A conditional time-intervals formulation of the real-time Railway Traffic Management Problem

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

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

Grégory Marlière[a], Sonia Sobieraj Richard[a], Paola Pellegrini[a], Joaquin Rodriguez[a,]

[a]*COSYS-ESTAS, Univ. Gustave Eiffel, Campus de Lille, 20 rue Elisée Reclus, Villeneuve d'Ascq, F-59650, France*

Abstract

This paper tackles the real-time Railway Traffic Management Problem (rtRTMP). It is the problem of finding optimal choices for train schedules and routes to minimize delays due to conflicts. We present a new Constraint Programming (CP) algorithm for the rtRTMP. The new formulation at the basis of this algorithm exploits the concept of conditional time-interval variables provided in the CP Optimizer library. A time-interval variable assumes a value representing either the time interval in which an activity is executed, or a quantity "$\perp$" indicating that the activity is non-executed. The new formulation exploits this new kind of variables, and specific constraint propagation techniques contribute to the efficiency of the algorithm. This efficiency is assessed in a wide experimental analysis based on five railway control areas. The algorithm performance is compared to the one of the state-of-the-art algorithm RECIFE-MILP based on a mixed-integer linear programming (MILP) formulation. Moreover, an hybridization of RECIFE-MILP and the proposed algorithm is proposed. It often outperforms the two individual approaches, while the opposite never happens.

Keywords:
Real Time Traffic Management, Train Dispatching Problem, rerouting and reordering trains, Minimize secondary delays, Constraint Propagation

1. Introduction

The design of railway services is a complex process. Here, the planning of train schedules and necessary resources is made at the tactical level. However, at the operational level, operators need to make control decisions to manage perturbations as late train arrivals, bad weather conditions or extra passenger flow. Indeed, at the tactical level, time allowances are introduced in train schedules, at station arrival and departure and other relevant control points. The values of these time allowances should make it possible to recover from the consequences of perturbations. Nevertheless, in railway networks with dense and heterogeneous traffic, time allowances are not always sufficient and delays due to perturbations propagate in a snowball (or domino) effect. Delays due to propagation are called secondary delays. Secondary delays occur when two trains running at the planned speed would claim a track section at the same time. When this happens, a conflict emerges. An order decision must be made, and one of the two competing trains must be slowed down or stopped. Conflicts are managed by dispatchers in charge of traffic within a specific control area. Dispatchers aim to reduce the negative impacts of conflicts by means of appropriate decisions through train retiming, rerouting and reordering. The problem of optimally making these decisions is formalized at microscopic level as the real-time Railway Traffic Management Problem (rtRTMP) (Pellegrini et al., 2014). The solution of the rtRTMP must satisfy the constraints imposed by the signalling system and other service needs such as train connections. The time available for making decisions is limited by the real-time nature of the problem. To consider the use of a solution algorithm for rtRTMP in a real-time traffic control loop, its computation time must be short. This time must be less than the period of the control loop. A review of published algorithms for rtRTMP used in a real-time traffic control loop performed in the European X2Rail-4 project is reported by Söhlke and Rodrigues (2022). The values envisaged for the period of the control loop go from 60 to 480 seconds. These values may seem large, but they must be considered with regard to the specificities of the rail traffic management process. Although full automation of traffic management can be expected in the future with advances in rtRTMP algorithms, some railway infrastructure managers are keen to keep the human-in-the-loop principle. With the human-in-the-loop, dispatchers are meant to assess and possibly modify algorithm decisions. In this situation, it is not possible to solicit the dispatcher with new solutions too frequently. How to choose

the best control loop period is an open question that will require extensive field experimentation. In the meantime, a period of a few minutes is generally considered to be reasonable. However, it shall be emphasized that the period of the real-time traffic control loop should not be confused with the response time of an algorithm designed to be triggered by a dispatcher for a specific situation. In the latter case, infrastructure managers wish a shorter response time, of around 30 seconds (Söhlke and Rodrigues, 2022). Indeed, experiments must also be carried out with dispatchers to find the compromise between the computation time wished and the quality of the solution provided.

A rich literature exists on approaches tackling various variants of the rtRTMP. We refer the interested reader to Lusby et al. (2011), Cacchiani et al. (2014), Fang et al. (2015) for relevant literature surveys. Following the classification proposed by Cacchiani et al. (2014), the rtRTMP variant we tackle focuses on dealing with perturbations (or disturbances), considers microscopic level of details of the railway system, and optimizes a train-centered objective function. The surveys point out that integer and mixed integer linear programming (ILP and MILP, respectively) are the most popular techniques used for this variant, along with graph models. Instead, constraint programming (CP) ones are very seldom used. Nevertheless, CP has some undeniable merits which make it interesting to develop a new CP model for the rtRTMP.

Among the advantages of CP, there is the ability to handle arbitrary (non-linear) constraints and to use propagation to tighten variable domains. CP also has the general advantage of well capturing the combinatorial complexity of the problem. However, a weakness of CP is that it has to perform the constraint propagation for each constraint individually to obtain a global view. This weakness can be compensated with high-level global constraints that represent a group of simpler constraints to improve constraint propagation.

Many benchmark studies between CP and MILP conclude that neither outperforms the other, except for specific problems. Since the 1990s, much research has been conducted to combine their complementary strengths (Hooker and van Hoeve, 2018). One approach for the integration of the two techniques is to create for a given CP model, a linear programming (LP) model from a set of constraints which represent a specific combinatorial structure. The linear relaxation of this LP model gives bounds and can be used to guide the search.

The development of the new CP model for rtRTMP presented in this

article is based on the CP Optimizer library (Laborie et al., 2018). It provides a component that exploits the linear relaxation of a scheduling model to handle temporal decisions.

The main contributions of this paper are summarized as follows:

- A new CP microscopic formulation of the rtRTMP based on CP conditional-interval variables is proposed and validated,
- In our model, train retiming, reordering and rerouting decisions are made considering characteristics of the signalling system such as track detection sections, block signal limits, multi-aspect signalling and route-lock sectional-release interlocking principle,
- The new CP model allows a compact formulation with relatively few variables and constraints, enabling to push the applicability to large problem instances,
- Global constraints and high-level temporal constraints between groups of activities have been introduced to allow partial assignment of routes, along with an improved constraint propagation on shared track sections between several train route options,
- The experimental study integrates a benchmarking between two different formulation paradigms, i.e., CP and MILP, for the rtRTMP, to highlight their strengths and weaknesses,
- Experiments are conducted based on a larger set of real-world railway instances in comparison with most existing publications. These are diversified instances of increasing difficulty, based on four infrastructure case studies and covering many different configurations such as junctions, stations and bi-directional line corridors,
- A fifth artificial control area representing a generic station has been designed and made available for result reproducibility and benchmarking as the data of the four real control areas are protected by confidentiality clauses,
- A solution algorithm using the hybridization of a CP and a MILP model is also proposed and validated. To the best of our knowledge this kind of hybridization has never been applied to solve the rtRTMP.

The paper is structured as follows. Section 2 reviews contributions to the three levels of control in the field of real-time rail traffic management. Sections 3 explains the concept of the conditional time-interval variables. Section 4 describes the general modeling principles we consider for the rtRTMP. Section 5 details RECIFE-CPI, including the novel formulation and the solution process. Section 6 sketches the functioning of the RECIFE-MILP algorithm. Section 7 reports on the experiments setup and discusses the results obtained when comparing RECIFE-CPI and RECIFE-MILP. Then, Section 8 proposes and assesses experimentally the hybridization of these algorithms. Finally, Section 9 reports our conclusions.

2. Literature review

The field of real-time rail traffic management has given rise to numerous research works over the past decades. This section gives a brief overview of some of the results obtained so far. Many rail networks organize real-time rail traffic management into three control levels : (1) train control level, (2) local area control level, and (3) network control level. To give a broader view of the context of the rtRTMP addressed in this article, the literature review is structured according to these three levels of control.

2.1. Level 1 - Train control

At train control level, the control variables are the train tractive and braking effort. Before the start of the service, the driver is informed of the main time points of the journey, in addition with other technical constraints of the track or the rolling stock. During the trip, the driver has to adhere to the planned time points as much as possible. In addition, the train driver may also try to minimize power consumption. To do so, the driver applies specific tractive and braking effort along the journey. The problem of finding the optimal values of tractive and braking effort and their sequence is named train trajectory planning problem. This problem of optimal train control is very complex, it has been addressed by many researchers from the 1960s and still has open issues. As in other application domains, three basic approaches address this optimal control problem: dynamic programming, indirect methods, and direct methods (Diehl et al., 2006).

Dynamic programming (DP) simplifies the problem by breaking it down into simpler subproblems, in a recursive manner. Approximations by discretization is needed to apply DP to discrete-time systems with continuous

state spaces. These approximations leads to exponential growth of the computation time (Wang et al., 2013).

Indirect methods can be sketched as "first optimize, then discretize". These methods derive from the original problem a boundary value problem in ordinary differential equations. Based on the Pontryagin Maximum Principle, the latter hard to solve problem is reformulated to obtain optimal driving regimes (i.e., traction, cruising, coasting, and braking) and their sequence (Howlett, 1990, 2000; Albrecht et al., 2016a,b). Most published articles use the indirect approach with (heuristic) constructive methods to find (sub)optimal driving strategies based on optimality conditions, with simplifications or assumptions on driving regimes (Goverde et al., 2021). The applicability of these models to real situations has been successfully achieved. Some models have been integrated into on-board driver advisory systems (DAS) deployed on some railway lines. The algorithms used in DAS are heuristic methods based on indirect methods due to their computational efficiency (Scheepmaker et al., 2017).

Direct methods can be sketched as "first discretize, then optimize". These methods transform the original problem into a finite dimensional non-linear programming problem. The latter is then solved with efficient numerical optimization methods. In the optimal train control literature, direct methods have been used only recently. Most approaches are based on pseudospectral methods (Wang et al., 2013; Wang and Goverde, 2016; Goverde et al., 2021).

When regenerative braking is possible, the optimal train control structure is extended with additional driving regimes: regenerative braking can be used for cruising or braking (Albrecht, 2010; Yang et al., 2015).

Some other researches on optimal train control consider the additional operational constraints due the signalling system and the interactions with other trains (Wang et al., 2014b; Wang and Goverde, 2016, 2017). These models of muti-trains control can be considered as a step toward traffic control of the upper control levels.

For more references and detailed analysis on train control optimization, the interested reader is referred to (Yang et al., 2016; Scheepmaker et al., 2017; Yin et al., 2017; Goverde et al., 2021).

2.2. Level 2 - Local area traffic control

At the local area level, control variables are mostly the departure/arrival time of trains at stations, the order of trains on track sections and the routes to be followed by trains. In specific situations, control variables can include

reservicing decisions, e.g., stop-skipping. The local control areas considered are junctions areas, stations areas or corridor lines. A dispatcher is in charge of analyzing the traffic of a local control area, identify trains involved in potential conflicts for track usage and make decisions to apply the appropriate control actions to reduce passenger inconvenience. The most used indicators are delays or travel time when passenger information is available. The literature is very abundant on this subject. Therefore, we present only a few representative publications. In the presentation, the first part concerns works which use a microscopic approach. The second one those which use a macroscopic approach. The third part is devoted to the integration of train control with local traffic control.

2.2.1. Microscopic approaches

The papers reviewed here consider microscopic approaches of local traffic control with train retiming, reordering rerouting actions. This optimization controller problem is known as rtRTMP.

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), Corman and Meng (2015), Fang et al. (2015) for recent literature surveys.

We focus on three main streams of research grouping approaches based on similar techniques: graph models, linear programming and CP approaches.

An important stream of research on the rtRTMP considers the alternative graph formulation defined during the European project COMBINE, in Mascis et al. (2002). In this formulation, each node of the graph corresponds to the moment when a train enters a section of track, named block, where only one train can circulate at each time instant. A fixed directed arc is defined for each train movement through a block, linking a pair of nodes. A pair of alternative directed arcs is defined for each pair of train movements through a common block whose running time intervals may overlap, i.e., where a conflict may occur. For each conflict, an arc is selected and to give priority to the train that minimizes delay propagation. The feasibility of the final schedule is checked by verifying that the selected arc do not create positive length cycles. The longest path between source and sink node represents the maximum train delay in the schedule. Without being exhaustive, we can mention D'Ariano et al. (2007), D'Ariano et al. (2008), Mannino and Mascis (2009), Corman et al. (2010b) as important contributions in this research stream. When also rerouting is considered, a metaheuristic algorithm is devoted to this facet of the problem, which iteratively defines train routes to

be used in the alternative graph (Samà et al., 2017).

(Xu et al., 2017) extended the AG model to take into account multi-aspect signalling, moreover an average speed choice variable for each signalling block allows to apply an adjusted blocking-time theory (Hansen, 2008) according to the chosen average speed.

The second research stream we recall here uses the solution of (M)ILP formulations of the problem as the main rationale of the approaches. These approaches can be classified according to the way in which time is modeled. In the first group, time is discretized in small intervals and time decisions are binary variables indicating if a train movement takes place or not in a specific interval. These are the so called time-indexed formulations. Caimi et al. (2012), Lusby et al. (2013), Meng and Zhou (2014) and more recently Reynolds et al. (2020) develop approaches based on time-indexed formulations of the rtRTMP. In the second group, continuous variables represent times, and the assume a value equal to the instant at which events (e.g., movements) occur. These are the continuous time formulations. Törnquist and Persson (2007), Khosravi et al. (2012), Pellegrini et al. (2014), Fischetti and Monaci (2017) and Luan et al. (2017a) are some relevant contributions proposing approaches based on continuous time formulations.

The stream of rtRTMP research proposing CP approaches is much thinner than for alternative graphs and (M)ILP. CP was initially used in railway for dealing with timetabling problems (Fukumori, 1980; Isaai and Singh, 2000; Kreuger et al., 2001) and capacity studies (Rivier et al., 2001; Ingolitto et al., 2004). To consider a microscopic representation of the infrastructure for the rtRTMP, Lamma et al. (1997) and Oliveira (2001) define a CP formulation that manages train movements at block level. Rodriguez and Kermad (1998) describes a formulation that manages train movements at the track detection section level. The extensions of this model enable to take into account unplanned accelerations/decelerations when trains are delayed (Rodriguez, 2007a), an effective handling of opposite direction conflicts (Rodriguez, 2007b) and an algorithm that implements a dynamic analysis of the search space called "texture measurement" (Rodriguez et al., 2010). Recently, Cappart and Schaus (2017) proposes a CP approach exploiting conditional time-interval variables, as we do in this paper. Specifically, train routing choices are modeled as an alternative global constraint, which links conditional time-intervals associated to each route choice. Another CP approach is later designed by Kumar et al. (2018). These two approaches are theoretically interesting, but they are shown to be capable of tackling only very

simple instances.

2.2.2. Macroscopic approaches

Preliminary works on traffic control optimization for local areas with macroscopic approaches only relate to metro lines. The schedules of these lines are based on the respect of headways which include the minimum train separation imposed by the signalling systems and a margin time. When train delays are above the margin time, the lines are unstable and give rise to the phenomenon of "bunching": train delays are propagated to the following trains and delays increase at each station due to the accumulation of passengers.

In Cury et al. (1980) and Araya and Sone (1984), an on-line optimization model for automated guideway transit systems is proposed which define retiming actions at stations to optimize trip time and discomfort of passengers.

Based on the previous formulations, Van Breusegem et al. (1991) define a control law for a conventional metro line by solving an unconstrained quadratic optimization problem.

In Fernandez et al. (2006) and Cucala et al. (2007), a model predictive control (MPC) framework is used to improve the efficiency of the dispatcher. An MPC strategy uses an explicit model of the system to be controlled, this model is used to predict its future behavior over a finite time interval. The on-line optimization control problem is solved by maximizing a performance criteria subject to constraints between inputs and outputs. Only the first step of the sequence of control actions is applied. The problem is solved again at the next step of the control loop using updated measurements and a shifted time interval (García et al., 1989).

More recently, Wang et al. (2022a) propose to tackle the joint train traffic regulation and passenger flow control problems. The formulation of the variation of train traffic is based on the one of Van Breusegem et al. (1991).

Adding or skipping stops, cancelling and short-turning trains are reservicing control actions to recover from delays when disturbances occur in urban lines. Many researchers have explored the benefits of these control actions.

