Annals of Operations Research
https://doi.org/10.1007/s10479-023-05784-7

ORIGINAL RESEARCH

Solving a real-life multi-skill resource-constrained multi-project scheduling problem

Rahman Torba[1,2] · Stéphane Dauzère-Pérès[1,3] · Claude Yugma[1] · Cédric Gallais[2] · Juliette Pouzet[2]

Received: 10 February 2023 / Accepted: 7 December 2023

Abstract

This paper addresses a multi-skill resource-constrained multi-project scheduling problem (MSRCMPSP) with different types of resources and complex industrial constraints, which originates from SNCF heavy maintenance factories. Two objective functions, that have been rarely addressed in the literature, are independently considered: (i) Minimization of the sum of the weighted tardiness of the projects and (ii) Minimization of the sum of the weighted duration of the projects. A time-indexed mixed-integer linear programming model is presented with both resource assignment and capacity constraints. To solve large instances with several thousand activities, a new memetic algorithm combining a novel hybrid simulated genetic algorithm with a simulated annealing is implemented. The memetic algorithm is compared with popular solution approaches. Computational experiments conducted on real instances and benchmark instances validate the efficiency of the proposed algorithm.

Keywords Scheduling · Integer linear programming · Metaheuristics · Multiple projects · Multiple skills · Maintenance

✉ Rahman Torba
r.torba@emse.fr

Stéphane Dauzère-Pérès
dauzere-peres@emse.fr

Claude Yugma
yugma@emse.fr

Cédric Gallais
cedric.gallais@sncf.fr

Juliette Pouzet
juliette.pouzet@sncf.fr

[1] Department of Manufacturing Sciences and Logistics, Mines Saint-Etienne, Univ Clermont Auvergne CNRS, UMR, LIMOS, 6158 Gardanne, France

[2] SNCF, Saint-Denis, France

[3] Department of Accounting and Operations Management, BI Norwegian Business School, Oslo, Norway

Published online: 30 January 2024

1 Introduction

In this paper, a multi-skill resource-constrained multi-project scheduling problem (MSRCMPSP) is modeled and solved. The problem is motivated by a real industrial issue at SNCF, the French national railway company, which carries out the heavy maintenance of its rolling stock in ten different factories. Several rolling stock units are maintained simultaneously, and each unit is considered as a project. To complete each project, a certain number of activities requiring multiple skilled resources must be performed. Different types of resources (maintenance operators and machines) with different characteristics and constraints are taken into account. Two original objective functions, that have rarely been addressed in the literature, are independently considered: (i) Minimization of the sum of the weighted tardiness of the projects (SWTP) and (ii) Minimization of the sum of the weighted duration of the projects (SWDP).

Project management has attracted the attention of many researchers over the years. Planning and scheduling are crucial for controlling the execution time of projects, managing the required resources, and meeting (customer) deadlines. Without a detailed efficient plan, activities would be poorly executed causing resource conflicts, waste of time and thus cost increase. In a world where market competition and customer expectations are very high, companies want to digitize their processes to better control production, optimize resource allocation and reduce costs (Moeuf et al. 2018). However, project scheduling is a very complex problem. The well-known Resource-Constrained Project Scheduling Problem (RCPSP) belongs to the class of hard optimization problems, and instances of more than 60 activities cannot be solved with exact methods in reasonable computational time (Koné et al. 2011). Furthermore, most companies produce several products which generally share the same resources. The survey of Lova et al. (2000) found that 84% of companies work with several projects. The multi-project version of the RCPSP is closer to real-word applications. Yet, most papers study the single project version of the RCPSP. The objective function is often the minimization of the makespan, i.e., the completion time of the last activity. The multi-project extension results in more complex problems because of the scarcity of shared resources, the interactions and the competition among different projects, project specific characteristics, deadlines and more elaborate objective functions (Browning and Yassine 2010).

We first show how to model a complex industrial problem by proposing a time-indexed MILP (Mixed-Integer Linear Programming) model with several constraints such as precedence constraints with lag times, time-dependent resource capacity constraints and machine assignment. In the model, each rolling stock unit is considered as a project, maintenance operations as activities requiring a certain number of human resources with different skills and one specific machine to be executed. Since resources have multiple skills and several rolling stock units are maintained simultaneously, the problem studied in this paper corresponds to the Multi-Skill Resource-Constrained Multi-Project Scheduling Problem (MSRCMPSP) (Pritsker et al. 1969; Bellenguez and Néron 2004). The problem with the considered constraints and objective functions has never been addressed in the literature.

Secondly, several solution approaches are proposed and tested to find the most effective one. To address large industrial instances involving hundreds of projects and thousands of operations, two greedy algorithms are first introduced: a serial scheduling generation scheme and a parallel scheduling generation scheme with different priority rules. These priority rules are determined using a global precedence graph, which is constructed from the precedence graph of each individual project (which is a graph showing the critical paths for each project). The solutions produced by the greedy algorithms are then further optimized using a memetic algorithm. Finally, by conducting various computational experiments, the potential gains are quantified.

The main contributions of this paper are summarized below.

- To our knowledge, this is the first attempt to tackle the integration of both multi-project and multi-skill versions of the RCPSP problem with an objective function other than the makespan. A time-indexed MILP formulation of the problem is provided.
- A new Memetic Algorithm (MA) and a new Simulated Genetic Algorithm (hSGA) are proposed for scheduling multiple projects with different resource constraints. The proposed solution approaches are compared, using large industrial instances, to the time-indexed MILP model, and to popular solution approaches of the literature (two constructive heuristics, Simulated Annealing (SA) and a Genetic Algorithm (GA)).
- The MA is evaluated on benchmark instances, and the computational results show that it stands as one of the most effective methods compared to the existing literature.
- The gap between theory and practice is reduced by considering more realistic objective functions and constraints. As highlighted by Hartmann and Briskorn (2022), Sánchez et al. (2022) and Rahman et al. (2020) real-world case studies are necessary to motivate more complex models.

The paper is organized as follows. Section 2 gives an overview of the literature on project scheduling problems. Section 3 describes the industrial problem and introduces the associated mathematical model. The solution approaches are presented in Section 4. Section 5 provides the numerical results used to evaluate the performance of the proposed approaches. Finally, Section 6 concludes the paper and gives some perspectives on future work.

2 Literature review

In this section, we review the literature on the multi-project and multi-skill extensions of the RCPSP. Section 2.1 recalls the classical RCPSP and then focuses on the multi-project extension. Section 2.2 reviews the research on project scheduling problems with multi-skilled resources. In Section 2.3, papers integrating both multiple projects and multiple skills are discussed. The limits of the existing literature are also discussed.

2.1 Scheduling problems with multiple projects

The resource-constrained project scheduling problem (RCPSP) is a complex optimization problem that involves scheduling a set of activities subject to precedence constraints and resource availability (Deblaere et al. 2011). It is an extension of the classical job shop scheduling problem and is NP-hard in nature (Blazewicz et al. 1983). Since its formulation by Pritsker et al. (1969), many extensions and solution methods have been proposed. A comprehensive overview of the different RCPSP problems and variants can be found in the literature such as Özdamar and Ulusoy (1995), Brucker et al. (1999), and more recently Habibi et al. (2018) and Hartmann and Briskorn (2022). Kolisch (1996) and Kolisch and Hartmann (2006) examine the performance of various heuristics and priority rules for the classical RCPSP. Lancaster and Ozbayrak (2007) focus on evolutionary algorithms, while Pellerin et al. (2020) provide a comprehensive review on recent hybrid metaheuristics.

A generalization of the RCPSP is to consider the resource-constrained multi-project scheduling problem (RCMPSP). This extension is interesting since it is a step forward on modeling real world problems (Lova et al. 2000). Although the same methods for modeling and solving the RCPSP can be applied to the RCMPSP (Drexl 1991), developing efficient algorithms is more challenging. Dealing with several projects simultaneously significantly increases the size of the problems to solve. The deadlines are relatively easier or harder to meet depending on the tightness of each project, the delay penalties, etc. Additional objective functions, such as the sum of the weighted tardiness of projects (Krüger and Scholl 2009) can be considered and thus exploring the problem structure knowledge may increase the efficiency of solution techniques. In fact, for the multi-project version of the RCPSP, there are two approaches to present the links between activities (Kurtulus and Davis 1982):

1. The single-project approach, that uses two dummy activities and precedence arcs to combine the projects into a single global project. The problem is then reduced from the RCMPSP to the RCPSP and the critical paths of the projects are lost.
2. The multi-project approach, that uses $(P + 1) * 2$ dummy activities (where P is the number of projects) and where each project has its own critical path(s). Given the objective functions considered in this paper, the multi-project approach is more appropriate and is used to compute priority rules.

Even if most papers on project scheduling focus on solution methods for the single-project version, there exist some work on the multi-project version. Pritsker et al. (1969) are the first to propose a zero-one programming approach for the RCMPSP. Later, Deckro et al. (1991) explore the model of Pritsker et al. (1969) and use a project decomposition approach to solve larger multi-project problems. Similarly, Vercellis (1994) consider a Lagrangian decomposition technique to solve a multi-project planning problem.

Because of the complexity of the RCMPSP, heuristics based on priority rules are widely studied in the literature. Kurtulus and Davis (1982) experiment six new priority rules (PRs). The authors show that priority rules computed using the multi-path method (critical paths for each project) outperform priority rules computed on a single global graph. Browning and Yassine (2010) analyze the performances of 20 PRs and consider various objective functions. The authors help project managers by characterizing the best priority rule based on four problem structure measures: Objective function, network complexity, resource distribution, and resource contention. Lova et al. (2000) use an iterative Forward-Backward heuristic to solve the RCMPSP and consider two time criteria (mean project delay and sum of duration of projects), and four non-time criteria (project splitting, in-process inventory, resource leveling and idle resources). Gonçalves et al. (2008) propose a random key genetic algorithm to solve the RCMPSP. The genes decode the priority of the activities (computed using slack times), the delay of each iteration g (when a new activity is scheduled) and the release date of each project.

The Multi-Mode Resource-Constrained Multi-Project Scheduling Problem (MRCMPSP) is an extension of the multiple project scheduling problem that has attracted the attention of many researchers (Chen et al. 2022). In the MISTA 2013 challenge, various solution methods were proposed to minimize, in lexicographical order, the makespan and the total project delay. For details on the presented methods, the reader can refer to Wauters et al. (2016). Other papers tackle the distributed (or decentralized) resource-constrained multi-project scheduling problem (DRCMPSP) (Confessore et al. 2007). Multi-agent based approaches are the most popular solution methods (Li et al. 2021).

2.2 Multiple skill resource allocation

The multi-skill resource-constrained project scheduling problem (MSRCPSP) is formalized in Bellenguez and Néron (2004). Each activity requires a certain number of (human) resources mastering a given skill. Since resources have multiple skills, not only which resources allocated to each activity must be decided, but the allocated resources must also have the required skills to execute the activity. Generally, skills can be classified into two types: Categorical and hierarchical skills (Snauwaert and Vanhoucke 2023). The categorical class makes no distinction between resources; they either possess the skill or do not. The hierarchical skill class offers additional information about resource efficiency. Resources with higher hierarchical skills can, for instance, process tasks faster. Moreover, certain activities can only be executed by a resource possessing a skill level that meets a specified minimum requirement (Bellenguez and Néron 2004).

These additional decisions and constraints make the problem even harder to solve than the classical RCPSP (Polo-Mejía et al. 2021; Almeida et al. 2019) and exact methods are only considered for small instances. Among the exact approaches that can be found in the literature, Correia and Saldanha-da Gama (2014) propose a MILP model with different valid inequalities. The considered objective functions are the total costs and the makespan. Li and Womer (2009) propose a hybrid MILP/CP Benders decomposition approach to solve the MSRCPSP where activities require only one multi-skill resource. Bellenguez-Morineau and Néron (2007) propose a branch and bound method with instances having at most 32 activities, 10 resources and 5 skills. Montoya et al. (2014) propose a branch and price algorithm with an activity and time decomposition approach and the makespan as objective function.

