TDS	set of trains, routes and track detection sections, respectively,
$R_t \subseteq R$	set of routes that can be used by train t,
TDS^r	set of track detection sections composing route r,
ty_t	type corresponding to train t (indicating characteristics as weight, length, engine power, etc.),
$TDS_t \subseteq TDS$	set of track detection sections that can be used by train t ($TDS_t = \bigcup_{r \in R_t} TDS^r$),
$PL \subset TDS$	set of track detection sections corresponding to platforms (if the control area includes a station),
$PL_{t,t'} \subset PL$	set of track detection sections corresponding to the possible departure platforms of a train t' which uses the same rolling stock as train t and results from the turnaround of train t,
$bs_{r,tds}$	block section including track detection section tds along route r,
$p_{r,tds}$	track detection sections preceding tds along route r,
$ref_{r,tds}$	reference track detection section for the blocking time reservation of tds along route r: first track detection section of the $n-2^{nd}$ block section preceding $bs_{r,tds}$, with n number of aspects characterizing the signaling system,
$rt_{ty,r,tds}$	running time of track detection section tds along route r for a train of type ty,
$ct_{ty,r,tds}$	clearing time of track detection section tds along route r for a train of type ty,
for_{bs}, rel_{bs}	formation and release time for block section bs, respectively,
$init_t$	earliest time at which train t can be operated: either expected arrival in the control area or expected departure from a platform within the control area,
$exit_t$	earliest time at which train t can reach its destination given $init_t$, the route assigned to t in the timetable and the intermediate stops,
$i(t,t')$	indicator function: 1 if trains t and t' use the same rolling stock and t' results from the turnaround of train t, 0 otherwise,
$ms_{t,t'}$	minimum separation between the arrival of a train t and the departure of another train t' using the same rolling stock,
$S_t, TDS_{t,s}$	set of stations where train t has a scheduled stop and set of track detection sections that can be used by t for stopping at station s,
$arr_{t,s}$	scheduled arrival times for train t at station s.

3.1 Decision variables

We define following decision time-interval variables :
for all triplets of $t \in T$, $r \in R_t$ and $tds \in TDS^r$:

$a^{t,r}_{tds,h}$:	optional time-interval variable which represents the running time activity of t's head through tds along r,
$a^{t,r}_{tds,b}$:	optional time-interval variable which represents the blocking time reservation activity of tds for t along r,

for all $t \in T$:

D^{arr}_t, D^{exit}_t :	delay suffered by train t at station arrivals (cumulated) and at the exit from the control area.

Moreover, we define binary variables for the route choices:
for all pairs of train $t \in T$ and route $r \in R_t$:

$$x_{t,r} = \begin{cases} 1 & \text{if } t \text{ uses } r, \\ 0 & \text{otherwise, not;} \end{cases}$$

The objective is the minimization of the total secondary delays suffered by trains at their departure from stations and exit from the control area:

$$\min \sum_{t \in T} (D^{arr}_t + D^{exit}_t) \tag{1}$$

To define the constraints, let us consider the following additional notations :

$s(a), e(a), d(a), pres(a)$	the start, end, duration and presence status for time-interval variable a, respectively,
$first(a^{t,r}_{tds,h}), last(a^{t,r}_{tds,h})$	boolean functions that return true if $a^{t,r}_{tds,h}$ is the first, respectively the last, head running activity of train t through the tds sequence for route r,
$\{(G^t_{prec}, G^t_{succ})\}$	set of pairs of groups of tds of train t $G^t_{prec} \in \mathcal{P}(TDS_t)^2$, $G^t_{succ} \in \mathcal{P}(TDS_t)$ with the folllowing property : each head running activity through a $tds \in G^t_{prec}$ (resp. $tds \in G^t_{succ}$) precedes (resp. follows) at least one head running activity through a $tds \in G^t_{succ}$ (resp. $tds \in G^t_{prec}$),
$prec(G)$	boolean function that returns true if $\exists (tds, tds') \in G$ such that the head running activities through tds and tds' are linked with a precedence constraint, otherwise false is returned.