Given an OD passenger demand, Wang et al. (2014c,a) address the problem of adding skip-stop decisions while minimizing energy consumption and passenger travel times. An MPC controller is used to generate the schedule. In Wang et al. (2015a), a new iterative convex programming (ICP) approach is proposed to solve the train scheduling problem.

Altazin et al. (2017) tackle the stop-skipping problem during perturbation situations. An integrated Integer Linear Programming model is proposed. Its objective function minimizes both recovery time and waiting time of passengers. In Altazin et al. (2020), a multi-objective optimization including all actions previously mentioned and combined with a macroscopic simulation is proposed. A tool embedding this algorithm has been tested by dispatchers and the very positive results have encouraged a future deployment in the control centers of Paris suburban lines.

2.2.3. Train control integration

The integration of train control and traffic control problems has been addressed with different approaches. A straightforward approach is the sequential one, which first solves the train scheduling problem with fixed (approximate) speed profiles then calculates the energy-efficient speed profiles of trains given the train orders on tracks (D'Ariano and Albrecht, 2006; Albrecht, 2009; Montrone et al., 2018).

A nearby approach is the iterative one with a closed-loop feedback control between the optimization model that feeds the optimized speed profiles to the scheduling optimization (Mazzarello and Ottaviani, 2007; Lüthi, 2009).

In Luan et al. (2017c, 2018a,c), three mathematical models address the integration of the two problems with retiming and reordering trains as traffic control actions.

A two-layer hierarchical MPC model of the two problems formulated with MILP models is proposed by Wang et al. (2022b), the multi-aspect signalling constraints inspired by Xu et al. (2017) are integrated in the lower-level train control formulation.

2.3. Level 3 - Network traffic control

Most of the railway networks have network traffic control centers. After perturbations which impact several local areas or (main) disruptions, operators are in charge to recover the normal traffic state as soon as possible.

Due to the extended area, solving a microscopic model over the whole network is not possible. This problem has been addressed more recently, the proposed approaches can be distinguished according to the assumptions made on the local areas (stations).

Schutter et al. (2002) propose to use an MPC framework for minimizing delays by breaking train connections at a cost in railway networks. The railway MPC problem is formulated as an extended linear complementary

problem. This formulation is easier to tackle as it involves the solution of a sequence of optimization problems with a convex feasible set.

Strotmann (2007) applies a microscopic model of alternative graph to the entire network, then decomposes the graph into different subgraphs corresponding to local areas. First, the local problems are solved with the subgraph models. The global feasibility of the local solutions is checked through a coordinator graph induced by the analysis of border elements of subgraphs. In case of infeasibility, a coordination procedure adds suitable constraints to the subgraphs to lead the global feasibility, the additional constraints are calculated from the mathematical properties of the coordinator graph. Further research improves the coordination procedure and the local solution method used (Corman et al., 2010a, 2012, 2014).

Kersbergen et al. (2016a) extend the switching max-plus-linear model of van den Boom and De Schutter (2006) to minimize delays at network level. The control actions considered are breaking connections, splitting joined trains, retiming, reordering and choosing the track between successive stations when two options are possible. A model predictive controller solves the dispatching problem formulated as a MILP model. A geography-based decomposition of the previous model is proposed in Kersbergen et al. (2016b). The partitions of the railway network are determined by solving a mixed integer quadratic program. The traffic of each sub-network is optimized by a proper MPC. There is no coordination control level, all sub-network controllers communicate and coordinate with each other through an iterative algorithm to reach a feasible solution. The efficiency of this geography-based decomposition is illustrated with perturbation scenarios on the entire Dutch network.

Wang et al. (2015b) develop an event-driven model for a urban rail transit network to define the optimal train retiming considering accurate passenger transfer behavior.

Luan et al. (2017b) propose three decomposition approaches to tackle the traffic management problem microscopically at network level, the microscopic model of Luan et al. (2017a) optimizes traffic at local areas level, like Kersbergen et al. (2016b). Then, a coordination agreement is obtained through the iterative exchange of values of coupling variables.

Toletti et al. (2020) proposes a coordination approach for local rescheduling algorithms applied to adjacent control areas. The local rescheduling algorithm is based on a resource conflict graphs as in Caimi et al. (2012). The coordination problem is modeled by applying a Lagrangian relaxation

to the complicating or coupling constraints as in Luan et al. (2020).

Cavone et al. (2022) propose an algorithm that combines MPC with both the macroscopic MILP model of Kersbergen et al. (2016b) and the mesoscopic MILP model of Blenkers (2015). The algorithm is tested on a disruption case study of the national Dutch railway network with 5 disruption durations combined with 20 short delays scenarios giving 100 instances.

2.4. Discussion and research motivations

From the review of the literature of the previous sections, it may be pointed out that the practical applicability of the results obtained at the train control level has been successfully achieved. In particular, this holds for the results that have been able to be transferred to DAS.

Demonstrating the applicability of the approaches for the local area and network control levels is much more difficult, mainly due to the complexity of setting up actual pilot tests.

We notice that most of the practical research in the field of rail traffic management automation considers either the detailed sequence of driving events at the train control level (c.f. Subsection 2.2.2), or the traffic events at the macroscopic or mesoscopic level, such as arrivals/departures of trains in stations (c.f. Section 2.1).

Almost all contributions in Section 2.2.2 consider only train retiming and reordering. However, they neglect events related to train interactions through the signalling system, such as a train passing within sight distance of a restrictive aspect of a signal, a train passing through block signal boundaries or the entry/exit of a train into/from a track detection section.

This level of detail of the signalling system is seldom used in the automatic control approaches due to tractability issues (Luan et al., 2017c, 2018a,b; Wang et al., 2022b). A frequent assumption for foregoing this level of detail is that the interactions through these signalling events have negligible impact on the choices of optimal traffic control actions, and therefore may also be neglected in the formulation of the optimal traffic control problem.

Indeed, there are many situations where the simplifying assumptions hold. For example, when one or many of the following conditions are met: timetables are designed with significant buffer times; the capacity of the infrastructure is sufficient to cover the traffic demand; the rail services are homogeneous; the configuration of the infrastructure makes it possible to separate incompatible traffic flows, thus making it possible to avoid almost all convergences and cutting-across conflicts.

Typical examples where these assumptions hold are on certain urban lines where overtaking is almost never used or on lines where each track is dedicated to one traffic direction. Most of the infrastructure case studies tackled in the articles that use macroscopic approaches mentioned in Subsection 2.2.2 meet these conditions. The envisaged future deployment of Altazin et al. (2020) results illustrates a successful validation of the simplifying assumptions and the proposed optimization control model.

Other constraints that are sometimes neglected are coupling constraints between trains due to platform turnaround or passenger transfer constraints at stations. The first assumption is valid when only one traffic direction is considered possible on a line, and the second one when the line is isolated of the network. Most of the articles considering microscopic approaches take these constraints into account. At the network traffic control level, only Schutter et al. (2002) and Kersbergen et al. (2016b) do so.

At the local area traffic control level, one of the most sensitive simplification made consists in ignoring rerouting control actions. As the rerouting actions make full use of the available capacity provided by alternative routes, this simplification can only be valid if the capacity on the default route is always sufficient or, on the contrary, when there is no useful capacity available on the alternative routes. Only the papers referenced in the microscopic approach Subsection 2.2.2 consider rerouting.

Among the articles mentioned in Section 2.3 on the network traffic control level, we point that Luan et al. (2017b) consider route choices by using the microscopic model of Luan et al. (2017a). We also notice that the station capacity constraint model of Blenkers (2015) used at lower level control in Cavone et al. (2022) amounts to a platform allocation problem. The experiments reported in Blenkers (2015) lead to long computation times, making the proposed model not practical for a real-time implementation. As the platform allocation problem in stations is a simplification of the rerouting problem, this illustrates the difficulty to consider rerouting in stations.

Several papers who consider rerouting control actions assume that route incompatibility of the railway interlocking systems is only managed with the route-lock route-release principle. However, modern interlocking systems deployed in almost all station areas use the route-lock sectional-release principle. Therefore, a model that does not apply this interlocking principle underestimates station capacity and may yield suboptimal solutions. To correctly model the route-lock sectional-release principle, it is necessary to be able to take into account the events of the entry/exit of a train into/from

track detection sections. The articles that consider the route-lock sectional-release principle are Rodriguez (2007b); Caimi et al. (2012); Pellegrini et al. (2014); Reynolds et al. (2020); Toletti et al. (2020).

At the network traffic control level, having a single model for the whole network is an issue. One approach to deal with this issue is to make simplifying assumptions about some parts of the network. For example, Schutter et al. (2002); van den Boom and De Schutter (2006); Kersbergen et al. (2016b) assume that there is no capacity constraint at stations, i.e., that stations have sufficient capacity to accommodate all arriving trains.

Another approach to deal with this issue is to decompose the network into smaller parts, coupling or coordination constraints are added to the optimization controller subproblem of each part. These coordination constraints are provided by neighboring problems (Kersbergen et al., 2016a; Wang et al., 2015b) or by a higher level "coordinator" (Strotmann, 2007; Corman et al., 2010a; Toletti et al., 2020; Cavone et al., 2022).

The assumption of an infinite capacity constraint on dispatching areas can be an issue for some practical applications. Today, the growing demand for rail traffic from multiple stakeholders is causing bottlenecks. Moreover, if the scheduled traffic is mixed, this increases the difficulty of traffic control in these bottlenecks. Bottleneck distribution areas can be complex junctions, stations or network corridors.

A well-known phenomenon in bottleneck dispatching areas is that a small train delay can lead to a multiple additional knock-on delays (Lüthi, 2009). Reducing the knock-on delays is a problem of major complexity in rail traffic management. The previously mentioned relationships between signalling events have a central role in the propagation speed of delays and the non-linearity of the consequences.

The research goal of this work is to propose a new formulation at the microscopic level of the real-time rail traffic management problem. This formulation can improve the efficiency of the solution method while taking into account the non-linearity behavior in bottleneck dispatching areas with heavy and mixed traffic. We want to evaluate to what extent the powerful algorithms provided by the CP conditional time-interval approach allow to better deal with the combinatorics related to the coupling of the train routing and ordering decisions. This evaluation must be carried out in the most diverse situations possible. For this, we need to consider a large number of instances that represent various infrastructure case studies.

From the perspective of the rail traffic control process, this work con-

tributes to finding new trade-offs between the level of detail of the control model of the optimization problem in the MPC framework, the geographical extent of the control areas, the extent of the control horizon and the limit of the computation time corresponding to the period of the control loop.

3. Conditional time-interval variables in CP

In this section, we briefly introduce the concept of conditional time-interval variable that is a key feature of the new CP formulation presented in this paper.

In many works in the scheduling field, the main decisions to be made consist in assigning resources to activities and in scheduling activities. However, in industrial applications, it can also be necessary to consider the choice of specific activities to be executed in the final schedule, for example when there are alternative production processes to satisfy to an order. This translates into the introduction of optional activities.

Vilím et al. (2005) introduce a tree data structure and a specific constraint propagation algorithm to deal with optional activities. This is later extended through the introduction of conditional time-interval variables in the IBM 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 rtRTMP, a time-interval variable is the period in which an activity may be executed; the concept of optional activity and that of time-interval variable are same. 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 $\{\perp\} \cup \{[s, e) | s, e \in \mathbb{Z}, s \leqslant e\}$. As 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 a, 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 if it was deleted. An execution or presence status noted $pres(a)$ 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 resources that correspond to two resource choices will be linked by a clause with a logical operator $\oplus$ (exclusive disjunction).

The temporal constraints state the different temporal positions of the start and end events of the time-interval variables, e.g., "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.

A new feature that has also motivated the proposal of our new CP formulation for the rtRTMP is the integration of temporal linear relaxation (Laborie and Rogerie, 2016) to the automatic search methods provided in the CP Optimizer library (Laborie et al., 2018). The objective function we use in the rtRTMP is an irregular one, i.e., it violates the property of non-decreasing function of tasks completion times, in other words, an optimal schedule does not necessarily execute all the activities as soon as possible. To mitigate this issue for the solution methods, it is important to provide good time placements of the activities. For this purpose, a linear relaxation of the problem is solved with CPLEX at the relaxation step of a Large neighborhood search (LNS) integrated as one component of the automatic search procedure of CP Optimizer. The solution of the linear relaxation provides indicative presence, start and end values for interval variables at the root node of a completion strategy to re-optimize the relaxed precedence constraints (Laborie and Rogerie, 2016).

The concept of time-interval variables and related algorithmic components such as the temporal linear relaxation and 2-SAT and STN constraint networks are specific to the CP Optimizer library, and no other CP library currently covers it.

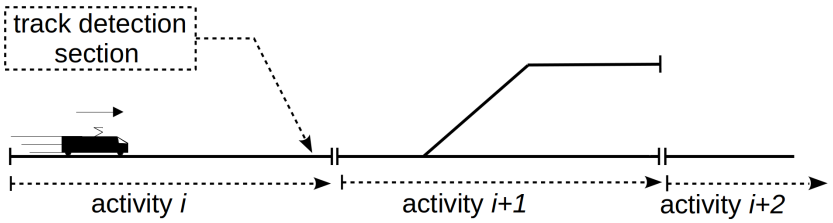


Figure 1: Train movement as a sequence of activities.

4. RECIFE modeling principles

As mentioned in the introduction, RECIFE is the French acronym for "Research on the capacity of railway infrastructures". It is the name of the first research project in which the modeling principles presented in this sections have been used for the rtRTMP (Rodriguez and Kermad, 1998). In this paper, we present a new CP-based algorithm for the rtRTMP, which exploits conditional time-interval variables. We name this algorithm RECIFE-CPI, where RECIFE stays for REcherche sur la Capacité d'Infrastructures FErroviaires (Research on the Capacity of Railway Infrastructures). RECIFE-CPI is an advanced version of a previous CP algorithm of the rtRTMP, named RECIFE-CP (Rodriguez and Kermad, 1998). It makes the best of additional modeling possibilities that have been developed in the last decade.

The main RECIFE modeling principles for rail traffic management problems are:

- The model must consider the railway infrastructure at a microscopic level to be as accurate as necessary,
- The management of train movements must be formulated as in a conventional scheduling problem.

The first principle stems from the central role of the signalling system in rail traffic control. This leads to the explicit model of the influence of the signalling system on traffic management decisions. Here, the microscopic representation of infrastructure comes down to considering the technical features of the signalling system like signal aspects, sight distance, block boundaries or track detection sections (tds).

For the second principle, a train journey can be described as a sequence of activities, or "jobs" in scheduling theory. As the smallest part of the track

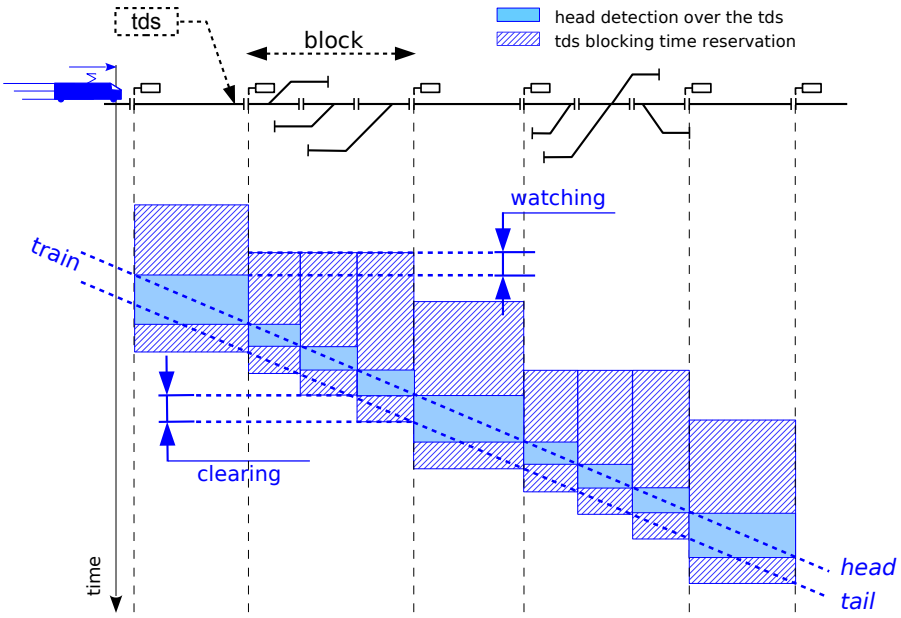


Figure 2: Train head detection activities and tds blocking time reservation.

considered in the infrastructure is a tds, activities are train movements on tds's as shown in Figure 1. In practice, a technical device is associated to each tds. It allows the detection of the occupation of this part of the railway infrastructure by a train. In many railway infrastructures, these are electric devices named "track circuits" and are part of the block signalling system that ensures the safe movement of trains.