As for the RCPSP, heuristics based on different priority rules and metaheuristics are also popular approaches to solve the MSRCPSP. Myszkowski et al. (2015) compare state-of-the art priority rules using data from a real world problem. They conclude that complex priority rules are not necessarily better than simple ones. Almeida et al. (2016) propose a parallel scheduling generation scheme (PSGS) with activity priority rules and resource weights to avoid a random resource selection. At each stage of the PSGS, a flow graph is implemented to assign resources. Javanmard et al. (2017) develop a genetic-based algorithm and a particle-swarm-based algorithm for the MSRCPSP with preemptive activities. Lin et al. (2020) implement a genetic programming algorithm (GP) used as a high-level strategy to select a sequence of 10 low-level heuristics (priority rule based heuristics).

For more details about multi-skill resource allocation problems, the reader is referred to the review of Afshar-Nadjafi (2021). The author reviews 160 articles published from 2000 to 2020 and classify the articles based on the objective functions, the mathematical formulations and the solution approaches.

2.3 Multiple skills and multiple projects scheduling problems

In the literature, few papers integrate multiple skills and multiple projects. Cui et al. (2021) consider the multi-mode and multi-skill resource-constrained multi-project scheduling problem in high-end equipment production. A variable neighborhood search metaheuristic is developed to solve the problem. As the objective function is the makespan, the problem can be reduced to a single project with additional project-based precedence constraints.

Some papers consider multiple project scheduling with multi-skilled resources but the only sequencing decisions that are made are the starting order of projects (Heimerl and Kolisch 2010; Felberbauer et al. 2019; Chen et al. 2022). Similarly, Haroune et al. (2022) study a multi-project scheduling and multi-skilled employee assignment problem with preemptive tasks. Each project is broken down into several tasks, but without explicit precedence constraints. A mixed-integer goal programming (MIGP) formulation to optimally solve small instances is implemented. A local search algorithm and a tabu search algorithm are developed to tackle large instances provided by an IT company.

To the best of our knowledge, problems with multiple projects, multi-skilled resources, explicit precedence constraints between activities and other objective functions than the makespan are not studied in the literature.

3 Problem modeling

In Sect. 3.1, the problem of heavy rolling stock maintenance is described in detail. Information about the activities and resources involved in the problem is also provided. In Sect. 3.2, the mathematical model for the MSRCMPSP is presented.

3.1 Problem description

Heavy maintenance refers to the renovation and/or modernization of trains as well as the repair of different components such as electronic cards, bogies, axles and rotors. The main challenges in this process include achieving economic savings and reducing the environmental impact of the railway industry, while also increasing passenger comfort and service quality. During heavy maintenance, the rolling stock is immobilized for several weeks. Since a rolling stock unit is very expensive, the primary objective of SNCF is to minimize the immobilization time of these units. This is highly desirable for clients as it enables the rolling stock units to be operational again as soon as possible. Additionally, by decreasing lead times, idle times for resources are minimized, maintenance costs are reduced and the maintenance process becomes more cost-effective.

The first objective, minimizing the immobilization time of rolling stock units, can be seen as a tactical objective. In fact, each year a plan is made for the next one or two years. The plan is communicated to the clients, and the due dates are fixed. In this way, the clients can build their timetable based on the availability of rolling stock units. But, at the operational level, there are many uncertainties and sometimes due to consecutive or large disturbances (e.g., new crashed rolling stock arrival) it is necessary to reschedule. In this case, respecting the due dates of customers, fixed at the tactical level, is the main objective for SNCF maintenance workshops since, to maintain the train timetable, the rolling stock units must be available on time.

When trains enter into the workshop, some technical tests and observations are performed to re-estimate the workload. Then, trains are uncoupled in several coaches. Components of coaches are dismantled and repaired in parallel. To reassemble the train, the activities of all coaches and components must be finished. Once coaches are coupled, the final activity consists of testing the train to check that safety and quality standards are respected. Figure 1 illustrates the activity precedence graph of a very simplified maintenance procedure of a rolling stock unit composed of only two coaches.

In the maintenance workshops, several rolling stock units, with different activities to process and deadlines to respect, are repaired simultaneously and compete for resources. Two kinds of renewable resources are considered: (i) Several teams having several maintenance operators with multiple categorical skills and daily variable capacities. An example of the characteristics of a team is shown in Table 1. The team has a different total capacity for each period and for each skill k_1 and k_2. The capacities of skills k_1 and k_2 are computed according to the availability and the skills of each operator. (ii) Locations in the maintenance center

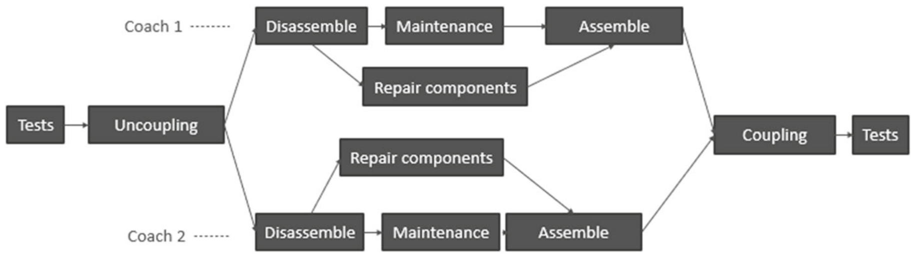


Fig. 1 Simplified activity precedence graph of rolling stock heavy maintenance

Table 1 Example of the capacities in working hours of a team with two skills k_1 and k_2

Period	Team capacity	Capacity skill k_1	Capacity skill k_2
$t = 1$	100	60	80
$t = 2$	90	50	70
$t = 3$	110	70	80

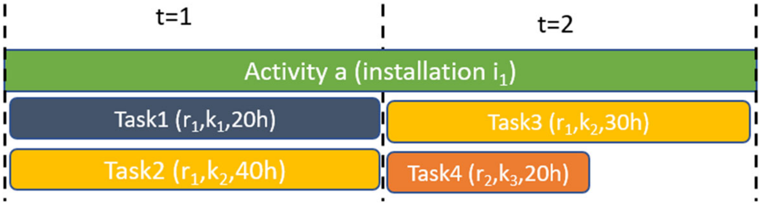


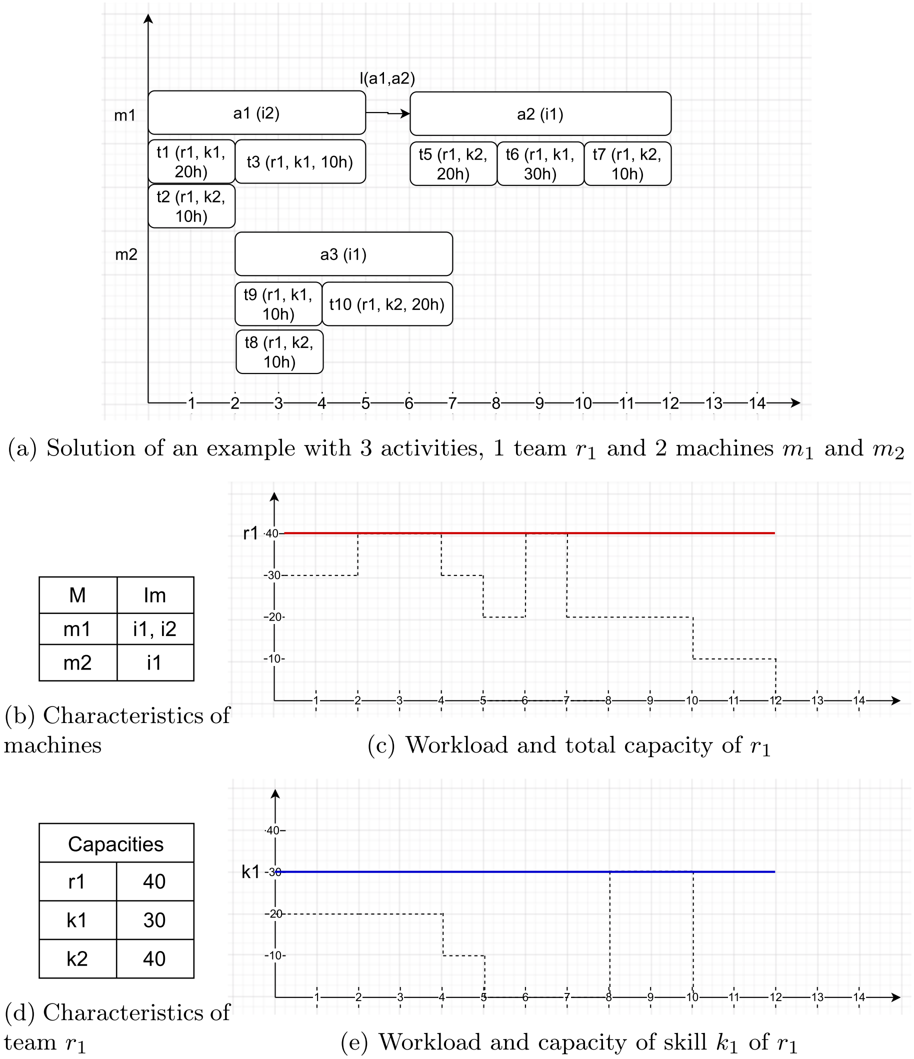
Fig. 2 Example of activity resource requirements

equipped with one or more installations (e.g. a garage pit and a roof access) but, for safety reasons, only one installation at a time can be occupied. A location can be seen as a renewable resource with multiple categorical skills (the skills being the installations at the location).

To process an activity, several resources are necessary. Each activity is composed of several sub-tasks with a fixed order. Each sub-task requires a workload of team $r \in \mathcal{R}$ with a skill $k \in \mathcal{K}$ (during the processing time of the sub-task and not the duration of the activity). To execute the set of sub-tasks, an installation is always necessary (e.g., a garage pit). An example of activity resource requirements is shown in Fig. 2:

- At period $t = 1$, activity a requires from team r_1 a total of workload of 60 h: 20 h of skill k_1, and 40 h of skill k_2. Human resources become available again at the completion of each sub-task.
- At period $t = 2$, activity a requires a workload of 30 h of skill k_2 from team r_1 and a workload of 20 h of skill k_3 from team r_2.
- An installation of type i_1 which, contrary to the human resources (teams), is necessary during the entire processing time of activity a, i.e., the processing time p_a.

Since a team has multiple operators and thus a given capacity for each period and skill, many sub-tasks can be processed in parallel (as it is the case for Task1 and Task2 in Fig. 2) as long as the capacities of the team are not exceeded. In fact, due to long-horizon scheduling (1 to 2 years) and the uncertainty on the number of operators, operators are not assigned. However, we ensure to not exceed a capacity threshold for each team $r \in \mathcal{R}$ and skill $k \in \mathcal{K}$ at each period $t \in \{1, ..., H\}$ (constraints (10) and (11)).



(a) Solution of an example with 3 activities, 1 team r_1 and 2 machines m_1 and m_2

M	Im
m1	i1, i2
m2	i1

(b) Characteristics of machines

(c) Workload and total capacity of r_1

Capacities	
r1	40
k1	30
k2	40

(d) Characteristics of team r_1

(e) Workload and capacity of skill k_1 of r_1

Fig. 3 Example of a solution of a problem with three activities

Furthermore, the workshop configuration is very unlikely to be modified on the considered scheduling horizon. Thus, we assign to each activity the location with the installation required to process this activity. Since the locations have several installations, this is a multi-skill resource assignment problem (Bellenguez and Néron (2004)).

In Fig. 3, a solution of an example with three activities, two machines m_1 and m_2 and one team r_1, is illustrated. A precedence constraint, with a lag time l_{a_1,a_2} of one time unit, exists between activity a_1 and activity a_2. Machine m_1 includes two installations i_1 and i_2, while machine m_2 contains only installation i_1 (Fig. 3b). The total capacity of r_1 and the capacities of its skills k_1 and k_2 are given in Fig. 3d. To simplify the example, we assume that the capacities remain constant over time.