The constraints are :

$$\sum_{r \in R_t} x_{t,r} = 1 \, \forall t \in T, \tag{2}$$

$$if(x_{t,r} = 1) \Rightarrow pres(a^{t,r}_{tds,h}) = 1 \forall t \in T, r \in R_t, tds \in TDS^r, \tag{3}$$

$$if(x_{t,r} = 1) \Rightarrow pres(a^{t,r}_{tds,b}) = 1 \forall t \in T, r \in R_t, tds \in TDS^r, \tag{4}$$

$$s(a^{t,r}_{tds,h}) \geqslant init_t \forall t \in T, r \in R_t, tds \in TDS^r, \tag{5}$$

[2]We use the notation $\mathcal{P}(S)$ to denote the power set of a set S.

$$d(a^{t,r}_{tds,h}) \geqslant rt_{ty,r,tds} \forall t \in T, r \in R_t, tds \in TDS^r, \tag{6}$$

$$s(a^{t,r}_{tds,h}) = e(a^{t,r}_{p_{r,tds},h}) \forall t \in T, r \in R_t, tds \in TDS^r, \tag{7}$$

$$e(a^{t,r}_{tds,b}) = e(a^{t,r}_{tds,h}) + ct_{ty,r,tds} + rel_{bs_{r,tds}} \forall t \in T, r \in R_t, tds \in TDS^r, \tag{8}$$

$$s(a^{t,r}_{tds,b}) = a^{t,r}_{ref_{r,tds},h} - for_{bs_{r,ref_{r,tds}}} \forall t \in T, r \in R_t, tds \in TDS^r, \tag{9}$$

$$\begin{gathered}\texttt{alternative}(a^t_G, a^{t,r_1}_{tds_1,h}, \ldots, a^{t,r_n}_{tds_n,h}), G = \{a^{t,r_1}_{tds_1,h}, \ldots, a^{t,r_n}_{tds_n,h}\} \\ \forall t \in T, G \in (G^t_{prec} \cup G^t_{succ}) : \neg prec(G)\end{gathered} \tag{10}$$

$$\begin{gathered}\texttt{span}(a^t_G, a^{t,r_1}_{tds_1,h}, \ldots, a^{t,r_n}_{tds_n,h}), G = \{a^{t,r_1}_{tds_1,h}, \ldots, a^{t,r_n}_{tds_n,h}\} \\ \forall t \in T, G \in (G^t_{prec} \cup G^t_{succ}) \ : \ prec(G)\end{gathered} \tag{11}$$

$$\begin{gathered}e(a^t_G) = s(a^t_{G'}) \\ \forall t \in T, (G, G') \in \{(G^t_{prec}, G^t_{succ})\}\end{gathered} \tag{12}$$

$$\begin{gathered}pres(a^{t',r'}_{tds,h}) = pres(a^{t,r}_{tds,h}) \\ \forall t, t' \in T, r \in R_t, r' \in R_{t'} : i(t,t') = 1 \wedge tds \in PL_{t,t'}\end{gathered} \tag{13}$$

$$\begin{gathered}s(a^{t',r'}_{tds,h}) \geqslant e(a^{t,r}_{tds,h}) + ms_{t,t'} \forall t, t' \in T, r \in R_t, r' \in R_{t'} : \\ i(t,t') = 1 \wedge last(a^{t,r}_{tds,h}) \wedge first(a^{t',r'}_{tds,h}) \wedge tds \in PL_{t,t'}\end{gathered} \tag{14}$$

$$\begin{gathered}s(a^{t',r'}_{tds,b}) = e(a^{t,r}_{tds,b}) \forall t, t' \in T, r \in R_t, r' \in R_{t'} : \\ i(t,t') = 1 \wedge last(a^{t,r}_{tds,h}) \wedge first(a^{t',r'}_{tds,h}) \wedge tds \in PL_{t,t'}\end{gathered} \tag{15}$$

$$\texttt{noOverlap}(a^{t,r}_{tds,b}) \forall t \in T, r \in R_t : tds \in TDS \tag{16}$$

$$D^{exit}_t = \sum_{\substack{r \in R_t, tds \in TDS_r: \\ last(a^{t,r}_{tds,h})}} e(a^{t,r}_{tds,h}) - exit_t \forall t \in T \tag{17}$$

$$D^{arr}_t = \sum_{r \in R_t} \sum_{\substack{s \in S_t, \\ tds \in TDS_{t,s}}} (s(a^{t,r}_{tds,h}) + rt_{ty,r,tds} - dep_{t,s} x_{t,r}) \forall t \in T \tag{18}$$