If we refer to Figure 1, the temporal relations among train movement, tds detection events and signalling system can be represented in a spatio-temporal diagram as shown in Figure 2. This diagram can be interpreted as a Gantt diagram of the use over time of the elementary components of the railway infrastructure. 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 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. The filled blue rectangles show the sequence of head detection activities. Nevertheless from the point of view of the usage of a tds, we must also consider (i) the behavior of the signalling system which

imposes a safety headway to another train that has to run on the tds and (ii) the length of the train. For requirement (i), the tds is "reserved" in advance to take into account the main characteristics of the block signalling system such as: the number of signalling aspects and the watching time (i.e., running time of the sight distance). For (ii), the clearing phase due to train length is represented by the extended striped rectangle. It lasts until the tail of the train is no longer detected in the tds.

When there are switches within a block, the existence of multiple tds's allows the interlocking system to release and set as soon as possible the sequence of incompatible routes and then safely optimize traffic. The sectional route release of the interlocking system is built in because the release event of tds coincides with the end of the clearing phase.

The temporal constraints of tds reservation, illustrated in Figure 2, comply with the blocking time theory constraints (Hansen, 2008) and will be called "blocking time" constraints (resp. activities) in the sequel of the paper.

Furthermore, it can be noted that RECIFE is also the name of the digital platform (Rodriguez et al., 2007) which brings together these types of models. The RECIFE-MILP algorithm (Pellegrini et al., 2014) is a key figure of the last decade. In addition to the different optimization algorithms, the platform includes a database of case studies, interface modules with railway simulators, a test bench for configurations in closed loop or open loop and statistical and graphical tools to process the results of experiments.

5. RECIFE-CPI algorithm

In this section, we detail the algorithm we propose in this paper. The first two subsections respectively describe resource and time constraints of RECIFE-CPI. The third subsection is devoted to high-level temporal constraints between groups of activities to improve constraint propagation. The last two subsections detail the problem formulation and the solution algorithm.

5.1. Resource constraints

The resources considered in the RECIFE modeling principles for the rtRTMP are a set of railway track sections corresponding to the tds's of the signalling system (c.f. Figure 1). During normal operations, after a route is set for a train, the latter runs with clear line indications through a

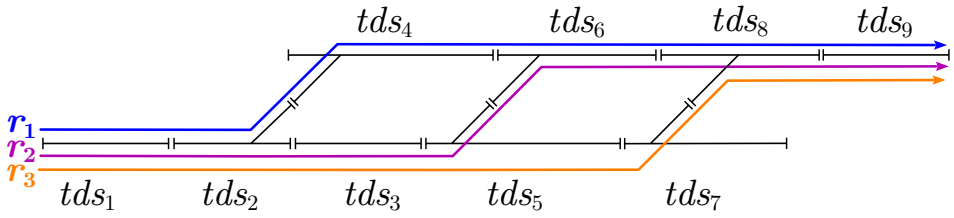


Figure 3: Example of tds sequences on three routes.

sequence of tds's at the planned speed, lower than or equal to the maximum authorized speed. Depending on the chosen route, the sequence of train head detection activities, as the one shown in Figure 2, may vary. In CP optimizer, a straightforward approach to model the consequent alternative combination of resources is to define optional activities (Laborie et al., 2018). More specifically for the rtRTMP, an optional "route activity" is associated with each alternative route of a train in Cappart and Schaus (2017), and only one route activity can have a presence status equal to 1. A route activity covers the sequence of tds optional activities associated with a route. The presence status of a route activity and of its sequence of tds activities are equal. A disadvantage of this approach is that it leads to "two-phase" search methods that choose routes for trains in a first step and choose order of trains in a second step. In Rodriguez et al. (2010), we showed that a search method that incrementally assigns tds's to trains (i.e., assigns route portions) and makes associated order decisions have better performance than two-phase ones. To allow the solver to use such incremental method, we define an oriented graph for each train. In this graph, nodes are optional head detection activities and edges represent precedence constraints between pairs of activities. In particular, an edge connects two activities if they follow each other on at least one route. A source (respectively sink) node is defined and connected to the activities of the entry (respectively exit) tds's of the control area. The value of the presence status of the activities must be set so as to obtain a sequence of activities corresponding to an allowed route. The proposed graph representation allows defining a single activity on a tds along different routes. This is done provided that all movement characteristics on this tds are the same on all the routes considered. For example, running and clearing times must be the same on different routes.

To illustrate the model, let us consider the example of a train that has

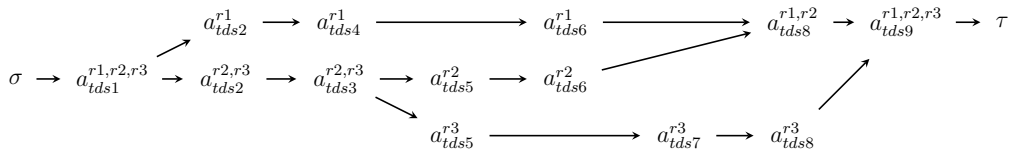


Figure 4: Graph of head detection activities for the routes of Figure 3.

three alternative routes r_1, r_2 and r_3, in Figure 3. We have three sequences of head detection activities for this example: one sequence of six activities for r_1 and two sequences of seven activities for r_2 and r_3. The train head detection activities on these sequences are represented in the graph in Figure 4. σ and τ are respectively the source and sink nodes. The train movement on tds_1 and tds_9 have the same characteristics, hence the corresponding activities generate a single node per tds. Conversely, for the movement through tds_2, two activities $a^{r_1}_{tds2}$ and a^{r_2,r_3}_{tds2} are defined because the minimum running time for r_1 is different from that for r_2 and r_3, as the train passes a switch in different positions.

In the resulting directed graph for each train, if the status of an activity is "present", it means that the corresponding tds has been chosen as a part of the final route within the solution under construction. However, to build a coherent solution, the values of the presence status of the activities included on the paths from the source to the sink node must satisfy the following consistency conditions:

1. At least one path must include all present activities and its corresponding tds sequence must be an allowed route,
2. If all activities included in one path are present then all other activities must be absent.

For the previous example, given the graph of activities shown in Figure 4, the status of the activities a^{r_1,r_2,r_3}_{tds1} and a^{r_1,r_2,r_3}_{tds9} is necessarily "present" because these activities must be in any solution.

If the status of activities a^{r_2,r_3}_{tds2}, a^{r_2,r_3}_{tds3} is also set to "present", then these values satisfy Condition 1 because both activities are included in path $< a^{r_1,r_2,r_3}_{tds1}, a^{r_2,r_3}_{tds2}, a^{r_2,r_3}_{tds3}, a^{r_2}_{tds5}, a^{r_2}_{tds6}, a^{r_1,r_2}_{tds_8}, a^{r_1,r_2,r_3}_{tds9} >$ corresponding to the allowed route r_2.

If the status of activities a^{r_2,r_3}_{tds2}, a^{r_2,r_3}_{tds3}, $a^{r_1}_{tds6}$ is "present", then these values do not satisfy Condition 1 because there is no path which includes all present

	presence statuses for										
	a^{r1}_{tds2}	a^{r1}_{tds4}	a^{r1}_{tds6}	$a^{r1,r2}_{tds8}$	$a^{r2,r3}_{tds2}$	$a^{r2,r3}_{tds3}$	a^{r2}_{tds5}	a^{r2}_{tds6}	a^{r3}_{tds5}	a^{r3}_{tds7}	a^{r3}_{tds8}
r_1	1	1	1	1	0	0	0	0	0	0	0
r_2	0	0	0	1	1	1	1	1	0	0	0
r_3	0	0	0	0	1	1	0	0	1	1	1

Table 1: The `allowedAssignments` constraint for presence statuses of activities for the routes of Figure 3

activities.

If the status of activities $a^{r_1,}_{tds_2}$,$a^{r_1}_{tds_4}$,$a^{r_1}_{tds_6}$,$a^{r_1,r_2}_{tds_8}$ is "present", all present activities are included in the path which corresponds to route r_1, and if the status of activities a^{r_2,r_3}_{tds2}, a^{r_2,r_3}_{tds3}, $a^{r_2}_{tds5}$, $a^{r_2}_{tds6}$, $a^{r_3}_{tds5}$, $a^{r_3}_{tds7}$ and $a^{r_3}_{tds8}$ is "absent", then these values satisfy Condition 2.

To ensure the above consistency conditions, we define `allowedAssignments` constraints that enumerate the tuples of allowed values for all presence statuses of head detection activities, for each allowed route. Table 1 shows these tuples for the three alternative routes of the example in Figure 3. Let us remark that, in general, not all paths of the graph of activities correspond to an allowed route. The number of `allowedAssignments` constraints amounts to the number of allowed routes.

A further constraint that has to be added states that the capacity of tds's is 1: they are unary resources, as during normal operation no more than one train can reserve a tds at a time. We model these constraints as `noOverlap` ones. The modeling of route choice through `allowedAssignments` and `noOverlap` constraints is very different from what had been previously done in RECIFE-CP. In particular, in RECIFE-CP a train route is modeled with a single sequence of activities regardless of the route chosen. The tds's in the control area are data structures associated with unary resources from the out-dated Ilog Scheduler library. To take into account all routes, first, the highest number of tds's in a route is identified. Then, a sequence with as many activities as this number is created. For each activity a_i of the sequence, a variable tds_{a_i} per possible tds assignments is defined, and a requirement constraint `requires`(a_i,tds_{a_i}) is set to link them. The domain values of tds_{a_i} is a subset of tds's and a table constraint enumerates the allowed tuple values for a train route variable r and tds variables tds_{a_i}. This table constraint is the assignment constraint that links a route choice to each activity's tds choices.

For the case of routes with a lower number of tds's, a "dummy" tds, denoted tds^* with infinite capacity is defined as a special value for the additional tds variables (Rodriguez, 2007a). Table 2 shows the assignment constraint for the example in Figure 3. Note that the dummy track detection section tds^* can be inserted in any position of the tds sequence. However, in general it is better to insert it in the middle of the sequence to maximize the number of variables with minimum size domain, and thus facilitate the early constraint propagation from the variables of these activities.

Variables	r	tds_{a_1}	tds_{a_2}	tds_{a_3}	tds_{a_4}	tds_{a_5}	tds_{a_6}	tds_{a_7}
Tuples	r_1	tds_1	tds_2	tds_4	tds^*	tds_6	tds_8	tds_9
	r_2	tds_1	tds_2	tds_3	tds_5	tds_6	tds_8	tds_9
	r_3	tds_1	tds_2	tds_3	tds_5	tds_7	tds_8	tds_9

Table 2: Table constraint of RECIFE-CP for the example of Figure 3.

5.2. Temporal constraints

In RECIFE-CPI, temporal constraints for each tds are defined on pairs of activities, first of all. The first activity associated with a tds is the head detection activity. An example of sequence of the head detection activities is illustrated by the filled blue rectangles in Figure 2. With the exception of the first activity, each activity of the sequence is linked to the previous one by a "no wait" constraint. These constraints are classic in job shop scheduling theory (Mascis and Pacciarelli, 2002), and they are implemented in CP Optimizer by the `startAtEnd` constraints.

The second activity associated with a tds, named blocking time reservation activity, includes the head detection activity and is extended to satisfy the blocking time constraints (c.f. Section 4). The start of the blocking time reservation activities of a tds belonging to a block is synchronized with the watching time of the signal controlling the entering of the block (c.f. Figure 2). It is formulated with a `startAtStart` constraint. The end of the blocking time reservation corresponds to the end of the head detection activity extended with a duration corresponding to the clearing of the tds by the train (c.f. Figure 2). It is formulated with a `endAtEnd` constraint. As for resource constraint, this formulation is quite different from the existing

RECIFE-CP one. There, only one activity was defined for each tds, and three additional variables were defined: the earliest start time of detection in the tds ($estd$), the earliest end time of detection in the tds ($eftd$) and the index of the activity to synchronize the start variable (s_i). Table constraints enumerated the allowed tuple values of these variables and the route choice variables. Appropriate values for these variables were defined for the case of a dummy value tds^* in the tds sequence of a route (c.f. Subsection 5.1). The blocking time constraints were formulated with these variables. For example, the blocking time synchronization constraint was formulated as $start(a_i) = end(a_{s_i}) - eftd_{s_i} - estd_{s_i}$ (Rodriguez, 2007a). Note that this is a non-linear constraint because the index of the variables used in the right hand side is the index variable s_i.

5.3. Global constraints on groups of activities

To improve the constraint propagation and therefore the performance of the RECIFE-CPI algorithm, we create a hierarchical model with global constraints on groups of activities. These global constraints allow the encapsulation of a group of activities in a high-level activity. High-level activities can be used with any temporal constraint in the same way as low level ones. They are particularly useful for strengthening resource and temporal constraints, in particular `allowedAssignments` constraints. Indeed, the unknown presence status of head detection activities does not allow a prompt constraint propagation and domain reductions of the interval variables. For the example of the graph in Figure 3, the activity $a_{tds3}^{r_2,r_3}$ is a point of route divergence and, in the basic model, its latest end time is not propagated to the following activities $a_{tds5}^{r_2}$ and $a_{tds5}^{r_3}$ as long their presence status is unknown. However, if these two activities are grouped as alternatives in a high-level one, then the latest end time of $a_{tds3}^{r_2,r_3}$ can be propagated to this high-level activity through a `startAtEnd` time constraint.

To create high-level activities, the precedents and the successors of each head detection activity of the graph are recursively analyzed with Algorithm 1. The algorithm generates all pairs of activity groups (G_{prec}, G_{succ}) that can be linked with a group constraint. Before setting the group constraints, redundant group pairs are removed. Two group constraints are possible: `span` and `alternative`. If there is no precedence relation between two activities of a group, the group constraint `alternative` is applied, otherwise the group constraint `span` is applied. The meaning of these group constraints is : `span`$(a_G, a_1, \ldots, a_n)$ states that activity a_G, if present, spans

ALGORITHM 1 : Generation of pairs of head detection activities groups

```
1: procedure GENPRECSUCC(a_i, G_prec, G_succ)
2:     G_succ = G_succ ∪ {a_i}
3:     Set_prec = getPrecSet(a_i)          # Get the set of previous activities
4:     for each a_j ∈ Set_prec do
5:         if a_j ∉ G_prec then
6:             G_prec = G_prec ∪ {a_j}
7:             Set_succ = getSuccSet(a_j)      # Get the set of next activities
8:             for each a_k ∈ Set_succ do
9:                 if a_k ∉ G_succ then
10:                    GENPRECSUCC(a_k, G_prec, G_succ)
```