In this solution, activity a_1 and activity a_2 are assigned to machine m_1 while activity a_2 is assigned to machine m_2 (Fig. 3a). Figure 3c shows the workload and the total capacity of team r_1. The workload and the capacity of skill k_1 of team r_1 is presented in Fig. 3e. Note that activity a_3 cannot start before $t = 2$, since the total capacity of team r_1 would be violated.

The goal is to find a resource feasible solution that minimizes the sum of the weighted tardiness of the projects or the sum of their weighted duration. The weights are given by the planners of SNCF (e.g. the weight of a high-speed train is usually larger than that of a regional train). The proposed mathematical model is presented in the next section.

3.2 Mathematical model

The following notations are considered to model the MSRCMPSP.

Parameters of the model:

- H, number of periods in the horizon,
- $\mathcal{E}$, set of N projects,
- dr_e, ready date of the project (engine) e,
- dd_e, due date of project e,
- w_e, weight of project e,
- $\mathcal{A}$, set of activities to schedule,
- p_a, processing time of activity a,
- $\mathcal{P}_a$, set of pairs of activities: $(a, a') \in \mathcal{P}_a$ means that activity $a \in \mathcal{A}$ must be processed before $a' \in \mathcal{A}$,
- $l_{a,a'}$, positive or negative lag time between $(a, a') \in \mathcal{P}_a$,
- $e(a)$, function that returns the project of activity a,
- $\mathcal{M}$, set of machines (locations in the workshop with one or several installations),
- $\mathcal{I}$, set of installations,
- $\mathcal{I}_m$, set of installations (skills) associated to machine m,
- $b_{a,i} = \begin{cases} 1 & \text{if installation } i \in \mathcal{I} \text{ is necessary to process activity } a, \\ 0 & \text{otherwise,} \end{cases}$
- $\mathcal{R}$, set of teams ,
- $\mathcal{K}$, set of skills,
- $W^{\overline{\mathcal{R}}}_{r,t}$, total capacity of team $r \in \mathcal{R}$ at period $t \in \{1, ..., H\}$,
- $W^{\overline{\mathcal{K}}}_{r,k,t}$, total capacity of team $r \in \mathcal{R}$ for skill $k \in \mathcal{K}$ at period $t \in \{1, ..., H\}$,
- $\alpha_{r,a} = \begin{cases} 1 & \text{if team } r \in \mathcal{R} \text{ is necessary to process activity } a \in \mathcal{A}, \\ 0 & \text{otherwise,} \end{cases}$
- $\phi_{k,a}(l)$, workload of skill $k \in \mathcal{K}$ necessary to process $a \in \mathcal{A}$ at period $l \in \{1, ..., p_a\}$.

Decision variables:

- $X_{m,a,t} = \begin{cases} 1 & \text{if } a \in \mathcal{A} \text{ start at } t \in \{1, ..., H\} \text{ and } m \in \mathcal{M} \text{ is assigned to } a, \\ 0 & \text{otherwise.} \end{cases}$
- $Y_{m,i,a} = \begin{cases} 1 & \text{if } m \in \mathcal{M} \text{ is assigned to } a \in \mathcal{A} \text{ with installation } i \in \mathcal{I}_m, \\ 0 & \text{otherwise.} \end{cases}$

To keep a similar structure as the model proposed in Pritsker et al. (1969), but also to ease the understanding of the mathematical model, the following auxiliary variables are defined.

Auxiliary variables:

- $S_{a,t} = \begin{cases} 1 & \text{if activity } a \in \mathcal{A} \text{ starts at } t \in \{1, ..., H\}, \\ 0 & \text{otherwise.} \end{cases}$

Let us note that $S_{a,t} = \sum_{m \in \mathcal{M}} X_{m,a,t}, \qquad \forall a \in \mathcal{A}, \forall t \in \{1, ..., H\}$.
Furthermore, the completion time and the tardiness of project $e \in \mathcal{E}$ are respectively defined as:

$$C_e = \max_{a \in \mathcal{A}; e(a)=e} (\sum_{t=1}^{H} t S_{a,t} + p_a)$$
$$T_e = \max(0, C_e - dd_e)$$

Using the notations introduced above and based on Pritsker et al. (1969) and Bellenguez and Néron (2004), the following MILP model is proposed.

$$Minimize \sum_{e=1}^{N} w_e T_e \tag{1}$$

or

$$Minimize \sum_{e=1}^{N} w_e C_e \tag{2}$$

subject to,

$$\sum_{t=1}^{dr_{e(a)}-1} S_{a,t} = 0 \qquad \forall a \in \mathcal{A} \tag{3}$$

$$\sum_{t=1}^{H} S_{a,t} = 1 \qquad \forall a \in \mathcal{A} \tag{4}$$

$$\sum_{t=1}^{H} t S_{a,t} + p_a + l_{a,a'} \leq \sum_{t=1}^{H} t S_{a',t} \qquad \forall (a, a') \in \mathcal{P}_a \tag{5}$$

$$\sum_{a \in \mathcal{A}} \sum_{t_1 = max(1, t - p_a)}^{t} X_{m,a,t_1} \leq 1 \qquad \forall m \in \mathcal{M}, \forall t \in \{1, ..., H\} \tag{6}$$

$$\sum_{m \in \mathcal{M}} X_{m,a,t} = S_{a,t} \qquad \forall t \in \{1, ..., H\}, \forall a \in \mathcal{A} \tag{7}$$

$$\sum_{t=1}^{H} X_{m,a,t} = \sum_{i \in \mathcal{I}_m} Y_{m,i,a} \qquad \forall m \in \mathcal{M}, \forall a \in \mathcal{A} \tag{8}$$

$$\sum_{m \in \mathcal{M}} Y_{m,i,a} = b_{a,i} \qquad \forall i \in \mathcal{I}, \forall a \in \mathcal{A} \tag{9}$$

$$\sum_{a \in \mathcal{A}} \sum_{t_1 = max(1, t - p_a)}^{t} \alpha_{r,a} \phi_{k,a} (t + 1 - t_1) S_{a,t_1} \leq W_{r,k,t}^{\overline{\mathcal{K}}}$$
$$\forall r \in \mathcal{R}, \forall k \in \mathcal{K}, \forall t \in \{1, ..., H\} \tag{10}$$

$$\sum_{k\in\mathcal{K}}\sum_{a\in\mathcal{A}}\sum_{t_1=max(1,t-p_a)}^{t} \alpha_{r,a}\phi_{k,a}(t+1-t_1)S_{a,t_1} \leq W^{\mathcal{R}}_{r,t}$$

$$\forall r \in \mathcal{R}, \forall t \in \{1, ..., H\} \quad (11)$$

$$S_{a,t} \in \{0,1\} \qquad \forall a \in \mathcal{A}, \forall t \in \{1, ..., H\} \quad (12)$$

$$X_{m,a,t} \in \{0,1\} \qquad \forall m \in \mathcal{M}, \forall a \in \mathcal{A}, \forall t \in \{1, ..., H\} \quad (13)$$

$$Y_{m,i,a} \in \{0,1\} \qquad \forall m \in \mathcal{M}, \forall i \in \mathcal{I}, \forall a \in \mathcal{A} \quad (14)$$

The objective functions (1) and (2), respectively minimize the weighted sum of the tardiness of the projects and the weighted sum of the duration of the projects. Constraints (3) ensure that activities cannot start before the project start date (note that $dr_{e(a)}$ can be replaced with the earliest starting date of each activity $a \in \mathcal{A}$). Constraints (4) are the non-preemption constraints, meaning that each activity has one and only one possible start date. Constraints (5) are the precedence constraints with minimum lag times (mainly transportation times of coaches from one location to another). An original disaggregated approach to write precedence constraints was proposed by Christofides et al. (1987) and Almeida et al. (2019):

$$\sum_{l=1}^{t+p_a-1} lS_{a,l} + p_a + l_{a,a'} \leq \sum_{l=1}^{H} lS_{a',l} \qquad \forall t \in \{1, ..., H\}, \forall (a,a') \in \mathcal{P}_a \quad (15)$$

The main advantage of the disaggregated approach is that it has a better linear relaxation and can be used to reinforce the model. Constraints (6) ensure the non-duplication of machines, i.e., that machine (location) $m \in \mathcal{M}$ is not assigned to more than one activity in a single period $t \in \{1, ..., H\}$. Constraints (7) ensure the synchronization of variables $X_{m,a,t}$ and auxiliary variables $S_{a,t}$. Constraints (8) guarantee that if machine $m \in \mathcal{M}$ is assigned to activity $a \in \mathcal{A}$, then exactly one of its installations (skills) is used. Constraints (9) ensure that the right installation is assigned to each activity a. Constraints (10) and (11) are the time-varying cumulative capacity constraints. Constraints (10) ensure that the total capacity of each skill $k \in \mathcal{K}$ of team $r \in \mathcal{R}$ is not exceeded, and constraints (11) ensure that the total capacity of each team $r \in \mathcal{R}$ is not exceeded (Table 1). Finally, constraints (12), (13) and (14) are the binary constraints of the decision variables. A similar model for the RCPSP with multiple skills (without constraints (10) and constraints (11)) is proposed in Almeida et al. (2019).

4 Solution approaches

Section 4.1 introduces two greedy algorithms and 7 priority rules. The general framework and the different components of the proposed memetic algorithm are presented in Section 4.2.

4.1 Greedy algorithms

In the literature, there are two methods to determine feasible schedules (also called constructive heuristics) for the RCPSP problem (Kolisch 1996): Serial Scheduling Generation Scheme (SSGS) and Parallel Scheduling Generation Scheme (PSGS). The SSGS performs activity-incrementation while the PSGS performs time-incrementation. In the SSGS method, at each stage g, an activity is selected among a set of eligible activities $\mathcal{E}_g$ (activities where each predecessor has already been scheduled), and scheduled at its first precedence and resource feasible time. The selection of the activity at stage g is based on one or several priority rules. The algorithm ends when all activities are scheduled, and the solution is a list with the activities in their scheduled order and with their starting times. The procedure of the algorithm is formalized in Algorithm 1.

Algorithm 1 Greedy algorithm: SSGS + Priority rule

```
for $g \leftarrow 0, 2, ..., n$ do
    Calculate the eligible set of activities $\mathcal{E}_g$;
    Select one activity $a \in \mathcal{E}_g$ using a priority rule;
    Calculate $t_0$, the earliest precedence feasible start time:
  $t_0 \leftarrow \max\{ES_a, \max_{a' \in Prec(a)} (S_{a'} + p_{a'})\}$;
    $S_a \leftarrow CheckResources(a, t_0)$;                ▷ Return the earliest resource feasible time
end for
```

In the PSGS method, at each stage g, the smallest completion time t_g among the active activities ($\mathcal{A}_{Active}$) is calculated. Then, the eligible activities at t_g in terms of precedence constraints are computed and sorted according to a priority rule. If not all eligible activities can be scheduled at t_g due to lack of resources, activities with the lowest priority are postponed to the next iteration (Algorithm 2). The algorithm ends when all activities are scheduled.

