A Conditional Time-Intervals formulation of the real-time Railway Traffic Management Problem

Grégory Marliere, Sonia Sobieraj Richard, Paola Pellegrini, Joaquin Rodriguez

Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 54-2 (2021) 187–194

A Conditional Time-Intervals formulation of the real-time Railway Traffic Management Problem

Grégory Marlière * **Sonia Sobieraj Richard** *
Paola Pellegrini ** **Joaquin Rodriguez** *

* *COSYS-ESTAS, Univ. Gustave Eiffel, Campus de Lille, F-59650 Villeneuve d'Ascq, France (e-mail: {gregory.marliere, sonia.sobieraj-richard, joaquin.rodriguez}@univ-eiffel.fr).*
** *COSYS-LEOST, Univ. Gustave Eiffel, Campus de Lille, F-59650 Villeneuve d'Ascq, France (e-mail: paola.pellegrini@univ-eiffel.fr).*

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 Constraint Based Scheduling (CBS) formulation of the rtRTMP. This new formulation is based on the concept of conditional time-interval variables provided in the Ilog CP-optimizer library. A time-interval variable is the time interval in which an activity is executed, but it can also be a specific value "⊥" meaning the activity is non-executed. The new formulation exploits this new kind of variables and specific constraint propagation algorithms which contribute to the efficiency of the solution methods. 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. Among more recent publications, we can also mention Bettinelli et al. (2017) which proposes a very fast heuristic solution algorithm but almost no information is provided on the considered instances. Kumar et al. (2018) developed a constraint programming (CP) model tested at India's largest train station, but rail routes are limited to two unary resources. Van Thielen et al. (2019) also proposes a heuristic which solves the problem in a very short time and which can be used for very large and complex networks. Josyula et al. (2020) shows that a mixed-integer programming (MIP) algorithm outperforms an heuristic for the five minimization objectives considered but the heuristic is faster in some specific instances.

The surveys point out that integer programming (IP) and MIP models are the most popular approaches along with graph models, while 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 network 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 require less memory for the instance formulation. It is also worthwhile mentioning that despite the

10.1016/j.ifacol.2021.06.046

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), named RECIFE-CP hereinafter, 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. Moreover, we initiate a comparison of the performance achievable by a CBS model with the ones of other algorithms.

To do so, in this paper, we present a new CBS model based on conditional time-interval variables of the rtRTMP, RECIFE-CPI. This formulation has been validated with experiments on a large set of instances. The performances of this new CBS model has been compared with the one of the state-of-the art RECIFE-MILP heuristic (Pellegrini et al., 2014).

2. CBS MODEL

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 model, 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 constraints.

This modelling approach for train scheduling was first proposed by Spzigel (1973), who formulated 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.

2.2 Microscroscopic model of the rtRTMP

We consider a microscroscopic model of the rtRTMP where train movements are controlled with a fixed block signalling system. It decomposes a train journey into a sequence of activities. Each activity is an elementary movement of the train through a track detection section

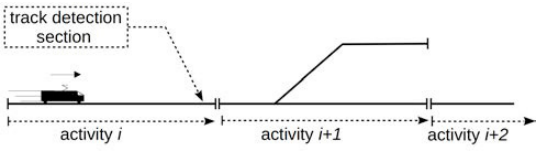


Fig. 1. Train movement as a sequence of activities.

(tds), as illustrated in Figure 1 (Rodriguez, 2007). 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 shall be detected by a tds at any point in time. Hence, tds's are modelled as unary resources. A unary resource is a resource that can be used by at most one activity at any point in time. However, an exception occurs if a train set is split 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

The temporal constraints between activities allow modeling main 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 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-CPI, we define two 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 one. The second activity associated to a tds includes the first one and is extended to contain the reservation time to consider the length of the train ("clearing time" in Figure 2) and comply with the blocking time theory constraints (Hansen, 2008). This second type of activities are shown with striped rectangles in Figure 2 for the case of a 3-aspect block signalling sytem. All tds's of a block must be reserved when the train reaches the sight distance point of the previous block. When there are switches within a block, the existence of multiple tds's allows the interlocking system to release and set as soon

[1] Additional characteristics are ommitted , e.g., time for clearing signal or release time, to simplify the presentation. For a complete list we refer the reader to Pellegrini et al. (2014)

