Train routing selection problem: Ant colony optimization versus integer linear programming

Bianca Pascariu, Marcella Sama, Paola Pellegrini, Andrea D'ariano, Dario Pacciarelli, Joaquin Rodriguez

▶ **To cite this version:**

Bianca Pascariu, Marcella Sama, Paola Pellegrini, Andrea D'ariano, Dario Pacciarelli, et al.. Train routing selection problem: Ant colony optimization versus integer linear programming. CTS 2021, 16th IFAC Symposium on Control in Transportation Systems, Jun 2021, Lille, France. pp167-172, 10.1016/j.ifacol.2021.06.060 . hal-03288731

HAL Id: hal-03288731
https://hal.science/hal-03288731

Submitted on 3 Dec 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution - NonCommercial - NoDerivatives 4.0 International License

Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 54-2 (2021) 167–172

Train routing selection problem: Ant colony optimization versus integer linear programming

Bianca Pascariu * **Marcella Samà** * **Paola Pellegrini** ** **Andrea D'Ariano** * **Dario Pacciarelli** * **Joaquin Rodriguez** ***

* *Roma Tre University, Department of Engineering, Via della Vasca Navale 79, 00146 Roma, Italy (e-mail: {bianca.pascariu, marcella.sama, andrea.dariano, dario.pascciarelli}@uniroma3.it).*

** *Univ. Lille Nord de France, Ifsttar, COSYS, LEOST, rue Élisée Reclus 20, 59666 Villeneuve d'Ascq, Lille, France (e-mail: paola.pellegrini@univ-eiffel.fr)*

*** *Univ. Lille Nord de France, Ifsttar, COSYS, ESTAS, rue Élisée Reclus 20, 59666 Villeneuve d'Ascq, Lille, France (e-mail: joaquin.rodriguez@univ-eiffel.fr)*

Abstract: The real-time Railway Traffic Management Problem (rtRTMP) is the problem of detecting and solving time-overlapping conflicting request done by multiple trains on the same track resources. It typically consists in taking retiming, reordering or rerouting train actions in such a way that the propagation of disturbances in the railway network is minimized. The rtRTMP is an NP-Hard problem and finding good strategy to simplifying its solution process is paramount to obtain good quality solutions in a short computation. Solving the Train Routing Selection Problem (TRSP) aims to do so, by limiting the number of routing variables through a pre-processing that selects the most promising routing alternatives among the available ones for each train in order to reduce the size of rtRTMP instances. This paper studies the performance of an Ant Colony Optimization (ACO) algorithm for the same problem. An integer linear programming formulation for the TRSP is presented and solved using a commercial software, and it is considered as a benchmark. Computational experiments are performed on two practical case studies of the French railway infrastructure: the line near the city of Rouen and the Lille terminal station area. ACO and the commercial solver perform comparably only on small instances and both are able to find optimal solutions. However, on larger instances, the ACO algorithm outperforms the commercial software, both in terms of computation time and solution quality.

Copyright © 2021 The Authors. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0)

Keywords: Rail Transportation; Public Transportation; Scheduling and optimization of transportation systems; Intelligent Transportation Systems