Algorithm 2 Greedy algorithm: PSGS + Priority rule

```
$t_g = 0$
while still activities to schedule do
    Calculate the eligible set of activities $\mathcal{E}_g$ at $t_g$;
    Sort $\mathcal{E}_g$ according to a priority rule;
    for $a \in \mathcal{E}_g$ do
        if $CheckResources(a, t_g) = t_g$ then
            Schedule $a$ at $t_g$;
        else
            Add $a$ to $\mathcal{E}_{g+1}$;
        end if
    end for
    $g \leftarrow g + 1$;
    $t_g \leftarrow \min\{S_a + p_a, a \in \mathcal{A}_{Active}\}$;
end while
```

Because of its relative ease of implementation, SSGS is more popular than PSGS, but the performances of each method seem to be problem related (Kolisch and Hartmann 2006). For more details about complexity, performances and priority rules used in SSGS and PSGS the reader is referred to Hartmann and Kolisch (2000) and Kolisch and Hartmann (2006). In this work, both SSGS and PSGS are adapted and implemented to solve the MSRCMPSP. The main differences compared to the classical SSGS and classical PSGS stand on the assignment of multi-skilled resources and the computation of priority rules since they depend on the tightness of each project. The procedure that allows the availability of resources to be checked so that an activity a may start at time t, is detailed in Algorithm 3. First, the resource availability for each skill $k \in \mathcal{K}$ and resource $r \in \mathcal{R}$ is checked for each period $t' \in \{t, t+1, ..., t+p_a\}$. Then, the first available machine $m \in \mathcal{M}$ is selected and assigned to activity a. Since machines have multiple skills, the selection of machines is based on the number of skills as in Almeida et al. (2019).

Seven priority rules are implemented:

- EF: The activity with the minimal earliest finish time is selected,
- ES: The activity with the minimal earliest starting time is selected,
- LF: The activity with the minimal latest finish time is selected,
- LS: The activity with the minimal latest start time is selected,
- Rand: Activities are selected in a uniformly distributed random manner,
- SA: The activity with the smallest duration is selected,
- SST: The activity with the smallest slack time is selected.

The priority rules considered in this paper are among the most common ones in the literature. They are computed using a precedence graph for each project.

Algorithm 3 Implementation of $CheckResources(a, t)$

```
for $t' \leftarrow t, t+1, ..., t+p_a$ do
    for $r_{a,k} \in \mathcal{R}_{a,t'-t}$ do    ▷ $\mathcal{R}_{a,t'-t}$ is the set of resources required by
                                                   activity $a$ at period $t'-t$ (Figure 2).
        Let $b_0$ be the remaining capacity of skill $k$ at $t'$ for resource $r$;
        Let $b_1$ be the remaining capacity of resource $r$ at $t'$;
        if $r_{a,k} > b_0$ or $r_{a,k} > b_1$ then
            $CheckResources(a, t+1)$;
        end if
    end for
end for
Let $i \in \mathcal{I}$ be the installation required by $a$;
if no machine $m \in \mathcal{M}$ with installation $i$ is available during $[t, t+p_a[$ then
    $CheckResources(a, t+1)$;
else
    Assign $m$ to activity $a$ and update remaining capacities;
end if
return t;
```

4.2 Memetic algorithm

The solutions determined by the greedy algorithms in Section 4.1 are improved using a memetic algorithm (MA). A memetic algorithm combines a Genetic Algorithm (GA) and a local search procedure which is usually applied after mutation or as a mutation operator on the new individuals (Moscato and Cotta 2003; Gonçalves et al. 2005).

In this paper, a hybridization of simulated annealing and genetic algorithm (hSGA) is combined with a conditional simulated annealing (SA) used as a local search procedure. The general framework of the proposed MA for the MSRCMPSP is presented in Algorithm 4. The algorithm starts by generating an initial population of solutions. To determine good initial solutions, and based on experimental results conducted in this research (Table 5), the SSGS with a randomized version of priority rule LS (using a geometric distribution with $p = 0.5$) as a priority rule is used. The randomization of LS ensures the diversification of the initial population. Then, while any of the stopping criteria is not met, two parents are randomly selected from the population. A one-point crossover operator is used to generate two children. Each child has a probability of P_m to mutate. The mutation operator implemented in this work randomly changes the position of an activity in the sequence by preserving the precedence feasibility. Contrary to the classical generational GA and to avoid the loss of chromosomes with high qualities, a steady-state replacement strategy (Syswerda 1991) based on a simulated annealing method is employed. The incremental replacement can lead to a low population diversity (Essafi et al. 2008) but simulated annealing helps to overcome this drawback. Each child is accepted or not according to the simulated annealing procedure. If the child is accepted, an individual among N_{worst} individuals is replaced with the accepted child. A common strategy is to replace the worst individual (Chen and Shahandashti 2009) but, in this case, the Metropolis acceptance criterion loses its sense. In fact, there is a high probability that the accepted child with a worse objective than the current population may be replaced before undergoing the crossover process. Accepting worse solutions into the population not only helps in the diversification of the population (which is crucial for GA) but also it helps to escape from a local optimum. To ensure global convergence, a cooling scheme is applied.

Algorithm 4 Memetic algorithm (MA)

```
Initialize $N_{pop}, P_c, P_m, T_{init}, C, N_{rem}, N_{iter}, GA_{iter}, N_{SA}, SA_{iter}$
Generate initial population using greedy algorithm;
while no stopping criterion satisfied do
    Randomly select two parents;
    Apply one-point crossover with probability $P_c$;
    Apply mutation with probability $P_m$;
    for each new child $c \in \{c_1, c_2\}$ do
        Check acceptance (Metropolis criterion with temperature $T_{init}$);
        if $c$ accepted then
            Insert $c$ in population;
            Randomly remove a solution from $N_{rem}$ worst solutions;
        end if
    end for
    $T_{init} \leftarrow T_{init} * C$                    ▷ Decrease cooling temperature
    $N_{iter} \leftarrow N_{iter} + 1$
    if $N_{iter} = GA_{iter}$ then
        Randomly select $N_{SA}$ individuals from the current population;
        Apply SA to each individual with $T_{init}$ as initial temperature, $C/10$ as cooling factor and $SA_{iter}$ as stopping criterion;
        $N_{iter} \leftarrow 0$
    end if
end while
return best solution;
```

The local search procedure of the proposed memetic algorithm is a SA algorithm. SA is known for its ability to escape local optima and many papers found that SA leads to very good results when solving complex scheduling problems (Bouleimen and Lecocq 2003; Yugma et al. 2012; Tamssaouet et al. 2022). The neighborhood function implemented in the SA procedure is the *Swap* activity move (Asta et al. 2016). To maintain the precedence feasibility of the sequence list, the move starts by randomly selecting an activity a_1. Then, a second activity a_2 between the positions of the latest predecessor and the earliest successor of a_1 (with $a_1 \neq a_2$) is selected. The move is accepted or not according to the Metropolis criterion. There are two main methods to apply the local search procedure in a memetic algorithm (Moscato and Cotta 2003): (i) As a mutation operator of the GA algorithm and (ii) After each iteration to all or a part of the population. Many papers underline that the quality of a population based metaheuristic results from the interplay between intensification and diversification (Sörensen and Sevaux 2006). Intensification may conduct the population to be very close to the few first improved individuals (using local search) since the mutation probability is usually small. Consequently the population loses its diversification. Diversification is very time consuming (Gonçalves et al. 2005; Sevaux and Dauzère-Pérès 2003). To overcome these drawbacks, in this work, the SA local search procedure is applied after every GA_{iter} iterations of the evolutionary process to N_{SA} random individuals of the population.

All the metaheuristics implemented in this paper work on a precedence feasible activity list. To evaluate the objective function, Algorithm 5 converts the activity list to a time feasible schedule.

Algorithm 5 Time schedule construction algorithm

```
Let $L$ be the (precedence feasible) list solution representation;
for $i \leftarrow 1, 2, ..., n$ do
    $a \leftarrow L(i)$;
    Calculate $t_0$, the earliest precedence feasible start time: $t_0 \leftarrow \max\{ES_a, \max_{a' \in Prec(a)} (S_{a'} + p_{a'})\}$;
    $S_a \leftarrow CheckResources(a, t_0)$;        ▷ returns the earliest resource feasible time
end for
```

5 Computational experiments

Section 5.1 presents the design of the computational experiments and how they were conducted. In Section 5.2, the MILP model is compared to MA on 10 small instances. Section 5.3 investigates the performances of SSGS, PSGS and the 7 priority rules. In Sect. 5.4, five solution approaches (Greedy, SA, GA, hSGA and MA) are compared on 27 large instances. The MA is evaluated on benchmark instances in Sect. 5.5. Finally, sensitivity analyses of the MA are discussed in Sect. 5.6.

5.1 Design of experiments

The performances of the proposed solution approaches are analyzed using real industrial instances of SNCF. The instances are extracted from the manufacturing execution system (MES) database of the maintenance centers and are feasible. All the numerical experiments are carried out on the two considered objective functions: Minimization of the weighted sum of the completion times of the projects and minimization of the weighted sum of the tardiness of the projects.

First, the MILP model is compared with the proposed MA on 10 small instances defined based on industrial data. Table 2 provides some properties of the small instances. The number of projects varies from 1 to 4 and the number of activities varies from 48 to 156. The serial/parallel indicator SP (defined in Vanhoucke et al. (2008) as I_2) is lower than 0.5, which means that many activities can be performed in parallel. Note that, since the instances consist of multiple projects, the SP values are computed using the global graph, which is formed by the precedence graphs of each individual project. The number of teams varies from 4 to 8 with approximately 3 skills per team. The average skill strength avgSS_k (Snauwaert and

Table 2 Characteristics of real industrial instances: $|\mathcal{E}|$ is the number of projects, $|\mathcal{A}|$ is the number of activities, SP is the serial/parallel indicator of the global project, avg$|Pred|$ is the average number of the predecessors for each activity, $|\mathcal{R}|$ is the number of global human resources (teams), avg$|\mathcal{K}_r|$ is the average number of skills per team, max$|\mathcal{K}_r|$ is the maximum number of skills per team, avgSS_k is the average skill strength of skills, $|\mathcal{M}|$ is the number of machines, avg$|\mathcal{I}_m|$ is the average number of installations (skills) per machine, max$|\mathcal{I}_m|$ is the maximum number of installations (skills) per machine

						Properties of resources						
Size	Instance	$\|\mathcal{E}\|$	$\|\mathcal{A}\|$	SP	Avg $\|Pred\|$	$\|\mathcal{R}\|$	Avg $\|\mathcal{K}_r\|$	Max $\|\mathcal{K}_r\|$	Avg SS_k	$\|\mathcal{M}\|$	Avg $\|\mathcal{I}_m\|$	Max $\|\mathcal{I}_m\|$
Small	N_1_48	1	48	0.47	1.2	7	3.0	5	0.25	38	1.3	4
	N_2_57	2	57	0.61	1.2	8	3.0	5	0.10	43	1.5	5
	N_2_96	2	96	0.23	1.2	7	3.0	5	0.12	38	1.3	4
	N_3_105	3	105	0.33	1.2	8	3.0	5	0.17	43	1.5	5
	N_3_144	3	144	0.24	1.2	7	3.0	5	0.17	38	1.3	4
	S_2_58	2	58	0.25	1.2	5	2.6	4	0.20	49	1.2	3
	S_2_98	2	98	0.17	1.2	4	3.6	5	0.16	40	1.2	2
	S_3_107	3	107	0.23	1.2	7	3.0	5	0.12	54	1.9	4
	S_3_127	3	127	0.22	1.2	7	3.0	5	0.13	54	1.9	4
	S_4_156	4	156	0.19	1.2	7	3.0	5	0.09	54	1.9	4
Large	B_21	35	3402	0.12	1.1	21	2.9	5	0.09	79	1.5	4
	B_21_22	49	5514	0.08	1.1	21	3.1	5	0.06	79	1.5	4
	B_22	14	2112	0.30	1.1	19	3.2	5	0.14	77	1.5	4
	N_21	381	5302	0.04	1.4	11	4.9	9	0.06	71	1.8	6
	N_21_1	336	4625	0.04	1.4	11	4.9	9	0.05	71	1.8	6

