Effective train routing selection for real-time traffic management: Improved model and ACO parallel computing

Bianca Pascariu, Marcella Sama, Paola Pellegrini, Andrea Dariano, Joaquin Rodriguez, Dario Pacciarelli

Effective train routing selection for real-time traffic management: improved model and ACO parallel computing

B. Pascariu, M. Samà, P. Pellegrini, A. D'Ariano, J. Rodriguez, D. Pacciarelli

Effective train routing selection for real-time traffic management: improved model and ACO parallel computing

B. Pascariu[1*], M. Samà[1], P. Pellegrini[2], A. D'Ariano[1], J. Rodriguez[3], D. Pacciarelli[1]

[1]Roma Tre University, Department of Engineering, Via della Vasca Navale 79, 00146 Rome, Italy [2]Univ. Lille Nord de France, Ifsttar, COSYS, LEOST, rue Élisée Reclus 20, 59666 Villeneuve d'Ascq, Lille, France [3]Univ. Lille Nord de France, Ifsttar, COSYS, ESTAS, rue Élisée Reclus 20, 59666 Villeneuve d'Ascq, Lille, France

* Corresponding author email: bianca.pascariu@uniroma3.it

Abstract

The real-time Railway Traffic Management Problem (rtRTMP) is the problem of detecting and solving time-overlapping conflicting requests made by multiple trains on the same track resources. This problem consists in retiming, reordering and rerouting trains in such a way that the propagation of disturbances in the railway network is minimized. The rtRTMP is an NP-complete problem and finding good strategies to simplify its solution process is paramount to obtain good quality results in a short computation time. Solving the Train Routing Selection Problem (TRSP) aims to reduce the size of rtRTMP instances by limiting the number of routing variables: during the pre-processing, the most promising routing alternatives among the available ones are selected for each train. Then, the selected alternatives are the only ones used for the rtRTMP. A first version of the TRSP has been recently proposed in the literature. This paper presents an improved TRSP model, where rolling stock re-utilization timing constraints and estimation of train delay propagation are taken into account. Additionally, a parallel Ant Colony Optimization (ACO) algorithm is proposed. We analyze the impact of the TRSP model and algorithm on the rtRTMP through a thorough computational campaign performed on a French case study with timetable disturbances and infrastructure disruptions. The presented model leads to a better correlation between TRSP and rtRTMP solutions, and the proposed ACO algorithm outperforms the state-of-the-art algorithm.

Keywords: Rail Transportation, Train Scheduling and Routing, Ant Colony Optimization, Parallel Computing, Disturbance Management.

1 Introduction

To ensure safe and regular train services, rail infrastructure managers design timetables periodically and in advance of their practical implementation (Tang et al., 2021). However, train operations are subordinated to a large number of parameters, making them vulnerable to disturbances. These may cause train conflicts, i.e., the simultaneous request to utilize the same infrastructure section by more than one train. Dispatchers are thus required to quickly detect new conflicts arising during operations and take actions to recover feasibility (Corman et al., 2014), typically by retiming, reordering and rerouting trains in such a way that the propagation of disturbances is minimized. This problem is known in literature as the real-time Railway Traffic Management Problem (rtRTMP).

Several models and algorithms have been developed to solve this problem and provide decision support systems to help dispatchers take more informed decisions (Törnquist, 2006). Still, the rtRTMP is NP-complete and good quality solutions must be found in the short computation times imposed by the nature of the problem. In fact, the instance size highly increases in larger stations and complex junctions, e.g., congested bottleneck areas. Large infrastructures, high numbers of trains and available routes result in a huge number of constraints and variables, thus it becomes challenging to find a good quality solution. To increase the tractability of rtRTMP instances, some approaches limit the size of the problem by intervening on the granularity used to model infrastructures and traffic flows (Pellegrini et al., 2019; Cavone et al., 2019) or focus on the solution process to properly drive the search (Pellegrini et al., 2015; Samà et al., 2016). Other approaches limit the number of variables by considering only what they perceive to be the most significant ones (Van Thielen et al., 2018). This paper focuses on the improvement of the rtRTMP solutions where optimized pre-processing is used to limit the number of routing variables.

In large stations, routing variables are among the factors that mostly affect the size of the rtRTMP search space (Boccia et al., 2013). Some approaches disregard them and use the timetable routes, limiting the rtRTMP to a pure scheduling problem. However, it is well established that rerouting is a strong action for improving rtRTMP solutions (D'Ariano et al., 2008; Corman et al., 2010; Pellegrini et al., 2016). Moreover, in case of infrastructure disruptions (Dollevoet et al., 2017) or maintenance (D'Ariano et al., 2019; Zhang et al., 2019), routing decisions might be indispensable. Some approaches tackle this issue by using effective algorithms that concurrently optimize train rerouting and rescheduling to solve the rtRTMP. Besides these solution methods, recent approaches show that limiting in a smart way the number of routing variables available for each train further improves the rtRTMP solutions. This pre-process requires solving the Train Routing Selection Problem (TRSP), and consists in selecting a feasible and optimized subset of alternative routes for each train. In the TRSP, the benefit of using specific subsets of routes in the rtRTMP is assessed by considering estimations of costs. Costs are often measured as a function of delays, but could in general account for very different indicators (Samà et al., 2015) such as train

travel times.

The need to consider routing subsets when deciding whether a certain assignment of routes to trains is feasible dates back to Kroon et al. (1997). They proved that train routing is an NP-complete problem as soon as each train has three routing possibilities. Over time, significant research efforts have been devoted in developing methods that solve train routing concurrently with rescheduling, while the TRSP has been a rather neglected topic. Some recent approaches use subsets of all possible alternative routes available, which are either based on guidelines set by infrastructure managers (Caimi et al., 2011), or chosen because considered the ones that will probably lead to the best quality solutions. Samà et al. (2016) extensively study this latter option: they propose the formalization of the TRSP by a construction graph, together with an integer linear program and a model to estimate the costs due to route choices in the rtRTMP, in terms of conflict occurrence and scheduling decision impacts. Moreover, they develop an Ant Colony Optimization (ACO) algorithm to solve the problem. Pascariu et al. (2021) show that the ACO-TRSP algorithm of Samà et al. (2016) is able to converge to globally optimal solutions. A solution is globally optimal when no other solutions with better objective value exist in the entire solution space. However, as the size of the instances increases, this algorithm starts struggling: it progressively improves the incumbent solution, but it does not converge to optimum in the limited computation time set.

In this paper, as a first contribution, we improve the TRSP model. Specifically, we aim to increase the correlation between TRSP and rtRTMP. By boosting the TRSP route choice through better conflict and scheduling cost estimations; we want to identify the best routes to be used by the rtRTMP solver, in order to find the highest quality solutions, i.e, those that minimize the total propagation of train delay. Two incremental upgrades are proposed for cost estimations: we strengthen the link between train movements performed by the same rolling stock, which we refer to as *rolling stock re-utilization*, such as turnarounds; we then assess the impact of delay propagation due to train conflicts. As a second contribution, we improve the TRSP solution process. On the one hand, we develop a parallel ACO-TRSP algorithm to speed-up the search space exploration in the available computation time. On the other hand, we propose and test two local search algorithms, based on parallel computing, to enlarge the solution search. The purpose is to escape from local minima, find better quality solutions, and possibly reach global optima.

To assess the benefits brought by these contributions, we carry out a thorough campaign of experiments on large-size instances, referring to the French station area of Lille Flandres, with timetable *disturbances* and infrastructure *disruptions*. A disturbance is a small perturbation of the timetable, while a disruption is a large disturbance (e.g., a track blockage) that requires major adjustments of planned traffic flow. Different variants of ACO-TRSP, incrementally including each of our algorithmic and modeling contributions, are applied to supply routing alternatives to the RECIFE-MILP solver for the rtRTMP (Pellegrini et al., 2015). We compare the performance of

these variants with the state-of-the-art algorithm introduced by Samà et al. (2016) and we analyze the statistical significance of the improvement achieved by each contribution.

The rest of this paper is structured as follows: Section 2 reviews the relevant literature on TRSP; Sections 3 and 4 provide a description of the rtRTMP and TRSP models used; Section 5 illustrates the novel TRSP cost estimation model, where rolling stock re-utilization and train delay propagation are considered; Section 6 describes the parallel ACO algorithm and the two local search algorithms; Section 7 shows the computational results and their discussion, and Section 8 summarizes the conclusions and suggests where to focus future research.

2 Literature review

The rtRTMP is the problem of detecting and solving time-overlapping conflicting requests done by multiple trains on the same track resources. It typically consists in taking train retiming, reordering and rerouting decisions in such a way that the propagation of disturbances in the railway network is minimized. The rtRTMP is an NP-complete problem (Kroon et al., 1997; Mascis & Pacciarelli, 2002), reason that has motivated over time a great research effort to find good strategies to simplify its modeling and solving process, to achieve high quality solutions in a short computation time.

Zhu & Goverde (2019) and Zhang et al. (2021) formulate the problem as a Mixed Integer Linear Program (MILP) at *macroscopic* level, i.e., aggregating the network mainly into nodes and edges, corresponding to stations and track segments. Šemrov et al. (2016) propose a reinforcement Q-learning algorithm for a simulation based model, that minimizes the total train delay induced by an initial disturbance. The problem is tackled at a *microscopic* level, i.e., addressing a detailed representation of the infrastructure, where each resource corresponds to a single block-section or even to a single track-circuit. However, the authors consider simplified modeling assumptions on train length, signalling and interlocking systems, that may not be compatible with the rail practice. Cavone et al. (2019) use a bi-level train rescheduling algorithm based on a mesoscopic MILP model, which minimizes train delays, cancellations, and shunting in stations. *Mesoscopic* approaches are a mix of macroscopic and microscopic ones, e.g., multiple block-sections are grouped into one. In all of these three types of works, the rtRTMP is studied as a pure train rescheduling problem, with no train rerouting decisions.

Recent works include train rerouting options to allow for improvement of a rtRTMP solution by redistributing traffic in the rail network. Binder et al. (2017) formulate the rtRTMP as a multi-objective Integer Linear Program (ILP), based on a space-time graph, considering the minimization of passenger dissatisfaction, operational costs and deviations from the timetable. Veelenturf et al. (2016) propose an ILP formulation, based on an event-activity network, to solve the rtRTMP while minimizing the number of cancelled and delayed trains. Both works (Binder et al., 2017; Veelen-

turf et al., 2016) use a macroscopic infrastructure representation to simplify the train scheduling problem, taking into account the possibility of rerouting trains, but disregarding routes inside stations or junctions to reduce the size of the instances. To allow rerouting in all possible areas, some approaches in the literature tackle the solution process to properly drive the search for good solutions. For example, Pellegrini et al. (2014) formulate the problem as a MILP model and solve it by computing an initial solution for the train scheduling problem (where, for each train, the timetable route is used) and then improve this solution by enlarging the search space while adding all available rerouting variables. D'Ariano et al. (2007), Corman et al. (2010) and Samà et al. (2017a), instead, model the problem by using alternative graphs (Mascis & Pacciarelli, 2002) and iteratively solve the train scheduling and routing problems separately. Furthermore, Gholami & Törnquist (2018) use a heuristic algorithm to solve the rtRTMP, based on a hybridization of the disjunctive and alternative graphs, modeling the rail infrastructure at a mesoscopic level.