1. INTRODUCTION

To ensure a safe and regular train service, rail infrastructure managers periodically and with large advance design timetables. However, train schedules and operations are subordinated to a large number of parameters, making it vulnerable to service disturbances. The dispatchers are thus required to quickly detect new conflicts arising during operations and take action to recover feasibility, typically by retiming, reordering or rerouting trains in such a way that the propagation of such disturbances is minimized. This problem is know in the literature as the real-time Railway Traffic Management Problem (rtRTMP). Several models and algorithms have been developed to solve the problem to provide Decision Support Systems to help dispatcher take more informed decisions (Cacchiani et al., 2014; Borndörfer et al., 2017). Still, the rtRTMP is an NP-Hard problem and simplifying its model or its solution process is a key aspect to obtain good quality solution in the short computation time imposed by the nature of the problem, especially when considering complex networks. Some approaches limit the size of the problem by intervening on the granularity used to model the infrastructures and the traffic flows. The minimum section of infrastructure is the track circuit, and a sequence of track-circuits between two consecutive signals forms a block-section. Then, instead of modelling each resource as a single block-section (Zhang et al., 2019) or track circuit (Pellegrini et al., 2019), as is typical in *microscopic* approaches, *macroscopic* approaches aggregate the network infrastructure mainly into nodes and arcs corresponding to stations and track segments (Lamorgese & Mannino, 2015), while *mesoscopic* ones use a mixture of microscopic and macroscopic, e.g., gouping multiple block-sections (Meng & Zhou, 2014). Some approaches focus on the solution process to properly drive the search of good solutions. For example, initial solutions are computed for the scheduling problem considering the timetable route that are then improved by either enlarging the search space adding rerouting variables (Pellegrini et al., 2015) or iteratively solving the scheduling and routing

2405-8963 Copyright © 2021 The Authors. This is an open access article under the CC BY-NC-ND license.
Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2021.06.060

problems separately (Samà et al., 2017). Others, limit the number of variables by considering only what they consider to be the most significant ones. Looking at retiming and reordering variables, Van Thielen et al. (2018) include only the ones with the highest impact, i.e., the ones involving the trains engaged in the initial conflicts. Often, rerouting variables are the ones most affecting the size of the search space (D'Ariano et al., 2019), thus some approaches disregard them, limiting the rtRTMP to a pure scheduling problem (**?**), others use subsets of all the 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) study this last option by proposing the Train Routing Selection Problem (TRSP). The TRSP is a pre-processing step for the rtRTMP where a feasible and optimized subset of alternative routes is selected for each train using an Ant Colony Optimization (ACO) algorithm (Dorigo & Stützle, 2004). This approach appears to be promising as it has been shown to provide a good starting point and a refined search space for the rtRTMP. However, the performance of the ACO algorithm has not been thoroughly tested nor evaluated, leaving unanswered the question whether ACO is actually suitable to tackle the TRSP.

To answer this question, the aim of this paper is to compare the performance of the ACO algorithm introduced in Samà et al. (2016) with the one of an actual competitor. To do so, we propose an integer linear programming (ILP) formulation of the TRSP. Then, we solve it on different case studies using a commercial software and compare the result with the ones obtained by ACO. The aim is to verify if the ACO algorithm can compete with an exact algorithm under two perspectives. We first assess its ability to be competitive with the commercial solver when the computation time is or not limited. Second, we observe ACO ability to identify the optimal solution of the TRSP, avoiding stagnation in local minima. The analysis is performed taking into account two French infrastructures: the line around the city of Rouen and the Lille terminal station area. For each infrastructure are considered different perturbed timetables, traffic types and time windows.

The rest of the paper is structured as follows: Section 2 presents a description of the TRSP, while Section 3 the specific formulation used in this paper. Section 4 briefly illustrates how the ACO algorithm works. Section 5 shows the computational results and their analysis on the two case studies while Section 6 summarizes the conclusions and suggests where the future research is promising.

2. PROBLEM DESCRIPTION

The TRSP solution represents an input to the rtRTMP that aims to reduce its search space. Therefore, the TRSP must be consistent with the general operational and technical constraints of the railway system, and oriented to the rtRTMP objective function. The operational constraints regard commercial requirements. They involve trains arrival, departure and dwell times at stations, as specified in the timetable. The technical constraints instead regard the trains running time and the *minimum headway times* imposed by the signaling system in each infrastructure section. The *minimum headway time* is the minimum time that shall separate two trains in order to avoid any overlap of the infrastructure utilization, following the blocking time theory (Hansen & Pachl, 2014). They also ensure that trains can comply with operational constraints in practice, at least in absence of traffic perturbations. During operations, unexpected events may cause train delays (*primary delay*), which may lead to an infrastructure utilization overlap by two trains. The simultaneous utilization of the same infrastructure section(s) will be prevented by the signaling system that will stop one of the two concurrent trains, which will inevitably suffer a delay. Solving the rtRTMP translates into providing an optimized rescheduling that minimizes the propagation of such delays. When rerouting trains is contemplated, the TRSP concurs to this objective as well.