Table 2 continued

						Properties of resources					
N_21_2	305	4342	0.04	1.4	11	4.9	9	0.06	71	1.8	6
N_21_3	355	4976	0.04	1.4	11	4.9	9	0.06	71	1.8	6
N_22	81	1988	0.09	1.3	9	3.1	5	0.06	52	2.0	6
N_22_ot	241	3906	0.05	1.4	11	5.1	9	0.07	68	1.9	6
N_22_1	202	3350	0.06	1.4	11	5.1	9	0.07	68	1.9	6
N_22_2	195	3224	0.06	1.4	10	5.2	9	0.08	68	1.9	6
N_22_3	211	3438	0.06	1.4	10	5.2	9	0.07	68	1.9	6
N_b_21	127	1843	0.10	1.4	10	2.7	4	0.07	56	2.0	6
N_b_21_1	115	1622	0.11	1.4	10	2.7	4	0.07	56	2.0	6
N_b_21_2	101	1540	0.12	1.4	10	2.7	4	0.09	56	2.0	6
N_b_21_3	125	1823	0.10	1.4	11	3.3	6	0.09	67	1.9	6
N_b_22_ot	67	1446	0.14	1.3	11	3.7	7	0.12	63	1.8	6
S_21	106	7119	0.09	1.1	13	3.7	7	0.04	107	1.9	5
S_21_1	91	5890	0.11	1.1	13	3.7	7	0.05	107	1.9	5
S_21_2	89	6106	0.11	1.1	12	3.7	7	0.05	107	1.9	5
S_21_3	93	6614	0.10	1.1	13	3.7	7	0.05	107	1.9	5
S_22	82	4506	0.15	1.1	14	3.6	7	0.05	107	2.0	5
S_22_1	68	3564	0.18	1.1	14	3.6	7	0.07	107	2.0	5
S_22_2	69	4162	0.16	1.1	14	3.6	7	0.06	107	2.0	5
S_22_3	66	3709	0.18	1.1	14	3.6	7	0.06	107	2.0	5
S_b_21	93	6629	0.10	1.1	12	3.8	7	0.05	93	2.1	5
S_b_22	66	4089	0.16	1.1	14	3.6	7	0.07	106	2.0	5

Vanhoucke 2023) is relatively small, which indicates that all the required skills are scarce, posing a significant challenge when solving the instances. The number of machines varies from 38 to 49 and with an average number of installations per machine from 1.2 to 1.9. For the small instances, very few machines (one or two) were critical. To diversify the instances, data from two different heavy maintenance centers are used.

The MILP model is solved by the standard solver IBM ILOG CPLEX 20.1 with default parameters. The memetic algorithm is implemented in C++ and the numerical experiments were carried out on a personal computer with a 1.60 GHz processor and 16 Gb RAM. To compare the convergence efficiency over time, the computational time is limited first to 600 s and then to 3600 s. The results are summarized in Sect. 5.2 and show that MA outperforms the MILP model on both the quality of solutions and the convergence efficiency.

Then, computational experiments are conducted on 27 large instances with 14 to 380 projects and 1539 to 7119 activities. The large instances correspond to real cases and are provided from three heavy maintenance centers which repair different types of rolling stock units. Since many projects are considered simultaneously, a large number of activities can be executed in parallel. Consequently, the SP values of the large instances are even smaller in comparison to the SP values of the small instances. The number of teams varies from 9 to 21 with an average number of skills per team between 2.9 and 5.2. The average skill strength is very small (lower than 0.14) because, at the tactical level, the resources are sized to reduce the idle times. The number of machines varies from 52 to 107 with an average number of installations per machine between 1.5 and 2.1. Additional properties of the instances can be found in Table 2.

In Sect. 5.3, SSGS is compared to PSGS with the 7 priority rules described in Sect. 4.1. The results show that SSGS outperforms PSGS and that LS is the best priority rule for both objective functions. Five algorithms are compared in Sect. 5.4: Greedy, SA, GA, hSGA and MA. The algorithms are implemented in C++ and empirically parameterized in the following manner:

- Greedy: Refers to SSGS+LS,
- SA: Simulated Annealing algorithm with the following parameters:
  - Initial temperature: $T_{init} = 24$ for SWTP and $T_{init} = 48$ for SWDP,
  - Cooling factor: $C = 0.99999$,
- GA: Genetic Algorithm with the following parameters:
  - Population size: $N_{pop} = 120$,
  - One-point crossover probability: $P_c = 0.95$,
  - Mutation probability: $P_m = 0.75$,
  - Linear ranking selection (Hartmann 1998) method is used for selecting the next generation,
- hSGA: hybrid Simulated Genetic Algorithm with the same parameters as above and $N_{rem} = 40$,
- MA: Memetic Algorithm with the same parameters as hSGA and $GA_{iter} = 2500$ iterations, $N_{SA} = 4$ and $SA_{iter} = 2000$.

The parameter tuning is done step by step. First the SA parameters are tuned as in Knopp et al. (2017). By computing the first 100 moves which deteriorates the solution, the 2% percentile on some training instances was calculated. We found that, in most of the cases, the 2% percentile is approximately 24 for the SWTP objective and approximately 48 for the SWDP objective. Note that 24 correspond to 24 hours, which is the duration of most of the activities. Secondly, by empirically testing different values (as in Hartmann (1998)) of the population size, mutation and crossover probabilities, the parameters of the GA were set. The mutation probability (0.75) is notably larger than what is commonly found in papers of the literature that use GA for solving the RCPSP. However, it seems that, for complex and large problems, it is usually better to have a high mutation probability (Elloumi and Fortemps 2010; Murata et al. 1996). For the hGSA, the same parameters of SA and GA are used but; to keep a diversified population and to exploit the advantages of the simulated annealing approach, several values of N_{rem} (10, 20, 30, 40, 50) were tested. The best results were obtained for $N_{rem} = 40$. Finally, for the MA, the value of GA_{iter} is the number of iterations required for the GA algorithm to converge towards a good solution. After reaching approximately 2 500 iterations (equivalent to 5 000 schedules), the convergence of the GA significantly decelerates. Adjusting parameters N_{SA} and SA_{iter} is challenging, because large values may lead to premature convergence (Sörensen and Sevaux 2006). Consequently, small values are preferred and have been tested by trying different values (2, 4, 6, 8, 10 for N_{SA} and 2000, 4000, 6000, 8000, 10000 for SA_{iter}).

The computational time limits were chosen to evaluate the quality of the approaches in a limited amount of time (600 s) and if longer computational times are allowed (1800 and 3600 s). The requirements in terms of computational times from the planners of SNCF depend on how the optimization is used (for generating an initial complete schedule or to perform simulations by varying input parameters).

5.2 Comparison of MILP and MA

In this section, the performances of the MILP model and the proposed MA are compared. Table 3 shows the numerical results and the percentage improvement PI (in %) after 300 s of computational time, while Table 4 shows the results obtained after 3600 s of computational time.

PI is defined as follows:

$$PI(\%) = \frac{MILP_{Obj} - MA_{Obj}}{MILP_{Obj}} * 100,$$

where MA_{Obj} is the objective value of the best solution determined by MA and $MILP_{Obj}$ is the objective value found by CPLEX. Column LB is the lower bound found by CPLEX after 3600 s, column Gap is the gap between LB and the objective value of the MILP, while Gap$_{MA}$ is the gap between LB and the solution found by MA. The cells filled with "–" indicate that no feasible solution is found by CPLEX, while the cells filled with "*" indicate that the solution found by CPLEX is optimal.

Regarding the SWDP objective function, Table 3 shows a clear dominance of MA, which obtains the best solution for 8 instances out of 10 and equivalent solutions for the 2 remaining instances. The improvement becomes particularly significant when the instances have more than 90 activities. CPLEX obtained an optimal solution in less than 300 s for only 2 instances. For Instance N_3_144, the solver could not find a feasible solution. For Instances N_3_144 and S_4_156, Gap$_{MA}$ is relatively large (6% and 7.5%), but as CPLEX struggles to find good solutions, it is likely that the lower bound LB is of poor quality.

Regarding the SWTP objective function, MA found the best solution for 8 instances out of 10 and equivalent solution for Instance S_2_58 with a large PI (>78%) for instances with more than 90 activities. For Instance N_2_57, the solver outperforms the proposed MA. Instance N_2_57 has the smallest avgSS_k and also the average number of skills per machine

Table 3 MILP vs. MA—Numerical results and percentage of improvement with CPU time limited to 300s

	SWDP						SWTP					
Instance	LB	MILP	Gap	MA	Gap$_{MA}$	PI (%)	LB	MILP	Gap	MA	Gap$_{MA}$	PI (%)
N_1_48	621	**621***	0.0	**621**	0.0	0	21	57	63.2	**21**	0.0	63.2
N_2_57	1110	1206	8.0	**1110**	0.0	8	105	**105***	0.0	117	10.3	−11.4
S_2_58	1088	**1088***	0.0	**1088**	0.0	0	70	**70***	0.0	**70**	0.0	0.0
N_2_96	1648	3678	55.2	**1710**	3.6	53.5	164	–	–	**222**	26.1	–
S_2_98	1027	1456	29.5	**1052**	2.4	27.7	45	2696	98.3	**56**	19.6	97.9
N_3_105	2154	5307	59.4	**2163**	0.4	59.2	15	1170	98.7	**21**	28.6	98.2
S_3_107	1592	3156	49.6	**1608**	1.0	49	143	864	83.4	**188**	23.9	78.2
S_3_127	1588	2418	34.3	**1614**	1.6	33.3	31	3738	99.2	**44**	29.5	98.8
N_3_144	3838	–	–	**4083**	6.0	–	13	–	–	**33**	60.6	-
S_4_156	2001	4132	51.6	**2164**	7.5	47.6	165	–	–	**198**	16.7	-

Table 4 MILP vs. MA—Objective functions and percentage of improvement with CPU time limited to 3600 s

	SWDP						SWTP					
Instance	LB	MILP	Gap	MA	Gap_{MA}	PI (%)	LB	MILP	Gap	MA	Gap_{MA}	PI (%)
N_1_48	621	**621***	0.0	**621**	0.0	0	21	**21***	0.0	**21**	0.0	0.0
N_2_57	1110	**1110***	0.0	**1110**	0.0	0	105	**105***	0.0	**105**	0.0	0.0
S_2_58	1088	**1088***	0.0	**1088**	0.0	0	70	**70***	0.0	**70**	0.0	0.0
N_2_96	1648	2922	43.6	**1698**	2.9	41.9	168	1458	88.5	**198**	15.2	86.4
S_2_98	1027	1264	18.8	**1064**	3.5	15.8	52	164	68.3	**56**	7.1	65.9
N_3_105	2154	2991	28.0	**2163**	0.4	27.7	15	585	97.4	**21**	28.6	96.4
S_3_107	1592	1896	16.0	**1596**	0.3	15.8	143	552	74.1	**152**	5.9	72.5
S_3_127	1588	2102	24.5	**1602**	0.9	23.8	31	344	91.0	**44**	29.5	87.2
N_3_144	3838	–	–	**3927**	2.3	–	13	–	–	**18**	27.8	–
S_4_156	2001	3272	38.8	**2116**	5.4	35.3	165	984	83.2	**186**	11.3	81.1

is larger than in other instances with less than 60 activities, which makes it harder to solve to optimality.

However, after 3600 s of computational time, MA obtains the optimal solution for Instance N_2_57. All instances with less than 60 activities were optimally solved by both CPLEX and MA. For the other instances, MA outperforms CPLEX with a PI larger than 15% for the SWDP objective function and larger than 65.9% for the SWTP objective function. For Instance N_3_144, the solver still could not find a feasible solution while MA reduces its gap from the LB from 6% to 2.3% after 3600 s, which indicates the capability of MA to address hard instances. Concerning the SWTP objective function, large gaps between the solution of MA and the lower bound for the Instances S_3_127, N_3_105 and N_3_144 (respectively 29.5%, 28.6% and 27.8%) can be observed. This is likely due to the small values of the objective function and the poor quality of the lower bounds.

Tables 3 and 4 illustrate the superiority of the proposed MA over CPLEX for the MSRCMPSP. The percentage improvement increases with the size of the instances. The performances of CPLEX could be improved by better tuning the parameters or applying a warm start for example. However, as the real instances include several thousand activities, CPLEX is not adapted to solve the industrial instances. Memory limitations actually prevented us from obtaining even a lower bound for the large real instances.