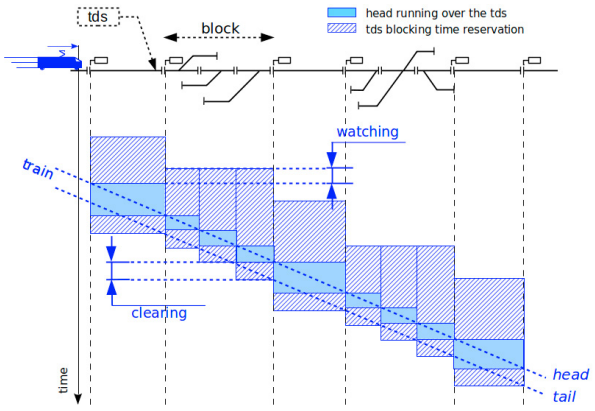


Fig. 2. Head running activity and tds blocking time reservation

as possible the sequence of incompatible routes and then safely optimise traffic. The sectional route release of the interlocking system is modelled with separated activities for each tds of a block section (c.f. Figure 2).

In the previous RECIFE-CP (Rodriguez, 2007), we consider only one activity for each tds and the blocking time theory constraints are expressed according the start and end of these activities.

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 Conditional time-interval variables

In this section, we briefly introduce the concept of conditional time-interval variables. In many works in the scheduling field, the main decisions are assigning resources to activities and scheduling activities. However, in industrial applications, it can be also necessary to consider the choice of specific activities that will be executed in the final schedule, for example when there are alternative production processes in response to an order. This translates into the introduction of *optional activities*.

Vilím et al. (2005) introduced 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 a period of interest in a schedule. In many cases, as in the problem modelled here, a time-interval variable is the period in which an activity is executed.

Let $\perp$ be a value indicating that the period of interest is not present in the solution schedule i.e., the corresponding 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 $\underline{a}$ denote a fixed time-interval variable, then $\underline{a} = \perp$ means that the activity is non-executed (not present in the solution schedule); $\underline{a} = [s, e)$ 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 it was 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.

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.

2.5 Optional activities for alternative route choices

For each route choice we define one specific sequence of optional head running activities. It should be noted that each blocking time reservation activity covers a head running activity, thus there is also a sequence of optional blocking time reservation activities "enveloping" the sequence of head running activities.

To illustrate the model, let us first consider the example of a train that has two alternative routes r_1 and r_2 in Figure 3. 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 fully exploit 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 to the last tds activity gives a sequence of activities for r_1 and a different one for r_2. In the example, the activities for the run through tds_1 and tds_7 are merged as they have the same characteristics for r_1 and r_2 as illustrated in Figure 4. Conversely, for the 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_2 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.

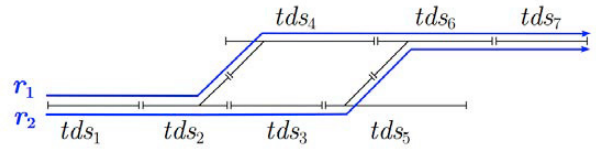


Fig. 3. Example of tds route sequences

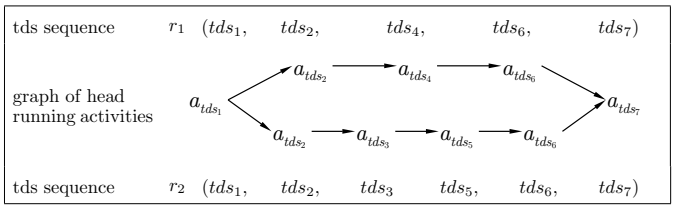


Fig. 4. Sequences of activities and tds's for the two routes of Figure 3

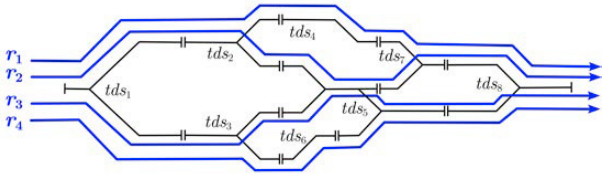


Fig. 5. Example of tds sequences to illustrate group constraints

It can be noted that the RECIFE-CP approach for the re-routing decisions is very different : a train run is modelled with only *one sequence of activities* whatever the chosen route. To do so, "dummy" tds's are introduced and the blocking time constraints result in a tricky formulation (Rodriguez, 2007).

2.6 Global constraints on groups of activities

To improve the constraint propagation and hence the performance of the solution method, we create 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 activity. Derived high-level activities can be used with any temporal constraint in the same way as low-level ones.