The TRSP is then a combinatorial optimization problem in which, given an initial set of trains each with a set of alternative routes, the goal is to choose a subset of routes for each train that will help the rtRTMP into providing more quickly better quality solution. A *train route* is an ordered list of all infrastructure sections a train passes through to reach the locations where it is scheduled to stop, i.e., its *stopping points*. A *set of alternative train routes* includes all those with equal stopping points but crossing different infrastructure sections.

3. PROBLEM FORMULATION

Given a railway infrastructure with a set R of routes, and a corresponding timetable, a set T of n trains is present within a certain time window. For each train $t \in T$ a set $R_t \subset R$ of alternative routes $r \in R_t$ is given. For a pair of trains $t, v \in T$, we define their routes $r_t \in R_t, r_v \in R_v$ *coherent* in case of no rolling stock constraint (e.g., turnaround, join or split) between them; if such constraint exists, they are coherent when the last infrastructure section of the first train corresponds to the first infrastructure section of the second train. A combination of train routes is *feasible* if each pair of train routes is coherent and for each train one route has been selected. The objective function of our TRSP formulation uses the concept of *potential delay* (Samà et al., 2016), which estimates by pairs the consecutive delay of trains on specific routes. Trains scheduling and ordering are not explicitly defined, so the delay of a train is propagated only within the considered pair and not to other trains. Given an infrastructure, a perturbed timetable and a time window, the question is whether there exist p feasible combinations of train routes which minimize the total potential delay.

3.1 Graph representation

We model the TRSP according to the *construction graph* $G = (C, L)$ proposed by Samà et al. (2016), illustrated with an example in Fig. 1. A vertex $c_i \in C$ in the graph represents an alternative route of a train $t \in T$. Given n trains, the vertices are grouped according to each set $R_t \subset R$ of alternative train routes. G is then an n-partite graph such that $\cup_{i=1}^{n} R_i = C$ and $\cap_{i=1}^{n} R_i = \emptyset$. Two vertices $c_i, c_j \in C$ are connected by an edge $l_{ij} \in L$ if they

represent coherent routes and belong to different trains. The n-vertex cliques in this graph identify the set $\Gamma \subset C$ of all feasible combinations of train routes. A clique is a subset of vertices, of a not-oriented graph, such that every two distinct vertices in the clique are adjacent (Solnon & Bridge, 2006). Therefore, a subset $S_k \subset \Gamma$ is a set of k feasible combinations of train routes and thus a TRSP solution. Fig. 1 highlights an example of n-vertex clique in bold. Each vertex $c_i \in C$ and edge $l_{ij} \in L$ in the graph has a non-negative cost: the unavoidable delay u_i due to the longer running time of a route compared to the timetable one; the potential delay w_{ij} suffered when two routes are jointly considered.

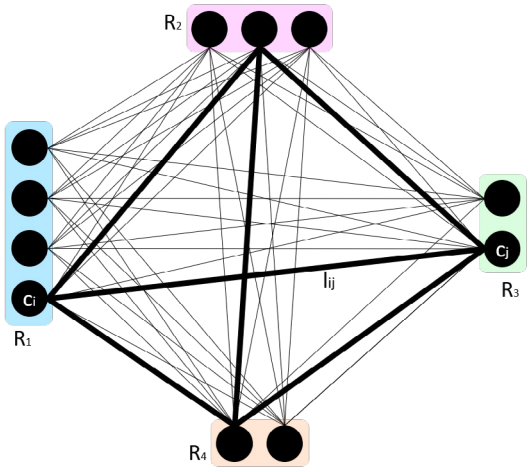

Fig. 1. Example of a construction graph $G = (C, L)$

3.2 ILP formulation

In order to model the problem of building the minimum cost n-vertex clique as an integer linear program, we introduce the binary decision variables r_i and z_{ij}, respectively $\forall c_i \in C$ and $\forall l_{ij} \in L$ as follows. We remind that arcs in G are not-oriented, i.e., $l_{ij} = l_{ji}$ and they both correspond to the arc connecting c_i and c_j.