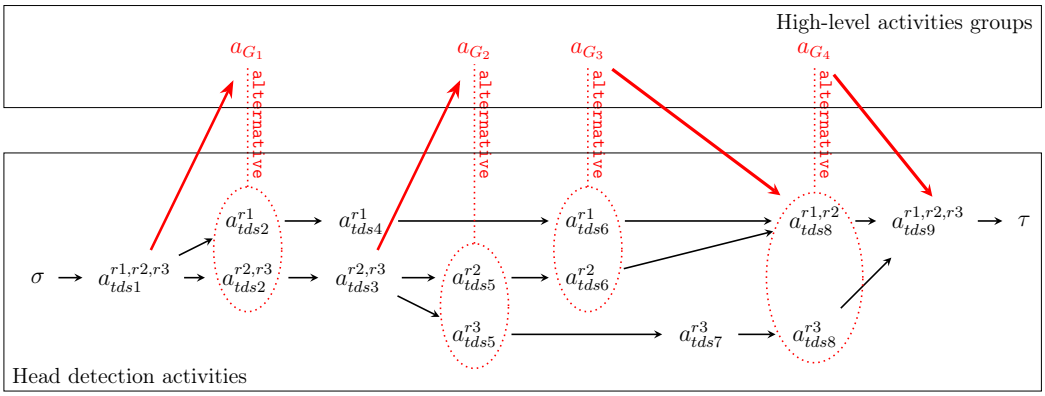
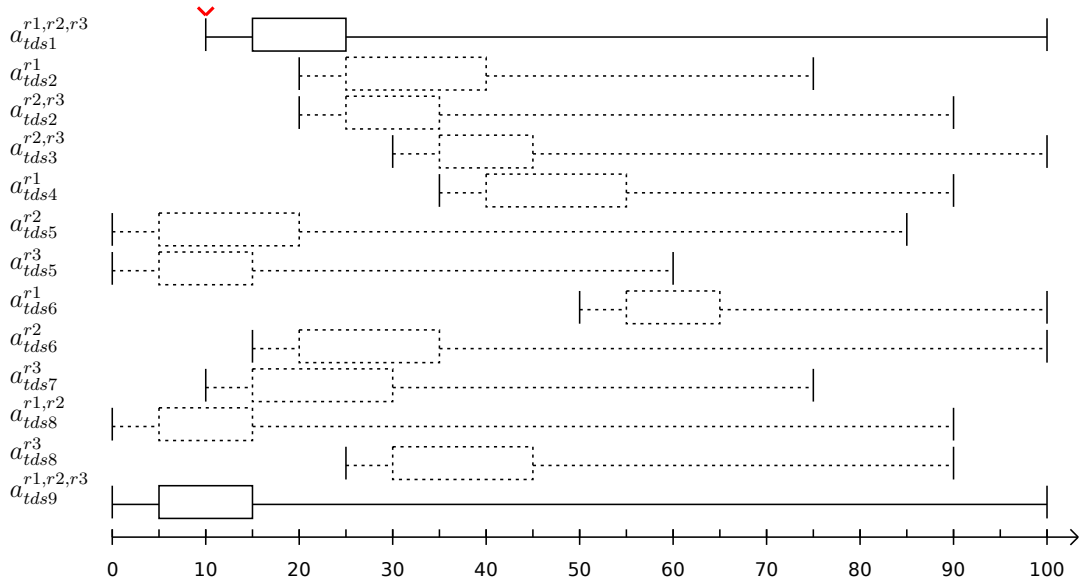


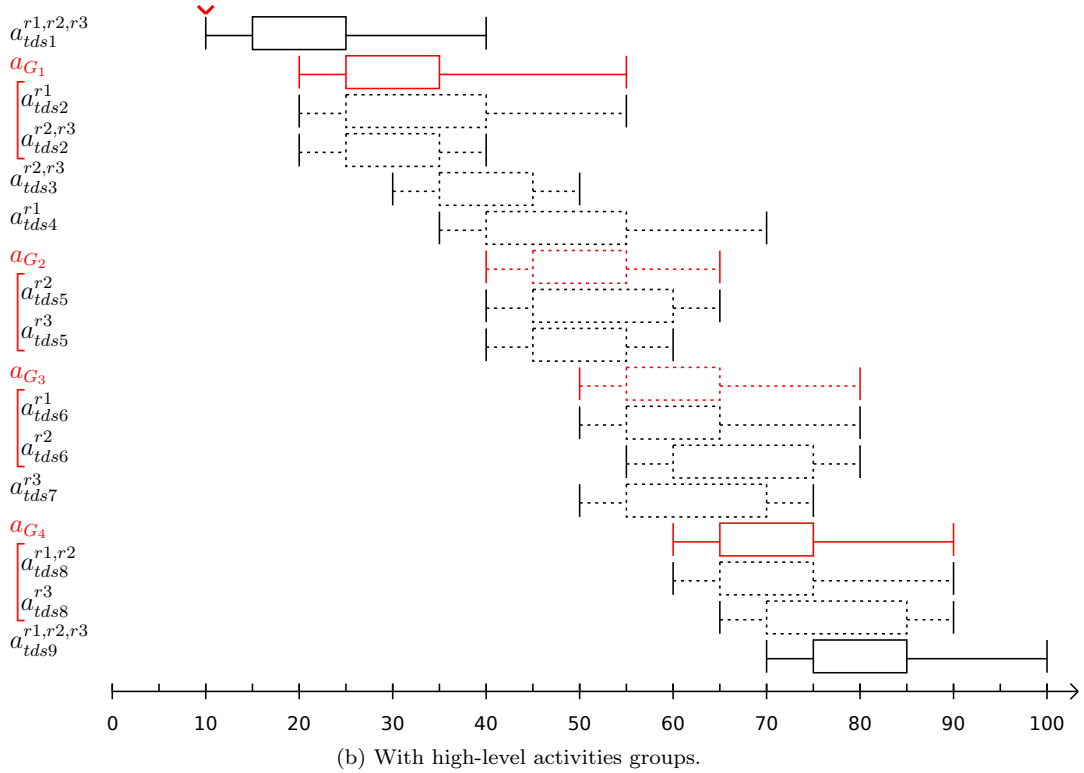
Figure 5: High level activities definition for the example of Figure 3.

over all present activities of the set $\{a_1, \ldots, a_n\}$; `alternative`$(a_G, a_1, \ldots, a_n)$ states that if activity a_G is present then exactly one of activities $\{a_1, \ldots, a_n\}$ is present, and a_G starts and ends together with the chosen one. Activity a_G is absent if and only if none of the activities $\{a_1, \ldots, a_n\}$ is present (Laborie and Rogerie, 2016).

To illustrate the definition of high-level activities and group constraints, let us consider the example depicted in Figure 3 and the associated graph of head detection activities in Figure 4. Figure 5 shows this graph enriched with high-level activity groups. The activities of each group are shown with red dotted shapes linked by a red dotted line to the corresponding group activity. The links are named with the group constraint used. The first high-level activity a_{G_1} is linked by an `alternative` constraint to



(a) Without high-level activities groups.



(b) With high-level activities groups.

Figure 6: Domain reduction of interval variables of activities for the example of Figure 3.

the group $G_1 = \{a^{r1}_{tds2}, a^{r2,r3}_{tds2}\}$ allowing to state the precedence constraint `startAtEnd`(a_{G_1},$a^{r1,r2,r3}_{tds1}$) and the logical constraint `samePresence`($a_{G_1}, a^{r1,r2,r3}_{tds1}$). The latter means that if $a^{r1,r2,r3}_{tds1}$ is present, a_{G_1} must be present and therefore we have to make a choice between a^{r1}_{tds2} and $a^{r2,r3}_{tds2}$. The red arrows represent these two types of constraints with high-level activities.

For an intuition on the strengthening of the resource and temporal constraint propagation with high-level activities, let us consider a numerical instance associated to the previous example of Figure 5. Let [0,100] be the time interval corresponding to the optimization horizon, 10 and 15 being the minimum duration of all head detection activities depending on the state of the switch the train runs through, and 10 being the earliest time at which the train can be operated. Figure 6a shows the domain reduction of the time-interval variables of activities after setting the expected arrival time of the train in the control area (marked with a red arrow) and after propagation of resource and temporal constraints but without definition of high-level activities. The graphical conventions are similar to those of a Gantt chart: each row is associated with an activity, the x-axis represents time, time interval domains are represented by "horizontal boxplots", the size of the inner "box" represents the minimum duration of the activity, the end marks represent the earliest start and latest end times of the activity, if the state of the activity is present, the box and whiskers are drawn as a solid line, but if the activity state is unknown (i.e. the domain includes $\perp$), they are drawn as a dotted line. The lines of the Gantt chart for activities $a^{r_1,}_{tds_2}$, $a^{r_2,r_3}_{tds_2}$, $a^{r_2,r_3}_{tds_3}$, $a^{r_1}_{tds_4}$, $a^{r_1}_{tds_6}$ show that the earliest start times are updated by the `startAtEnd` constraint propagation. However, the rows for activities $a^{r_2}_{tds5}$ and $a^{r_3}_{tds5}$ do not show any change in the earliest start times. This illustrates the previously mentioned issue of the lack of constraint propagation due to insufficient information on the relationship between the presence statuses of activities at the points of divergence of routes such as the one of $a^{r_2,r_3}_{tds_3}$, $a^{r_2}_{tds5}$ and $a^{r_3}_{tds5}$. We can observe the same issue for the constraint propagation at the points of convergence of the routes like that of $a^{r_1}_{tds6}$, $a^{r_2}_{tds6}$ and a^{r_1,r_2}_{tds8} where there is no update of the latest end times. The definition of group constraints with the auxiliary constraints `startAtEnd` and `samePresence` make up for the lack of information on the presence statuses of activities at the points of convergence and divergence of routes. These additional constraints facilitate the constraint propagation and thus the reduction of the domains of the time-interval variables. Figure 6b shows the results of adding four group constraints and clearly illustrates the strengthening of constraint propagation that leads to a considerable domain

reductions, which reduces the number of backtracks of search methods and thus increases their efficiency.

Let us remark that in the example of Figure 5, we only consider a simple infrastructure layout. When a track detection section includes more than one switch, the corresponding head detection activity can be associated with several points of route convergence and divergence. Figure 7 illustrates an infrastructure example of this kind of situation. Such tricky cases can be treated through `span` group constraints.

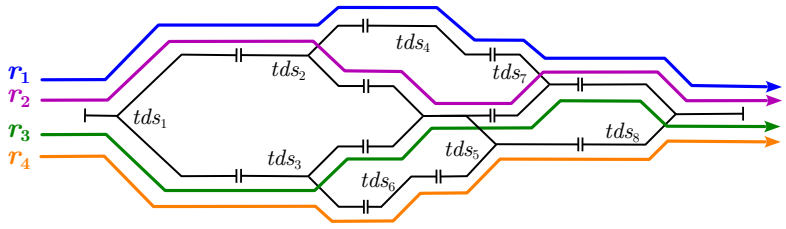


Figure 7: Example of infrastructure requiring `span` group constraints.

5.4. Formulation

For the formulation, we use a notation close the one introduced by Pellegrini et al. (2014) for RECIFE-MILP and reported in Table 3 for parameters and Table 4 for variables and functions.

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,
$ref_{r,tds}$	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 signalling system,
$rt_{ty,r,tds}$	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,
$arr_{t,s}, dw_{t,s}$	scheduled arrival time and minimum dwell time for train t at station s,
$S^{t,r}_{pres}$	tuple of presence status values of head detection activities of train t for route r,
$\{(G^t_{prec}, G^t_{succ})\}$	set of pairs of head detection activities groups generated with Algorithm 1,
$\mathcal{G}^t_{alt}$	set of groups of head detection activities of train t linked by an `alternative` group constraint,
$\mathcal{G}^t_{span}$	set of groups of head detection activities of train t linked by a `span` group constraint.

Table 3: Parameter definitions

$\boldsymbol{a^{t,r}_{tds,h}}$	time-interval variable which represents the head detection activity of t on tds along r,
$\boldsymbol{a^{t,r}_{tds,b}}$	time-interval variable which represents the blocking time reservation activity of tds for t along r,
$\boldsymbol{D^{arr}_t}, \boldsymbol{D^{exit}_t}$	delay suffered by train t at station arrivals (cumulative) and at the exit from the control area,
$first(\boldsymbol{a^{t,r}_{tds,h}})$	boolean functions that return true if $\boldsymbol{a^{t,r}_{tds,h}}$ is the first head detection activity of train t through the tds sequence for route r,
$last(\boldsymbol{a^{t,r}_{tds,h}})$	boolean functions that return true if $\boldsymbol{a^{t,r}_{tds,h}}$ is the last, head detection activity of train t through the tds sequence for route r,
$s(\boldsymbol{a}), e(\boldsymbol{a}), d(\boldsymbol{a})$	function that return the start, end and duration variables for time-interval variable $\boldsymbol{a}$, respectively,
$pres(\boldsymbol{a})$	function that return the presence status variable for time-interval variable $\boldsymbol{a}$.

Table 4: Variable and funtion definitions

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

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

The constraints are :

$$\texttt{allowedAssignments}(\bigcup_{r \in R_t, tds \in TDS^r} pres(\boldsymbol{a^{t,r}_{tds,h}}), \bigcup_{r \in R_t} S^{t,r}_{pres}) \ \forall t \in T, \tag{2}$$

$$s(\boldsymbol{a^{t,r}_{tds,h}}) \geqslant init_t \forall t \in T, r \in R_t, tds \in TDS^r, \tag{3}$$

$$d(\boldsymbol{a^{t,r}_{tds,h}}) \geqslant rt_{ty,r,tds} \forall t \in T, r \in R_t, tds \in TDS^r, \tag{4}$$

$$\texttt{startAtEnd}(\boldsymbol{a^{t,r}_{tds,h}}, \boldsymbol{a^{t,r}_{p_{r,tds},h}}) \forall t \in T, r \in R_t, tds \in TDS^r, \tag{5}$$

$$\texttt{endAtEnd}(\boldsymbol{a^{t,r}_{tds,b}}, \boldsymbol{a^{t,r}_{tds,h}}, ct_{ty,r,tds} + rel_{bs_{r,tds}}) \forall t \in T, r \in R_t, tds \in TDS^r, \tag{6}$$

$$\texttt{startAtStart}(\boldsymbol{a}^{\boldsymbol{t,r}}_{\boldsymbol{tds,b}}, \boldsymbol{a}^{\boldsymbol{t,r}}_{\boldsymbol{ref}_{\boldsymbol{r,tds}},\boldsymbol{h}}, -for_{bs_{r,ref_{r,tds}}}) \forall t \in T, r \in R_t, tds \in TDS^r, \tag{7}$$

$$\texttt{alternative}(\boldsymbol{a}^{\boldsymbol{t}}_{\boldsymbol{G}}, \boldsymbol{a}^{\boldsymbol{t,r_1}}_{\boldsymbol{tds_1},\boldsymbol{h}}, \ldots, \boldsymbol{a}^{\boldsymbol{t,r_n}}_{\boldsymbol{tds_n},\boldsymbol{h}}) \forall t \in T, \{\boldsymbol{a}^{\boldsymbol{t,r_1}}_{\boldsymbol{tds_1},\boldsymbol{h}}, \ldots, \boldsymbol{a}^{\boldsymbol{t,r_n}}_{\boldsymbol{tds_n},\boldsymbol{h}}\} \in \mathcal{G}^t_{alt}, \tag{8}$$

$$\texttt{span}(\boldsymbol{a}^{\boldsymbol{t}}_{\boldsymbol{G}}, \boldsymbol{a}^{\boldsymbol{t,r_1}}_{\boldsymbol{tds_1},\boldsymbol{h}}, \ldots, \boldsymbol{a}^{\boldsymbol{t,r_n}}_{\boldsymbol{tds_n},\boldsymbol{h}}) \forall t \in T, \{\boldsymbol{a}^{\boldsymbol{t,r_1}}_{\boldsymbol{tds_1},\boldsymbol{h}}, \ldots, \boldsymbol{a}^{\boldsymbol{t,r_n}}_{\boldsymbol{tds_n},\boldsymbol{h}}\} \in \mathcal{G}^t_{span}, \tag{9}$$

$$\texttt{startAtEnd}(\boldsymbol{a}^{\boldsymbol{t}}_{\boldsymbol{G}}, \boldsymbol{a}^{\boldsymbol{t}}_{\boldsymbol{G'}}) \forall t \in T, (G, G') \in \{(G^t_{prec}, G^t_{succ})\}, \tag{10}$$

$$pres(\boldsymbol{a}^{\boldsymbol{t',r'}}_{\boldsymbol{tds,h}}) = pres(\boldsymbol{a}^{\boldsymbol{t,r}}_{\boldsymbol{tds,h}})$$
$$\forall t, t' \in T, r \in R_t, r' \in R_{t'} : i(t,t') = 1 \wedge tds \in PL_{t,t'}, \tag{11}$$

$$s(\boldsymbol{a}^{\boldsymbol{t',r'}}_{\boldsymbol{tds,h}}) \geqslant e(\boldsymbol{a}^{\boldsymbol{t,r}}_{\boldsymbol{tds,h}}) + ms_{t,t'} \forall t, t' \in T, r \in R_t, r' \in R_{t'} :$$
$$i(t,t') = 1 \wedge last(\boldsymbol{a}^{\boldsymbol{t,r}}_{\boldsymbol{tds,h}}) \wedge first(\boldsymbol{a}^{\boldsymbol{t',r'}}_{\boldsymbol{tds,h}}) \wedge tds \in PL_{t,t'}, \tag{12}$$

$$s(\boldsymbol{a}^{\boldsymbol{t',r'}}_{\boldsymbol{tds,b}}) = e(\boldsymbol{a}^{\boldsymbol{t,r}}_{\boldsymbol{tds,b}}) \forall t, t' \in T, r \in R_t, r' \in R_{t'} :$$
$$i(t,t') = 1 \wedge last(\boldsymbol{a}^{\boldsymbol{t,r}}_{\boldsymbol{tds,h}}) \wedge first(\boldsymbol{a}^{\boldsymbol{t',r'}}_{\boldsymbol{tds,h}}) \wedge tds \in PL_{t,t'}, \tag{13}$$

$$\texttt{noOverlap}(\bigcup_{t \in T, r \in R_t, tds \in TDS_t} \boldsymbol{a}^{\boldsymbol{t,r}}_{\boldsymbol{tds,b}}) \forall tds \in TDS, \tag{14}$$

$$\boldsymbol{D}^{\boldsymbol{exit}}_{\boldsymbol{t}} = \sum_{\substack{r \in R_t, tds \in TDS_r: \\ last(\boldsymbol{a}^{\boldsymbol{t,r}}_{\boldsymbol{tds,h}})}} e(\boldsymbol{a}^{\boldsymbol{t,r}}_{\boldsymbol{tds,h}}) - exit_t \forall t \in T, \tag{15}$$

$$\boldsymbol{D}^{\boldsymbol{arr}}_{\boldsymbol{t}} = \sum_{r \in R_t} \sum_{\substack{s \in S_t, \\ tds \in TDS_{t,s}}} (s(\boldsymbol{a}^{\boldsymbol{t,r}}_{\boldsymbol{tds,h}}) + (rt_{ty,r,tds} - arr_{t,s}) pres(\boldsymbol{a}^{\boldsymbol{t,r}}_{\boldsymbol{h}})) \forall t \in T, \tag{16}$$