Another approach to increase the computational efficiency when solving the rtRTMP is to reduce the number of variables. Looking at train retiming and reordering variables, Van Thielen et al. (2018) only include those variables involving the trains engaged in initial timetable disturbances, i.e., those not caused by delay propagation. Pellegrini et al. (2019) reformulate the microscopic RECIFE-MILP model presented in Pellegrini et al. (2014) by modifying the constraints and reducing the number of train scheduling variables, by means of valid inequalities that link routing and scheduling variables. Toletti et al. (2020) solve the rtRTMP for railway systems by using a decomposition and coordination framework, modelled according to the resource conflict graph of Caimi et al. (2011). They use a MILP commercial solver, which includes train routing variables as well, and an ad-hoc developed column generation approach to solve the local train rescheduling problems at a microscopic level. Fischetti & Monaci (2017) limit, in a pre-processing of a MILP solver, the binary variables associated with alternative routes for each train. The pre-processing proposed is not optimized as they consider (for each train) the timetable route and a randomly chosen alternative route. Samà et al. (2016) formalize the Train Routing Selection Problem (TRSP) as a pre-processing step of the rtRTMP, in which a feasible and optimized subset of alternative routes is selected for each train.

Both Samà et al. (2016) and Fischetti & Monaci (2017) show that a partial routing flexibility is to be preferred to a complete routing flexibility in the rtRTMP: the alternative routings available for each train strongly affect the problem size and the required computation time. Traditional TRSP approaches use subsets of alternative routes predefined by infrastructure managers (Caimi et al., 2011; Bettinelli et al., 2017), while others use optimization algorithms to select the best routes. Samà et al. (2016) develop an Ant Colony Optimization (ACO) algorithm (Dorigo & Stützle, 2004) with the objective function that approximates the one of the rtRTMP. Pascariu et al. (2021) use a mixed-integer linear programming model to prove that this ACO algorithm (Samà et al., 2016)

converges to the optimum for small instances, while large instances are more challenging to be solved to optimum. Moreover, Samà et al. (2017b) analyze the impact of solving the TRSP at tactical and operational levels. For the former, the routing selection is based on records of past perturbed traffic. While for second one, the selection is based on the real-time traffic perturbation. The study shows that the operational level allows to better consider the effect of the routing selection, rather than the tactical one.

It is evident how a wide trend exists toward developing models that simplify the rtRTMP solution process. Along with this trend, the TRSP represents a useful option, but more effective and reliable algorithms could be used for large networks. The high number of alternative train routes available in these instances, which can reach hundreds of routings per train, leads to a huge solution space. The ACO-TRSP of Samà et al. (2016) finds difficult the effective exploration of such a large solution space in real-time. A possible research opportunity is offered by parallel computing. Although parallel programming is well established in operations research, rather few and very recent implementations exist for railway traffic management algorithms. Bettinelli et al. (2017) use parallel computing to perform independent executions of a greedy algorithm, which schedules trains on a time-space network by different criteria, and select the best option. Josyula et al. (2018) propose a parallel computing algorithm to simultaneously explore different parts of a train-conflict binary tree.

In this paper, we contribute to fill the research gap represented by the lack of an advanced approach that can optimally solve large instances by considering the infrastructure at a microscopic level. We integrate the research trend on the TRSP by improving its current state-of-the-art algorithm, namely ACO-TRSP by Samà et al. (2016). We exploit modeling enhancements by including rolling stock re-utilization constraints and by improving the estimation of train delay propagation, to identify the best routes to be used by the rtRTMP solver. Furthermore, we propose a parallel Ant Colony Optimization algorithm to speed up the TRSP search space exploration, and two local search algorithms to diversify the solution search.

3 Real-time Railway Traffic Management Problem

Rail traffic management typically relies on operational and technical constraints. The operational constraints regard requirements imposed by the infrastructure manager and involve train arrival, departure and *dwell times* at stations, i.e., the time allocated for passengers boarding/alighting, as specified in the timetable. In case of rolling stock re-utilization, additional time is allocated between the arrival and the departure of a train to allow turnaround, joint or split operations. The technical constraints regard the train *running times*, which are the times required by each train to traverse infrastructure sections, as well as the minimum *headway times* imposed by the signaling system in

each infrastructure section. The minimum headway time is the minimum time that must separate a pair of trains to avoid any overlap on the infrastructure section and any *deadlocks*, according to the blocking time theory (Hansen & Pachl, 2014). A deadlock occurs when a train cannot continue on its route, under any circumstances, because it is blocked by another (Pachl, 2007). Under the fixed block signaling system, electrical *track-circuits* represent the minimum section of infrastructure able to detect the presence of a train. A sequence of track-circuits between two consecutive signals is called *block-section*. In rail operations, each track-circuit is allocated exclusively to a specific train, and blocked for the other ones, for a given duration of time called *infrastructure utilization*, which is usually longer than the physical train occupation. The times at which a train begins and ends utilizing a track-circuit, called respectively *reservation* and *release* times, depend on the interlocking and signaling systems. In this paper, the *route lock-sectional release system* is considered as interlocking system, in which all the track-circuits of a block-section are reserved at the same time, while they are released one-by-one after track-free detection, according to the transit of each train. As signaling system, we consider the *three-aspect system*, whereby a green aspect signal indicates free way for the next block-section, a yellow aspect warns the train to start braking, because the next signal may be red, and a red aspect indicates the train must stop, since the next block-section is still to be released by the preceding train. According to this system, the two block-sections following a green aspect signal are reserved at the same time to ensure that trains can travel without unexpected brakings, since each train needs a full block-section to stop.

The operational constraints ensure that trains can comply with technical constraints in practice, at least in absence of traffic perturbations (Goverde et al., 2016). However, during operations, unexpected events may cause train delays (*primary delays*), which can lead to an infrastructure utilization overlap by two trains, if they both travel at their planned speed. The simultaneous utilization of the same infrastructure section(s) is prevented by the signaling system, which will stop one of the two concurrent trains. The second train will inevitably suffer a delay, called *secondary delay*. Solving the rtRTMP translates into providing an optimized plan of operations, that minimizes the propagation of train secondary delays. This classical problem in the field of railway operations research consists in defining the passing orders, the arrival and departure times of trains in stations and in selecting their route across the network, given a perturbed timetable. A *train route* is an ordered list of infrastructure sections that a train might traverse to reach the locations of its scheduled stops, i.e., its *stopping points*. We remark that a stopping point may include several alternative tracks, where the stop itself takes place. A set of *alternative routes* can be assigned to each train and be selected among the available ones in the rail network such that all routes share the same stopping points, while traversing different infrastructure sections.

Several models exist in literature for the rtRTMP. In this paper, we use the MILP formulation of Pellegrini et al. (2014) and the RECIFE-MILP solver of Pellegrini et al. (2015). While a

brief overview of this model is given in the reminder of this section, for the more comprehensive formulation we refer to the original paper (Pellegrini et al., 2014). RECIFE-MILP (microscopically) models the rail infrastructure on a track-circuit level. Given a timetable perturbed by train delays or infrastructure disruptions, the solver reschedules the trains in the timetable, while minimizing the sum of secondary delays for all trains at their exit from the considered infrastructure. This mathematical formulation uses non-negative continuous variables for modeling the start and end times of each track-circuit utilization and other timing events, e.g., the entrance times in the network and the arrival and departure times at stations or at other relevant points. For train ordering and routing decisions, the formulation employs binary variables. The train ordering (binary) variables establish the precedence relationship between trains on each shared track-circuit. The routing variables indicate whether a train uses a specific route among its alternative ones. Train scheduling variables are managed by means of *disjunctive constraints* (or *capacity constraints*), including a big-M coefficient. The constraints ensure that the infrastructure utilization times (for which two or more trains require the same track-circuit) do not overlap, unless these trains are subject to *rolling stock re-utilization constraints*. The latter constraints model turn-around, join, or split operations, which we refer to as rolling stock re-utilization operations. These constraints impose that a train service ending at one platform shall be followed by another train service that starts in the same location. Between the two linked arrival and departure times, the track-circuits (relevant to the platform track) are kept utilized, to perform the rolling-stock re-utilization operation. RECIFE-MILP solves the rtRTMP in two optimization steps. In (at most) the first 30 seconds of computation, the solver tackles the train scheduling problem, assuming that each train is using its timetable route. The best train scheduling solution is used to reduce the value of the big-M coefficient, and as starting solution in the second step, which jointly addresses the train scheduling and routing problems.

4 Train Routing Selection Problem

The TRSP is a combinatorial optimization problem in which, given the set of alternative routes available for each train travelling in the rail network, a subset of train routes is selected to improve the computation efficiency of the rtRTMP solving process.

Let us consider a set T of n trains requiring to traverse a railway infrastructure within a given time window. For each train $t \in T$, the set of all possible alternative route assignments is given. For each pair of trains, we define their routes *coherent* when no rolling stock re-utilization constraint exists between them; otherwise, if such constraint exists, they are coherent when the last infrastructure section of the first train route corresponds to the first infrastructure section of the second train route, i.e., the two routes satisfy the given rolling stock constraint from an

infrastructure point of view. A combination of train routes is *feasible* if one route has been selected for each train and each pair of train routes is coherent. The objective function of the TRSP formulation uses the concept of *potential delay* (Samà et al., 2016), which estimates the cost associated to the secondary delay caused by the train route choices. Given an infrastructure and a perturbed timetable, the TRSP question is which $p \geq 1$ feasible combinations of train routes better minimize the total potential train delay.

We model the TRSP according to the *construction graph* $G = (C, L)$ proposed by Samà et al. (2016).

Vertex $c_i \in C$ represents an alternative route belonging to train $t \in T$. We use the index i to refer to a specific vertex among all possible vertices in the graph G. Given n trains, the vertices are grouped into n partitions such that, for each train $t \in T$, we have an independent set $C_t \subset C$ of (all) alternative train routes. G is thus an n-partite graph such that $\cup_{t=1}^{n} C_t = C$ and $\cap_{t=1}^{n} C_t = \emptyset$. Two vertices $c_i, c_j \in C$ are connected by edge $l_{ij} \in L$ if they represent coherent train routes and belong to different trains. The n-vertex cliques in this graph identify the set Γ of feasible combinations of train routes. Subset $S_p \subset \Gamma$ is a set of p feasible combinations of train routes and thus a TRSP solution. The quality of each single clique is evaluated by the sum of all its vertices and edges costs. Vertex $c_i \in C$ and edge $l_{ij} \in L$ in the graph G have a non-negative cost: a potential delay u_i is associated to each c_i, expressing the longer running time of the corresponding train on that route compared to the timetable one; the potential delay w_{ij} encountered when two routes c_i, c_j are jointly considered is associated to edge l_{ij}. Solving the TRSP translates into selecting the set S_p of minimum cost cliques. From this set, a non-empty subset of at most p routes is obtained for each train, which is then used in the rtRTMP as the set of the only possible alternative train routes.

In Figure 1, the construction graph G for a 4-train example is presented. The graph is 4-partite: each train $t \in T$, with $t = 1, 2, 3, 4$, has a specific set of alternative routes, respectively C_1, C_2, C_3, C_4. Each vertex c_i in a partition, indicated by a black dot, represents an alternative train route. The index i indicates a specific vertex among all possible vertices in the partition. A 4-vertex clique is highlighted in bold. By definition, the clique is a complete (induced) subgraph of G, i.e., each pair of vertices c_i, c_j in the clique are connected by edge l_{ij}. This ensures all selected routes are coherent, and can be jointly selected. Each clique vertex belongs to a different partition, providing for each train a specific route assignment.

5 TRSP cost estimation model

As stated in Section 4, the quality of each clique in the construction graph G can be expressed in terms of the costs associated to each vertex $c_i \in C$ and to each edge $l_{ij} \in L$. In this section,

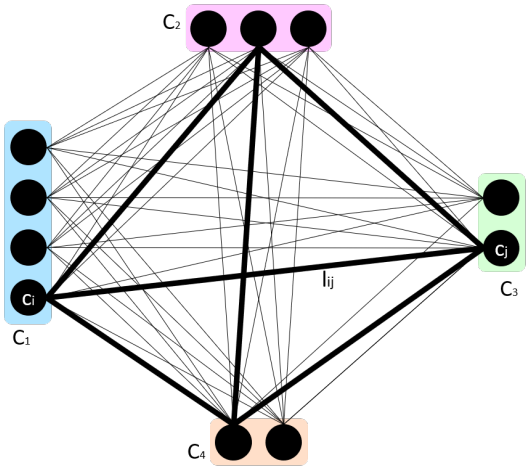

Figure 1: Example of construction graph $G = (C, L)$

we address the research question on how these costs can be calculated to enforce that the routes selected in each TRSP solution allow the minimization of the rtRTMP objective function. We introduce a TRSP cost estimation model, which proposes a good approximation of train delays by using a given set of routes in the rtRTMP solution, to build a strong correlation between TRSP and rtRTMP solutions.