Constraints (2) ensure that exactly one route is used by each train.
Constraints (3) and (4) link the choice of a route r and the presence of the corresponding activities, i.e., if route r is chosen all the activities must be executed (be present in the solution schedule) .
Constraints (5) state that trains cannot be operated earlier than $init_t$.
Constraints (6) impose that the duration of the running time head activities are greater than the running time of track detection section tds along route r for a train of type ty .
Constraints (7) impose a precedence constraints between running time head activities of a train.
For constraints (8), the blocking time reservation lasts after the tail of the train clears tds, which corresponds to the start of the head running plus a clearing time for the type of train ty plus the block section release time.
Constraints (9) state that the blocking time reservation activity is synchronized with the time the head of the train is detected by the reference track detection section $ref_{r,tds}$ minus the route formation time.
Constraints (10) and (11) link a group of activities $G = \{a^{t,r_1}_{tds_1,h}, \ldots, a^{t,r_n}_{tds_n,h}\}$ into a high-

level activity a_G^t according to the presence of precedence constraints between low-level activities. High-level activities are linked to low-level activities by `span` or `alternative` constraints.
Constraints (12) state the precedence constraints between high-level activities.
Constraints (13) ensure local coherence: trains using the same rolling stock must use the same platform where they turnaround.
Constraints (14) ensure that a minimum separation time must separate the arrival and departure of trains using the same rolling stock for a turnaround.
Constraints (15) ensure the tds where the turnaround takes place is utilized for the whole time between t''s arrival and t's departure. Thus, the first activity blocking time reservation of t' starts when the last activity blocking time reservation of t ends.
Constraints (16) ensure that the blocking time activities of a shared tds do not overlap.
Constaints (17) and (18) state that the values of the delays D_t^{exit} and D_t^{arr} of a train t is the difference between the actual and the scheduled times at the exit of the infrastructure, respectively at the arrival at stop stations.

4 Solution method

The solution method uses the algorithm of Vilím et al. (2015) for scheduling problems which combines a Failure-Directed Search (FDS) with Self-Adapting Large Neighborhood Search (SA-LNS).

First, SA-LNS (Laborie and Godard, 2007) aims to find a good quality solution quickly. It is an iterative improvement method with following steps:

1. Start with an existing solution (heuristic or CP search)
2. Select a Large Neighborhood (LN) and a Completion Strategy (CS)
3. Apply LN to relax part of the solution and fix the rest
4. Apply CS to improve solution using a limited search tree
5. If time limit is reached then stop else go to 2

SA-LNS uses the following components to improve the search:

- *Constraint propagation algorithms* for the logical and the precedence constraints networks (Vilím et al., 2005),
- Enhanced selection of LN and CS: *machine learning techniques* to portfolios of LN and CS that quickly converge on good solutions (Laborie and Godard, 2007),
- *Temporal Linear Relaxation*: use CPLEX's LP solver for a solution to a relaxed version of the problem to guide heuristics (Laborie and Rogerie, 2016).

FDS is activated when the search space seems to be small enough, and SA-LNS has difficulties improving the current solution. It builds a complete search tree and it drives the search into conflicts in order to prove that the current branch is infeasible. It uses a *restart scheme with nogoods*.

		Junction	**Line**	**Terminal stations**	
		# 1	*# 2*	*# 3*	*# 4*
		Gonesse	*MLJ-Rouen*	*Lille*	*StLazare*
Infrastructure	*Length (km)*	15	80	7	4.5
	Routes	37	187	2409	84
	Blocks	79	157	829	197
	Track Circuits	89	236	299	212
	Stations	0	13	1	4
	Platforms	0	33	17	51
Timetable	*Trains/Day*	336	237	589	1212
	Routes alternatives/Train	5-13	1-24	1-71	1-9
	Turnarounds	0	6	298	606

Table 1: Case-studies characteristics

5 Experiments

5.1 Case-studies

In the experimental analysis, we tested our formulation on perturbations of real instances representing four French control areas with different characteristics: a junction with mixed traffic, a line with intermediate stops, and two passenger terminal stations with high density traffic. Namely, they cover the Gonesse junction north of Paris (# 1), the line between Mante-La-Jolie and Rouen-Rive-Droite (# 2) and the Lille-Flandres (# 3) and Paris–Saint-Lazare (# 4) stations. Their characteristics are detailed in Table 1 and their layout in Appendix A. Notes that the second line of Table 1 gives the values of parameter R and the heighth line gives the bounds of the number of routes per train (bounds of $|R_t|$).