5.3 Performance of greedy algorithms

In this section, we investigate the performances of SSGS and PSGS tested with the 7 priority rules detailed in Sect. 4.1. Table 5 summarizes the numerical results obtained using the 27 large instances (the detailed results for each instance can be found in Table 11 of Appendix A). Column "#best" gives the number of instances where the corresponding priority rule found the best solution. Column "avg gap (%)" shows the average gap (in %) from the best solution and "max gap (%)" is the maximal gap among the 27 instances. The computational times are not given since both SSGS and PSGS need less than 1 (approximately 5ms) to compute a solution.

The first observation is that the serial scheduling scheme is superior to the parallel scheduling scheme. Column "#best" of Table 5 shows that SSGS found a better solution than PSGS for the 27 considered instances. The average gap of PSGS is at least 47% for the SWDP objective and at least 90% for the SWTP objective. Kolisch and Hartmann (2006) and Hartmann and Kolisch (2000) concluded that SSGS is better than PSGS for complex problems. In this paper, this significant gap can be explained by the large size of the problem. Having several projects with different characteristics and deadlines makes the problem very complex to solve. The gap is even larger for the SWTP objective function since scheduling in parallel is very myopic regarding the deadline tightness of projects.

Among the 7 considered priority rules, the best one is LS with 22 best solutions found out of 27 for both objective functions. The second best priority rule is LF which found 4 best solutions out of 27 instances for SWDP and 5 best solutions out of 27 instances for SWTP. LS is also the most robust priority rule since LS has the smallest average gap and the smallest maximum gap for both objective functions. Hence, SSGS+LS is used to generate the initial solution of the different metaheuristics. When a population of initial solutions is needed, a randomized version of LS is used as a priority rule to generate the individuals. The numerical results obtained by the metaheuristics are discussed in the next section.

Table 5 SSGS vs. PSGS: Performance of priority rules and associated gap from the best solution found by SSGS or PSGS

		SWDP			SWTP		
Algorithm	PR	#best	Avg gap (%)	max gap(%)	#best	Avg gap (%)	Max gap (%)
SSGS	EF	0	8.9	13.0	0	54.4	83.3
	ES	1	4.0	7.9	0	40.0	78.7
	LF	4	3.9	9.1	5	22.2	60.2
	LS	**22**	**0.4**	**4.8**	**22**	**0.9**	**8.3**
	Rand	0	20.3	31.4	0	78.8	94.3
	SA	0	36.7	58.9	0	88.2	95.2
	SST	0	17.2	26.2	0	71.9	91.4
PSGS	EF	0	47.3	57.3	0	90.7	98.1
	ES	0	47.7	61.6	0	90.7	98.5
	LF	0	47.5	61.5	0	90.3	98.1
	LS	0	46.3	56.9	0	90.1	98.5
	Rand	0	57.1	72.7	0	93.9	99.2
	SA	0	56.9	76.0	0	93.8	99.3
	SST	0	58.7	80.5	0	93.3	99.2

5.4 Comparison of heuristics

Five algorithms are compared in this section: Greedy, SA, GA, hSGA and MA. The results are presented after 600, 1800 and 3600 s for both considered objective functions. The detailed results for each large instance are presented in the 6 tables of Appendix B which have a similar structure (Tables 12, 13, 14, 15, 16, and 17). The tables show the objective value for each algorithm and the associated gap (column "%Gap") from the best solution highlighted in bold.

A summary of the results can be found in Table 6 for the SWDP objective and in Table 8 for the SWTP objective. Column "#best" provides the number of best solutions found by the corresponding algorithm over the 27 large instances. To analyze the efficiency and the robustness of each algorithm, the average, the standard deviation and the maximal gap from the best solution are also provided.

Let us first analyze the results obtained for the SWDP objective. When the computational time is limited to 600 s, from Tables 12 and 6, note that, out of 27 instances MA finds the best solution for 19 instances and hSGA finds the best solution for 9 instances. No best solution is determined by the other algorithms. Consequently, MA and hSGA outperform the Greedy algorithm, GA and SA. MA is slightly better than hSGA since the average gap for MA is equal to 0.1% while the average gap of hSGA is around 0.4%. When compared to the greedy algorithm which is used to generate the initial solutions, MA improves the solutions by around 8%. The maximal gap of the Greedy algorithm is 15.6%. Even if SSGS+LS (i.e., the Greedy algorithm) seems to have good results in Sect. 5.3 compared to PSGS and the other priority rules, still a significant improvement is obtained with the proposed MA.

Regarding the maximal gap, MA is also robust because its worst gap is around 0.8%. The worst gap is obtained for Instance S_21 which has the larger number of activities. In the evolutionary process, many decisions are made randomly. Thus, for large instances it

Table 6 Summary of results on the performances of the metaheuristics for objective function SWDP

CPU	Algorithm	#best	Avg gap (%)	Std gap (%)	Max gap (%)
600	SSGS	0	8.1	2.9	15.6
	SA	0	1.7	0.8	3.5
	GA	0	4.0	1.5	6.8
	hSGA	9	0.4	0.4	1.5
	MA	**19**	**0.1**	**0.2**	**0.8**
1800	SSGS	0	8.6	2.9	16.5
	SA	1	1.4	0.8	2.8
	GA	0	4.0	1.4	6.5
	hSGA	8	0.4	0.4	1.4
	MA	**18**	**0.1**	**0.3**	**1.1**
3600	SSGS	0	8.8	3.0	16.6
	SA	1	1.2	0.8	2.8
	GA	0	4.0	1.5	6.6
	hSGA	8	0.4	0.4	1.7
	MA	**18**	**0.1**	**0.2**	**1.0**

is more difficult to control the convergence. Since GA has worse results compared to SA (with respectively an average gap of 1.7% and 4.0% in Table 6), the fact that the simulated annealing procedure helps to escape local optimum is very important when dealing with large instances and complex problems. Similar conclusions were drawn in Tamssaouet et al. (2022).

Furthermore, after 1800 s of computational time, SA closes the gap from the best solutions. Table 6 shows that SA has an average gap of 1.7% and a maximal gap of 3.5% after 600 s but that after 1800 s the average gap reduces to 1.4% and the maximum gap to 2.8%. SA even finds the best solution for Instance N_b_21_3 (Table 13). However, MA remains the best metaheuristic with 18 best solutions out of 27, followed by hSGA with 8 best solutions. The gap of the Greedy algorithm increases (from 8.1% to 8.6%), meaning that the metaheuristics can still improve the solution after 600 s. In particular, the gap difference between the greedy Algorithm and MA increases from 8% to 8.5%.

Similar observations can be derived when the computational time is limited to 3600 s. SA still closes the gap (from 1.4% to 1.2%) and the number of best solutions found is the same as after 1800 s. However, the best solutions determined by MA, hSGA and SA are not for the same instances as in Table 13. In Table 13, the best solution found by SA was for Instance N_b_21_3 but, in Table 14 the best solution found by SA is for Instance N_22_3. Another difference can be observed for Instances N_b_22_ot and N_b_21_1 where MA overcomes hSGA. However, for Instances S_21_2 and S_22_3, hSGA overcome MA. The gap of the Greedy algorithm still increases from 8.6% after 1800 s to 8.8% after 3600 s. Details on the percentage of improvement (PI) over time are illustrated in Table 7. These results confirm the complexity of the problem and that finding the optimal solution is difficult.

Considering the results presented so far, MA is the most appropriate approach to solve the MSRCMPSP with SWDP as objective function. SA can close the gap over time, which makes SA a very good candidate for the local search procedure of a memetic algorithm.

Table 7 Convergence performances of the metaheuristics: Mean of the PI (in %) of the initial solutions provided by SSGS

	SWDP				SWTP			
CPU	SA	GA	hSGA	MA	SA	GA	hSGA	MA
600	6.5	4.3	7.7	8.0	43.1	44.4	49.2	50.0
1800	7.2	4.8	8.2	8.5	45.3	45.6	50.3	50.9
3600	7.7	5.0	8.5	8.7	46.5	46.0	51.0	51.4

Table 8 Summary of results on the performances of the metaheuristics for objective function SWTP

CPU	Algorithm	#best	Avg gap (%)	Std gap (%)	Max gap (%)
600	SSGS	0	51.0	13.3	73.5
	SA	0	14.6	8.1	29.1
	GA	1	12.4	9.6	33.3
	hSGA	14	3.7	5.4	21.2
	MA	**14**	**2.3**	**3.5**	**11.6**
1800	SSGS	0	52.1	13.2	73.5
	SA	0	13.0	7.7	28.7
	GA	2	12.3	9.8	33.4
	hSGA	13	3.8	5.3	18.8
	MA	**14**	**2.5**	**3.6**	**11.1**
3600	SSGS	0	52.6	13.2	73.5
	SA	0	11.9	6.9	27.0
	GA	0	12.5	9.6	31.9
	hSGA	14	3.5	5.0	19.2
	MA	**15**	**2.5**	**3.4**	**8.8**

The same analysis is conducted for the SWTP objective function. Table 8 presents the summary of the results for the 27 instances.

After 600 s of computational time, MA and hSGA both find the best solution for 14 instances. One best solution was found by GA for Instance S_22_2 but the gap with the solution found by MA is only 0.9%, which represents less than one day of total delay for 69 projects (Table 15). MA remains the best approach with an average gap of 2.3%, while the second best approach is hSGA with an average gap of 3.7%. MA also has the best maximal gap (11.6%) and the best standard deviation gap (3.5%), confirming the robustness of MA compared to the other approaches. In particular, the average gap of the Greedy algorithm is 51%, which implies that the tardiness is approximately reduced by a factor of 2. SA and GA have good results as well with an average gap smaller than 15% but their efficiency is not stable since the maximum gap is respectively 33.3% for SA and 29.3% for GA. Once again, this instability can be justified by the complexity of the problem with many projects, activities, teams, machines, skills and constraints. When neighborhood exploration is too random, it may be difficult for the metaheuristics to select promising moves. Table 5 in Section 5.3 shows significant gaps as well, i.e., SSGS and PSGS determine solutions that are far from an optimal solution.

When the computational time is limited to 1800 s, several observations can be made (Table 16):

- hSGA finds 13 best solutions whereas MA still finds 14 best solutions out of 27 and GA determines one additional best solution, i.e., 2 best solutions.
- As for the SWDP objective, hSGA converges better than MA for some instances towards the best solution after 1800 s (for example Instance S_21_3). On the contrary, MA overcomes hSGA for Instance N_21_2.
- SA closes the gap from 14.6% after 600 s to 13% after 1800 s. The other metaheuristics do not considerably improve the solutions since the gap of the Greedy algorithm deteriorates by only 1%. MA and hSGA struggle to improve the solutions (Table 7) and consequently, their average gap slightly deteriorates (from 2.3% to 2.5% for MA and from 3.7% to 3.8% for hSGA).

However, MA improves the maximal gap, in particular when the computational time is limited to 3600 s. The maximal gap of MA decreases from 11.6% after 600 s to 8.8% after 3600 s, which confirms its robustness. SA, GA and hSGA still have significant maximal gaps (respectively 27%, 31.9% and 19.2%). hSGA overcomes GA for Instances S_22_2 and S_21_2 for which GA finds a better solution after 1800 s (Table 16). As for the SWDP objective, accepting worse solutions using the Metropolis criterion allows local optima for the SWTP objective to be escaped.

In view of the results, the intensification in the memetic algorithm, ensured by the local search procedure, increases the average quality of solutions and the robustness of the proposed solution approach. Hence, MA is the best approach to solve the MSRCMPSP. However, in terms of best solutions, hSGA and MA are quite equivalent when SWTP is considered as objective function. These results illustrate that there is still room for improvement and in particular in the intensification mechanism ensured by SA, as SA consistently improves its average gap (for example from 13.0% after 1800 s to 11.9% after 3600 s). An interesting perspective would be to better choose the visited neighborhood. The evaluation of moves is very time-consuming, by discarding uninteresting moves considerable computational time can be saved (Dauzère-Pérès and Paulli 1997; Mati et al. 2011). Hence, with the same time allocated to the intensification part, a more promising neighborhood could be explored. This would probably improve the convergence and the results of the proposed memetic algorithm and the results of SA as well.