Constraints (2) ensure consistency of the presence status values of head run-

ning activities with the set of allowed routes. Constraints (3) state that trains cannot be operated earlier than $init_t$. Constraints (4) 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 (5) impose a precedence relation between running time head activities of a train. For Constraints (6), 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 (7) 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 (8) and (9) link a group of activities $G = \{\boldsymbol{a}^{t,r_1}_{\boldsymbol{tds_1},h}, \ldots, \boldsymbol{a}^{t,r_n}_{\boldsymbol{tds_n},h}\}$ into a high-level activity $\boldsymbol{a}^{t}_{\boldsymbol{G}}$ 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 (10) state the precedence constraints between high-level activities. Constraints (11) ensure local coherence: trains using the same rolling stock must use the same platform when they perform the turnaround. Constraints (12) ensure that a minimum separation time must separate the arrival and departure of trains using the same rolling stock. Constraints (13) 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 (14) ensure that blocking time activities on a tds do not overlap. Constraints (15) and (16) state that the values of the delays $\boldsymbol{D}^{\boldsymbol{exit}}_{\boldsymbol{t}}$ and $\boldsymbol{D}^{\boldsymbol{arr}}_{\boldsymbol{t}}$ 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.

5.5. Solution algorithm

The RECIFE-CPI solution algorithm follows a two-step approach based on the warm-start method used in RECIFE-MILP (Pellegrini et al., 2014). As the number of alternative routes is one of the causes of the difficulty of the rtRTMP, in the first step, the route fixed in the timetable is imposed to each train. A pure scheduling problem is hence solved.

In the second step, the first-step solution is used as a warm start for tackling the problem with all the alternative routes. Note that in both steps, we solved a routing and scheduling problem. In the first step, the parameter

of the set of allowed routes for a train is reduced to a singleton, while in the second step the parameter considers all allowed routes.

For the second step, we consider the automatic search methods provided in CP-optimizer library. It uses the algorithm of Vilím et al. (2015) for scheduling problems which combines Self-Adapting Large Neighborhood Search (SA-LNS) with Failure-Directed Search (FDS). The former (Laborie and Godard, 2007) aims to find a good quality solution quickly. It is an iterative improvement method with the 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 the 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: apply 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 SA-LNS has difficulties improving the current solution the search space seems to be small enough. 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.

6. RECIFE-MILP algorithm

As RECIFE-CPI, RECIFE-MILP aims at solving the rtRTMP, simultaneously optimizing routing and scheduling decisions. It applies the warm-start method sketched in Section 5.5. It consists in solving two MILP formulations using a commercial solver for a limited time. If the optimal solutions are reached before this time, then the algorithm either passes to the next step

or it stops. Otherwise, it returns the best solution found in the available time disregard the absence of optimality proof. In the first step, the MILP formulation only considers scheduling decisions: trains must traverse all the tds's in the route they are assigned in the timetable respecting operational constraints. Continuous variables represent times and binary variables represent passing orders for pairs of trains on common sequences of tds's. Disjunctive constraints are imposed by exploiting these variables. For each tds possibly used by a pair of trains, two big-M disjunctive constraints are created. Travel time coherence is imposed by controlling running times on each tds for each route it can be used on. In particular, if a tds can be used along ten routes by a train, ten variables and constraints will be created to control travel time there. Once the solution of this scheduling problem is returned, optimal or not, it is used as a warm start for tackling an extended MILP formulation in the second step. Here, in addition to scheduling decisions modeled as in the first step, routing decisions are made, represented by binary variables indicating the choice (or not) of a route by a train. Some pre and post-processing are operated for the two optimization steps, as detailed in Pellegrini et al. (2015) and Pellegrini et al. (2019).

The detailed formulations used in RECIFE-MILP are reported in Pellegrini et al. (2014) and Pellegrini et al. (2015).

7. Experimental comparison of RECIFE-CPI and RECIFE-MILP

In this section, we present an experimental analysis aimed to assess the performance of RECIFE-CPI (CPI). Specifically, we consider RECIFE-MILP (MILP) as a benchmark and we compare the two algorithms on five control areas. Such a thorough comparison is very seldom proposed in the literature, if ever. However, we consider it pertinent here to verify whether and how control areas with different characteristics may favour an algorithm in particular.

In the following, we detail the characteristics of the five control areas and traffic perturbation scenarios we tackle, we describe the experimental setting, and we present results.

7.1. Control areas

The control areas we consider have different characteristics: a line with intermediate stops, a junction with mixed traffic, a passing station with typical characteristics and two passenger terminal stations with high density traffic.

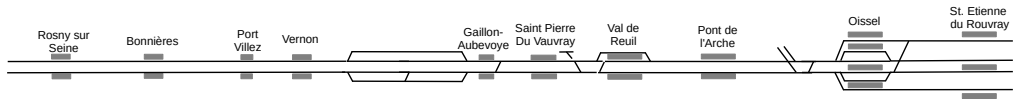
	RSS-SER	Gonesse	GenStation	Lille	StLazare
Infrastructure					
Length (km)	80	15	8	7	4,5
Routes	187	37	71	2409	84
Blocks	157	79	60	829	197
tds	236	89	34	299	212
Timetable					
Trains/Day	237	336	400	589	1212
Routes/Train	13 (1-24)	7 (5-13)	18	12 (1-71)	5 (1-9)
tds/Route/Train	57 (5-97)	28 (16-34)	13	21 (9-33)	18 (13-25)

Table 5: Control area characteristics. For the last two rows, we indicate the average and, in parentheses, minimum and maximum values observed. Remark that in GenStation all trains have exactly 18 routes and all routes have exactly 13 tds's.

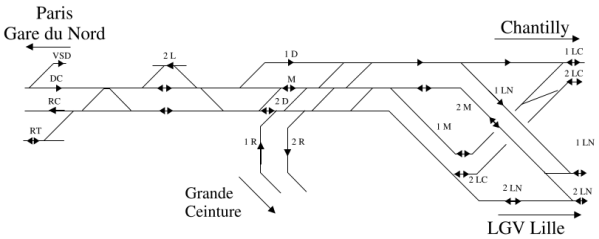
Namely, they cover: the 80-km portion of the Paris-Le Havre line between Rosny sur Seine and Saint-Etienne de Rouvray (RSS-SER), the Pierrefitte-Gonesse junction north of Paris (Gonesse), an artificial generic station with a very classic topology (GenStation), and the control areas including the Lille-Flandres (Lille) and Paris–Saint-Lazare (StLazare) terminal stations. Their characteristics are detailed in Table 5 and their layout is in Figure 8. Additional comments on these characteristics are as follows:

- RSS-SER is a double-track line including ten small stations and only a few possibilities for changing tracks between stations. The presence of the relatively large stations at Oissel and St. Etienne du Rouvray, though, implies that some trains may have up to 24 alternative routes, which makes the rerouting problem quite challenging. These routes are often very long: trains have routes 57 tds long, in average. The difference here can be very large: the train with the shortest routes has them including around five tds's, while the one with the longest has them made of about 97. The presence of mixed traffic, spanning from freight to the high-speed trains, is a further challenge as running times change quite significantly from one type of train to another.

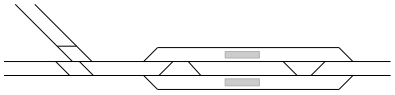
- Gonesse presents the same peculiarity of mixed traffic. This peculiarity is also particularly interesting for its infrastructure topology. Despite its rather short length, this control area represents an interesting challenge as train routes are very articulated and they overlap in various ways. The complex interlocking that makes this possible implies train



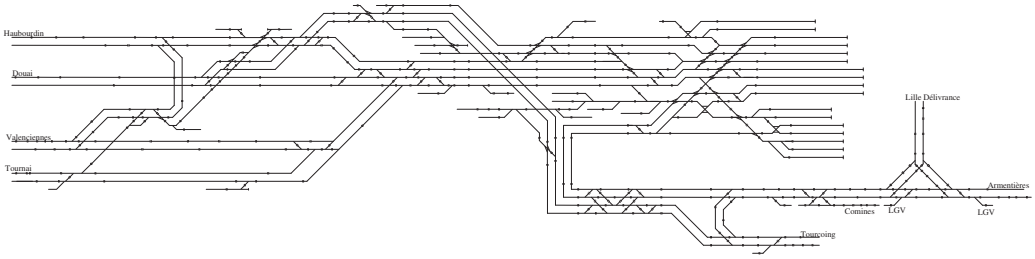
(a) RSS-SER



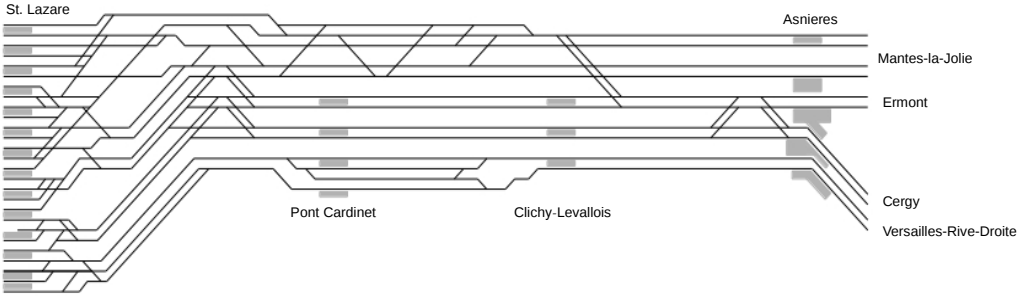
(b) Gonesse



(c) GenStation



(d) Lille



(e) StLazare

Figure 8: Infrastructure layout of the five control areas considered.

route decomposition in large numbers of tds's: in average this number is 28, about half than in RSS-SER although the infrastructure length is about a fifth. The differences here are much smaller too, as the train with the shortest routes have them including about 16 tds's and this value grows to a maximum of 34. Most of the trains here can span over most of the infrastructure by changing their route.

- GenStation represents a station that can be very often found in reality. Here, all trains have several alternative routes, and they overlap quite substantially in their central part, i.e., in the immediate vicinity of the platforms. Differently from the other control areas, all trains have the same number of routes, and they are equally long in terms of tds's. No train starts or terminates its service in this control area: no turnaround occurs here.

- Lille is a major terminal station with a rather articulated topology. Here, a very large number of alternative routes is available for each train, all with a number of tds's comparable to Gonesse. As shown in Figure 8d, the infrastructure is such that trains can use almost any of the 17 platforms independently on their origin or destination line, and often a few possibilities are available for connecting a platform to a line track. These possibilities are often almost indistinguishable for about half of their length, while they vary quite substantially when getting closer to platforms. Being a terminal station, trains very often meet others running in the opposite direction. Moreover, rolling-stock reutilization relations exist between pairs or sets of trains: all trains are subject to a turnaround. This further complicates traffic management, as decisions made on different trains can be interdependent although no conflict between these trains occurs.

- StLazare has similar complications: it is a terminal station where much fewer train routes are available, but where the number of trains to be dealt with is extremely high. Here, the infrastructure brings quite a strong flow separation: trains travelling on a specific line can only use a pair of alternative tracks to reach the station, and hence only a few platforms (Figure 8e). This explains the much smaller number of routes per train appearing in Table 5. Trains have routes of similar length to Lille and Gonesse in terms of tds's. Three small stations also belong to this control area, where some trains make intermediate stops.

While data representing traffic in real control areas are protected by confidentiality engagement, the instances concerning the GenStation one are publicly available at the address: `http://recife.univ-eiffel.fr/sharedData/`.

7.2. Traffic perturbation scenarios

For each of the five control areas, we generate 30 perturbation scenarios: starting from the original daily timetable, 20% of randomly selected trains are subject to a random entrance delay between 5 and 15 minutes. For the real control areas, the timetable is an actual 24-hour one. For GenStation, we generate the daily timetable by propagating the same pattern of trains along the day. To cover a variety of instance sizes, for each perturbation scenario, we select 13 instances. Each instance includes the first $|T|$ trains entering the control area from 6am, with $|T|$ varying from 10 to 130 and considering a ten train step. Therefore, we consider 390 instances (13 sets of 30 instances) for each control area, i.e., 1950 instances in total.

The specific time horizons considered for each infrastructure are as follows:

- For RSS-SER, the instances include trains entering the control area in a time horizon between 6am and 5:47pm,
- For Gonesse, between 6am and 1:12pm,
- For GenStation between 6am and 11:31am,
- For Lille between 6am and 9:20am,
- For StLazare between 6am and 7:50am.

Indeed, the duration of these time horizons is very different across control areas, going from less than two hours in StLazare, to 3 hours and a half in Lille, to more than seven and almost eleven hours in Gonesse and RSS-SER. Undoubtedly, tackling the rtRTMP on an 11-hour time horizon is not very meaningful in practice, as perturbations occurring so far in time can hardly be predicted. However, we consider increasing the number of trains consistently across control areas interesting for comparing algorithmic performance.

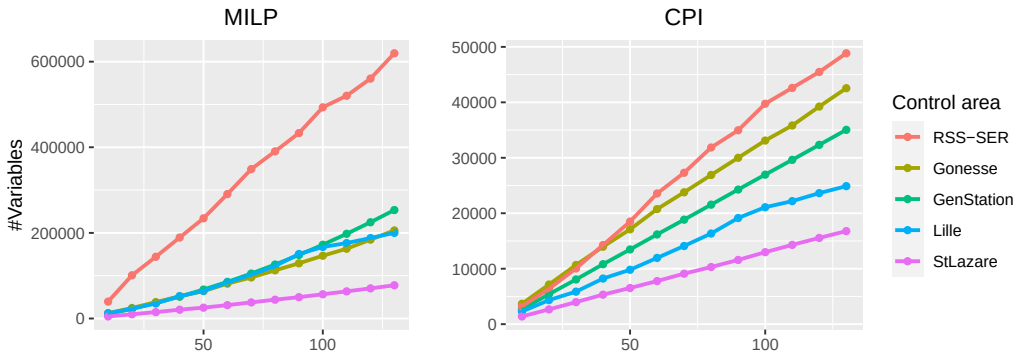


Figure 9: Average number of variables for each control area in MILP and CPI.

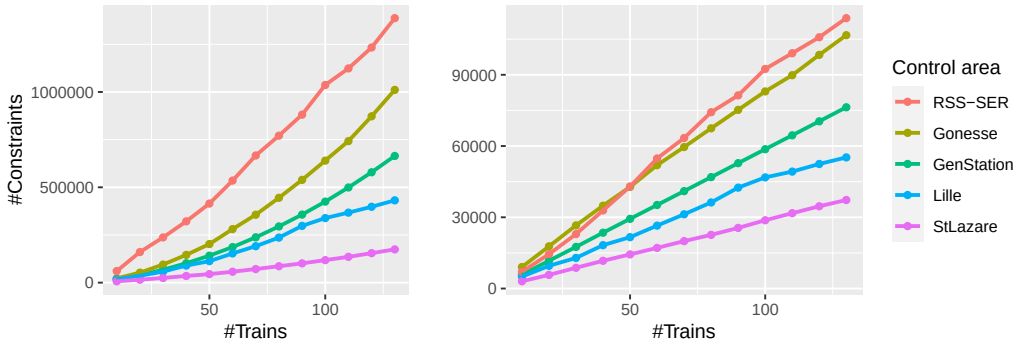


Figure 10: Average number of constraints for each control area in MILP and CPI.