Let us consider train $t \in T$ and the set $C_t \subset C$ of vertices associated to the alternative routes available for t. Let c_d be the default route used by the corresponding train in the timetable. For each $c_i \in C_t$, the positive cost u_i represents the potential delay due to the longer running time required to travel on that route compared to the timetable one. This cost is computed as $u_i = max(0, run_{c_i} - run_{c_d})$, where run_{c_i} indicates the minimum running time of c_i and run_{c_d} that of c_d. We remark that this equation assumes no buffer, i.e., allowance to exceed the default running time: as soon as route c_i has its running time longer than the default one, the use of the former generates a train delay. If a buffer exists, this equation can be trivially modified to take it into account.

The cost assigned to each edge l_{ij} is the potential train delay arising when two coherent route assignments c_i and c_j are jointly used. The seminal paper on the TRSP (Samà et al., 2016) formulates this cost by assessing the minimum infrastructure utilization overlap of the two trains according to their routes, and based on the train ordering decisions. This cost computation considers the train starting time and the undisturbed running time until the potential train conflict (to be solved). We next refer to this cost as *Base* cost.

In this paper, we propose the integration of the Base cost with two new building blocks:

- *Re-utilization* considers the rolling stock re-utilization constraints, not just for defining edges between coherent pairs of vertices, but also to compute potential delays;

- *Propagation* introduces the accurate estimation of the potential delays propagation among all trains in a clique.

The calculation of both costs is carried out by two steps. First, we propose a new method to assign a fixed cost w_{ij} to each edge of the construction graph G, by considering the interaction of a single pair of route assignments at a time. Depending on which additional building block is used (whether none, Re-utilization or Propagation), the computation of fixed costs changes. Appendix A shows a formal description of this method. Second, a clique-dependent cost w_{ij}^s is proposed by using Propagation building block. This cost is computed after clique $s \in \Gamma$ has been generated, as detailed in Procedure 1. This procedure assigns an additional cost to each edge of clique s, to better estimate the delay propagation caused by all routes in this clique.

Let us first focus on the fixed cost computation by Base cost. Here, if two route assignments concern trains that use the same rolling stock, no conflict arises between the concerned trains and this fixed cost is set to zero. Otherwise, we examine the set of *common resources* (res) between the considered route assignments. A common resource is the set of consecutive track-circuits shared by a pair of route assignments. If the set of common resources is not empty, the fixed cost is equal to the minimum utilization overlap (O) of the two trains computed by using the considered routes, according to the train ordering decisions. The utilization overlap is the maximum, among all common resources, of the time difference between the utilization end (eU) by the first passing train and utilization start (sU) by the second passing train. This represents the time that the second train waits before entering the considered common resource. By taking the maximum value of utilization overlap, we make a conservative choice, supposing that the trains will face the worst case situation. Then, we consider the best scheduling option, i.e., the train ordering for which the waiting time is minimum. In the Base cost, the start and end times of common resource utilization are computed by considering running and blocking times according to the undisturbed timetable.

When the new Re-utilization building block is considered, the fixed costs account for the *temporal coherence* of the trains that are using the same rolling stock. Let us consider the pair of trains $t, t' \in T$ with t' following t. The temporal coherence consists of two conditions: (i) respecting the minimum time, called *processTime*, between the arrival of t and the departure of t', necessary to perform the rolling stock re-utilization operations; (ii) ensuring that the common resource, where the re-utilization takes place, cannot be occupied by any other train between the arrival of t and the departure of t'.

In condition (i), the fixed cost is equal to the potential train delay that will be suffered by the second train, due to the late arrival time of the first train. Let $\overline{res}$ be the involved common resource, where the re-utilization operations occur. The potential train delay is equal to the (not negative) difference between the *processTime* and the time elapsing between the utilization start (sU) of $\overline{res}$ by the second train and the utilization end (eU) of the first train.

Condition (ii) is addressed for each pair of trains t and v that are not using the same rolling stock. For each common resource res, we check if either t or v is subject to a rolling stock re-utilization constraint with another train t' on a common resource $\overline{res}$, which overlaps res. If t [v] precedes t' in the rolling stock re-utilization constraint, the utilization end of res by t [v] is set equal to the utilization end of $\overline{res}$ by t': the common infrastructure resource will be usable by v [t] only after the departure of t', when t [v] passes first. We remark that when the re-utilization constraint involves more than two trains, we consider the utilization end of $\overline{res}$ by the last train involved in this constraint. On the contrary, if t [v] follows t' in the re-utilization process, the utilization start of res by t [v] is set equal to the utilization start of $\overline{res}$ by t'. In the latter case, when more than two trains are linked by re-utilization, the first utilization start is considered for $\overline{res}$.

Let us consider Figure 2 to give an illustrative example of the proposed Re-utilization building block. Here, we have four trains t_1, t_2, t_3, t_4, which enter the network according to their index

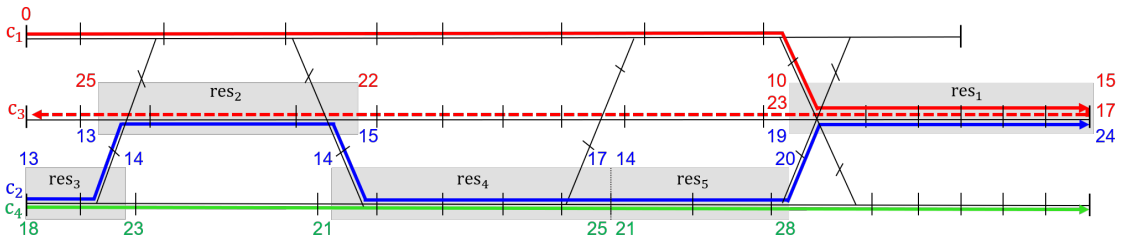

Figure 2: Example of four route assignments c_1, c_2, c_3, c_4.

order. For each train, we have a possible route assignment, respectively c_1 (red solid line), c_2 (blue line), c_3 (red dashed line) and c_4 (green line). The common resources between each pair of route assignments are highlighted in grey. In Figure 2, the start and end utilization times are shown for each train. For t_1 and t_3, we have a rolling stock re-utilization constraint, i.e., a turn-around operation; t_1 precedes t_3 and the re-utilization operation takes place on res_1. Due to this constraint, t_3 will start its utilization of res_1 after that t_1 ends its own utilization. Therefore, in the Base cost, $w_{1,3} = 0$, as no conflict occurs. By applying the Re-utilization building block, $w_{1,3} = max(0, processTime - (sU_{c_3,res_1} - eU_{c_1,res_1}))$: supposing t_1 ends its utilization of res_1 at 12, t_3 starts at 17, and $processTime$ is 5, $w_{1,3} = 0$ as for the Base cost. However, if t_1 is late and ends its utilization at 15, $w_{1,3} = 3$. This potential delay of t_3 is the effect of condition (i), to satisfy the required time for the rolling stock re-utilization operation. As for condition (ii), the impact is visible when calculating the costs of pairs (c_1, c_2) and (c_2, c_3), as they share res_1, i.e., the resource where the rolling stock re-utilization between t_1 and t_3 takes place. Trains (t_1, t_2) travel along (c_1, c_2) in the same direction. When t_1 precedes t_2 on res_1, Eq. (1) is used to calculate the

utilization overlap, according to the Base cost.

$$O^{res_1}_{c_1c_2} = eU_{c_1,res_1} - sU_{c_2,res_1} = 15 - 19 = -4 \quad (1)$$

When applying Re-utilization building block, Eq. (2) replaces Eq. (1): if t_1 utilize res_1 before t_2, then t_2 must wait for t_3 to release res_1.

$$O^{res_1}_{c_1c_2} = eU_{c_3,\overline{res}} - sU_{c_2,res_1} = 23 - 19 = 4 \quad (2)$$

The same procedure is used when considering the opposite train order, i.e., (t_2, t_1) on res_1. In this case, the Re-utilization building block has no impact on the utilization overlap $O^{res_1}_{c_2c_1} = 14$. The fixed cost $w_{1,2}$ is equal to 4, i.e., the minimum value obtained by evaluating the two train ordering possibilities.

After computing w_{ij}, we distinguish three cases:

1. c_i, c_j do not have common resources, thus $w_{ij} = 0$;
2. c_i, c_j have common resources and a positive potential delay, i.e., $w_{ij} > 0$;
3. c_i, c_j have common resources but no potential delay. Despite the absence of a potential delay, they may suffer a delay propagation, due to traffic perturbations. In this case, the fixed cost w_{ij} is set to 1, to disadvantage this choice over the absence of common resources (case 1), unless the Propagation building block is considered.

In case 3, no potential delay occurs when the trains related to c_i, c_j utilize a common resource without overlapping in time. In this case, a null or negative value of $w_{i,j}$ indicates that the related trains have no potential delays and the minimum distance separating their utilization of the common resources is equal to the absolute value of $w_{i,j}$ (buffer time). By applying the Propagation building block, w_{ij} keeps a negative or null value, to be used in the clique-dependent cost computation.

The Propagation building block aims to improve the ability of capturing the impact of knock-on delays in the cost computation. Besides the above difference in the fixed cost computation method, we propose a clique-dependent cost, as shown in Procedure 1, which estimates the extent to which train route assignments are vulnerable to delay propagation, for a given clique.

Procedure 1 takes as input: clique s, the list of common resources res between each pair of route assignments $c_i, c_j \in s$, plus the fixed cost w_{ij} for each edge $l_{ij} \in s$. Given the clique s, the clique-dependent cost w^s_{ij} of each edge $l_{ij} \in s$ is initialized with the corresponding fixed cost w_{ij}. If w^s_{ij} is greater than zero, we first identify the train that suffers the potential delay between i and j: this is the train that is chosen (in the fixed cost estimation method) as the one passing second on the corresponding common resource. Let us consider $i[j]$ as the delayed train, we propagate its potential delay w^s_{ij} to the edges connecting its route $c_{i[j]}$ with any other route $c_h \in s$ such that c_h

Procedure 1: Clique-dependent cost computation

Data: Clique s; every common resource *res* for each pair $c_i, c_j \in s$; w_{ij} for each $l_{ij} \in s$

Result: Clique-dependent cost w^s_{ij} for each l_{ij} in clique s

1 **forall** $l_{ij} \in s$ **do**
2 $\quad w^s_{ij} = w_{ij}$
3 **end**
4 **forall** $l_{ij} \in s$ *in increasing order of train entrance time* **do**
5 $\quad$ **if** $w^s_{ij} > 0$ *and the delay is assigned to the train running on* $c_i[c_j]$ **then**
6 $\quad\quad$ **forall** $l_{i[j]h}$*:* $\exists$*res between* $c_{i[j]}, c_h$ **do**
7 $\quad\quad\quad$ **if** $w^s_{i[j]h} < 0$ **then**
8 $\quad\quad\quad\quad w^s_{i[j]h} = w^s_{ij} + w^s_{i[j]h}$
9 $\quad\quad\quad$ **else**
10 $\quad\quad\quad\quad w^s_{i[j]h} = w^s_{i[j]h} + max(0, w^s_{ij} - w^s_{i[j]h})$
11 $\quad\quad\quad$ **end**
12 $\quad\quad$ **end**
13 $\quad$ **end**
14 **end**
15 **forall** $l_{ij} \in s$*:* $w^s_{ij} \leq 0$ *and* $\exists$*res between* c_i, c_j **do**
16 $\quad w^s_{ij} = 1$
17 **end**

is different from c_i and c_j and has common resources with $c_{i[j]}$. This is done by Procedure 1 at lines 6-13. When $w^s_{i[j]h}$ is negative (at line 8 in Procedure 1), the propagated delay w^s_{ij} reduces the buffer time between the two involved trains on the common resource, (eventually) causing a positive potential train delay. Otherwise, when $w^s_{i[j]h}$ is not negative (at line 10 in Procedure 1), the propagated delay w^s_{ij} increases w^s_{ih} by the difference $w^s_{ij} - w^s_{ih}$, if positive.