5.5 Comparison on benchmark datasets

To the best of our knowledge a benchmark dataset with both multiple projects and multiple skills does not exist in the literature. Since our objective functions are only meaningful in the context of multiple projects, we conducted experiments on the instances of the Multi-Project Scheduling Problem library (MPSPLib: http://www.mpsplib.com/, July 2023). The library contains 20 datasets with a total of 140 instances built by combining several RCPSP instances of Kolisch and Sprecher (1997). The number of projects varies from 2 to 20 and the number of activities varies from 60 to 2 400. For more details about the characteristics of these datasets, the reader is referred to Wauters et al. (2015). The memetic algorithm is compared with the sequential learning-based metaheuristic of Wauters et al. (2015), which is reported to be one of the best (decentralized) solution approaches of the literature (Bredael and Vanhoucke 2022). Their solution approach consists of a sequence learning game played by several project managers. Each manager learns both a local activity list (using reinforcement learning) and a global activity list containing all the activities of all projects. The global activity list is build by adding all the activities of the first project, then the activities of the second project, and so on. Finally, the global list is transformed into a feasible schedule by using the classical serial generation scheme. To ensure a fair comparison, the same stopping criterion (i.e. 100 000 schedules) is used. After preliminary experiments and since the objective values are smaller than the objective values of the real instances, an initial temperature $T_{init} = 2.5$ was set. The other parameters of the proposed memetic algorithm remain unchanged.

Table 9 presents the results of the first configuration (ADP1, Time1 in seconds, and Gap1 in %). The proposed memetic algorithm shows good performance when the average project delay (ADP) is small (less than 30). For example, better solutions are found by MA for Instances MP120_20, MP30_2, MP30_5, MP90_2 and MP90_5. On the contrary, when the ADP is large, MA does not perform well.

The problem subsets with large ADP have an significant resource Average Utilization Factor (AUF) (Kurtulus and Davis 1982 and the MPSPLib web site). In this case, and as it is underlined in Asta et al. (2016) and Bredael and Vanhoucke (2022), the prioritization of the projects is very important. At the SNCF maintenance centers, the arrival times of the rolling stock (i.e. the starting time of the projects) are decided at a tactical level to avoid overloading the available resources. Hence, the prioritization of the projects was not really necessary. However, we have implemented several neighborhood operators and, among them, a novel one which handles the project priorities (more precisely the priorities of the different phases of a project). The novel neighborhood operator (*SortActivities*) consists of selecting a sublist from the global activity list, with a random size uniformly distributed between 3 and $|\mathcal{A}|/2$. The activities of the sublist are then sorted by project delay while conserving the initial order of the activities of a same project. For example, if the following sublist is selected: $[1(1), 2(2), 3(2), 4(1)]$ (where $a(n)$ means that the activity a of project n is in the sublist), and project 1 has a larger delay compared to project 2, after applying the *SortActivities* neighborhood function, the sublist becomes: $[1(1), 4(1), 2(2), 3(2)]$. The *SortActivities* neighborhood function is added as a mutation operator in the memetic algorithm of Sect. 4.2. The memetic algorithm has now two mutation operators with the same probability of being selected. The original one is kept to help the exploration of additional solutions by the *SortActivities* function. This is the second configuration of our proposed memetic algorithm and the results (ADP2, Time2 (in seconds), and Gap2 (in %)) are detailed in Table 9. The gap is considerably reduced for the instances with large ADP values. For Problem subset MP120_20AC, Gap1 was −108% and the second configuration reduces the gap to −4.4%. Generally speaking, the average gap over all instances is reduced from −32.2% to 6.4%. The memetic algorithm with the second configuration obtains better results for 15 datasets out of 20 compared to the learning approach of Wauters et al. (2015), leading to an average improvement over all instances of 6.4%. In terms of computational times, the proposed memetic algorithm is on average at least 3 times faster for the first configuration and 4 times faster with the second configuration (note that, when the solution quality is poor, Algorithm 3 requires more time, which explains the difference of computational times between Time1 and Time2).

Let us note that, for the instances with small ADP values, the second configuration still finds better solutions compared to the learning approach of Wauters et al. (2015), but does worse than the first configuration (for Instances MP30_2, MP90_2, MP90_5 and MP30_5). In view of the results, the *SortActivities* mutation operator should be used when the problem has a large AUF. That is why the *SortActivities* mutation operator is not useful to solve the real instances of SNCF.

The column GapBest gives the best gap between Gap1 and Gap2. If the appropriate configuration of the memetic algorithm is chosen, an overall average improvement of the ADP of 7.7% can be achieved. Even if our algorithm was not initially built to solve the

Table 9 Comparison of average project delay (ADP) with Wauters et al. (2015)

	Wauters et al. (2015)		This paper						
Problem subset	APD	Time	APD1	Time1	Gap1	APD2	Time2	Gap2	GapBest
P30_2	11.2	21.4	10.0	29.1	10.7	10.9	13.3	2.7	10.7
MP30_5	15.4	79.7	13.5	71.6	12.5	14.2	43.2	7.5	12.5
MP90_2AC	104.3	144.3	124.5	46.2	−19.4	98.3	21.5	5.7	5.7
MP90_2	5.3	83.4	4.6	41.8	13.2	5.1	22.6	3.8	13.2
MP120_2	49.4	184.2	54.7	71.9	−10.7	45.3	35.1	8.3	8.3
MP120_2AC	35.2	154.2	41.1	53.3	−16.8	30.1	27.2	14.3	14.3
MP30_10	52.0	250.8	66.5	280.6	−27.9	50.9	159.1	2.1	2.1
MP90_5	7.8	297.2	4.8	143.0	37.9	5.2	97.1	33.3	37.9
MP90_5AC	244.6	553.1	391.0	169.5	−59.8	237.8	63.6	2.8	2.8
MP120_5	48.5	673.6	56.4	349.2	−16.2	42.7	187.4	12.0	12.0
MP30_20	111.4	868.4	161.8	1390.2	−45.2	112.1	912.6	−0.7	−0.7
MP120_5AC	178.8	871.1	295.5	215.0	−65.3	177.4	81.9	0.8	0.8
MP90_10	31.8	1448.6	39.1	920.7	−23.0	29.6	528.8	6.9	6.9
MP90_10AC	169.4	1414.5	289.1	360.0	−70.7	173.3	133.3	−2.3	−2.3
MP120_10AC	96.9	2618.5	193.8	483.5	−100.0	98.3	192.4	−1.5	−1.5
MP120_10	100.0	2110.6	133.1	1661.5	−33.1	97.9	789.6	2.1	2.1
MP90_20	17.6	3305.2	26.5	1729.3	−50.8	14.7	1731.3	16.5	16.5
MP90_20AC	85.4	4252.1	155.1	380.2	−81.7	87.6	214.2	−2.6	−2.6
MP120_20AC	158.6	8586.0	329.9	1113.7	−108.0	165.7	355.5	−4.4	−4.4
MP120_20	28.2	6946.4	25.1	1917.0	11.2	22.6	2696.0	19.9	19.9
Mean	77.6	1743.2	120.8	571.4	−32.2	76.0	415.3	6.4	7.7

Table 10 Sensitivity of MA: Impact of input parameters with CPU time limited to 600 s

		SWDP					SWTP		
N_{pop}	T_{init}	N_{rem}	P_m	Avg gap (%)	N_{pop}	T_{init}	N_{rem}	P_m	Avg gap (%)
60	48	40	0.75	1.3	60	24	40	0.75	14.1
80	48	40	0.75	0.2	80	24	40	0.75	9.7
120	48	40	0.75	0.2	120	24	40	0.75	2.8
180	48	40	0.75	0.3	180	24	40	0.75	2.7
240	48	40	0.75	0.8	240	24	40	0.75	2.9
120	12	40	0.75	1.6	120	12	40	0.75	3.1
120	24	40	0.75	0.5	120	24	40	0.75	2.8
120	36	40	0.75	0.4	120	36	40	0.75	5.8
120	48	40	0.75	0.2	120	48	40	0.75	6.3
120	60	40	0.75	0.4	120	60	40	0.75	11.3
120	48	10	0.75	2.7	120	24	10	0.75	15.3
120	48	20	0.75	1.8	120	24	20	0.75	12.8
120	48	30	0.75	0.7	120	24	30	0.75	8.4
120	48	40	0.75	0.2	120	24	40	0.75	2.8
120	48	50	0.75	0.3	120	24	50	0.75	2.1
120	48	40	0.05	3.1	120	24	40	0.05	9.5
120	48	40	0.1	2.8	120	24	40	0.1	7.3
120	48	40	0.5	1.3	120	24	40	0.5	4.6
120	48	40	0.75	0.2	120	24	40	0.75	2.8
120	48	40	0.85	0.3	120	24	40	0.85	2.0

MSPSLib instances, it shows very promising results and could be generalized to solve other related problems.

5.6 Sensitivity analyses

An efficient hybrid metaheuristic requires the configuration of many parameters (Pellerin et al. 2020). Table 10 reports the average gap from the best solution obtained by the different configurations of MA. The computational experiments are conducted on the real instances of SNCF and MA is stopped after 600 s. In this sensitivity analysis, for the sake of brevity and, since they have been identified as the most influential factors during preliminary experiments, we exclusively focus on the parameters N_{pop}, T_{init}, N_{rem} and P_m.

In each row of Table 10, only a single parameter is changed from the initial configuration of MA. Regarding the SWPD objective function, the largest average gaps are obtained for small values of P_m and N_{rem} (resp. 3.1% and 2.7%), which are both useful for keeping a diversified population. Similar conclusions can be drawn when the objective function is SWTP.

Regarding the population size, note that, for the SWTP objective function, a larger value (180) is more suitable while, for the SWDP objective function, a lower population (between 80 and 120) performs better. The primary reason could be that numerous individuals yield the same objective values for the SWTP objective function. In fact, a neighborhood move can lead to a different solution but it may not impact the tardiness of the projects and hence, a larger population is necessary to prevent the population to be homogeneous (Van Peteghem and Vanhoucke 2010). This observation could also indicate why the suggested MA demonstrates improved performance with larger values of P_m and N_{rem} for the SWTP objective function (resp. 0.85 and 50) compared to the SWDP objective function (resp. 0.75 and 40).

This analysis shows the importance of a proper parameter tuning for the success of a (hybrid) metaheuristic. Except for the initial temperature, we chose a uniform configuration for both of the considered objective functions to facilitate the use of the algorithm.

6 Conclusions

This paper studies an original industrial RCPSP problem with multiple skills and multiple projects, which corresponds to a real case in the heavy maintenance centers of railway rolling stock. Two objective functions are considered: (i) Minimization of the sum of the weighted tardiness of the projects and (ii) Minimization the sum of the weighted duration of the projects. A time-indexed MILP formulation of the problem is provided, and two constructive heuristics (serial and parallel scheduling generation schemes) are implemented and tested with 7 priority rules on 27 large real instances. The solutions provided by SSGS+LS are used as initial solutions and are improved using four metaheuristics: SA, GA, hSGA and MA. Several computational experiments were conducted in this study. The first experiment compared the MILP model and the proposed memetic algorithm (MA) on 10 small instances defined using industrial data. The results showed that MA outperforms the MILP model after 300 and 3600 s of computational time, in particular for instances with more than 90 activities. Larger gaps were observed when the objective function was to minimize the sum of the weighted tardiness of the projects (SWTP). In the second experiment, the performance of 7 priority rules implemented in the serial scheduling generation scheme (SSGS) and the parallel scheduling generation scheme (PSGS) were evaluated using 27 large instances. SSGS found the best solution for the 27 instances, while none were found by PSGS. Among the priority rules, the Latest Start (LS) rule had the best performance, with 22 best solutions found out of 27 for both objective functions.