5.2 Experiments settings

The experiments involve RECIFE-CP2 (named CP) and RECIFE-MILP (named MILP) in order to compare their performances in various cases.
Both algorithms are configured to perform a two-step approach:

- in the first step, a maximum of 10 seconds CPU time is allocated for "fixed-route" solution, which means that the route fixed in the timetable is used for each train,
- in the second step, the best solution of the previous step is used for initializing the "all-route" resolution, which means that all possible routes are used.

A limit of 180s CPU time is imposed for the resolution of these two steps on an Intel(R) Xeon(R) CPU E5-2643 v4 @ 3.40GHz, 24 cores, 128go RAM.
For each control area, we used two methods to increase incrementally instance difficulties:

- Horizon size variation,
- Perturbation rate variation.

5.3 Horizon size variation

For each of the 4 control areas, we generates 30 disruption scenarios: starting from the original timetable, 20% of randomly selected trains are delayed with a value in the interval between 5 and 15 minutes.

To cover a variety of instances difficulty, for each disruption scenario, we have selected 12 morning time intervals starting at 8 am with duration from 10 minutes to 120 minutes with 10 minute step.

In this experimental set, 1440 problem instances are solved by each algorithm.

5.4 Perturbation rate variation

For each of the 4 control areas, one hour horizon scenario starting at 8 am is considered. Starting from the original timetable, the rate of randomly selected delayed trains for assigning the 5 to 15 minutes delay varies from 10% to 60% . The percentage is increased at a 10% step. We generates 30 scenarios for each of the perturbation percentage value.

In this experimental set, 720 problem instances are solved by each algorithm.

6 Results

On Figures 7 and 8, we introduce three types of graphs in order to explain the results, separately for each case study and as a function of horizon size and perturbation rate:

(a) in the first column, the curves indicate the mean frequency at which an algorithm returns the best solution among those found by both CP and MILP. The green squares (respectively the red circles) presents the mean performance of CP (respectively MILP). For example, in Figure 7, for case study #1, CP and MILP provide 100 % of the best solutions for 10 minutes horizon instance set: they always return solutions with the same values. In the same figure, CP, respectively MILP, provides 70 %, respectively 53 %, of the best solutions for the 120 minutes horizon instance set.

(b) in the second column, reports the boxplots show the distribution of the differences of objective values between the two algorithms. Observing these distributions, we can better undestand the performance results of (a) curves. For example, in Figure 7, for case study #2, in the curve (a), CP, respectively MILP, provides 80 %, respectively 100 %, of the best solutions for the 70 minutes horizon instance set. However, for this instance set, (b) boxplots indicate that the objective values of both algorithms are very close as the median of the differences is close to zero. The y axis reports the difference between the best objective value given by MILP minus the one given by CP. This means that the points above the origin are those for which CP provides better solutions than MILP,

(c) in the third column, the curves quantify the frequency at which an algorithm is able to prove the optimality of the returned solution during the allowed CPU time.

6.1 Horizon size variation

Figure 7 allows a comparison of the solution quality given by CP and MILP on each case study for the horizon size variation. The x-axis represents the horizon size in minutes, which