In Procedure 1, edges are considered by following the order in which trains enter the infrastructure. We propagate the clique-dependent cost, instead of the fixed one, to allow *knock-forward* propagation: a clique-dependent cost w^s_{ij}, which is positive due the delay propagation (at lines 6-12 in Procedure 1), is further propagated to the other edges. Once the delay propagation is completed, as in the above case 3, we still set to one the cost w^s_{ij} of those edges linking two route assignments with common resources but no potential delay.

Re-utilization and Propagation building blocks are incrementally applied to Base cost. In the reminder of the paper, Re-utilization cost is the extension of Base cost, and Propagation cost is the extension of Re-utilization cost.

Now let us show how Propagation building block is applied to the trains in the example of

Figure 2. We assume that the route assignments c_1, c_2, c_3, c_4 form a clique s. For each edge $l_{ij} \in s$, the clique-dependent cost w^s_{ij} is initialized with the corresponding fixed cost w_{ij}: $w^s_{1,2} = 4$, $w^s_{1,3} = 3$, $w^s_{1,4} = w^s_{3,4} = 0$, $w^s_{2,3} = 12$, and $w^s_{2,4} = -1$. When considering the train entrance order in the network, the first potential train delay (which is propagated) is $w^s_{1,2} = 4$, that is suffered by train t_2. Eqs. (3)-(4) propagate this potential delay to the edges connecting c_2 with c_3 and c_4, as they have common resources.

$$w^s_{2,3} = w^s_{2,3} + max(0, w^s_{1,2} - w^s_{2,3}) = 12 + max(0, 4 - 12) = 12 \tag{3}$$

$$w^s_{2,4} = w^s_{2,4} + w^s_{1,2} = 4 - 1 = 3 \tag{4}$$

Following, the potential delay $w^s_{1,3}$ (suffered by t_3) is propagated to $l_{2,3}$, without modifying its clique-dependent cost, since the delay $w^s_{2,3}$ is larger than the one propagated. $w^s_{2,3} = 12$ (suffered by t_2) is propagated via Eqs. (5)-(6) to the edges connecting c_2 to c_1 and c_4. A knock-forward delay propagation occurs in Eq. (6): the positive potential delay $w^s_{2,4}$, caused by the previous propagation due to $w^s_{1,2}$, further increases by the propagation of $w^s_{2,3}$. Since c_4 has no common resource with c_1 and c_3, the potential delay $w^s_{2,4}$ is not propagated.

$$w^s_{1,2} = w^s_{1,2} + max(0, w^s_{2,3} - w^s_{1,2}) = 4 + max(0, 12 - 4) = 12 \tag{5}$$

$$w^s_{2,4} = w^s_{2,4} + max(0, w^s_{2,3} - w^s_{2,4}) = 3 + max(0, 12 - 3) = 12 \tag{6}$$

6 Parallel Ant Colony Optimization for TRSP

Ant colony optimization (ACO) is an algorithm inspired by nature, which exploits ant foraging behavior to solve hard combinatorial optimization problems (Dorigo et al., 1996). Indirect communication between the ants enables them to find the shortest paths between their nest and food sources, without having to try every possible path. The algorithm adopts the concept of *pheromone trails* to keep track of the cumulative knowledge on the solution quality. Together with the *heuristic information*, a greedy measure of the quality of problem components, the pheromone iteratively guides the solution space exploration to progressively improve the incumbent solution.

Samà et al. (2016) applies (for the first time) ACO to the TRSP, inspired by the ACO algorithm developed for the maximum clique problem by Solnon & Bridge (2006). Their ACO-TRSP metaheuristic implements a sequential execution of three main procedures, which we refer to as *ant walk*, *daemon actions*, and *pheromone update*. They respectively correspond to: ant generation and initial solution construction; local search to favor the ant search process; modification of the pheromone trail deposited by ants (from the beginning of the search process) on each edge of the construction graph G, and its evaporation throughout the algorithm execution. Pascariu et al. (2021) show that the ACO algorithm in Samà et al. (2016) is able to find the global optimum.

However, for the largest instances, their algorithm does not converge to the global optimum within the short computation time availability, due to the real-time nature of the TRSP (i.e., typically around 30 seconds). To improve the algorithm performance on the largest instances, we propose a parallel version of ACO-TRSP. Parallel computing uses various processing units to execute multiple tasks at the same time. In the following, we will refer to the algorithm introduced by Samà et al. (2016) as the sequential ACO (sACO) and to the parallel version of the algorithm proposed in this paper as the parallel ACO (pACO).

The parallelization technique relies on the multi-threading OpenMP *fork-join* model (Chapman et al., 2008), according to which the investigated program starts as a single thread, called *master thread*, while at a designated point of its execution this branches into a number of threads, creating a *parallel region*, and joins later to resume the sequential execution on the master thread. Within the parallel region, each thread has both its *private memory* and a *shared memory*, whose access is available to all threads. During the sequential execution, the master thread uses the shared memory, while in the parallel region the master gets (as the other threads) its own (private) memory.

Figure 3 illustrates the flowchart with detailed operations of pACO. The algorithm takes as input: (i) the construction graph $G = (C, L)$; (ii) the maximum computation time $timeLimit$; (iii) the number (p) of cliques that compose the solution; (iv) the number $nThreads$ to be used; (v) additional user-defined ACO-specific parameters, such as pheromone exponential weight α, heuristic exponential weight β, pheromone evaporation rate ρ, number $nAnts$ of ants in the colony, maximum and minimum pheromone bounds τ_{max} and τ_{min}. Based on these input data, pACO operates iteratively and returns the best solution found S_p after the time limit elapse or when a clique with null cost is identified. Each iteration involves the consecutive execution of the above procedures, highlighted in Figure 3 by different colours: *ant walk* in yellow, *daemon actions* in green, and *pheromone update* in peach. The reminder of this section provides a detailed explanation of each procedure.

In *ant walk*, the *nbAnts* ants of the colony operate on multiple threads in parallel. Each ant a is assigned to a specific thread and builds clique $s_a \in \Gamma$ in the construction graph G, by means of its private memory. For each ant, the first vertex $c_i \in C$ is randomly selected and inserted in s_a. The rest of its clique is built incrementally by adding one vertex at a time among the available candidates $c_h \in Candidates$, according to the so-called *random proportional rule* $\frac{\tau(c_i,s_a)^\alpha \eta(c_i,s_a)^\beta}{\sum_{c_h \in Candidates} \tau(c_h,s_a)^\alpha \eta(c_h,s_a)^\beta}$. In this formula, $\tau(c_i, s_a)$ indicates the sum of the pheromone trails associated to each l_{ij} such that $c_j \in s_a$, with α measuring the influence of pheromone trails in the random proportional rule. Furthermore, $\eta(c_i, s_a)$ indicates the sum of the heuristic information associated with each l_{ij} such that $c_j \in s_a$, where the heuristic information is computed as $\eta(l_{ij}) = 1/(1 + w_{ij} + u_i)$, while β measures the heuristic information influence in the random proportional rule. The set *Candidates*

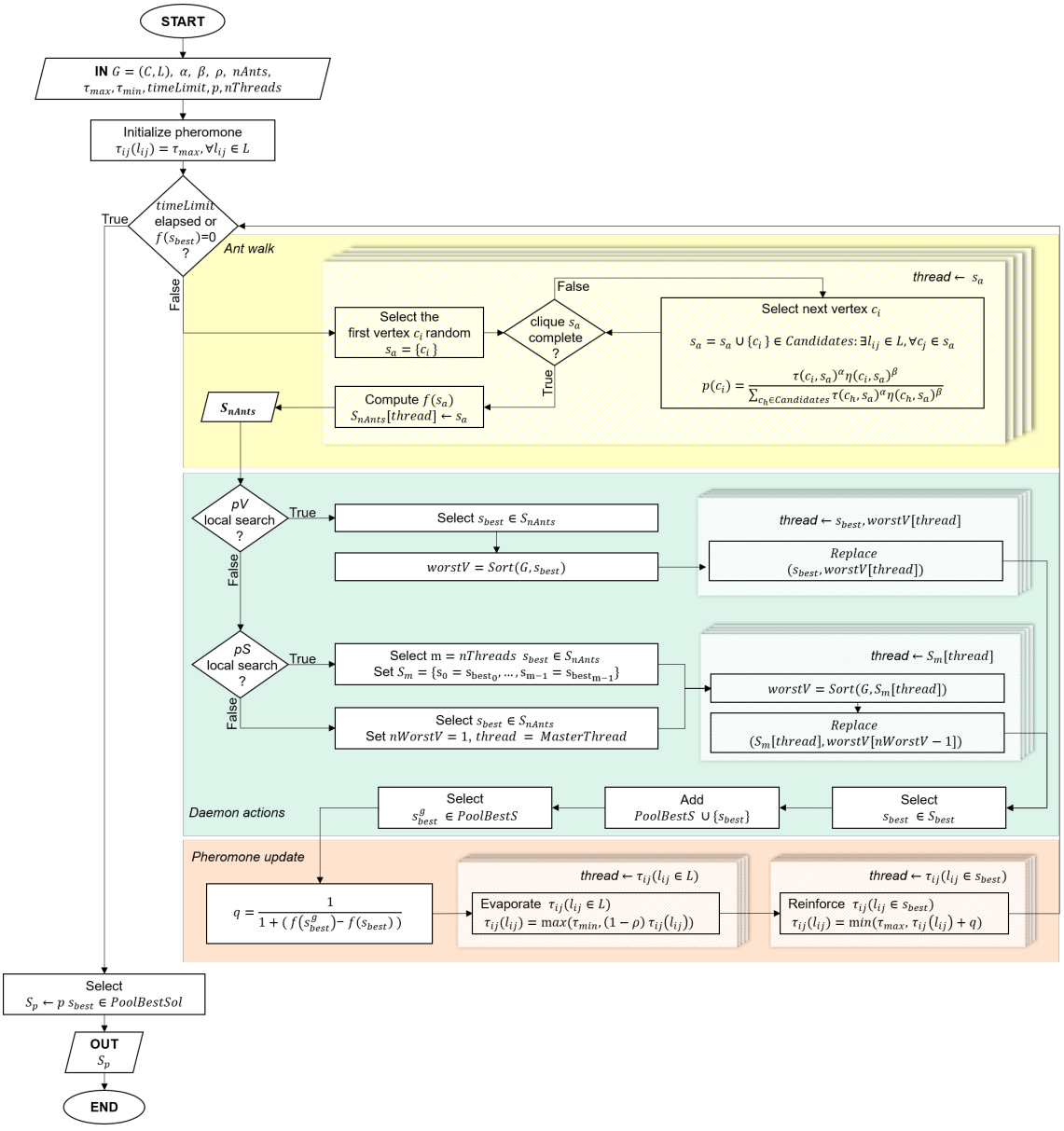

Figure 3: pACO flowchart.

contains the not-yet-chosen vertices, that are connected to the vertices already included in the partial clique s_a, i.e., the ones that can lead to a clique of the required cardinality. Once a clique s_a has been built, its cost $f(s_a)$ is computed as the sum of u_i and w_{ij} of each $c_i, l_{ij} \in s_a$. The clique s_a is then stored in the set S_{nAnts}, along with the cliques found by the other ants.