7.3. Model size

Figures 9 and 10 show the average number of variables and constraints in the second optimization step of MILP and CPI, i.e., when both rerouting and rescheduling are considered.

First of all, let us remark that the scale of the vertical axis is different for the four plots, so as to be able to see the difference between control areas. For the number of both variables and constraints, MILP shows values that are one order of magnitude larger than the ones of CPI. The difference is not surprising if one considers that the number of disjunctive variables and constraints in MILP grows exponentially in the number of trains. Moreover, as mentioned in Section 6, time coherence is ensured considering running times separately along each route, and this has a big impact when several

routes are available. This type of variables and constraints is actually the main contributor to the observed values.

The order in which control areas appear is quite preserved in all observations. In particular, RSS-SER is the control area where the number of variables and constraints is the highest, mostly due to its longest length: here trains have many routes and routes have many tds's, which explains the extremely large number of time coherence relations to be ensured. Then Gonesse, GenStation, Lille and StLazare follow, in this order. Their number of variables are very similar for MILP, while a clear distinction is visible for CPI and for the number of constraints. The difference among the routes that can be used by trains may explain this order.In particular, as discussed in Section 7.1:

- In StLazare all routes for a train are quite similar thanks to flow separation,
- In Lille, they are similar at one end of the control area and span over the whole station, on the other hand,
- In GenStation, the central part of the routes, around platforms, is the area where most differences exist,
- In Gonesse routes can be different from the very beginning to the very end.

To understand the link between these observations and the size of the models, imagine that a train can use two routes that have 10 tds's each, none of which in common. For this train, both CPI and MILP need to ensure time coherence on 20 tds's. If another train has the same alternative routes, there will also be constraints to ensure the non-overlap of different train utilizations on each of them. If the two routes are equal on the first 5 tds's, MILP will keep creating 20 variables and constraints for time coherence, while CPI only 15. As for the non-overlap, only 15 tds's will need to be considered by both models.

In summary, the difference among control areas is definitely remarkable. However, most of this difference is dependent on continuous variables and easy constraints. Hence, it does not translate into a relevant difference in the efficiency of the two algorithms in the five control areas.

7.4. Experimental settings

As mentioned in Section 5.5 and 6, both CPI and MILP follow a two-step approach: first a pure scheduling problem is solved, having each train using the route planned in the timetable; then the routing and scheduling problem is solved exploiting the first solution as a warm start. In this analysis, for each instance, we set the first step time limit to 30 seconds, as done in Pellegrini et al. (2015) and several following papers (Pellegrini et al., 2016; Samà et al., 2016; Pellegrini et al., 2019). If a solution is proven optimal before this time has elapsed, then the first step is stopped immediately. Moreover, if no feasible solution is available after 30 seconds, the first step is continued until at least one is found. The second step time limit is the complement to 180 seconds of the time consumed in the first step. This value is typically considered pertinent for this type of algorithms, which can be deployed to make traffic management decisions under the control of a human dispatcher (Quaglietta et al., 2016). In such a deployment, algorithms need to be sufficiently quick to be able to respond promptly to the predicted traffic perturbations, but can take a short time to do so as in any case human reaction is not instantaneous: dispatchers need some time to validate or refuse decisions, and they cannot be presented with a new set of decisions every few seconds. Three minutes is considered a good compromise between response speed and solution space exploration possibilities. We run all experiments on an Intel(R) Xeon(R) CPU E5-2643 v4 @ 3.40GHz, 24 cores, 128go RAM. We use CPLEX 12.6 as MILP solver and its CP optimizer for CPI.

7.5. Results

Table 6 shows the average objective function value obtained by CPI and MILP for each control area and number of trains, after the first and the second optimization step. Values in bold indicate that the corresponding performance is significantly better than the one of the competitor according to the Wilcoxon rank-sum test with a 95% confidence level. Recall that if the difference between the two algorithms is significant, for example in favor of CPI, it means that if we draw a novel sample from the same distribution of results, i.e., if we solve a further instance with similar characteristics, we can expect that CPI will perform better than MILP. If the difference is not statistically significant, instead, we have no reason to expect a better performance of one of them on the further sample: roughly speaking, half of the times CPI will be better, and half of the times it will be MILP. Remark that the significance of the difference is in no relation with the magnitude of

Table 6: Mean of objective values: minutes of total delay. Values are in bold if the performance is significantly the best according to the Wilcoxon rank-sum test with a 95% confidence level.

		# **Trains**												
		10	20	30	40	50	60	70	80	90	100	110	120	130
RSS-SER														
CPI	Step 1	57	98	121	185	215	238	290	306	357	396	431	441	496
	Step 2	56	97	115	**177**	**206**	**231**	**279**	299	356	395	431	441	496
MILP	Step 1	57	98	121	185	215	238	285	**301**	**349**	**383**	**416**	**425**	**478**
	Step 2	56	97	118	182	214	238	285	301	**349**	**383**	**416**	**425**	**478**
Gonesse														
CPI	Step 1	22	39	61	74	93	127	153	167	196	215	247	257	280
	Step 2	22	39	60	72	**91**	**125**	**152**	167	196	215	247	257	280
MILP	Step 1	22	39	61	74	93	127	153	167	196	215	246	256	**279**
	Step 2	22	39	59	73	92	126	153	167	196	215	246	256	**279**
GenStation														
CPI	Step 1	27	57	83	115	140	173	197	227	257	273	319	343	375
	Step 2	24	**53**	**77**	**105**	**127**	**159**	**180**	**209**	**238**	**259**	**315**	343	375
MILP	Step 1	27	57	83	115	140	173	197	227	257	273	319	343	375
	Step 2	24	53	78	110	134	172	197	227	257	273	319	343	375
Lille														
CPI	Step 1	19	36	72	118	146	172	212	270	322	343	407	437	497
	Step 2	18	30	62	**103**	133	160	**191**	248	290	328	403	435	491
MILP	Step 1	19	36	72	118	144	169	**204**	**247**	**281**	**304**	**356**	**377**	**413**
	Step 2	18	30	62	106	134	162	201	246	281	**304**	**356**	**377**	**413**
StLazare														
CPI	Step 1	46	71	127	160	204	262	295	340	411	463	509	574	631
	Step 2	44	69	117	145	184	237	262	**297**	**356**	**393**	**443**	**482**	**536**
MILP	Step 1	46	71	127	160	204	260	293	**334**	399	**447**	**497**	**552**	**605**
	Step 2	44	69	117	145	183	**236**	262	307	391	443	497	551	605

this difference itself. Indeed, there are cases in which a statistically significant difference is in average very small. For example, in 50 train Gonesse instances, CPI is significantly better than MILP although the difference is in average of only one minute. However, the difference is in favor of CPI in the majority of observations. Considering one control area at a time, we can observe that:

- In RSS-SER the two algorithms are equivalent in step 1 up to 60 trains,

then CPI is worse. In step 2, CPI improves its solution up to 100 trains while MILP improves its solution up to 50 trains. This is why CPI appears the best algorithm for instances with up to 80 trains. For higher numbers, the improvement of CPI in step 2 is not sufficient to compensate for the worse performance of the step 1, hence MILP achieves the best performance,

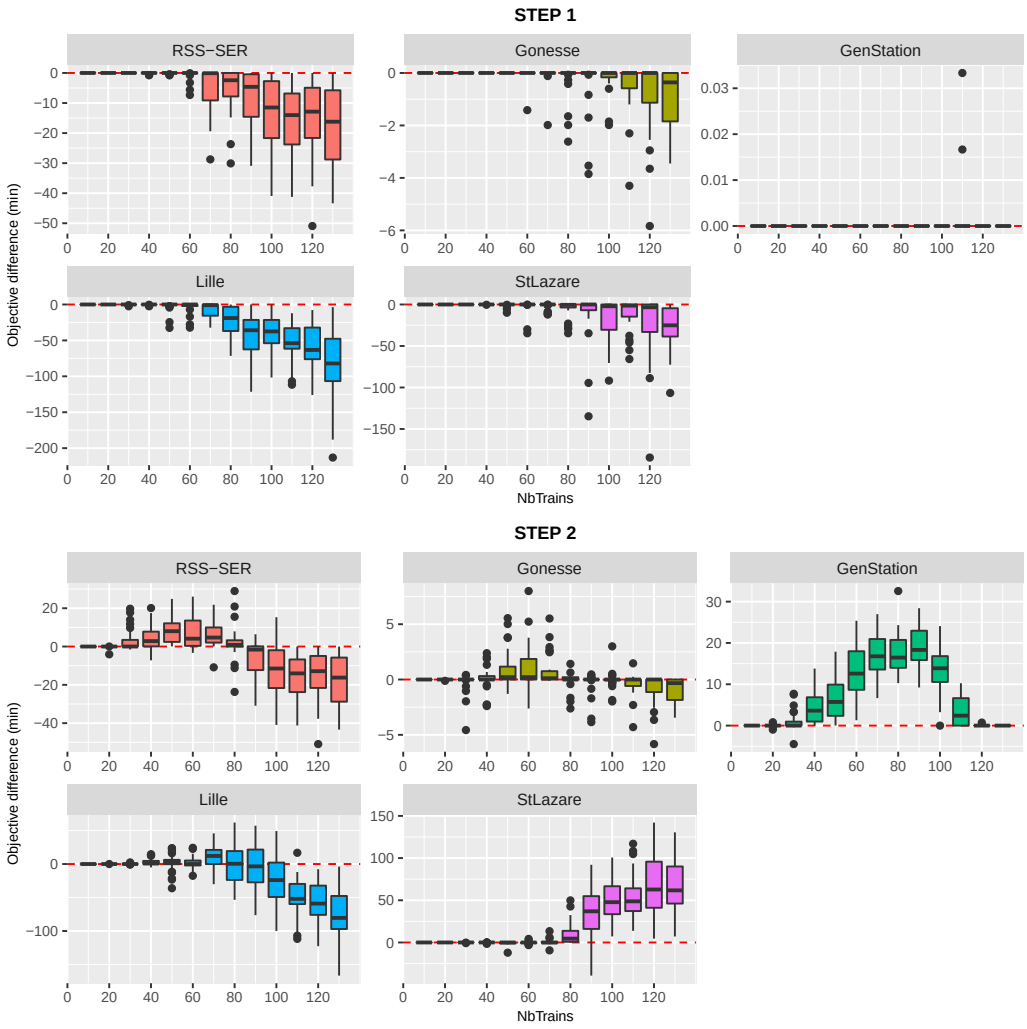
- In Gonesse, although the algorithms perform differently in quite some instances, the magnitude of the difference is so small that the average objective function value is virtually indistinguishable,

- In GenStation, CPI and MILP are equivalent in step 1 for all instances. In step 2, both algorithms improve their solution as in RSS-SER, with CPI improving slightly more for instances from 20 trains to 110 trains and MILP not improving at all for large ones (above 70 trains). The performance regain of MILP is not visible here, differently from RSS-SER, due to the higher tractability of the pure scheduling problem (step 1) in this control area, which removes its advantage over CPI for the largest instances,

- In Lille, the behavior is similar to the one in RSS-SER: the two algorithms are equivalent in step 1 up to 40 trains, then CPI performance gets worse, the difference reaching a value of 17% in average with the 130 train instances. In step 2, CPI improves its solution up to 130 trains while MILP improves its solution up to 80 trains. The differences in improvement between the algorithms in the two steps fully explains the fact that for instances with more than 80 trains CPI no longer gives the best solution after step 2. For example, with 90 train instances, CPI finds step 1 solutions 13% worse than the ones of MILP; then it improves them by 10% in step 2, while MILP does not manage to improve over its own. However, a 3% average difference in favor of MILP remains at the end.

- In StLazare, the algorithms are equivalent with up to 70 trains, then CPI becomes the best. CPI fails to achieve the best results in the first step for large instances. However, it strongly improves its solutions in the second step, much more than MILP, and this absorbs the worse scheduling performance. As for GenStation, the higher tractability of

Figure 11: MILP and CPI performance comparison: Best objective value (MILP) - Best objective value (CPI). STEP 1: pure scheduling. STEP 2: scheduling and routing.



the scheduling problem in this control area removes the MILP advantage over CPI for the largest instances.

Figure 11 graphically shows the comparison of the solution quality achieved by CPI and MILP. A separate graphic is dedicated to each control area. In each of them, a set of boxplots show the difference in the objective function value obtained by MILP and CPI for each set of instances, i.e., for each number of trains: a box depicts the distribution of the obtained values, from the

25-th to the 75-th percentile, with a thick line in between to represent the median. Whiskers show the presence of further relevant observations, while dots stand for outliers. Negative values indicate better results returned by MILP. The first set of five graphics show the performance of the first step of each algorithm, when only scheduling is optimized. The second set of graphics concerns the second step, when also rerouting is considered. This figure confirms the observations made on Table 6, and it shows that the relations on the average values are not due to distribution outliers. This representation highlights the fact that the relative performance of the two algorithms is the same across control areas. What mostly differs is the threshold on the number of trains for which this performance switches from being in favor of CPI to being in favor of MILP. The switch in favor of MILP is only partially visible in GenStation, where for the largest instances the two algorithms are equivalent. It is even less detectable in StLazare. However, the trends followed by the boxplots for these two control areas let us conjecture that a further increase in the number of trains may eventually lead to the elsewhere observed behavior.

In summary, these results show that CPI and MILP are equivalent for small instances: they both find good scheduling solutions and manage to improve them when also rerouting is possible. When the number of trains increases and the instances become more and more difficult, the behavior of CPI and MILP changes. On the one hand, CPI often fails to return the optimal, or a very good sub-optimal, pure scheduling solution in the first step. In the second step, it manages to improve this solution by adding rerouting, up to a point in which the instances become too difficult to do so: for the largest instances, the first step solution is equal to the final one. On the other hand, MILP almost always returns the optimal solutions of the pure scheduling problem in the first step, being often better than CPI. However, its capability to improve this solution through the addition of rerouting is worse than the one of CPI, as it starts failing any relevant improvement in the second step already with intermediate numbers of trains. This is coherent with the observations reported in the literature on the contribution of many rerouting possibilities to the difficulty of instances for MILP (Pellegrini et al., 2015). It is also in agreement with the observations made in Section 7.3 on the number of variables and constraints. Indeed, MILP model size often becomes huge even with intermediate numbers of trains, while CPI manages to keep this size under control for larger instances.

Let us now consider the computation time used during step 1. The mean

computation time and a quality indicator for CPI relative to MILP solutions at the end of step 1 are reported in Table 7. Computation time values smaller than 30 seconds show that optimal solutions are found before the time limit elapses. The CPI quality indicator Gap is the average percent deviation from the optimal or the best solution produced by MILP algorithm at the end of step 1.

The MILP algorithm uses much less than 30 seconds (0.2 to 10 seconds) for almost all instances: it finds the optimal or near optimal solution for all instances in step 1.

This happens much less often for CPI, which manages to complete solution optimality proofs only for about one third of the instances. Even in these instances, the difference in computation time with respect to MILP is more than an order of magnitude.

In the sets of instances for which CPI almost never proves optimality (computation time of 20 to 30 seconds in Table 7) in step 1, we can identify three groups of instances:

- In the first group, the gap is above 1%, meaning that either CPI finds the solution proven optimal by MILP but does not complete its own proof, or it finds a solution whose quality is close to the MILP one. This group includes the RSS-SER instances from 30 to 60 trains, the Gonesse and Generic Junction instances over 60 and 70 trains, respectively, the Lille instances from 30 to 40 trains and the StLazare instances from 30 to 50 trains. For almost all these instances, CPI also reaches better or equivalent quality solutions after step 2.

- The second group of instances has a low gap value, between 1 % and 4%. The group includes RSS-SER instances from 70 to 130 trains, Lille from 50 to 80 trains and StLazare from 60 to 130 trains. As the CPI starting solution in step 2 is still close to the one of MILP, the efficiency of CPI allows to return a better final solution than MILP for Lille and StLazare instances but not for RSS-SER. A possible explanation can be that RSS-SER has a corridor structure, while Lille and StLazare are terminal stations. The routing dimension has probably less influence on the quality of the solution for corridor infrastructure instances than for terminal station instances. The former is "closer" to a scheduling problem and therefore less convenient for CPI.