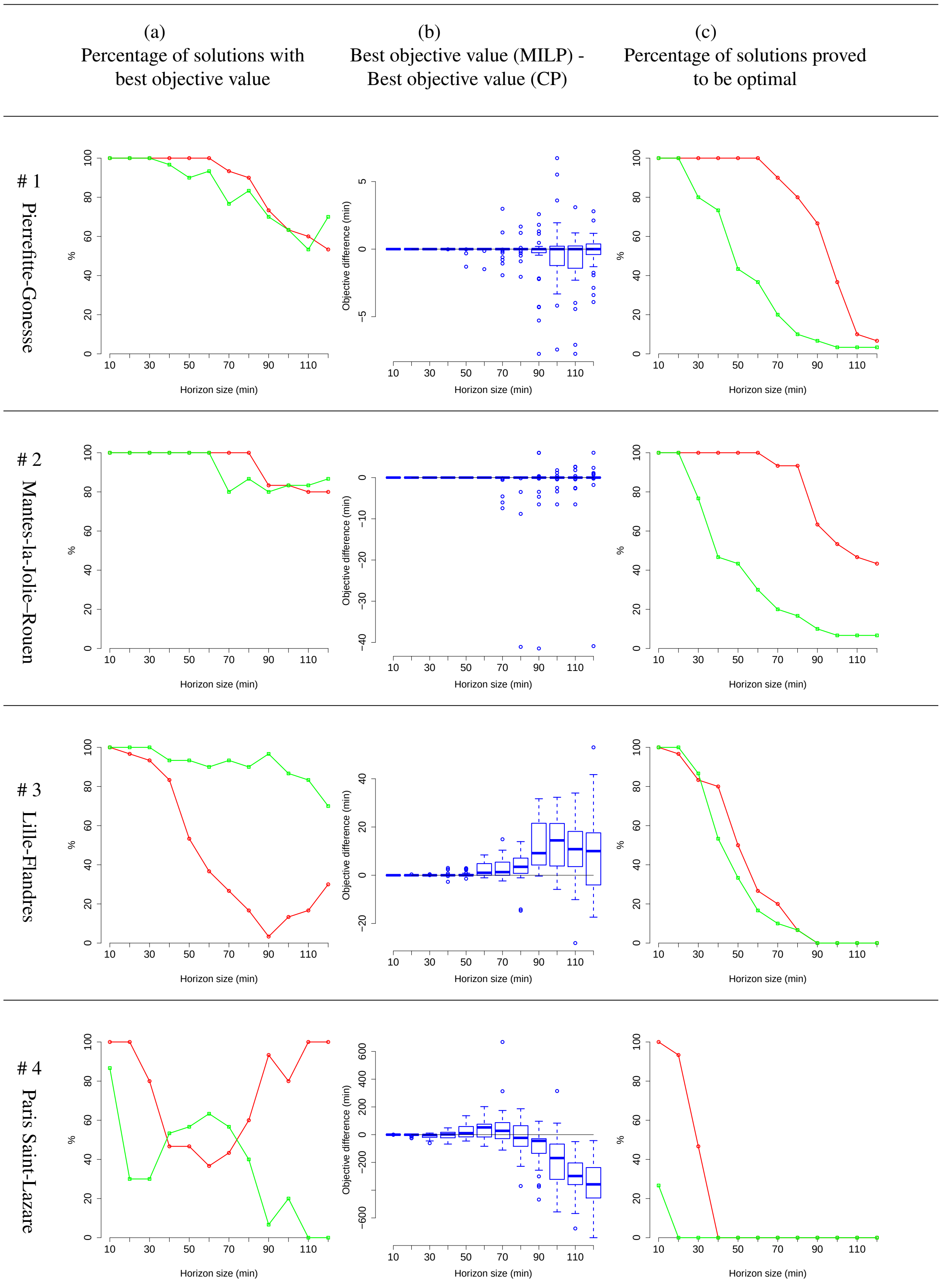


Figure 7: Experimental results for horizon size variation. In column 1 and 3, red circles for MILP, green squares for CP. In column 2, difference MILP minus CP.

is incrementally increased for this set of experiments.

These results show some general tendencies, common for all case studies:

- MILP has a better ability to prove optimality than CP, it outperforms CP for almost all horizon size problems according to this indicator, shown in column 3.. The # 1, # 2 # 3, # 4 case study instances are of increasing difficulty from this point of view: the success frequency of MILP decreases sharply after 90, 40, 30 and 20 minutes horizon size respectively,
- Regarding the best objective values indicator shown in column 1, CP and MILP have generally the same performance for small horizon sizes. As the horizon size increases, CP outperforms MILP starting from the following sizes:
  - around 110 minutes in # 1,
  - around 100 minutes in # 2,
  - before 10 minutes in # 3, note that after 90 minutes, the frequency of MILP increase,
  - around 40 minutes in # 4, note that after 60 minutes, the frequency of MILP increase and another crossing that reverses the performances order of the algorithms occurs around 70 min.

Further analysis shows that, for difficult instances of case studies (#3 and #4), MILP is not able to provide a solution during the all-routes solution phase. Therefore, the solution initially found during the fixed-route search phase is returned. Conversely, in these instances, CP provides poor solutions during the fixed-route search phase and is able to improve them during the all-routes solution phase. However, the improvement is not large enough to overtake the quality of the MILP ones.