The introduced multi-thread processing system has the advantage of being very fast, but it brings some issues: the concurrency of various parallel processes leads to a race condition, i.e., an unintentional data sharing which causes a wrong algorithmic behaviour. In our algorithm, since the solution set S_{nAnts} is placed in the shared memory, each thread can access it and save (in *write* mode) its solution (for later use by the master thread). By default, two or more threads can access and save their solution in the same memory area, eventually causing an overwriting. We manage the risk of overwriting the results returned by parallel threads by allocating (to each of them) a specific area of the memory dedicated to S_{nAnts}. Additional information shared by the ants (regarding, e.g., the construction graph and the pheromone matrix) is placed in the shared memory. Threads *read-only* this global information without modifying it, hence no race condition might occur.

The S_{nAnts} cliques found by the ant colony are processed in the *daemon actions*, where a local search method is applied on the best clique(s) to search for better neighbouring clique(s). Two alternative local search, which employ parallel computing, are proposed: pV stands for parallel vertices, while pS for parallel cliques. These local search methods rely on two approaches that we call *Sort* and *Replace*. Given the construction graph G and a clique s, *Sort* builds the following ordered list $worstV$ of vertices $c_i \in s$: the worst vertex c_w, i.e., the one with the highest cost $f(c_w)$, is the first in the list.

The cost computation of each vertex $c_i \in s$ depends on the adopted cost estimation model. When *Propagation* cost applies, c_i impacts on the cost of the clique s, thus its cost $f(c_i)$ is computed as the difference between the cost of cliques s and $s \setminus \{c_i\}$. Otherwise, $f(c_i)$ is computed as in Samà et al. (2016), i.e., as the sum of u_i and the cost of all edges incident to c_i. Regarding *Replace*, given a vertex $c_w \in worstV$ together with graph G and clique s, this method replaces vertex c_w with a better one c_b, belonging to the same partition of G, i.e., $c_b, c_w \in C_t$ with C_t the partition related to train t, such that a lower cost clique is obtained.

When the pV local search is considered, the best clique $s_{best} \in S_{nAnts}$ (found during the current iteration) is selected. *Sort* is then applied to s_{best} and its m ($min(nThreads, nTrains)$) neighbors are explored (*Replace*) by replacing a vertex in $worstV$, one for each thread. To avoid any race condition, S_m is a collection of m copies of s_{best}, such that when each vertex is replaced with a better one, the clique stored in S_m is changed accordingly, while s_{best} remains unchanged.

When the pS local search is considered, the m ($nThreads$) best cliques (found in the current iteration) are selected to form set S_m. Then, each clique in S_m is assigned to a different thread

and the replacement of its worst vertex is performed by *Replace*.

If none of these local search methods is used, the local search introduced by Samà et al. (2016) is applied. Here, the best clique $s_{best} \in S_{nAnts}$ and its worst vertex are only considered in the master thread.

At the end of the local search, the best clique $s_{best} \in S_m$ is chosen and stored in the ordered list $PoolBestS$. Its position in this list is based on the corresponding clique cost.

The best clique $s_{best} \in S_m$ is used in the *pheromone update* to modify the pheromone trails, together with the current best clique s^g_{best}. This procedure consists of an evaporation and a reinforcement action. In the former, a portion $(1 - \rho)$ of pheromones evaporates from each edge $l_{ij} \in G$, with $0 < \rho < 1$ measuring the pheromone evaporation rate. Then in the reinforcement, an additional pheromone $q = 1/(1 + f(s^g_{best}) - f(s_{best}))$ is deposited on the edges belonging to s_{best}. Upper (τ_{max}) and lower (τ_{min}) bounds are imposed on the pheromone trail, to avoid the search stagnation on a few areas of the search space (Stützle & Hoos, 2000). Both evaporation and reinforcement actions are performed in parallel: a specific memory area of the pheromone matrix, which is placed in the shared memory, is allocated to each thread.

7 Computational experiments

This section presents our computational analysis of the proposed TRSP methodology, which is carried out on the case study of Lille Flandres station area, France. The infrastructure layout, shown in Figure 4, is 12-km long and composed of 299 track-circuits, 734 block sections, and 2409 train routes. Lille Flandres station is a complex terminal station, linked to national and international lines, with 17 platforms shared by local, intercity, and high speed trains.

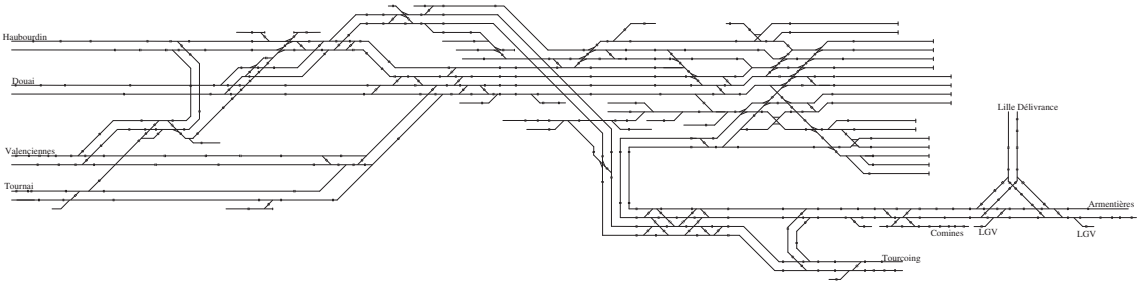

Figure 4: Lille Flandres station area

Starting from the one-day timetable, which has an average of 39 trains per hour, the test instances are obtained as follows. We perturb this initial timetable by applying a train delay between 5 and 15 minutes to the entrance time of 20% of trains. The trains to be delayed and the corresponding delays are randomly selected. We generate 16 perturbed timetables. For each

of them, we propose 10 instances corresponding to a 60-minute time window of train traffic flow. We select this instances by starting the time windows from ten different time instants, randomly drawn during the peak-hour time periods 7:30-9:00 and 18:30-20:00. We thus consider a total of 160 instances. To avoid overfitting, we use 100 of these instances to test the model and algorithm performance, 30 to assess the speed-up technique based on parallel computing, and 30 for the algorithm tuning.

The computational analysis is conducted in four phases. Section 7.1 evaluates the impact of the proposed cost estimation models on the correlation between the TRSP and the rtRTMP. The latter problem is modelled as in Pellegrini et al. (2014). Section 7.2 studies the performance of the parallel ACO-TRSP algorithm when using a different numbers of threads. In Section 7.3, the parallel ACO-TRSP algorithm is tuned to find the best local search methods and ACO-specific parameters. In Section 7.4, the proposed model improvements are compared with the state-of-the-art algorithm, while the TRSP solutions are given as input to the rtRTMP solver. Here, we use the ACO-TRSP approach as a pre-processing phase to the rtRTMP, that is then solved via the RECIFE-MILP of Pellegrini et al. (2015). For the rtRTMP, we consider the minimization of the total secondary delay as the objective function, and we take into account disturbances and disruptions. For the disturbances, we consider the train delays introduced during the generation of instances. The disruptions are obtained by considering the same instances (subject to disturbances) plus the failure of a track-circuit, which causes the infeasibility of around 60% of the available train routes. In our study, since the TRSP is designed to be solved right before the train rescheduling problem, the computation time need to be compatible with real-time train operations. We thus set (*a priori*) a computation time limit of 30 seconds, as in Samà et al. (2016).

All the experiments are performed on an Intel Xeon 24 core 2.2 GHz processor with 1.5 TB RAM, under Linux Ubuntu distribution. The ACO-TRSP is implemented in C++, while CPLEX 12.6 is used by RECIFE-MILP.

7.1 Comparison among cost estimation models

This section studies the correlation between the TRSP and the rtRTMP when Re-utilization and Propagation costs are used. We run the experiments on the 100 instances introduced in Section 7. After generating a random rtRTMP solution for each instance, we compute the clique cost corresponding to the route assignments in the solution.

Figure 5 shows the clique costs correlation to the rtRTMP objective function value. Three scatter plots are presented, each one reporting the data sets obtained with: Base, Re-utilization, or Propagation cost method. The rtRTMP solution values are plotted in blue dots in ascending order, with the corresponding clique cost in orange dots. We assume that the TRSP is coherent with the rtRTMP when the two sets of dots overlap or are very close to each other. A gradual improvement

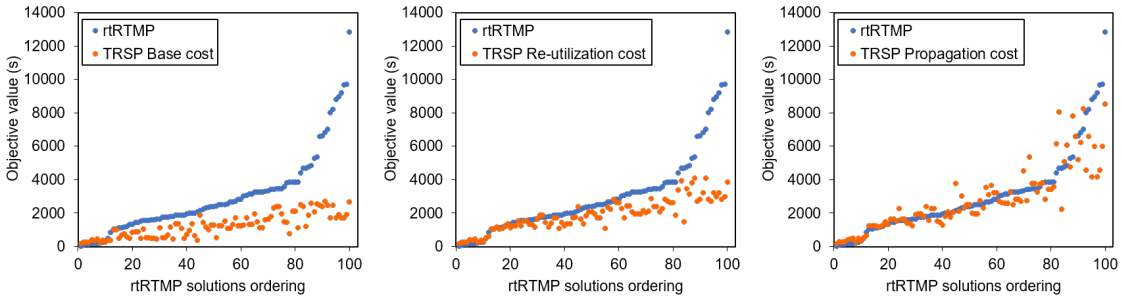

Figure 5: Correlation between TRSP (orange) and rtRTMP (blue) objective function values, when using Base, Re-utilization and Propagation costs

in the clique costs correlation with the rtRTMP objective value is noticeable when moving from the left graph to the right graph, as the two sets of dots are getting closer. Base cost follows the increasing trend of the rtRTMP solution. However, most rtRTMP solution values are underestimated by this cost, which is only able to precisely assess the cliques for which the corresponding rtRTMP objective function is close to zero. With the Re-utilization cost, the rtRTMP solution values continue to be underestimated, but we can observe an upward shift of the TRSP dots, thanks to the inclusion of the potential train delay due to rolling stock re-utilization. As for the Base cost, for smaller objective function values, Re-utilization cost is coherent with the rtRTMP. However, this holds for a larger number of solutions: Base cost more or less correctly assesses around 15 solutions over the 100 tested, while Re-utilization accomplishes this for around 40 solutions. For the remaining solutions, Re-utilization cost cannot properly capture the rtRTMP objective value. These are the solutions with the highest rtRTMP objective function value, which are most likely characterized by train delay propagation. The Propagation cost is targeted for managing these cases, and it records objective values closer to those of the rtRTMP, though some error persists. This is mostly due to the calculation method for the train delay propagation, which does not actually solve the train rescheduling problem, but it can be considered as a good approximation method.

Table 1 shows the numerical results on the correlation analysis between the clique cost value and the rtRTMP objective function value. We perform a linear regression and extrapolate the Pearson's correlation coefficient R with a Confidence Interval (CI) of 95%. In Table 1, the first column indicates the method used to compute the clique cost value. The following two columns specify the correlation coefficient R, and the bounds of the confidence interval. By definition, the correlation coefficient ranges between -1 and 1. Given the rtRTMP solution value and the clique cost, a positive coefficient indicates that: when the value of the rtRTMP solution increases, the value of the clique cost is also increasing. Vice versa, for negative values of R, when the

rtRTMP solution value increases, the clique cost decreases. The correlation gets stronger when the coefficient value is close to the extreme range values.

Table 1: Pearson's correlation coefficient R with 95% CI

TRSP cost	R	CI
Base	0.75	{0.64; 0.82}
Re-utilization	0.80	{0.72; 0.86}
Propagation	0.84	{0.77; 0.89}

The results in Table 1 show a significant positive linear correlation between the TRSP and the rtRTMP solutions, since all R values are positive. The lowest correlation level of Base cost is gradually improved by Re-utilization and Propagation costs, as shown in Table 1 by the increase of R coefficient and the reduction of confidence interval values. Among the three costs, Propagation achieves the best R value (0.84), which confirms its ability to better model the correlation between the TRSP and rtRTMP solutions compared to the other methods.