- The third group of instances has a significant gap value between 8 %

Table 7: Mean computation time of MILP and CPI at the end of step 1, and CPI gap.

#trains	*10*	*20*	*30*	*40*	*50*	*60*	*70*	*80*	*90*	*100*	*110*	*120*	*130*
RSS-SER													
MILP CPU (s)	0.3	0.6	1.3	2	2.9	4.7	7.6	10.3	14.6	19.5	23.4	26	29.3
CPI CPU (s)	2.6	11.4	21.7	26.3	29.3	28.4	27.2	30	30	30	30	30	30
CPI Gap (%)	0 %	0 %	0 %	0 %	0 %	0 %	2 %	2 %	2 %	3 %	4 %	4 %	4 %
Gonesse													
MILP CPU (s)	0.3	0.6	1.1	1.9	3.6	6.3	9.1	10.9	13.1	13.1	15.9	18.8	20.8
CPI CPU (s)	0.6	7	18.3	25.4	27.3	30	30	30	30	30	30	30	30
CPI Gap (%)	0 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %
Generic Junction													
MILP CPU (s)	0.2	0.4	0.5	0.9	1.5	2.4	3.5	3.8	4.5	5.6	7.8	8.8	10.9
CPI CPU (s)	0.5	6.2	16.4	24.5	29.7	29.9	30	30	30	30	30	30	30
CPI Gap (%)	0 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %
Lille													
MILP CPU (s)	0.2	0.3	0.6	1	1.8	2.8	5.6	15.7	24.2	27.2	27.8	29.8	29.3
CPI CPU (s)	0.4	13.1	25.7	29.1	30	30	30	30	30	30	30	30	30
CPI Gap (%)	0 %	0 %	0 %	0 %	2 %	2 %	4 %	8 %	13 %	11 %	13 %	14 %	17 %
StLazare													
MILP CPU (s)	0.2	0.3	0.4	0.6	1	1.4	2	3.2	4.9	6.8	9.7	11.8	16.4
CPI CPU (s)	2	9.9	26.1	27.2	30	30	30	30	30	30	30	30	30
CPI Gap (%)	0 %	0 %	0 %	0 %	0 %	1 %	1 %	2 %	3 %	3 %	2 %	4 %	4 %

and 17%, it includes the Lille instances from 80 to 130 trains. The efficiency of CPI in step 2 is not sufficient to reach the MILP solution of step 1, therefore MILP always gives the best solution for this group of instances.

To conclude on the effectiveness of RECIFE-CPI, we recall that the RECIFE-MILP algorithm has been tested on several industrial projects and presents very good performance resulting from many years of research, which makes it a state-of-the-art model and algorithm for the rtRTMP. In short, the overall results show that RECIFE-CPI has equivalent performance to RECIFE-MILP in small-sized instances, better performance in intermediate-sized instances, and worse performance on some larger-sized instances (RSS-SER, Lille). Therefore, we cannot conclude that RECIFE-CPI outperforms RECIFE-MILP, but we can conclude that it has very good performance and shows some strengths that can be exploited in a hybridization approach.

8. RECIFE-CPI and RECIFE-MILP hybridization

The results presented in Section 7.5 suggest that CPI and MILP have different strengths which make them alternatively the best option to solve

different instances. On the one hand, MILP appears to be better at solving the pure scheduling problem in the first optimization step. On the other hand, CPI appears more effective in dealing with the overall rerouting and scheduling problem, when it is not penalized by the quality of the first step solution.

These observations suggest that an hybridization of the two algorithms may be the way to go to exploit their respective strengths. Several possibilities can be imagined to do so. In this paper, we make a first step towards a full hybridization by defining an algorithm that starts with the first optimization step of MILP and uses the obtained solution as a warm start for the second step of CPI. We name this algorithm Hybrid. Intuitively, Hybrid exploits the pure scheduling efficiency of MILP and makes the best of the rerouting ability of CPI, by warm starting it with an excellent solution and having it running for a longer computational time than in the pure constraint programming algorithm, as explained at the end of Section 7.5.

We test Hybrid in the same experimental setup as described in Section 7, considering all five control areas and the increasing number of trains for each of them.

8.1. Results

Figure 12 depicts the results of these experiments. For each control area, a graphic represents the average performance of CPI, MILP and Hybrid for each number of trains in an instance. For each set of instances, squares show when algorithms find the optimal solution at the first step (abscissa equal to 0) and how much worse their solution is when they fail to do so (negative abscissa value). Indeed, MILP almost always proves optimality, and hence Hybrid too, while CPI returns suboptimal solutions for the largest instances. These squares are joined to circles by horizontal lines. Circles show the percentage improvement achieved in the second step with respect to the optimal first step solution. The longest the line, the larger the contribution of the second step. These graphics depict the CPI and MILP behavior discussed in Section 7.5. Moreover, they show that Hybrid combines the two algorithms' strengths: it always finds the optimal pure scheduling solution in the first step, and improves it in the second step unless for extremely large and complex instances. Moreover, it exploits the short computational time used by MILP in first step: in the second step it uses CPI for solving the rerouting and scheduling problem, for a longer time than what CPI itself can do. The additional time can be quite high, as shown in Table 7. The result is that

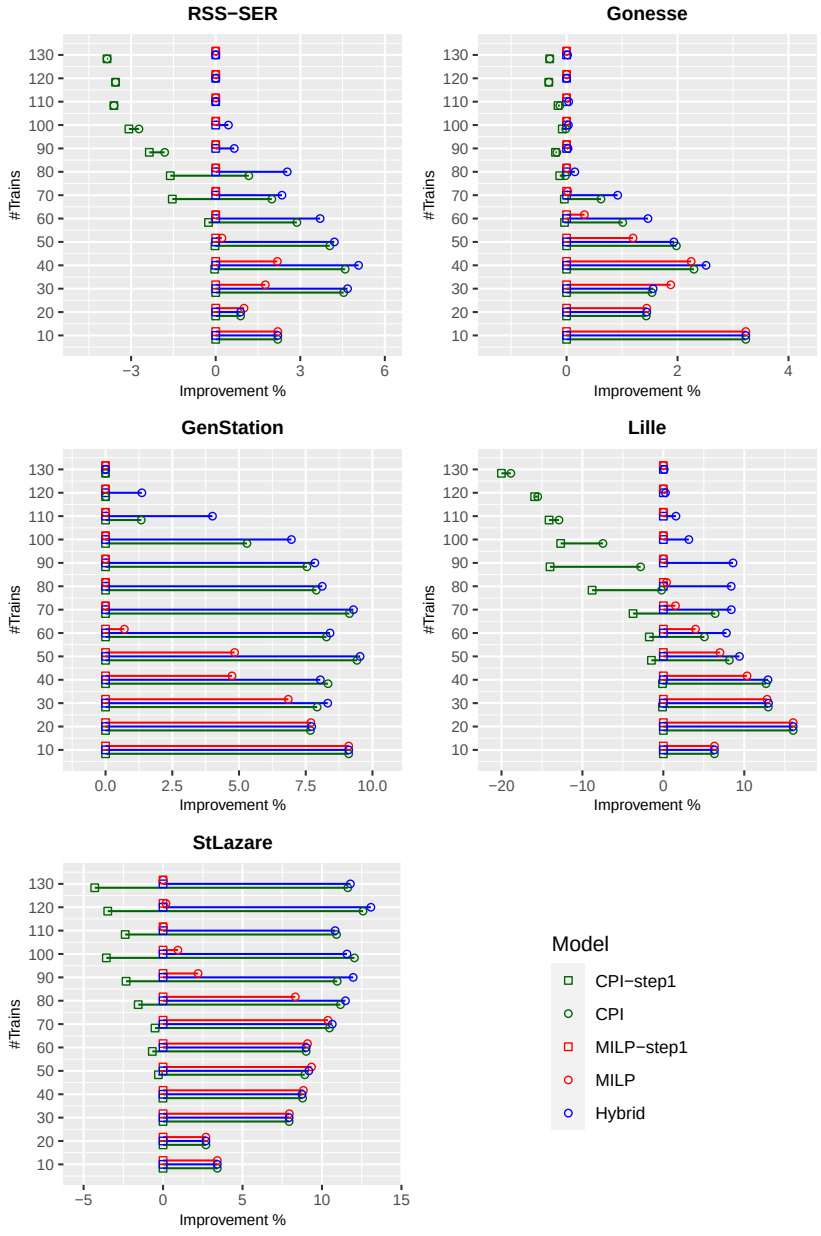


Figure 12: Results of CPI, MILP and Hybrid: percentage improvement with respect to the best solution found in the first optimization step.

Table 8: Results of the Wilcoxon rank-sum test on the difference of final solutions returned by CPI, MILP and Hybrid. A letter indicates that the difference in the results is statistically significant in favor of Hybrid for a set of instance: M indicates that the difference is significant when Hybrid is compared with MILP, C that it is such when the comparison is made with CPI. Neither MILP nor CPI are ever significantly better than Hybrid.

	# **trains**											
case study	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**	**110**	**120**	**130**
RSS-SER			M	M	M	C,M	C,M	C,M	C	C	C	C
Gonesse				M	M	C,M	C			C		C
GenStation			M	M	M	M	C,M	C,M	C,M	C,M		
Lille			M		C,M	M	C,M	C,M	C,M	C	C	C
StLazare							C,M	C,M	M	M	C,M	M

Hybrid often achieves better results of both CPI and MILP, and it is never outperformed. This is confirmed in Table 8, where we report the results of the Wilcoxon rank-sum tests performed for each set of instances. Indeed, the results are in many cases statistically better for Hybrid than for either CPI or MILP, while the opposite never happens. The difference is more striking for intermediate-size instances. When instances are small enough for CPI to find the optimal solution in the first step and large enough for MILP not to be able to improve in the second (e.g., RSS-SER with 40 to 60 trains), Hybrid is better than MILP but not than CPI. As instances grow, it outperforms both. Finally, when instances are too large both for CPI first step and for MILP second step, Hybrid is better than CPI but not than MILP (e.g., RSS-SET with more than 100 trains), of with whom it shares the pure scheduling optimal solution returned at the end.

9. Conclusion

In this paper, we proposed a new Constraint Programming formulation for the real-time Railway Traffic Management Problem. It is based on the concept of time-interval variables which simplifies the modeling of alternative route choices and the blocking time constraints. The solution of this formulation is the basis of a new algorithm named RECIFE-CPI.

We compared RECIFE-CPI with a state-of-the-art algorithm for the rtRTMP, namely RECIFE-MILP. We performed this comparison in a thorough experimental analysis based on data representing traffic perturbations in four real French control areas and in a fictive one, which we name Generic

Station. We considered 13 sets of instances for each control area, with increasing numbers of trains.

The results of this extremely large experimental analysis show that RECIFE-CPI achieves results that are often comparable with the ones of RECIFE-MILP. For largest instances, the former shows a weaker behavior than the latter for solving pure scheduling problems, and a stronger one for solving scheduling and routing ones. Despite the very different characteristics of the cases studies, the relative performance of the two algorithms remains substantially the same across them. What changes when considering different control areas is the number of trains that corresponds to the threshold for having one of the two algorithms performing well or badly for the scheduling or routing problems: "large" instances have different numbers of trains in different control areas, but similar algorithm behaviors are observable when tackling them.

To make our results reproducible, we made the instances concerning the Generic Station control area publicly available. The publication of rtRTMP instances is extremely rare in the field, mostly due to confidentiality issues. Such issues forbid publishing the instances representing the four real control areas used in this paper. To the best of our knowledge, no public data considering a microscopic infrastructure representation exist. Hence, we consider the use of the Generic Station instances particularly relevant in this paper, although it lacks the charm of real data shared by the other control areas. In the experimental analysis, we showed that the difficulty of these instances is comparable to real ones with similar characteristics, i.e., representing traffic in control areas where trains have many short alternative routes available. For example, the Pierrefitte-Gonesse junction instances challenge our algorithms in a way that is comparable to the Generic Station ones. In particular, they are quite tractable when only scheduling is optimized: we are able to find the optimal solution to instances including a very large number of trains in a few seconds when they all travel along the route planned in the infrastructure. Instead, the instances become difficult when also rerouting is considered. The published instances may hopefully become a test-bed for future algorithm comparisons.

After detecting strengths and weaknesses of RECIFE-CPI and RECIFE-MILP, we exploited the former to define a hybrid algorithm. It starts by exploiting RECIFE-MILP pure scheduling strength, and ends by exploiting RECIFE-CPI scheduling and routing one. This hybridization outperforms the two original algorithms for many sets of instances, while the opposite

never happens. Nonetheless, for extremely large and complex instances, no algorithm is capable of exploiting rerouting to improve over the pure scheduling optimal solution in the available computational time.

Future work will be devoted to the investigation of further hybridization possibilities between RECIFE-CPI and RECIFE-MILP. This may allow pushing the boundaries on the characteristics of tractable instances, be it in the number of trains or in the size and type of control areas managed.

Approaches used in other applications are relevant to consider. Among them, those used for wire routing in VLSI design and communication network routing may be particularly interesting. Indeed, as the rtRTMP, these are NP-Hard combinatorial problems, which include constraints on limited resources that must be shared by multiple entities (Lengauer, 1990; Chen et al., 2000). Approaches to solving these problems may help in a pre-processing phase to reduce the set of routes to the most promising ones for each train. This may simplify the problem solution and allow the use of methods as, e.g., the approximation algorithms developed in Terlaky et al. (2008).

Finally, the CPI solution method used in this paper is a general-purpose method. By making use of the specific properties of the rtRTMP, speed-ups can be expected. Therefore, in the future we will develop tailored CPI solution methods for the rtRTMP.

References

Albrecht, A., Howlett, P., Pudney, P., Vu, X., Zhou, P., 2016a. The key principles of optimal train control – Part 1: Formulation of the model, strategies of optimal type, evolutionary lines, location of optimal switching points. Transportation Research Part B: Methodological 94, 482–508.

Albrecht, A., Howlett, P., Pudney, P., Vu, X., Zhou, P., 2016b. The key principles of optimal train control – Part 2: Existence of an optimal strategy, the local energy minimization principle, uniqueness, computational techniques. Transportation Research Part B: Methodological 94, 509–538.

Albrecht, T., 2009. The influence of anticipating train driving on the dispatching process in railway conflict situations. Networks and Spatial Economics 9, 85–101.

Albrecht, T., 2010. Reducing power peaks and energy consumption in rail

transit systems by simultaneous train running time control. WIT Transactions on State-of-the-art in Science and Engineering 39.

Altazin, E., Dauzère-Pérès, S., Ramond, F., Tréfond, S., 2017. Rescheduling through stop-skipping in dense railway systems. Transportation Research Part C: Emerging Technologies 79, 73–84.

Altazin, E., Dauzère-Pérès, S., Ramond, F., Tréfond, S., 2020. A multi-objective optimization-simulation approach for real time rescheduling in dense railway systems. European Journal of Operational Research 286, 662–672.

Araya, S., Sone, S., 1984. Traffic dynamics of automated transit systems with pre-established schedules. IEEE Transactions on Systems, Man, and Cybernetics SMC-14, 677–687.

Blenkers, L., 2015. Railway disruption management,. Master's thesis. Delft Center Syst. Control, Delft Univ. Technol. Delft, The Netherlands.

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

Caimi, G., Fuchsberger, M., Laumanns, M., Lüthi, M., 2012. A model predictive control approach for discrete-time rescheduling in complex central railway station areas. Computers & Operations Research 39, 2578 – 2593.

Cappart, Q., Schaus, P., 2017. Rescheduling railway traffic on real time situations using time-interval variables, in: Salvagnin, D., Lombardi, M. (Eds.), Integration of AI and OR Techniques in Constraint Programming, Springer International Publishing, Cham. pp. 312–327.

Cavone, G., van den Boom, T., Blenkers, L., Dotoli, M., Seatzu, C., De Schutter, B., 2022. An mpc-based rescheduling algorithm for disruptions and disturbances in large-scale railway networks. IEEE Transactions on Automation Science and Engineering 19, 99–112.

Chen, S., Günlük, O., Yener, B., 2000. The multicast packing problem. IEEE/ACM Transactions on Networking 8, 311–318.