$$r_i = \begin{cases} 1 & \text{if node } c_i \in C \text{ is selected} \\ 0 & \text{otherwise} \end{cases}$$

$$z_{ij} = \begin{cases} 1 & \text{if edge } l_{ij} \in L \text{ is selected} \\ 0 & \text{otherwise} \end{cases}$$

The optimization problem, where the objective is the minimization of the total potential delay suffered by trains due to the route choice, states as follows:

$$min \sum_{c_i \in C} r_i \; u_i + \sum_{l_{ij}} z_{ij} \; w_{ij} \tag{1}$$

$$\sum_{c_i \in R_t} r_i = 1 \qquad \forall R_t \subset C \tag{2}$$

$$\sum_{c_j \in R_{v:v\in[t,n]}} z_{ij} = (n-t) r_i \qquad \forall t \in T: \; t = 1, ..., n, \forall c_i \in R_t \tag{3a}$$

$$\sum_{c_j \in R_{v:v\in[0,t)}} z_{ji} = (t-1) r_i \qquad \forall t \in T: \; t = 1, ..., n, \forall c_i \in R_t \tag{3b}$$

$$\sum_{c_i \in S_k} r_i \leq n - 1 \qquad \forall S_k, \forall k: \; k = 1, ..., p-1 \tag{4}$$

Constraints (2) ensure that to each train is assigned exactly one route/node. Constraints (3a) and (3b) guarantee that only arcs incident on selected nodes are selected. Considering the sub-sets $R_t \subset C$ sorted according to, e.g., the order in which trains enter the infrastructure. Train reordering is not explicitly addressed by the TRSP, so train order is invariant. Constraints (3a) refer to the arcs connecting the node c_i with nodes of the following trains, while (3b) with the preceding ones. Compared to the formulation in Samà et al. (2016), these two constraints allow to reduce by 50% the number of z_{ij} considered, resulting in a smaller formulation. If $p > 1$ feasible combinations of train routes (cliques) are desired, the problem is iteratively solved. At each iteration a new Constraint (4) is added, excluding from the search space identical combinations to the S_k. For the purposes of this work we consider $p = 1$.

4. ACO

Ant colony optimization (ACO) is a meta-heuristic that exploits the ant foraging behavior to solve hard combinatorial optimization problems (Dorigo & Stützle, 2004). Indirect communication between the ants via pheromone trails enables them to find shortest paths between their nest and food sources. The meta-heuristic adopts such pheromone concept, referring to a numerical matrix, to keep track of solutions' quality. Together with a heuristic information, the pheromone iteratively guides the solution space exploration from an incumbent solution to an hopefully better one. We take as reference for our work the ACO-TRSP algorithm implemented in Samà et al. (2016) to solve the TRSP, inspired by the ACO algorithm elaborated for the maximum clique problem by Solnon & Bridge (2006). At each iteration, each ant of the colony independently builds a solution based on the construction graph G. Specifically, solutions are based incrementally: one component at a time is selected based on a probability biased by pheromone trail and heuristic information, both weighted by integer parameters. After that, a local search is performed on the best clique found by the colony and the so-obtained solution is used to update the pheromone trail. An upper and lower bounds are imposed on the pheromone trails, following the MAX–MIN Ant System approach (Stützle & Hoos, 2000). During the update, some pheromone evaporates from all links of G and some additional pheromone is deposited on the links belonging to the best solution. ACO performance depends on the values of user-defined parameters such as: α, the pheromone factor weight; β, the heuristic factor weight; ρ, the pheromone evaporation rate; and $nAnts$ the number of ants in the colony. Following Samà et al. (2016), we use the setting $\alpha = 2$, $\beta = 2$, $\rho = 0.05$ and $nAnts = 150$.