7.2 ACO-TRSP algorithm: serial versus parallel computing

In this subsection, the impact of the ACO-TRSP parallelization is carefully assessed. For this type of experiments, we consider 30 instances from the set described in the introduction of Section 7. For each instance, we solve the TRSP with pACO by using a growing number of threads, between 1 and 24. The use of one thread corresponds to the application of sACO. We run the ACO algorithm for 30 seconds and record the best objective function value at each iteration. As for the ACO parameters, we use the sequential local search introduced in Section 6, the Base cost and the following parameters setting for all configurations of threads: $nAnts = 200$, $\alpha = 2$, $\beta = 2$, $\rho = 0.05$, $\tau_{min} = 0.01$, and $\tau_{max} = 6$. In a preliminary computational analysis, we have assessed various settings of these parameters. We note that different settings do not notably impact the conclusions that we can draw on the ACO performance achievable when using a different number of threads.

Figure 6 shows the evolution of the average pACO objective function values. Here, each line corresponds to a different thread configuration. In all curves in Figure 6, we can distinguish two phases of the algorithm behavior. In the first phase, each curve follows a descending trend, indicating that diversification actions lead to a rapid improvement of the best-known solution. In the second phase, called convergence phase, each curve settles, while a rather slow improvement is observed. With a small number of threads, the algorithm stagnates in local minima. As the number of threads increases, significant improvements in the objective values are obtained. Starting from 16 threads, the curves in Figure 6 maintain their initial descending trend for a longer computation time, indicating that the algorithm parallelization improves the diversification actions. Using 24

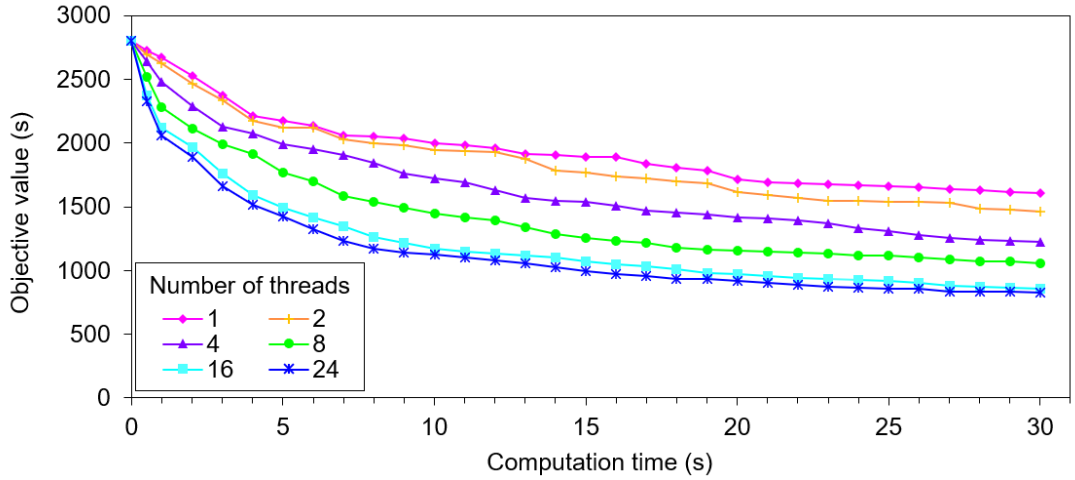

Figure 6: Evolution of pACO performance when using a different number of threads

threads appears to be the best choice, since this seems gives the best results at the end of the computation time. However, the difference between the curves is quite limited when using 16 or 24 threads.

7.3 Algorithm parameters tuning

As mentioned in Section 6, ACO algorithms can have several user-defined specific parameters, such as: pheromone exponential weight α, heuristic exponential weight β, pheromone evaporation rate ρ, number $nAnts$ of ants in the colony, maximum and minimum pheromone bounds τ_{max} and τ_{min}. We use IRACE (Iterated Racing for Automatic Algorithm Configuration), proposed by López-Ibáñez et al. (2016), to fine tune them. IRACE is an open source package, which implements an iterated racing procedure to select the best parameters configuration from a set of available value settings. Furthermore, we use IRACE to select which of the proposed local search methods offers the best performance for the investigated instances. We refer the interested reader to López-Ibáñez et al. (2016) for a more comprehensive description of the IRACE package, its user guide and applications.

We use 30 instances from the set described in the introduction of Section 7, different from those used in the other computational experiments. We consider the pACO algorithm setting that runs on 24 threads and the use of Propagation cost method. Table 2 shows the considered parameter configurations. The best configuration resulting from the tuning on the 30 instances is highlighted in bold in Table 2.

Table 2: pACO parameters tuning

Parameters	Values
$nAnts$	100, 150, 200, 250, 300, 350,**400**
α	1, 2, 3, **4**
β	3, 4, **5**, 6, 7, 8, 9, 10
ρ	0.03, 0.07, 0.09, 0.1, 0.3, 0.5,**0.6**, 0.7
τ_{max}	3, **4**, 6, 7, 8, 9, 10
τ_{min}	**0.001**, 0.003, 0.005, 0.01, 0.03, 0.05
Local search	pV, $\boldsymbol{pS}$, *serial*

7.4 Comparison among algorithms for the rtRTMP

In this subsection, we assess the potential of the TRSP improvements (proposed in Sections 5 and 6) on the rtRTMP solutions. This computational analysis is performed on 200 instances, 100 under timetable disturbances and 100 under infrastructure disruptions. The former are the instances used in Section 7.1. The latter are obtained by considering the former instances plus the unavailability of a track-circuit.

Considering that RECIFE-MILP (Pellegrini et al., 2015) is used as the rtRTMP solver and our paper is a follow-up work of Samà et al. (2016), we assume as state-of-the-art:

- *RECIFE 180s*, the rtRTMP solved by the RECIFE-MILP solver (Pellegrini et al., 2015) within 180 seconds of computation. For each train, all possible routes are considered, while the timetable routes are used as the default ones to compute the initial solution (warm-start) in the RECIFE-MILP solver;

- *sACO Base*, the TRSP-rtRTMP algorithm proposed in Samà et al. (2016), which uses the Base cost and the sequential ACO algorithm to solve the TRSP within 30 seconds. The p best train route assignments (selected from the TRSP solution) are provided as input to RECIFE-MILP, that needs to solve the rtRTMP within the remaining 150 seconds. The initial solution in RECIFE-MILP solver is obtained via the timetable (default) routes. When the routes are unfeasible, due to the disruption, the best found clique is considered as the initial one.

We solve the rtRTMP by using the p best train route assignments provided by the TRSP cost methods proposed in Section 5 and solved by using the parallel ACO algorithm. We next refer to each cost algorithm combination as follows: *pACO Base*, *pACO Re-utilization*, and *pACO Propagation*. Given the best correlation between the TRSP and rtRTMP solutions obtained with Re-utilization and Propagation costs, RECIFE-MILP computes its warm-start with the train routes in the best clique found by the pACO Re-utilization and pACO Propagation. We fix the maximum compu-

tation time to 30 seconds and 150 seconds to search for the best TRSP and rtRTMP solutions, as in Samà et al. (2016).

We compare the performance of our cost-algorithm combinations with the state-of-the-art approaches in Tables 3 and 4. Furthermore, we assess the performance compared to the optimal solutions in Figure 7. The optimal solutions (and the best-known bounds) for the tested instances are obtained by solving the rtRTMP via RECIFE-MILP within 24-hour computation time limit (RECIFE 24h). RECIFE 24h uses the same solving method as RECIFE 180s.

Tables 3 and 4 report the average objective values of the rtRTMP solutions (obtained by the various methods), respectively, in case of disturbances and disruptions. We test several route assignment values provided by the TRSP, varying p from 1 to 30, to study the rtRTMP solutions while increasing the search space. The first column of Tables 3 and 4 indicates the value of parameter p. The remaining columns specify the employed solution approach. RECIFE 180s takes as input either *all* the routes (when p=ALL) or the timetable routes (when $p = 1$). Columns 3-6 report on the TRSP-rtRTMP solutions. Specifically, RECIFE solves the rtRTMP by using the p best route assignments selected by the ACO-TRSP solver. In Tables 3 and 4, the best result for each approach is highlighted in bold.

Table 3: Average rtRTMP objective value (in seconds), in case of disturbances

p	RECIFE 180s	sACO Base	pACO Base	pACO Re-utilization	pACO Propagation
all	**2015**	-	-	-	-
1	2169	2169	2169	1112	1033
2	-	1336	1066	933	877
3	-	1253	1054	900	863
5	-	**1231**	1031	878	836
10	-	1377	**988**	**843**	**833**
20	-	1577	1260	902	897
30	-	1662	1275	939	889

From the results in Table 3, we observe that optimizing train rerouting always improves the average objective value. The worst performance is achieved when the timetable routes are only available. Limiting the train routing flexibility, as for sACO Base, gives better results than considering the full routing flexibility, as for RECIFE 180s ALL. A carefully selected number of train routes allows for a more fruitful search in the solution space. The parallelization, introduced in pACO Base, steadily improves the ACO performance. As expected, a wide exploration of the TRSP solution space corresponds to a remarkably better quality of the rtRTMP solutions. It is also evident that as the correlation between TRSP and rtRTMP solutions is boosted, the qual-

ity of the rtRTMP solution increases. For $p = 1$, pACO Re-utilization and pACO Propagation provide better train route configurations than the timetable ones. The results obtained by pACO Re-utilization, for any route assignment and p, outperform the best ones obtained by pACO Base (with $p = 5$). We conclude that the consideration of rolling stock re-utilization information (by pACO Re-utilization) is helpful during the train routes selection to then recommend routes to the rtRTMP that lead to better results. The highest correlation between the TRSP and rtRTMP solutions, obtained by pACO Propagation, leads to better routes than pACO Re-utilization. With pACO Propagation, the solution quality improve for any p-value, with the best overall results obtained when considering 10 routes per train. When the number of available routes increases ($p = 30$), the search space becomes too large for the rtRTMP solver (RECIFE-MILP) to find solutions as good as those obtained for smaller p.

Table 4: Average rtRTMP objective value (in seconds), in case of disruptions

p	RECIFE 180s	sACO Base	pACO Base	pACO Re-utilization	pACO Propagation
all	**2839**	-	-	-	-
1	infeasible	5458	4507	1117	1139
2	-	2809	2437	1035	1009
3	-	**2644**	2198	1021	973
5	-	2895	2027	1035	960
10	-	3543	1991	970	**923**
20	-	3641	**1617**	**943**	956
30	-	4136	2258	992	991

For the disruption case, Table 4 shows that, when considering the timetable routes only, RECIFE 180s with $p = 1$ leads to infeasible schedules for all the 100 instances: in each of these instances, there is at least one infeasible route. Train rerouting is thus necessary, while the ability to provide good (initial) train route assignments becomes fundamental to avoid large train delays. In this context, sACO Base finds better quality solutions than RECIFE 180s with all available routes, as shown in Table 4. This result confirms that the consideration of a limited number of train route assignments is better than the full train routing flexibility. However, with $p > 3$, sACO Base finds worse solutions than those obtained with a smaller p, due to its limited ability to explore the solution space and find good train route assignments. This trend does not occur for pACO Base, which explores the solution space extensively, and its solution quality improves up to $p = 20$.