6.2 Perturbation rate variation

The results reported in Figure 8 consider another difficulty parameter, namely the rate of perturbations with a fixed one hour size horizon. The x-axis of the plotted curves is the percentage of delayed trains: six configurations of perturbed scenarios are reported with 10 % to 60 % of delayed trains.

The results show different trends according to the case studies:

- For the instances of case studies # 1 and # 2, the best objective curves are close and does not show an important variation as for the case of horizon size variation. Regarding the optimality proof frequencies, MILP have reaches values above 80% , whatever the level of perturbations. This is not the case for CP whose frequencies decrease sharply after the first increase of perturbation level.
- For the instances of case studies # 3 and # 4, the one hour horizon scenarios considered are difficult to solve to the optimum, whatever the level of perturbation. The optimality proof rate of MILP for the instances of case study # 3 decreases under 10 % after 30 % of delayed trains, and the optimality is never proven for instances of case study # 4.

  Regarding the best objective indicator, CP either keeps good frequencies all along the level of perturbation, either improves over MILP above 20 % rate of delayed trains.

(a) Percentage of solutions with best objective value

(b) Best objective value (MILP) - Best objective value (CP)

(c) Percentage of solutions proved to be optimal

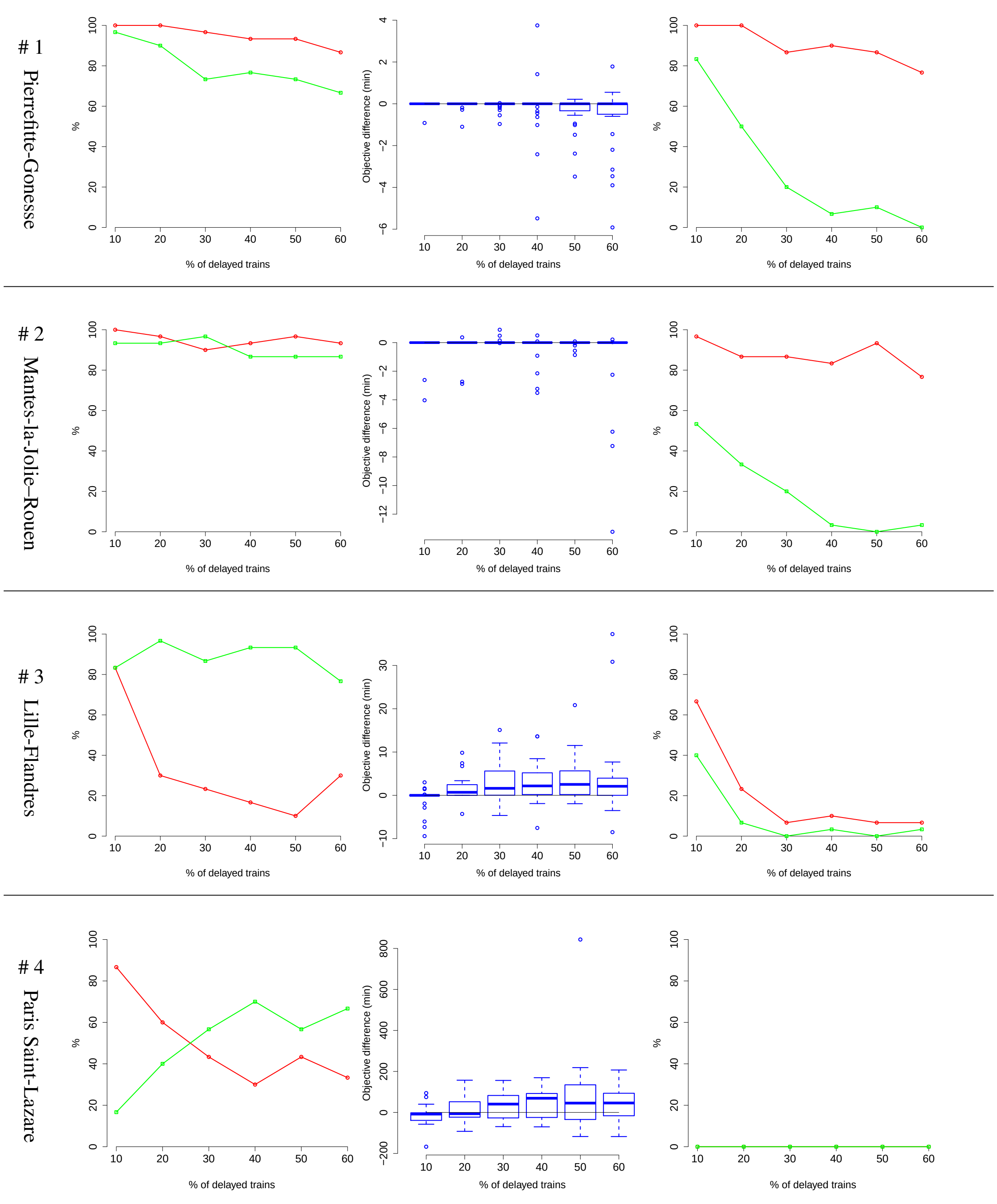


Figure 8: Experimental results for perturbation rate variation. In column 1 and 3, red circles for MILP, green squares for CP. In column 2, difference MILP minus CP.

The boxplots confirm the trend: the more we have perturbations, the more CP gives better objective values than MILP.

7 Conclusion

In this paper, we propose a new formulation of a Constraint Programming for the real-time Railway Traffic Management Problem. It is based on the concept of Time-interval variables which simplifies the formulation of optional activities. The solution method integrates Constraint Programming and Mathematical Programming techniques. Preliminary results show good performance of the proposed approach compared with the state-of-the art RECIFE-MILP algorithm.

The model is fed by two types of information: static information and dynamic information. Static information includes tds characteristics, block limits, number of aspects, platforms positions, train set parameters ... Dynamic information includes scheduled arrival and departure of trains in stations, running and clearing times of tds for trains and train delays. All information was provided by French railways and information on delays are supposed to be provided by a traffic state prediction module a few minutes before the start of the scenario. The prediction module can be based on simulation, statistics or an artificial intelligence learning technique.