5. COMPUTATIONAL RESULTS

This section presents the computational results regarding the performance evaluation of the ACO algorithm described in Section 4, in comparison to the ILP formulation proposed in Section 3. The two approaches are implemented and tested on two representative case studies, which can be classified as small and large size.

5.1 Test cases

The computational experiments have been performed in a laboratory environment for two French case studies based on real-world data. The first case study, shown in Figure 2, regards the 27-km-long railway line around the city of Rouen. The line is mainly on double-track with several intermediate stations, each with up to six platforms. The infrastructure is composed of 190 track-circuits, 189 block sections, and 11347 routes. The second case study is the 12-km-long Lille station area shown in Figure 3, with 299 track-circuits, 734 block sections, and 2409 routes. The Lille station is a terminal station linked to national and international lines, with 17 platforms used by local, intercity and high speed trains. A typical daily timetable has 186 trains for the Rouen case study and 509 for the Lille one. In our computational experiments, we consider two sets of traffic scenarios for both the Rouen and the Lille case study. They are obtained by perturbing a real one-day timetable. To 20% of the trains, randomly selected, are applied a random delay between 5 and 15 minutes at their entry point. For each perturbed timetable, starting from 10 different time instants randomly taken during time periods 7:30-9:00 and 18:30-20:00, we consider 4 sets of time windows: 30, 40, 50 and 60 minutes.

Table 1 shows the characteristics of each group of instances generated. Each row presents average values over 20 instances. Column 1 indicates the network the group of instances refers to. Columns 2 specifies the width of the time windows considered, while columns 3-5 show the size of the instances in terms of average number of trains, components and links.

Table 1. Instances characteristics

Infr.	Time Window	# Trains	$\|C\|$ $\|r_i\|$	$\|L\|$ $\|z_{ij}\|$
Rouen	30 min	7	439	79,699
Rouen	40 min	9	607	153,406
Rouen	50 min	12	717	216,566
Rouen	60 min	13	807	283,033
Lille	30 min	22	3,695	7,048,908
Lille	40 min	28	4,781	11,851,212
Lille	50 min	34	5,661	16,873,485
Lille	60 min	39	6,368	21,365,433

We perform the experiments on an Intel Xeon 22 core 2.2 GHz processor with 1.5 TB RAM, under Windows distribution. We use the IBM ILOG CPLEX Concert Technology for C++ (IBM ILOG CPLEX version 12.9) as ILP solver.

Since the TRSP is mainly designed to be solved in real-time as a pre-processing step for the rtRTMP, we give ACO algorithm a computation time of 30 seconds. To perform a satisfactory analysis, we use CPLEX on the TRSP ILP formulation with three different maximum computation time settings: 30 seconds, for real-time application, 180 seconds, to assess if it may be useful on other applications; and four hours in order to find the optimal solution for the tested instances.

5.2 Results

Table 2 presents the results so obtained, for both the Rouen and Lille instances. Column 1 displays the time windows considered as in Table 1. Column 2 shows the algorithm identifier, that is ACO, CPLEX with a computation time limit of 30 seconds (C. 30s), 180 seconds (C. 180s) or four hours (C. 4h). Columns 3 and 4 present respectively the number of instances for which the corresponding algorithm finds a feasible solution, and the number of instances for which it finds (proves) an optimal integer solution. Column 5 shows the average optimality gap calculated over the non optimal instances as $(\textit{upper bound} - \textit{lower bound})/\textit{upper bound}$, using as lower bound the best known value. Columns 6 and 7 indicate the average values of the objective function and computational time.

To evaluate the quality of ACO solutions, we compare them with the optimal solutions obtained by CPLEX or, in case it is not known, with the best lower bound. The average percentage difference of the objective value between the benchmark C. 4h and the other algorithms is also shown in Fig. 4. For a straightforward comparison of the algorithms, the objective value of C. 4h is considered to the denominator of the difference.