Besides the parallelization technique, the new Re-utilization and Propagation costs allow to find significantly better train route assignments compared to pACO Base. pACO Re-utilization halves the average objective value of pACO Base, while pACO Propagation often brings additional

improvements over pACO Re-utilization. pACO Propagation returns an average objective value slightly larger than pACO Re-utilization for $p = 1$. The clique-dependent cost assessment may cause a worse correlation of the TRSP solution with the rtRTMP one: when RECIFE-MILP uses all train route assignments in the best clique, the Propagation cost is less effective. This is due to the discrepancy between the train ordering decisions in few (4) TRSP solutions and their corresponding rtRTMP solutions. Train ordering discrepancy occurs when the train sequencing decisions, taken during the estimation of edge costs in the TRSP, do not correspond to the ones in the rtRTMP solution. It is possible to encounter this discrepancy in our TRSP method because the feasibility of all train ordering decisions can only be certified during the rtRTMP solution process.

Tables 3 and 4 show that pACO Propagation identifies the region of the search space with the best rtRTMP solutions: this approach always leads to a better rtRTMP average solution than pACO Re-utilization for $p < 20$. However, as the number of available train routes increases, part of this advantage is lost, especially when the search space becomes large for RECIFE-MILP and it is difficult to find the optimal solution within the given time limit. Optimal solutions are relative to the search space defined by the available train routes. For example, when $p = 20$ in Table 4, the average objective value obtained by pACO Re-utilization is better than the one of pACO Propagation. With pACO Re-utilization, RECIFE-MILP proves the optimality for 86 over 100 solutions, but this number is only 51 in case of pACO Propagation. In the case of 26 optimal pACO Re-utilization solutions for which pACO Propagation does not compute the optimal ones, the latter find a worst-value solution. While, when both pACO Propagation and pACO Re-utilization compute an optimal solution, the value of the pACO Propagation solution is always better. The value of the optimal solution can change because the available train routes, which generate the search space, are different. This is confirmed by the results obtained with $p = 30$: 37 and 35 optimal solutions are found, respectively, for pACO Re-utilization and pACO Propagation. The pACO Propagation solutions have better values, but since the difference between the values with those of pACO Re-utilization are small, the average values are similar.

The bar plots in Figure 7 show the performance of the studied approaches compared to the global optimal solutions obtained by RECIFE 24h. This analysis is performed in case of (a) disturbances and (b) disruptions. Each bar represents the average optimality gap: *(Best Obj. Value - Lower Bound)/Best Obj.Value*, in which *Best Obj. Value* indicates the corresponding best solution (value in bold in Tables 3 and 4), while *Lower Bound* is the best-known lower bound value obtained by RECIFE 24h.

From Figure 7, we can observe that RECIFE 180s presents the largest optimality gap, 70.4% (77.5%) in case of disturbances (disruptions). The larger gap of the disrupted instances indicates that these are more challenging to be solved by the rtRTMP solver. By limiting the number of routing variables via sACO Base, the rtRTMP solution values improve, thus reducing the optimality

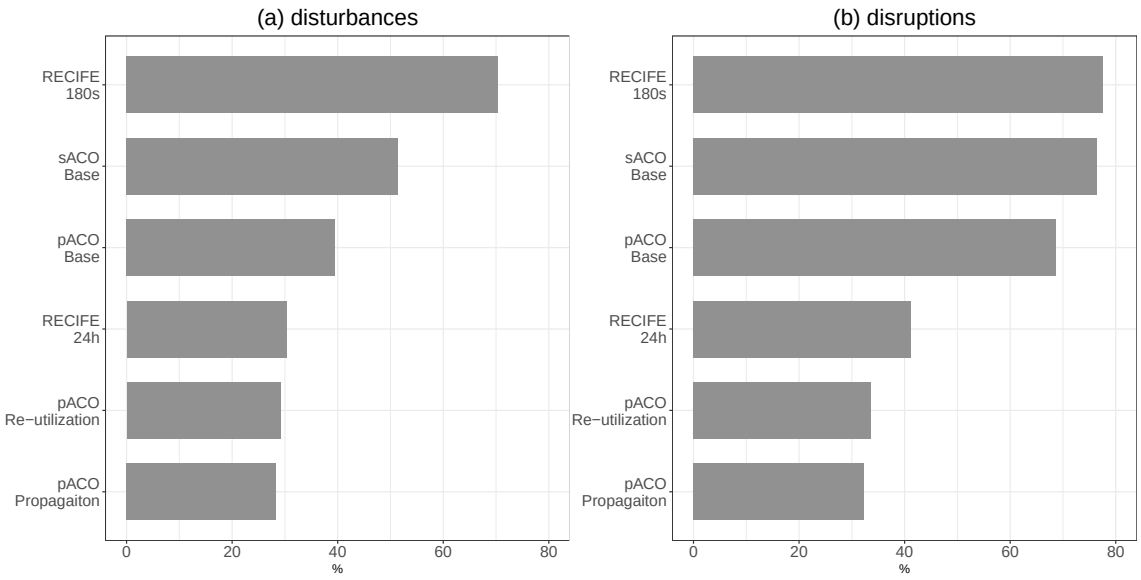

Figure 7: Average optimality gap of the rtRTMP objective value for the studied approaches, in case of (a) disturbances or (b) disruptions in the rail network

gap to 51.5% (76.4%). An improvement of 12.1% (23.5%) compared to the approach of Samà et al. (2016) is achieved when parallel computing is applied in pACO Base. The computational results obtained with the ACO speed-up reduce the optimality gap to 39.5% (68.6%). Re-utilization and Propagation costs effectively boost the rtRTMP solutions, even compared to RECIFE 24h. Specifically, RECIFE 24h certifies the optimum in 13 (3) out of 100 disturbed (disrupted) instances, confirming the complexity of solving these instances. When applying pACO Re-utilization, 5 (0) of the 13 (3) optimal solutions are found. However, the suboptimal solutions of RECIFE 24h are often worse than those of pACO Re-utilization and Propagation, leading to lower optimality gaps in these cases. When Propagation cost is applied, the solutions slightly improve: 6 (1) optima are found, and the optimality gap decreases to 28.3% (32.3%).

We next perform the Wilcoxon signed rank test (Woolson et al., 2007) with a confidence interval (CI) of 95% and 75% to ensure that the improvement to the rtRTMP solution brought by pACO Propagation is statistically significant. We compute the difference between the rtRTMP objective value obtained by each solution approach and the one obtained by pACO Propagation. If the pseudo-median μ resulting from this test is equal to zero (null hypothesis), no significant improvement is recorded. Positive values of μ and of the lower (LCI) and upper (UCI) bounds of the CI means that pACO Propagation improves significantly the rtRTMP solutions, vice versa, it does not with negative values. For the sake of clarity, the detailed computational results of the Wilcoxon signed rank test are provided in Appendix B. This test shows that pACO Propagation improvement is statistically significant in case of disruptions, with a 75% confidence level, when compared to pACO Re-utilization solutions. In all the other cases, both for disturbances

and disruptions, the improvement brought by pACO Propagation is statistically significant with a 95% confidence level. Therefore, we recommend the use of pACO Propagation when solving the rtRTMP.

8 Conclusions and further research

In this paper, we present an improved TRSP model and algorithm. The TRSP solutions can be used to limit the number of routing variables in the rtRTMP. The latter is an NP-complete problem, strongly affected by the (often) high number of routing variables and by the short available computation time. Our contributions address the challenge of improving the rtRTMP solution quality. We enhance the TRSP model by including rolling stock re-utilization constraints and by improving the ability to estimate the train delay propagation. Furthermore, we propose a parallel ACO algorithm to speed up the TRSP solution process. We introduce two local search algorithms to diversify the solution search. We analyze the impact of each contribution on solving the rtRTMP through a comprehensive computational campaign performed on the Lille Flandres station area in France, with the study of timetable disturbances and infrastructure disruptions.

Compared to the state-of-the-art method (Samà et al., 2016), the proposed model improvements lead to a better correlation between TRSP and rtRTMP solutions, which translates into the ability to recommend higher quality train routing combinations for the rtRTMP. Additionally, the proposed ACO algorithm performs a better exploration of the TRSP solution space, improving its convergence toward quality solutions. As a result, our TRSP approach defines a good search space for the rtRTMP, letting us to obtain significantly better solutions than the state-of-the-art method: the rtRTMP objective values improve by 32.3% and 65.1% in case of disturbed and disrupted traffic situations in Lille.

Future work should address how changes in the used rtRTMP model, solution method, rail infrastructure representation, and objective function may be considered in the TRSP formulation. Furthermore, other TRSP solution techniques could be designed.

Acknowledgements

This work was partially supported by the "SPECIES Scholarships" with a three-month mobility grant for Ph.D. candidates.

References

Binder, S., Maknoon, Y., & Bierlaire, M., 2017. The multi-objective railway timetable rescheduling problem. *Transportation Research Part C*, 78, 78-94.

Boccia, M., Mannino, C., & Vasilyev, I., 2013. The dispatching problem on multitrack territories: Heuristic approaches based on mixed integer linear programming. *Networks*, 62(4), 315–326.

Bettinelli, A., Santini, A., Vigo, D.,2017. "A real-time conflict solution algorithm for the train rescheduling problem". *Transportation Research Part B*, 106, 237–265.

Caimi, G., Chudak, F., Fuchsberger, M., Laumanns, M., Zenklusen, R., 2011. A new resource-constrained multicommodity flow model for conflict-free train routing and scheduling. *Transportation Science*, 45 (2), 212–227.

Cavone, G., Blenkers, L., van den Boom, T., Dotoli, M., Seatzu, C., De Schutter, B., 2019. Railway disruption: a bi-level rescheduling algorithm. *In 2019 6th International Conference on Control, Decision and Information Technologies (CoDIT), IEEE*, 54-59.

Chapman, B., Jost, G., Van Der Pas, R., 2008. *Using OpenMP: portable shared memory parallel programming (Vol. 10)*. MIT press.

Corman, F., D'Ariano, A., Pacciarelli, D., & Pranzo, M., 2010. A tabu search algorithm for rerouting trains during rail operations. *Transportation Research Part B*, 44(1), 175-192.

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.

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 (2), 643–657.

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(4), 405-419.

D'Ariano, A., Meng, L., Centulio, G., Corman, F., 2019. Integrated stochastic optimization approaches for tactical scheduling of trains and railway infrastructure maintenance. *Computers & Industrial Engineering*, 127, 1315–1335.

Dollevoet, T., Huisman, D., Kroon, L. G., Veelenturf, L. P., & Wagenaar, J. C., 2017. Application of an iterative framework for real-time railway rescheduling. *Computers & Operations Research*, 78, 203–217.

Dorigo, M., Maniezzo, V. and Colorni, A., 1996. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(1),29–41.

Dorigo, M., Stützle, T., 2004. Ant Colony Optimization, *MIT Press, Massachusetts Institute of Technology, Cambridge.*

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(1), 258-264.

Gholami, O., & Törnquist Krasemann, J., 2018. A heuristic approach to solving the train traffic re-scheduling problem in real time. *Algorithms*, 11(4), 55.

Goverde, R. M., Bešinović, N., Binder, A., Cacchiani, V., Quaglietta, E., Roberti, R., Toth, P., 2016. A three-level framework for performance-based railway timetabling. *Transportation Research Part C*, 67, 62–83.

Hansen, I.A., Pachl, J., 2014. Railway timetabling & operations. Hamburg: Eurailpress.

Josyula, S. P., Krasemann, J. T., Lundberg, L, 2018. A parallel algorithm for train rescheduling. *Transportation Research Part C*, 95, 545–569.

Kroon, L. G., Romeijn, H. E., & Zwaneveld, P. J., 1997. Routing trains through railway stations: complexity issues. *European Journal of Operational Research*, 98(3), 485–498.

López-Ibánez, M., Dubois-Lacoste, J., Cáceres, L. P., Birattari, M., & Stützle, T., 2016. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3, 43-58.

Mascis, A., Pacciarelli, D., 2002. Job shop scheduling with blocking and no-wait constraints. *European Journal of Operational Research,* , 143 (3), 498–517.

Pachl, J.,2007. Avoiding deadlocks in synchronous railway simulations. *In 2nd international seminar on railway operations modelling and analysis* , 1–10.