Corman, F., D'Ariano, A., Pacciarelli, D., Pranzo, M., 2010a. Centralized versus distributed systems to reschedule trains in two dispatching areas. Public Transport 2, 219–247.

Corman, F., D'Ariano, A., Pacciarelli, D., Pranzo, M., 2010b. A tabu search algorithm for rerouting trains during rail operations. Transportation Research Part B: Methodological 44, 175–192.

Corman, F., D'Ariano, A., Pacciarelli, D., Pranzo, M., 2012. Optimal inter-area coordination of train rescheduling decisions. Transportation Research Part E: Logistics and Transportation Review 48, 71–88. Select Papers from the 19th International Symposium on Transportation and Traffic Theory.

Corman, F., D'Ariano, A., Pacciarelli, D., Pranzo, M., 2014. Dispatching and coordination in multi-area railway traffic management. Computers & Operations Research 44, 146 – 160.

Corman, F., Meng, L., 2015. A review of online dynamic models and algorithms for railway traffic management. IEEE Transactions on Intelligent Transportation Systems 16, 1274–1284.

Cucala, A.P., Fernandez, A., de Cuadra, F., Pilo, E., 2007. An optimisation-based traffic regulator for metro lines. WIT Transactions on The Built Environment 96.

Cury, J., Gomide, F., Mendes, M., 1980. A methodology for generation of optimal schedules for an underground railway system. IEEE Transactions on Automatic Control 25, 217–222.

D'Ariano, A., Albrecht, T., 2006. Running time re-optimization during real-time timetable perturbations. WIT Transactions on The Built Environment 88.

D'Ariano, A., Corman, F., Pacciarelli, D., Pranzo, M., 2008. Reordering and local rerouting strategies to manage train traffic in real time. Transportation Science 42, 405–419.

D'Ariano, A., Pacciarelli, D., Pranzo, M., 2007. A branch and bound algorithm for scheduling trains in a railway network. European Journal of Operational Research 183, 643–657.

Diehl, M., Bock, H., Diedam, H., Wieber, P.B., 2006. Fast Motions in Biomechanics and Robotics: Optimization and Feedback Control. Springer Berlin Heidelberg, Berlin, Heidelberg. chapter Fast Direct Multiple Shooting Algorithms for Optimal Robot Control. pp. 65–93.

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

Fernandez, A., Cucala, A.P., Vitoriano, B., de Cuadra, F., 2006. Predictive traffic regulation for metro loop lines based on quadratic programming. Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit 220, 79–89.

Fischetti, M., Monaci, M., 2017. Using a general-purpose mixed-integer linear programming solver for the practical solution of real-time train rescheduling. European Journal of Operational Research 263, 258 – 264.

Fukumori, K., 1980. Fundamental scheme for train scheduling: Application of range-constriction search. Technical Report A.I. Memo No. 596. Massachusetts Institute of Technology Artificial Intelligence Laboratory. USA.

García, C.E., Prett, D.M., Morari, M., 1989. Model predictive control: Theory and practice – a survey. Automatica 25, 335–348.

Goverde, R.M., Scheepmaker, G.M., Wang, P., 2021. Pseudospectral optimal train control. European Journal of Operational Research 292, 353–375.

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

Hooker, J.N., van Hoeve, W.J., 2018. Constraint programming and operations research. Constraints 23, 172–195.

Howlett, P., 1990. An optimal strategy for the control of a train. The Journal of the Australian Mathematical Society. Series B. Applied Mathematics 31, 454–471.

Howlett, P., 2000. The optimal control of a train. Annals of Operations Research 98, 65–87.

Ingolitto, L., Barber, F., Tormos, P., Lova, A., Salido, M., Abril, M., 2004. An efficient method to schedule new trains on a heavily loaded railway network, in: Reyes, C., Gonzales, J. (Eds.), Advances in Artificial Intelligence, IBERAMIA 2004, 9th Ibero-American Conference on AI, Springer Verlag, Puebla, Mexico. pp. 164–173.

Isaai, M., Singh, M., 2000. An intelligent constraint-based search method for a single-line passenger-train scheduling problems, in: The Internation Conference on the Practical Application of Constraint Technologies and Logic Programming, pp. 79–91.

Kersbergen, B., Rudan, J., van den Boom, T., De Schutter, B., 2016a. Towards railway traffic management using switching max-plus-linear systems. Discrete Event Dynamic Systems 26, 183–223.

Kersbergen, B., van den Boom, T., De Schutter, B., 2016b. Distributed model predictive control for railway traffic management. Transportation Research Part C: Emerging Technologies 68, 462–489.

Khosravi, B., Bennell, J.A., Potts, C.N., 2012. Train Scheduling and Rescheduling in the UK with a Modified Shifting Bottleneck Procedure, in: Delling, D., Liberti, L. (Eds.), 12th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany. pp. 120–131.

Kreuger, P., Carlsson, M., Sjöland, T., Aström, E., 2001. Sequence dependent task extensions for trip scheduling. Technical Report SICS-T-2001/14-SE. Swedish Institute of Computer Science (SICS),.

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

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

Laborie, P., Rogerie, J., 2008. Reasoning with conditional time-intervals, in: Twenty-First International FLAIRS Conference, pp. 555–560.

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

Laborie, P., Rogerie, J., Shaw, P., Vilím, P., 2018. IBM ILOG CP Optimizer for scheduling. Constraints 23, 210–250.

Lamma, E., Mello, P., Milano, M., 1997. A distributed constraint-based scheduler. Artificial Intelligence in Engineering 11, 91–105.

Lengauer, T., 1990. Combinatorial Algorithms for Integrated Circuit Layout. XApplicable Theory in Computer Science. John Wiley & Sons ed., Vieweg+Teubner Verlag Wiesbaden.

Luan, X., Corman, F., Meng, L., 2017a. Non-discriminatory train dispatching in a rail transport market with multiple competing and collaborative train operating companies. Transportation Research Part C: Emerging Technologies 80, 148–174.

Luan, X., Corman, F., Meng, L., 2017b. Non-discriminatory train dispatching in a rail transport market with multiple competing and collaborative train operating companies. Transportation Research Part C: Emerging Technologies 80, 148–174.

Luan, X., Corman, F., Wang, Y., Meng, L., Lodewijks, G., 2017c. Integrated optimization of traffic management and train control for rail networks, in: 7th International Conference on Railway Operations Modelling and Analysis - RailLille2017, Lille, France, pp. 1413–1432.

Luan, X., De Schutter, B., Meng, L., Corman, F., 2020. Decomposition and distributed optimization of real-time traffic management for large-scale railway networks. Transportation Research Part B: Methodological 141, 72 – 97.

Luan, X., Schutter, B.D., van den Boom, T., Corman, F., Lodewijks, G., 2018a. Distributed optimization for real-time railway traffic management. IFAC-PapersOnLine 51, 106–111. 15th IFAC Symposium on Control in Transportation Systems CTS 2018.

Luan, X., Wang, Y., De Schutter, B., Meng, L., Lodewijks, G., Corman, F., 2018b. Integration of real-time traffic management and train control for

rail networks - part 1: Optimization problems and solution approaches. Transportation Research Part B: Methodological 115, 41–71.

Luan, X., Wang, Y., De Schutter, B., Meng, L., Lodewijks, G., Corman, F., 2018c. Integration of real-time traffic management and train control for rail networks - part 2: Extensions towards energy-efficient train operations. Transportation Research Part B: Methodological 115, 72–94.

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

Lusby, R.M., Larsen, J., Ehrgott, M., Ryan, D.M., 2013. A set packing inspired method for real-time junction train routing. Computers & Operations Research 40, 713–724.

Lüthi, M., 2009. Improving the Efficiency of Heavily Used Railway Networks through Integrated Real-Time Rescheduling. Diss. ETH No. 18615. Institute for Operations Research, ETH Zurich.

Mannino, C., Mascis, A., 2009. Optimal real-time traffic control in metro stations. Operations Research 57, 1026–1039.

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

Mascis, A., Pacciarelli, D., Pranzo, M., 2002. Models and algorithms for traffic management of rail networks. Technical Report DIA-74-2002. Dipartimento di Informatica e Automazione, Universitá Roma Tre.

Mazzarello, M., Ottaviani, E., 2007. A traffic management system for real-time traffic optimisation in railways. Transportation Research Part B: Methodological 41, 246–274.

Meng, L., Zhou, X., 2014. Simultaneous train rerouting and rescheduling on an n-track network: A model reformulation with network-based cumulative flow variables. Transportation Research Part B: Methodological 67, 208 – 234.

Montrone, T., Pellegrini, P., Nobili, P., 2018. Real-time energy consumption minimization in railway networks. Transportation Research Part D: Transport and Environment 65, 524–539.

Oliveira, E.S., 2001. Solving Single-Track Railway Scheduling Problem Using Constraint Programming. Ph.D. thesis. School of Computing - University of Leeds.

Pellegrini, P., Marlière, G., Pesenti, R., Rodriguez, J., 2015. RECIFE-MILP: An Effective MILP-Based Heuristic for the Real-Time Railway Traffic Management Problem. IEEE Transactions on Intelligent Transportation Systems 16, 2609–2619.

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

Pellegrini, P., Marlière, G., Rodriguez, J., 2016. A detailed analysis of the actual impact of real-time railway traffic management optimization. Journal of Rail Transport Planning & Management 6, 13–31.

Pellegrini, P., Pesenti, R., Rodriguez, J., 2019. Efficient train re-routing and rescheduling: Valid inequalities and reformulation of RECIFE-MILP. Transportation Research Part B: Methodological 120, 33 – 48.

Quaglietta, E., Pellegrini, P., Goverde, R.M., Albrecht, T., Jaekel, B., Marliere, G., Rodriguez, J., Dollevoet, T., Ambrogio, B., Carcasole, D., Giaroli, M., 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.

Reynolds, E., Ehrgott, M., Maher, S.J., Patman, A., , Wang, J.Y., 2020. A multicommodity flow model for rerouting and retiming trains in real-time to reduce reactionary delay in complex station areas. Optimization Online , 1–37.

Rivier, R., Lucchini, L., Curchod, A., 2001. Analysing the capacity of railway networks: Summing up the experience, in: 9 th World Conference on Transportation Research, pp. 1–15.

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

Rodriguez, J., 2007b. A study of the use of state resources in a constraint-based model for routing and scheduling trains, in: 2nd International Seminar on Railway Operations Modelling and Analysis, Hannover, Germany. pp. 1–14.

Rodriguez, J., Delorme, X., Gandibleux, X., Marlière, G., Bartusiak, R., Degoutin, F., Sobieraj, S., 2007. RECIFE : modèles et outils pour l'analyse de la capacité ferroviaire. Recherche Transports Sécurité 95, 19–36. (in French).

Rodriguez, J., Kermad, L., 1998. Constraint programming for real-time train circulation management problem in railway nodes, Computers in Railways VI. WIT Transactions on The Built Environment 37.

Rodriguez, J., Marlière, G., Sobieraj, S., 2010. A constraint-based scheduling model for optimal train dispatching, in: Joint Rail Conference, Urbana-Champaign, Illinois, USA. pp. 399–406.

Samà, M., D'Ariano, A., Corman, F., Pacciarelli, D., 2017. A variable neighbourhood search for fast train scheduling and routing during disturbed railway traffic situations. Computers & Operations Research 78, 480–499.

Samà, M., Pellegrini, P., D'Ariano, A., Rodriguez, J., Pacciarelli, D., 2016. Ant colony optimization for the real-time train routing selection problem. Transportation Research Part B: Methodological 85, 89–108.

Scheepmaker, G.M., Goverde, R.M., Kroon, L.G., 2017. Review of energy-efficient train control and timetabling. European Journal of Operational Research 257, 355–376.

Schutter, B.D., van den Boom, T., Hegyi, A., 2002. Model predictive control approach for recovery from delays in railway systems. Transportation Research Record 1793, 15–20.

Söhlke, A., Rodrigues, R., 2022. Deliverable D8.2 Standardized and automated conflict handling solution. Technical Report. Shift2Rail - X2RAIL-4 project.

Strotmann, C., 2007. Railway scheduling problems and their decomposition. Ph.D. thesis. Universität Osnabrück.

Terlaky, T., Vannelli, A., Zhang, H., 2008. On routing in VLSI design and communication networks. Discrete Applied Mathematics 156, 2178–2194.

Toletti, A., Laumanns, M., Weidmann, U., 2020. Coordinated railway traffic rescheduling with the resource conflict graph model. Journal of Rail Transport Planning & Management 15, 100173.

Törnquist, J., Persson, J.A., 2007. N-tracked railway traffic re-scheduling during disturbances. Transportation Research Part B: Methodological 41, 342–362.

Van Breusegem, V., Campion, G., Bastin, G., 1991. Traffic modeling and state feedback control for metro lines. IEEE Transactions on Automatic Control 36, 770–784.

van den Boom, T., De Schutter, B., 2006. Modelling and control of discrete event systems using switching max-plus-linear systems. Control Engineering Practice 14, 1199–1211. The Seventh Workshop On Discrete Event Systems (WODES2004).

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

Vilím, P., Laborie, P., Shaw, P., 2015. Failure-directed search for constraint-based scheduling, in: International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, Springer, Cham. pp. 437–453.

Wang, P., Goverde, R.M., 2016. Multiple-phase train trajectory optimization with signalling and operational constraints. Transportation Research Part C: Emerging Technologies 69, 255–275.

Wang, P., Goverde, R.M., 2017. Multi-train trajectory optimization for energy efficiency and delay recovery on single-track railway lines. Transportation Research Part B: Methodological 105, 340–361.

Wang, X., Li, S., Tang, T., Yang, L., 2022a. Event-triggered predictive control for automatic train regulation and passenger flow in metro rail systems. IEEE Transactions on Intelligent Transportation Systems 23, 1782–1795.

Wang, Y., De Schutter, B., van den Boom, T.J.J., Ning, B., Tang, T., 2014a. Efficient bilevel approach for urban rail transit operation with stop-skipping. IEEE Transactions on Intelligent Transportation Systems 15, 2658–2670.

Wang, Y., De Schutter, B., van den Boom, T.J., Ning, B., 2013. Optimal trajectory planning for trains – a pseudospectral method and a mixed integer linear programming approach. Transportation Research Part C: Emerging Technologies 29, 97–114.

Wang, Y., De Schutter, B., van den Boom, T.J., Ning, B., 2014b. Optimal trajectory planning for trains under fixed and moving signaling systems using mixed integer linear programming. Control Engineering Practice 22, 44–56.

Wang, Y., De Schutter, B., van den Boom, T.J., Ning, B., Tang, T., 2014c. Origin-destination dependent train scheduling problem with stop-skipping for urban rail transit systems, in: Proceedings of the 93rd Annual Meeting of the Transportation Research Board, Washington, DC. pp. 14–188.

Wang, Y., Ning, B., Tang, T., van den Boom, T.J.J., De Schutter, B., 2015a. Efficient real-time train scheduling for urban rail transit systems using iterative convex programming. IEEE Transactions on Intelligent Transportation Systems 16, 3337–3352.

Wang, Y., Tang, T., Ning, B., van den Boom, T.J., De Schutter, B., 2015b. Passenger-demands-oriented train scheduling for an urban rail transit network. Transportation Research Part C: Emerging Technologies 60, 1–23.

Wang, Y., Zhu, S., Li, S., Yang, L., De Schutter, B., 2022b. Hierarchical model predictive control for on-line high-speed railway delay management and train control in a dynamic operations environment. IEEE Transactions on Control Systems Technology 30, 2344–2359.

Xu, P., Corman, F., Peng, Q., Luan, X., 2017. A train rescheduling model integrating speed management during disruptions of high-speed traffic under a quasi-moving block system. Transportation Research Part B: Methodological 104, 638–666.

Yang, X., Chen, A., Li, X., Ning, B., Tang, T., 2015. An energy-efficient scheduling approach to improve the utilization of regenerative energy for

metro systems. Transportation Research Part C: Emerging Technologies 57, 13–29.

Yang, X., Li, X., Ning, B., Tang, T., 2016. A survey on energy-efficient train operation for urban rail transit. IEEE Transactions on Intelligent Transportation Systems 17, 2–13.

Yin, J., Tang, T., Yang, L., Xun, J., Huang, Y., Gao, Z., 2017. Research and development of automatic train operation for railway transportation systems: A survey. Transportation Research Part C: Emerging Technologies 85, 548–572.