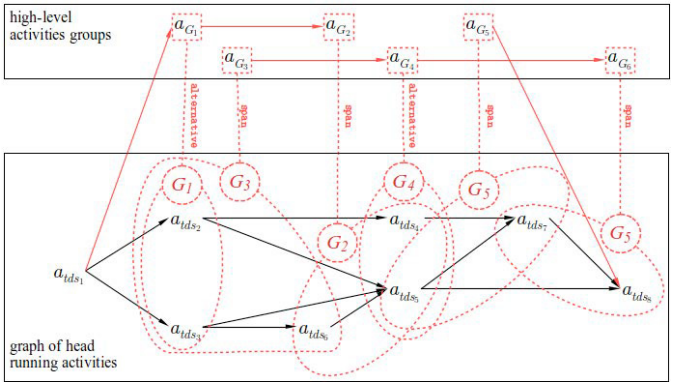


Fig. 6. Graph of activities for the example of Figure 5

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 $\{a_1, \ldots, a_n\}$; and the second group **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 and only if none of activities $\{a_1, \ldots, a_n\}$ is executed (Laborie and Rogerie, 2016).

To illustrate the definition of high-level activities and the group constraints, let us consider the 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 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 $a_{tds_1} \prec a_{G_1}$. 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).

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 tds's, respectively,
$R_t \subseteq R$	set of routes that can be used by train t,
TDS^r	set of tds's composing route r,
ty_t	type corresponding to train t (indicating characteristics as weight, length, engine power, etc.),
$TDS_t \subseteq TDS$	set of tds's that can be used by train t ($TDS_t = \bigcup_{r \in R_t} TDS^r$),
$PL \subset TDS$	set of tds's corresponding to platforms (if the control area includes a station),
$PL_{t,t'} \subset PL$	set of tds's 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,
$p_{r,tds}$	tds's preceding tds along route r,
$rt_{ty,r,tds}$	running time of tds along route r for a train of type ty,
$ct_{ty,r,tds}$	clearing time of 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 tds's 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 the 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 $r \in R_t$:

$$x_{t,r} = \begin{cases} 1 \text{ if } t \text{ uses } r, \\ 0 \text{ otherwise.}; \end{cases}$$

The objective is the minimization of the total 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 constraints, let us consider the following additional notation:

$s(a), e(a),$ $d(a), pres(a)$ 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's 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, and false otherwise.

The constraints are:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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 its 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 relation 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 end 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 according to the interlocking system $ref_{r,tds}$ 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^t_G 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 when they perform the turnaround. Constraints (14) ensure that a minimum separation time must separate the arrival and departure of trains using the same rolling stock. Constraints (15) ensure the tds where the turnaround takes place is utilized for the whole time between t''s arrival and ts 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^{exit}_t and D^{arr}_t 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 we consider is the one implemented in Ilog CP-optimizer library. It 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.*

5. EXPERIMENTS

5.1 Case-studies

In the experimental analysis, we test 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 Mantes-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 is in Appendix A. Notes that the second line of Table 1 gives the values of $|R|$ and the eighth line gives the bounds of the number of routes per train (bounds of $|R_t|$).

5.2 Experimental settings

The experiments involve RECIFE-CPI (named CPI) and RECIFE-MILP (named MILP) in order to compare their performance in various cases.

Both algorithms are configured to perform a two-step approach:

- in the first step, a maximum of 10 seconds CPU wall-clock 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 wall-clock time is imposed for the solution of these two steps on an Intel(R) Xeon(R) CPU E5-2643 v4 @ 3.40GHz, 24 cores, 128go RAM.

5.3 Perturbation scenarios

For each of the 4 control areas, we generate 30 perturbation 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 perturbation scenario, we select 12 morning time horizons starting at 8 am with duration from 10 minutes to 120 minutes with 10 minute step. Therefore we consider 360 instances for each case study.

In total, 1440 problem instances are solved by each algorithm.