Pascariu, B., Samà, M., Pellegrini, P., D'Ariano, A., Pacciarelli, D., Rodriguez, J., 2021. Train routing selection problem: Ant colony optimization versus integer linear programming. IFAC-PapersOnLine, 54(2), 167-172.

Pellegrini, P., Marlière, G., Rodriguez, J., 2014. Optimal train routing and scheduling for managing traffic perturbations in complex junctions. *Transportation Research Part B*, 59 (1), 58–80.

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 Trans. on Intelligent Transportation Systems,* , 16 (5), 2609–2619.

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(1), 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*, 120 (1), 33–48.

Samà, M., Meloni, C., D'Ariano, A., & Corman, F., 2015. A multi-criteria decision support methodology for real-time train scheduling. *Journal of Rail Transport Planning & Management* , 5(3), 146-162.

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*, 85 (1), 89–108.

Samà, M., D'Ariano, A., Corman, F., Pacciarelli, D. 2017a. 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., 2017b. On the tactical and operational train routing selection problem. *Transportation Research Part C*, 76, 1–15.

Šemrov, D., Marsetič, R., Žura, M., Todorovski, L., & Srdic, A., 2016. Reinforcement learning approach for train rescheduling on a single-track railway. *Transportation Research Part B*, 86, 250-267.

Solnon, C., Bridge, D., 2006. An Ant Colony Optimization Meta-Heuristic for Subset Selection Problems. In *System Engineering using Particle Swarm Optimization*. Nova Science publisher, 7–29.

Stützle, T., Hoos, H.H., 2000. MAX-MIN ant system. *Future Generation Computer Systems*, 16 (8), 889–914.

Tang, L., D'Ariano, A., Xu, X., Li, Y., Ding, X., & Samà, M., 2021. Scheduling local and express trains in suburban rail transit lines: Mixed–integer nonlinear programming and adaptive genetic algorithm. *Computers & Operations Research*, 135, 105436.

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. (2006). Computer-based decision support for railway traffic scheduling and dispatching: A review of models and algorithms. *In 5th Workshop on Algorithmic Methods and Models for Optimization of Railways (ATMOS'05)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

Van Thielen, S., Corman, F., Vansteenwegen, P., 2018. Considering a dynamic impact zone for real-time railway traffic management. *Transportation Research Part B*, 111 (1), 39–59.

Veelenturf, L. P., Kidd, M. P., Cacchiani, V., Kroon, L. G., & Toth, P., 2016. A railway timetable rescheduling approach for handling large-scale disruptions. *Transportation Science* , 50(3), 841-862.

Woolson, R. F., 2007. Wilcoxon signed-rank test. Wiley encyclopedia of clinical trials, 1-3.

Zhang, H., Li, S., Wang, Y., Wang, Y., & Yang, L., 2021. Real-time optimization strategy for single-track high-speed train rescheduling with disturbance uncertainties: A scenario-based chance-constrained model predictive control approach. *Computers & Operations Research*, 127, 105135.

Zhang, Y., D'Ariano, A., He, B., Peng, Q., 2019. Microscopic optimization model and algorithm for integrating train timetabling and track maintenance task scheduling. *Transportation Research Part B*, 127, 237–278.

Zhu, Y., Goverde, R. M., 2019. Railway timetable rescheduling with flexible stopping and flexible short-turning during disruptions. *Transportation Research Part B*, 123, 149-181.

Appendix A

Procedure A.1 details the fixed cost computation method. This procedure is common to Base cost, Re-utilization and Propagation building blocks. Indeed, a parameter of the procedure indicates which additional building block is to be used, whether none, Re-utilization or Propagation. Depending on this input, the computation differs.

Considering as input the construction graph $G = (C, L)$, and every common infrastructure resource res between each of pair route assignments $c_i, c_j \in C$ linked by edge $l_{ij} \in L$, the fixed cost w_{ij} for edge l_{ij} is computed as follows. Let us first focus on the Base cost computation. Here, if two route assignments concern trains that use the same rolling stock (i.e., $\Re(c_i, c_j) = 1$ or $\Re(c_j, c_i) = 1$, if the train using c_i precedes the one using c_j, or vice versa), no potential conflict arises between the concerned trains and the fixed cost w_{ij} for edge l_{ij} is set to zero. Otherwise, we analyze the set of common resources (res) between the considered route assignments.

When c_i, c_j are not mutually involved in a rolling stock re-utilization constraint, the infrastructure utilization overlap (O), due to the two possible ordering decisions, is computed for each res. The infrastructure utilization overlap $O^{res}_{c_i c_j}$, occurring when c_i [c_j] is traversed before c_j [c_i] on the common resource res, is the time difference between $eU_{c_i,res}$ [$eU_{c_j,res}$] and $sU_{c_j,res}$ [$sU_{c_i,res}$], i.e., the time at which res ends being utilized by c_i [c_j] and starts being utilized by c_j [c_i]. Among the utilization overlaps of all the common resources related to pair c_i, c_j, the maximum value is chosen for each train ordering decision $O_{c_i c_j}, O_{c_j c_i}$. The final value of w_{ij} is then obtained by selecting the train ordering decision which minimizes the maximum infrastructure utilization overlap. When no common resource exists between a pair c_i, c_j, its cost $w_{ij} = 0$.

When using the Re-utilization building block, the *temporal coherence* of the trains connected by turnaround, join or split operations is estimated as follows. Let us consider the pair of trains $t, t' \in T$ with t' following t. The temporal coherence conditions consist of: (i) respecting the minimum time, called *processTime*, between the arrival of t and the departure of t', necessary to perform the rolling stock re-utilization operations; (ii) ensuring that the involved common resource (where the re-utilization takes place) cannot be occupied by any other train between the arrival of t and the departure of t'.

Procedure A.1 addresses condition (i) at line 5. Here, the fixed cost is equal to the delay that will be suffered by the following train, due to the late arrival time of the first train. Let $\overline{res}$ be the involved common resource, where the re-utilization operations occur. The train delay is equal to the (non negative) difference between the *processTime* and the time elapsed between the utilization start (sU) of $\overline{res}$ by the following train and the utilization end (eU) of $\overline{res}$ by the first train.

Condition (ii) is addressed in Procedure A.1 at lines 9-16. This is considered for each common resource between trains t and v that use route assignments c_i and c_j, respectively. We note that t and v do not use the same rolling stock. We need to check if either t or v is subject to rolling

Procedure A.1: Fixed cost computation

Data: Construction graph $G = (C, L)$; common resources *res* for each pair $c_i, c_j \in C$ connected by edge $l_{ij} \in L$; Re-utilization and Propagation building blocks

Result: Fix cost w_{ij} of each edge l_{ij} in the graph construction graph G

forall $l_{ij} \in G$ **do**
  $O_{c_i c_j} = O_{c_j c_i} = -bigInt$, $w_{ij} = 0$
  if $\mathfrak{R}(c_i, c_j) = 1$ *or* $\mathfrak{R}(c_j, c_i) = 1]$ **then**
    if *not Base cost* **then**
      $w_{ij} = max(0, processTime - (sU_{c_j[i],\overline{res}} - eU_{c_i[j],\overline{res}}))$
    end
  else if $\exists$ *res between* c_i, c_j **then**
    forall *res between* c_i, c_j **do**
      if *not Base cost* **then**
        forall $\mathfrak{R}(c_{i[j]}, c_h) : c_{i[j]} \in C_{t[v]}, c_h \in C_{t'}$ *and* $\overline{res} \cap res \neq \emptyset$ **do**
          $eU_{c_{i[j]},res} = max(eU_{c_{i[j]},\overline{res}}, eU_{c_h,res})$
        end
        forall $\mathfrak{R}(c_h, c_{i[j]}) : c_{i[j]} \in C_{t[v]}, c_h \in C_{t'}$ *and* $\overline{res} \cap res \neq \emptyset$ **do**
          $sU_{c_{i[j]},res} = min(sU_{c_{i[j]},\overline{res}}, sU_{c_h,res})$
        end
      end
      $O^{res}_{c_i c_j} = eU_{c_i,res} - sU_{c_j,res}$
      $O^{res}_{c_j c_i} = eU_{c_j,res} - sU_{c_i,res}$
      $O_{c_i c_j} = max(O_{c_i c_j}, O^{res}_{c_i c_j})$
      $O_{c_j c_i} = max(O_{c_j c_i}, O^{res}_{c_j c_i})$
    end
    $w_{ij} = min(O_{c_i c_j}, O_{c_j c_i})$
    if $w_{ij} \leq 0$ *and not Propagation* **then**
      $w_{ij} = 1$
    end
  end
end

stock re-utilization constraints with train t' on a common resource $\overline{res}$, which physically overlaps res, when t' uses route assignment c_h. If t $[v]$ precedes t' in the re-utilization constraint, the utilization end of res by t $[v]$ is set equal to the utilization end of $\overline{res}$ by t': the common resource will only be usable by v $[t]$ after the departure of t', when t $[v]$ passes first. When the re-utilization constraint involves more than two trains, the utilization of $\overline{res}$ ends when the last train involved in the constraint finishes using it. On the contrary, if t $[v]$ follows t' in the re-utilization constraint, the utilization start of res by t $[v]$ is set equal to the utilization start of $\overline{res}$ by t'. In the latter case, when more than two trains are involved in the re-utilization constraint, the utilization of $\overline{res}$ starts when the first train of the constraint starts using it.

When c_i, c_j have common resources but no potential delay, Base and Re-utilization building block set the fixed cost $w_{i,j} = 1$ at line 24 of Procedure A.1. No potential delay occurs when the trains related to c_i, c_j utilize a common resource without overlapping in time. A negative value of $w_{i,j}$ indicates that the related trains have no potential delays and the minimum distance separating their utilization of the common resources is equal to the absolute value of $w_{i,j}$. Propagation building block keeps the negative value $w_{i,j}$ for computing the clique-dependent cost (see Section 5).

Appendix B

We perform the Wilcoxon signed rank test (Woolson et al., 2007) to analyse if pACO Propagation solutions are statistically better then those obtained with the other rtRTMP solution methods investigated in this paper, i.e, RECIFE 180s, sACO Base, pACO BAse, pACO Re-utilization. We measure the improvement achieved by pACO Propagation as the difference between the objective value obtained by each solution approach and the one obtained by pACO Propagation. We use the test to determine whether the pseudo-median of the sample μ is equal to zero (null hypothesis) with a Confidence Interval (CI) of 95%. When this hypothesis is true, no significant improvement is recorded. Positive values of μ and of the lower (LCI) and upper (UCI) bounds of the CI means that the sample objective function improves significantly, vice versa it does not with negative values.

Tables B.1 and B.2 show the results in case of disturbed and disrupted instances. The first column of these tables indicates the rtTMP solution method used in the comparison. The probability value (p-value), in the second column, is a statistical measure describing how likely the data would have occurred by random chance. The remaining columns represent the above mentioned pseudo-median μ, and the lower (LCI) and upper (UCI) bounds of the CI. In Table B.2, the value within the brackets shows the result obtained with a confidence interval of 75%.

Table B.1: Statistical results of pACO Propagation improvement in case of disturbances

Approach	p-value	μ	LCI	UCI
RECIFE 180s ALL	5.78E-18	1096.86	940.00	1279.50
sACO Base	1.78E-11	299.50	198.00	421.50
pACO Base	2.72E-02	29.00	3.00	58.50
pACO Re-utilization	3.74E-02	22.00	1.00	43.00

Table B.2: Statistical results of pACO Propagation improvement in case of disruptions

Approach	p-value	μ	LCI	UCI
RECIFE 180s ALL	3.96E-18	1760.00	1568.00	1999.00
sACO Base	1.96E-17	1525.00	1067.50	2048.50
pACO Base	1.18E-07	483.50	172.50	773.50
pACO Re-utilization	5.70E-02	27.50	-1.00 (11.50)	57.00

.