From Column 3 in Table 2, we can see that each algorithm finds a solution to all Rouen instances. The same does not happen for the Lille ones. Here, the growing number of variables severely affects the difficulty of finding feasible solutions in a limited computation time. CPLEX is unable to solve any instance in 30 seconds and it manages to

Table 2. Results

T.W. (min)	Solver	Solved (/20)	Opt. (/20)	Gap (%)	Obj. v. (s)	Comp. time (s)
			Rouen			
30	ACO	20	20	0	99	30
	C. 30s	20	20(17)	0	99	10
	C. 180s	20	20(19)	0	99	22
	C. 4h	20	20(20)	0	99	55
40	ACO	20	20	0	128	30
	C. 30s	20	14(7)	36	204	24
	C. 180s	20	17(10)	37	150	110
	C. 4h	20	20(20)	0	128	504
50	ACO	20	19	46	275	30
	C. 30s	20	7(2)	49	696	30
	C. 180s	20	9(3)	16	368	160
	C. 4h	20	20(20)	0	249	3853
60	ACO	20	14	79	351	30
	C. 30s	20	4(1)	68	1463	30
	C. 180s	20	7(1)	43	564	172
	C. 4h	20	14(14)	82	378	5755
			Lille			
30	ACO	20	0	100	235	30
	C. 30s	0	0	-	-	30
	C. 180s	7	0	100	631	180
	C. 4h	20	0	100	480	14400
40	ACO	20	0	100	448	30
	C. 30s	0	0	-	-	30
	C. 180s	8	0	100	1864	180
	C. 4h	19	0	100	798	14400
50	ACO	20	0	100	625	30
	C. 30s	0	0	-	-	30
	C. 180s	7	0	100	2321	180
	C. 4h	19	0	100	1061	14400
60	ACO	20	0	100	790	30
	C. 30s	0	0	-	-	-
	C. 180s	0	0	-	-	180
	C. 4h	13	0	100	1078	14400