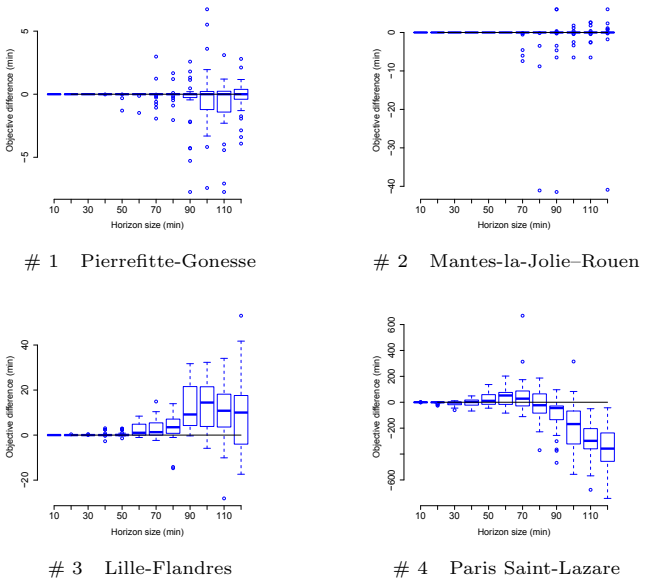
6. RESULTS

Figure 7 allows a comparison of the solution quality given by CPI and MILP on each case study. The x-axis represents the horizon size in minutes. The boxplots show the distribution of the differences of objective values between the two algorithms. Observing these distributions, we can analyze the algorithms' performance. For cases studies #1 and #2, the boxplots indicate that the objective values of both algorithms are very close: the median of the differences is close to zero. For cases study #3, the performance of both algorithms are also very close for instances until 70 minutes horizon, then CPI outperforms MILP. Conversely, for case study #4, both algorithms are also very close for instances until 80 minutes horizon, then MILP outperforms CPI. Further analysis shows that, for difficult instances (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. In these instances, CPI provides poor solutions during the fixed-route search phase but is able to improve them during the all-routes solution phase. However, the improvement is not sufficient for it to outperform MILP.

Table 1. Case-studies characteristics

	Junction	**Line**	**Stations**	
	# 1 Gonesse	*# 2 MLJ-Rouen*	*# 3 Lille*	*# 4 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/Train	5-13	1-24	1-71	1-9
Turnarounds	0	6	298	606

Fig. 7. Experimental results: Best objective value (MILP) - Best objective value (CPI)



7. CONCLUSION

In this paper, we proposed a new Constraint Programming model for the real-time Railway Traffic Management Problem. It is based on the concept of Time-interval variables which simplifies the formulation of alternative route choices and the blocking time constraints. 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.

This research is a contribution toward the proof of the applicability and relevance of use of a microscopic model to tackle real-time management 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 was used in a closed-loop with a simulation environment and tested on different European networks.

As perspectives of this research, we will exploit the use of state resources to better manage opposite direction conflicts, hence to improve our model. In addition, an in-depth analysis of weaknesses and strengths of both RECIFE-MILP and RECIFE-CPI will likely allow the proposal of a hybrid very well performing solution approach.

REFERENCES

Bettinelli, A., Santini, A., and Vigo, D. (2017). A real-time conflict solution algorithm for the train

rescheduling problem. *Transportation Research Part B-methodological*, 106, 237–265.

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*, 166–175.

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

Josyula, S.P., Krasemann, J.T., and Lundberg, L. (2020). An evaluation framework and algorithms for train rescheduling. *Algorithms*, 13(12).

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.

Kumar, R., Sen, G., Kar, S., and Tiwari, M.K. (2018). Station dispatching problem for a large terminal: A constraint programming approach. *INFORMS Journal on Applied Analytics*, 48(6), 510–528.

Laborie, P. and Godard, D. (2007). Application to single-mode scheduling problems. In *3rd Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA)*., 276284.

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

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

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: Methodological*, 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: Methodological*, 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 M.e. Ross (ed.), *OR'72*, 343–352. North Holland Publishing Co.

Van Thielen, S., Corman, F., and Vansteenwegen, P. (2019). Towards a conflict prevention strategy applicable for real-time railway traffic management. *Journal of Rail Transport Planning & Management*, 11, 100139.

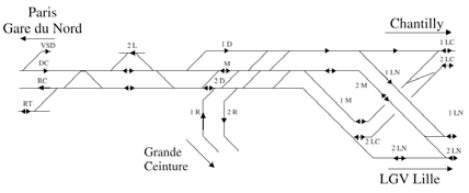
Vilím, P., Barták, R., and Čepek, O. (2005). Extension of O(N Log N) Filtering Algorithms for the Unary Resource Constraint to Optional Activities. *Constraints*, 10(4), 403–425.

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*, 437453. Springer, Cham.

Appendix A. CASE-STUDIES LAYOUTS

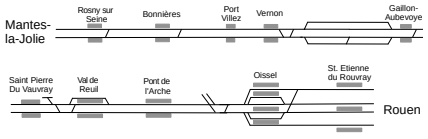
Pierrefitte-Gonesse

1



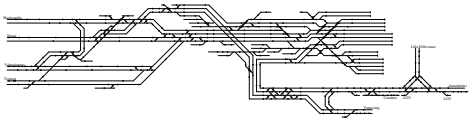
Mantes-la-Jolie–Rouen

2



Lille-Flandres

3



Paris Saint-Lazare

4