This research is an addtional contribution toward the applicability and relevance of the approache of a microscopic model to tackle real-time control of traffic perturbations. Previously, the output of the European project ON-TIME Quaglietta et al. (2016) provided a proof-of-concept of a framework where the RECIFE-MILP algorithm were used in a closed-loop with a simulation environment and tested on different networks of European infrastructure managers.

As perspectives of this research, we will exploit the use of state resources to better manage opposite direction conflicts, hence to improve algorithm performance. In addition, an in-depth analysis of weaknesses and strengths of RECIFE-MILP and RECIFE-CP should allow the proposal of a hybrid solution approach.

References

Cacchiani, V., Huisman, D., Kidd, M., Kroon, L., Toth, P., Veelenturf, L., and Wagenaar, J. (2014). An overview of recovery models and algorithms for real-time railway rescheduling. *Transportation Research Part B: Methodological*, 63:15 – 37.

Fang, W., Yang, S., and Yao, X. (2015). A survey on problem models and solution approaches to rescheduling in railway networks. *IEEE Transactions on Intelligent Transportation Systems*, 16(6):2997–3016.

Großmann, P., Hölldobler, S., Manthey, N., Nachtigall, K., Opitz, J., and Steinke, P. (2012). Solving periodic event scheduling problems with SAT. In *Advanced Research in Applied Artificial Intelligence - 25th International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2012, Dalian, China, June 9-12, 2012. Proceedings*, pages 166–175.

Hansen, I. A.; Pachl, J., editor (2008). *Railway Timetable & Traffic - Analysis, Modelling, Simulation*. Eurailpress.

Kroon, L., Huisman, D., Abbink, E., Fioole, P.-J., Fischetti, M., Maróti, G., Schrijver, A., Steenbeek, A., and Ybema, R. (2009). The new Dutch timetable: The OR revolution. *Interfaces*, 39(1):6–17.

Laborie, P. and Godard, D. (2007). Application to single-mode scheduling problems. In *3rd Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA).*, page 276–284.

Laborie, P. and Rogerie, J. (2008). Reasoning with conditional time-intervals. In *Twenty-First International FLAIRS Conference*.

Laborie, P. and Rogerie, J. (2016). Temporal linear relaxation in IBM ILOG CP Optimizer. *Journal of Scheduling*, 19(4):391–400.

Lusby, R. M., Larsen, J., Ehrgott, M., and Ryan, D. (2011). Railway track allocation: models and methods. *OR Spectrum*, 33(4):843–883.