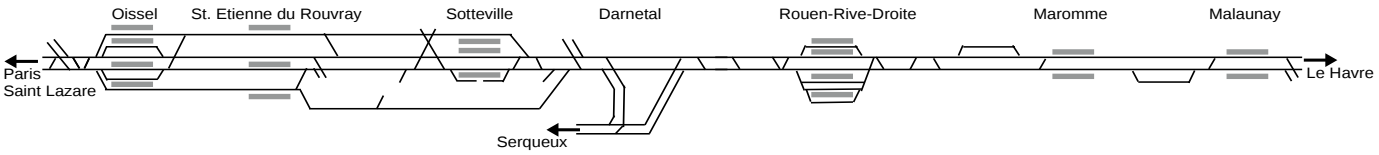

Fig. 2. Rouen network

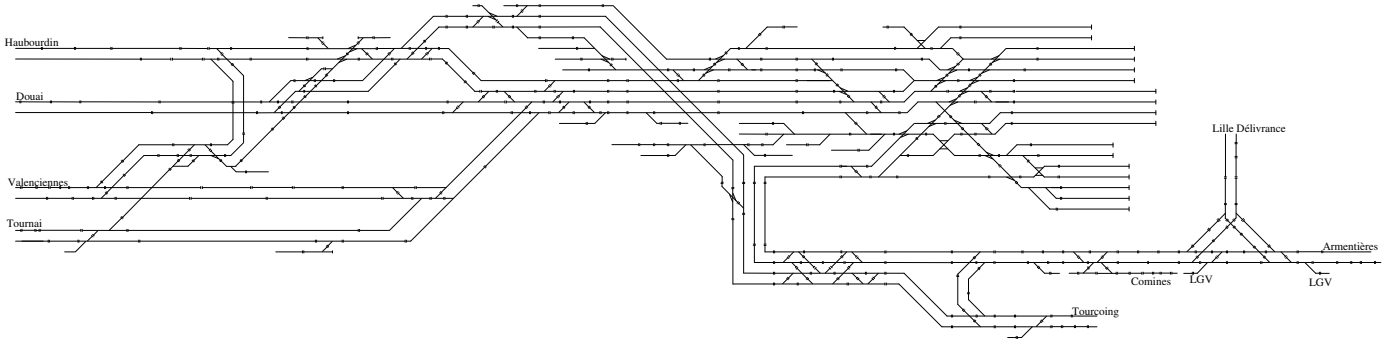

Fig. 3. Lille station area

deal with very few of them in 180 seconds, getting stuck in the presolve phase. In general, we remark that huge instances as in the Lille case are out of CPLEX's reach. By contrast, the ACO meta-heuristic is always able to provide a solution in a short time, even for the largest instances.

Focusing on Rouen instances, Table 2 shows that ACO is able to find almost all proven optimal solutions. As for the non-optimal ones, in the 50-minute time window, ACO fails to do so in one out of 20 instances. The relative performance worsening, however, does not appear to be a trend. In fact, for the 60-minute time window, ACO outperforms CPLEX disregard the computation time available. In particular, C. 4h finds an optimal solution for 14 of the 20 instances tackled. For the same instances, ACO returns the same solution as the former. Out of the remaining six instances, ACO finds better solutions than CPLEX in all but one cases. Regarding the real-time applicability, on the 30-minute time window instances, both CPLEX 30s and 180s find all the optimal solutions faster than ACO. The latter has always an average computational time of 30 seconds because it stops when it reaches the time limit or if it finds an objective value equal to zero. Cplex, on the other hand, stops the execution if it proves the optimality of a solution. However, with increasing problem size, neither C. 30s nor C. 180s is able to find all optimal solutions while ACO is, thus worsening their objective value in comparison to the latter. This can be seen both in Table 2 and in Fig. 4: when C. 30s does not find the optimum, its feasible objective values are very far from those of ACO; and also C. 180s, despite recording smaller values, they are about 30% worse in all time windows.

For the Lille test case, the optimality gap in Table 2 results to be 100% for all algorithms. As already mentioned, the huge number of binary variables reported in Table 1 leads to a difficult exploration of the solution tree by CPLEX. The linear relaxation of the formulation in Section 3 is very weak. In fact, the lower bound is always equal to zero. Fig. 4 shows ACO's performances outstand those of CPLEX, with much better objective functions on all Lille instances.

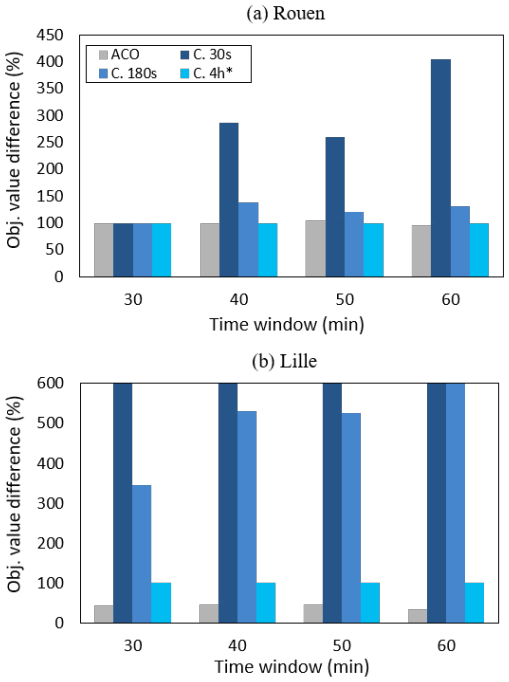

Fig. 4. Average percentage of objective value difference for (a) Rouen and (b) Lille instances (* is the reference).

C. 30s and 180s vary indefinitely in time windows where they found no solution.

We finally study the algorithm behavior in the given 30 seconds of computation. Fig. 5 shows the average variation of the objective value, computed as *(current objective value - final objective value)/final objective value* throughout the run. We show here the average over all time windows, as the algorithm behaviour does not significantly change from one to the other. We can observe a clear difference in the convergence speed of solutions in the two case studies. On the one hand, for Rouen, although ACO has a maximum computational time of 30 seconds, on average it converges on the best solution within the first few

seconds. Its computational time could then be reduced without loosing solutions quality. As seen in Table 2, the algorithm converges to the optimum most of the times, while for the instances for which the optimal solution is unknown, it may be trapped in local minima. The diversification of ACO algorithm should be increased here through a different parameter setting to escape these local minima. On the other hand, Fig. 5 confirms the difficulty of exploring the huge solution space of Lille instances. Indeed, the ACO algorithm improves the solution throughout the whole run. The graph shows greater slopes where major improvements are made, e.g., in the last part of the curve as the algorithm frequently finds the best solution in the final stage. A speed-up of the algorithm may allow to better explore the solution space, potentially leading to convergence also in this case.

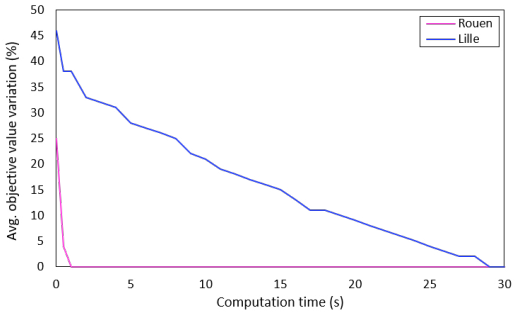

Fig. 5. Average variation of the objective value in time.

6. CONCLUSIONS AND FUTURE RESEARCH

This paper proposed an ILP formulation for the TRSP, solved using the CPLEX commercial solver. This is then used as benchmark to investigates the performances of an Ant Colony Optimization (ACO) algorithm for the TRSP. Computational experiments are performed on two practical case studies of the French railway infrastructure: the line near city of Rouen and the Lille terminal station area.

It results that ACO's performances outstand on average those of CPLEX, considering the computational time limit required for real-time application. In small instances, ACO solutions converge to the global optimum obtained by CPLEX in four hours of computation. In larger instances we do not have enough information to assess ACO absolute performance, as CPLEX fails to find optimum solutions. However, we note that as the size of the instances increases, ACO also begins to have difficulties in solving them. Although the algorithm progressively improves the solution throughout a single run, it does not converge in the available computation time. To improve its performances, future research will be focused on the speed up of the ACO algorithm, perhaps through parallelism, and a specific tuning of its parameters.

REFERENCES

Borndörfer, R., Klug, T., Lamorgese, L., Mannino, C., Reuther, M., Schlechte, T., 2017. Recent success stories on integrated optimization of railway systems. *Transportation Research Part C*, 74, 196–211.

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

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

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.

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

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

Josyula, S.P., Törnquist Krasemann, J., Lundberg, L. 2018. A parallel algorithm for train rescheduling. *Transportation Research Part C*, 98 (1), 545–568.

Lamorgese, L., Mannino, C., 2015. An exact decomposition approach for the real-time Train Dispatching problem. *Operations Research*, 63 (1), pp. 48–64

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

Meng, L., Zhou, X., 2014. Simultaneous train rerouting and rescheduling on an N-track network: A model reformulation with network-based cumulative flow variables. *Transportation Research Part B*, 67 (1), 208–234.

Pellegrini, P., Marlière, G., Pesenti, R., Rodriguez, J., 2015. RECIFE-MILP: An effective MILP-based heuristic for the real-time railway traffic management problem. *IEEE Transactions on Intelligent Transportation Systems*, **16 (5)** 2609–2619.

Pellegrini, P., Presenti, R., Rodriguez, J., 2019. OEfficient train re-routing and rescheduling: Valid inequalities and reformulation of RECIFE-MILP. *Transportation Research Part B*, **120 (1)** 33–48.

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. 2017. A variable neighbourhood search for fast train scheduling and routing during disturbed railway traffic situations. *Computers & Operations Research* , 78, 480–499

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.

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.

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.