Mascis, A. and Pacciarelli, D. (2002). Job-shop with blocking and no-wait constraints. *European Journal of Operational Research*, (143):498–517.

Pellegrini, P., Marlière, G., and Rodriguez, J. (2014). Optimal train routing and scheduling for managing traffic perturbations in complex junctions. *Transportation Research Part B: Mathodological*, 59:58–80.

Quaglietta, E., Pellegrini, P., Goverde, R. M., Albrecht, T., Jaekel, B., Marliere, G., Rodriguez, J., Dollevoet, T., Ambrogio, B., Carcasole, D., Giaroli, M., and Nicholson, G. (2016). The ON-TIME real-time railway traffic management framework: A proof-of-concept using a scalable standardised data communication architecture. *Transportation Research Part C: Emerging Technologies*, 63:23 – 50.

Rodriguez, J. (2007). A constraint programming model for real-time train scheduling at junctions. *Transportation Research Part B: Mathodological*, 41:231–245.

Schrijver, A. and Steenbeek, A. (1994). Timetable construction for railned. Technical report, C. W. I. Center for Mathematics and Computer Science Amsterdam. in Dutch.

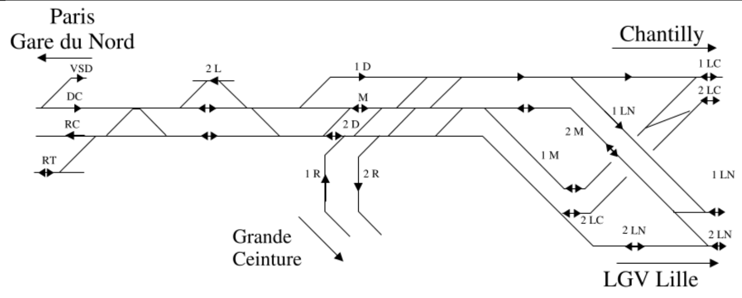
Spzigel, B. (1973). Optimal train scheduling on a single track railway. In Ross, M. e., editor, *OR'72*, pages 343–352. North Holland Publishing Co.

Vilím, P., Barták, R., and Čepek, O. (2005). Extension of O(N Log N) Filtering Algorithms for the Unary Resource Constraint to Optional Activities. *Constraints*, 10(4):403–425.

Vilím, P., Laborie, P., and Shaw, P. (2015). Failure-directed search for constraint-based scheduling. In *International Conference on AI and OR Techniques in Constriant Programming for Combinatorial Optimization Problems*, page 437–453. Springer, Cham.

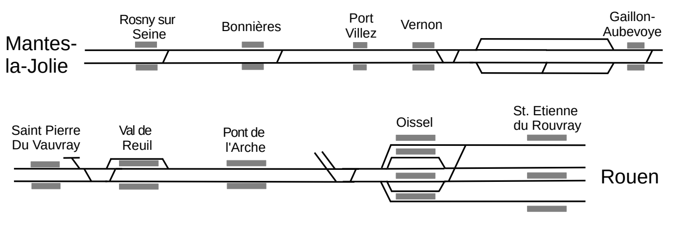
Appendix A - Case-studies layouts

1



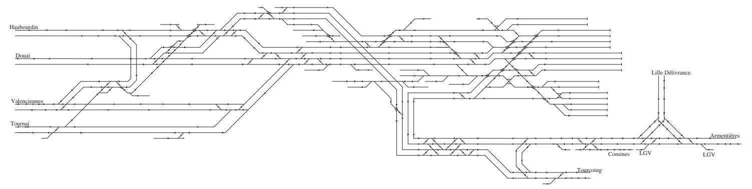
Pierrefitte-Gonesse

2



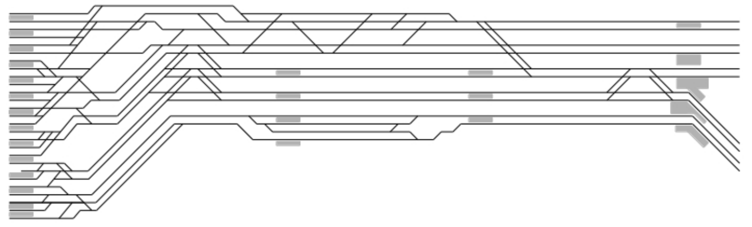
Mantes-la-Jolie–Rouen

3



Lille-Flandres

4



Paris Saint-Lazare