Finally, the performance of the memetic algorithm was compared to SSGS+LS, simulated annealing (SA), genetic algorithm (GA), and hybrid simulated annealing and genetic algorithm (hSGA) with computational times limited to 600, 1800, and 3600 s. MA has the best average gap and the smallest maximal gap, indicating a high level of stability. In particular, for the SWDP objective function, an average improvement of 8% of the initial solutions provided by the Greedy algorithm was observed. For the SWTP objective function, the total delay was reduced by a factor of 2. Furthermore, an overall average improvement of the ADP of 7.7% was achieved for the benchmark instances. In view of the results, the memetic algorithm is being integrated into the MES of SNCF maintenance centers. The scheduling horizon is decided by the planners, and all the activities starting beyond the starting date of the scheduling horizon are (re)scheduled.

Despite the promising results obtained by the memetic algorithm, several areas for improvement can be identified. One potential direction for future research is to conduct a more advanced sensitivity analysis of the proposed algorithm to optimize its configuration. The proposed approach requires the initialization of numerous parameters, and characterizing the most robust configuration can be a challenging task. A self-adapting component would be of great value for MA to solve other SNCF instances in the future, which may be different from the considered instances. Additionally, as previously discussed, a better exploration of promising neighborhood moves may save computational time and improve the convergence of the memetic algorithm.

Another area of future research is to consider the stochastic version of the problem. In the context of heavy maintenance, many operations are performed by human operators and processing times are uncertain. Moreover, uncertain additional tasks may also be encountered. Therefore, defining robust schedules would be helpful to meet customer deadlines. A possible robustness measure would be to maximize the chances of meeting project deadlines. Similar robustness measures, such as the service level, are proposed in Dauzère-Pérès et al. (2008).

Acknowledgements This work has been partially financed by the ANRT (Association Nationale de la Recherche et de la Technologie) through the PhD number 2021/0014 with CIFRE funds and a cooperation contract between SNCF and Mines Saint-Etienne.

Declarations

Conflicts of interest The authors have no relevant financial or non-financial interests to disclose.

Appendix A SSGS vs. PSGS: Detailed results on the performance of priority rules

Table 11 SSGS vs. PSGS: Performance of priority rules and associated gap from the best solution found by SSGS or PSGS (details for each instance)

		SWDP				SWTP			
Instance	PR	SSGS	%Gap	PSGS	%Gap	SSGS	%Gap	PSGS	%Gap
B_21	EF	24630	3.8	**37618**	37.0	2803	15.9	14844	84.1
	ES	23865	0.8	39279	39.7	2461	4.2	16510	85.7
	LF	***23683***	0.0	40052	40.9	2817	16.3	**14685**	83.9
	LS	24315	2.6	39223	39.6	***2357***	0.0	14841	84.1
	Rand	34333	31.0	38145	37.9	13820	82.9	19218	87.7
	SA	44752	47.1	43419	45.5	22456	89.5	20677	88.6
	SST	27484	13.8	44654	47.0	4305	45.2	24545	90.4
B_21_22	EF	40512	6.8	64308	41.3	6540	35.6	29385	85.7
	ES	38655	2.3	64781	41.7	5318	20.8	29873	85.9
	LF	***37757***	0.0	66343	43.1	4423	4.7	32118	86.9
	LS	39648	4.8	**63827**	40.8	***4214***	0.0	**27481**	84.7
	Rand	50745	25.6	75132	49.7	25964	83.8	35808	88.2
	SA	62223	39.3	86967	56.6	27981	84.9	52101	91.9
	SST	43536	13.3	84985	55.6	8474	50.3	39436	89.3
B_22	EF	14385	7.8	21013	36.9	2326	57.0	8845	88.7
	ES	13604	2.5	**20248**	34.5	1760	43.1	**8080**	87.6
	LF	13301	0.3	22048	39.9	***1001***	0.0	11265	91.1
	LS	***13259***	0.0	20764	36.1	1038	3.6	10258	90.2
	Rand	14301	7.3	22450	40.9	2284	56.2	12854	92.2
	SA	22894	42.1	29185	54.6	10954	90.9	17017	94.1
	SST	15704	15.6	32051	58.6	3703	73.0	14237	93.0
N_21	EF	126028	8.7	211432	45.6	14780	58.4	62085	90.1
	ES	118677	3.0	**202209**	43.1	9552	35.6	**52767**	88.3
	LF	121863	5.6	206036	44.2	10254	40.0	62444	90.2
	LS	***115060***	0.0	203812	43.5	***6150***	0.0	60020	89.8
	Rand	152451	24.5	292933	60.7	31433	80.4	125190	95.1
	SA	208886	44.9	273061	57.9	96524	93.6	121578	94.9
	SST	137554	16.4	303320	62.1	32915	81.3	100162	93.9
N_21_1	EF	120021	8.3	203515	45.9	19516	34.5	72364	82.3
	ES	112236	2.0	**189397**	41.9	13606	6.0	**58581**	78.2
	LF	116448	5.5	204187	46.1	14745	13.3	68648	81.4
	LS	***110022***	0.0	199111	44.7	***12791***	0.0	68837	81.4
	Rand	159900	31.2	266038	58.6	35401	63.9	117622	89.1
	SA	197827	44.4	253144	56.5	97634	86.9	120908	89.4
	SST	132789	17.1	261771	58.0	44622	71.3	123717	89.7

Table 11 continued

		SWDP				SWTP			
Instance	PR	SSGS	%Gap	PSGS	%Gap	SSGS	%Gap	PSGS	%Gap
N_21_2	EF	102208	8.2	169103	44.5	11408	50.7	48932	88.5
	ES	95181	1.5	171510	45.3	7019	19.9	51558	89.1
	LF	99497	5.7	175728	46.6	6621	15.1	54412	89.7
	LS	***93778***	0.0	**167026**	43.9	***5621***	0.0	**40852**	86.2
	Rand	111714	16.1	220777	57.5	22785	75.3	78779	92.9
	SA	158113	40.7	197210	52.4	65826	91.5	74517	92.5
	SST	106562	12.0	225685	58.4	15714	64.2	81507	93.1
N_21_3	EF	127789	6.9	222535	46.5	19760	41.0	79811	85.4
	ES	119335	0.3	**213107**	44.2	13175	11.5	70106	83.4
	LF	123025	3.3	228166	47.8	13092	11.0	**60491**	80.7
	LS	***118989***	0.0	220178	46.0	***11657***	0.0	74386	84.3
	Rand	157803	24.6	297886	60.1	43254	73.0	146218	92.0
	SA	227993	47.8	299121	60.2	119267	90.2	155780	92.5
	SST	147954	19.6	318400	62.6	45272	74.3	123376	90.6
N_22	EF	53345	8.0	111400	55.9	5971	82.2	56866	98.1
	ES	52118	5.8	127674	61.6	4989	78.7	72555	98.5
	LF	49903	1.7	127333	61.5	2668	60.2	**55580**	98.1
	LS	***49078***	0.0	**107352**	54.3	***1063***	0.0	70119	98.5
	Rand	64014	23.3	179505	72.7	18676	94.3	127516	99.2
	SA	65306	24.8	204734	76.0	17572	94.0	149821	99.3
	SST	59987	18.2	251311	80.5	6146	82.7	136541	99.2
N_22_ot	EF	71249	8.6	133743	51.3	6017	83.3	41395	97.6
	ES	67110	3.0	**118880**	45.2	2751	63.4	24466	95.9
	LF	67879	4.1	135783	52.1	1199	16.0	31204	96.8
	LS	***65088***	0.0	124235	47.6	***1007***	0.0	**22982**	95.6
	Rand	76826	15.3	149563	56.5	13047	92.3	54927	98.2
	SA	79790	18.4	139871	53.5	12697	92.1	46407	97.8
	SST	76135	14.5	150025	56.6	4025	75.0	33424	97.0
N_22_1	EF	61478	9.4	115416	51.7	6874	78.8	38575	96.2
	ES	57762	3.6	**101663**	45.2	4009	63.7	25398	94.3
	LF	58632	5.0	106274	47.6	2549	43.0	**21916**	93.4
	LS	***55692***	0.0	104366	46.6	***1454***	0.0	24669	94.1
	Rand	67217	17.1	137884	59.6	11286	87.1	59793	97.6
	SA	74121	24.9	125950	55.8	17548	91.7	46218	96.9
	SST	62071	10.3	127285	56.2	5111	71.6	37146	96.1

Table 11 continued

		SWDP				SWTP			
Instance	PR	SSGS	%Gap	PSGS	%Gap	SSGS	%Gap	PSGS	%Gap
N_22_2	EF	60931	9.3	110713	50.1	7296	73.8	35569	94.6
	ES	56884	2.9	111234	50.3	3910	51.0	33801	94.3
	LF	57873	4.5	111558	50.5	2531	24.4	31978	94.0
	LS	***55253***	0.0	**107552**	48.6	***1914***	0.0	**23790**	92.0
	Rand	65738	15.9	122832	55.0	7813	75.5	39741	95.2
	SA	69873	20.9	110550	50.0	14575	86.9	35319	94.6
	SST	64539	14.4	133928	58.7	4893	60.9	33844	94.3
N_22_3	EF	65507	5.9	113274	45.6	7828	64.3	33499	91.7
	ES	***61636***	0.0	120346	48.8	4937	43.4	39680	93.0
	LF	65041	5.2	119783	48.5	4088	31.6	39294	92.9
	LS	62421	1.3	**110156**	44.0	***2796***	0.0	**33448**	91.6
	Rand	69873	11.8	138131	55.4	13756	79.7	59199	95.3
	SA	74858	17.7	121460	49.3	15955	82.5	40673	93.1
	SST	65403	5.8	138713	55.6	5698	50.9	38426	92.7
N_b_21	EF	79474	11.1	**133253**	47.0	14993	63.8	**59634**	90.9
	ES	71628	1.3	156599	54.9	8870	38.7	82643	93.4
	LF	76638	7.8	156904	55.0	11629	53.3	66125	91.8
	LS	***70676***	0.0	148103	52.3	***5433***	0.0	76472	92.9
	Rand	97345	27.4	214791	67.1	53120	89.8	132559	95.9
	SA	172069	58.9	235297	70.0	109570	95.0	162092	96.6
	SST	87573	19.3	254039	72.2	50866	89.3	139575	96.1
N_b_21_1	EF	72514	12.4	127303	50.1	14294	61.2	62108	91.1
	ES	64853	2.0	130975	51.5	8551	35.2	65493	91.5
	LF	69793	9.0	126095	49.6	12777	56.6	66810	91.7
	LS	***63528***	0.0	**121198**	47.6	***5545***	0.0	**60983**	90.9
	Rand	92637	31.4	187810	66.2	41075	86.5	103080	94.6
	SA	146081	56.5	185548	65.8	90895	93.9	119810	95.4
	SST	79312	19.9	214241	70.3	48760	88.6	142482	96.1
N_b_21_2	EF	60057	10.6	116105	53.7	8926	69.5	55763	95.1
	ES	55562	3.3	108270	50.4	5373	49.3	**48239**	94.3
	LF	58445	8.1	114469	53.1	5070	46.2	53714	94.9
	LS	***53717***	0.0	**104743**	48.7	***2726***	0.0	52325	94.8
	Rand	66710	19.5	146696	63.4	17592	84.5	107668	97.5
	SA	99735	46.1	167657	68.0	49122	94.5	106769	97.4
	SST	64437	16.6	153217	64.9	16962	83.9	71302	96.2

Table 11 continued

		SWDP				SWTP			
Instance	PR	SSGS	%Gap	PSGS	%Gap	SSGS	%Gap	PSGS	%Gap
N_b_21_3	EF	78319	12.4	139506	50.8	14873	66.1	67648	92.6
	ES	70563	2.8	150814	54.5	8831	43.0	78587	93.6
	LF	75476	9.1	137230	50.0	12257	58.9	72620	93.1
	LS	***68586***	0.0	**132185**	48.1	***5037***	0.0	**67557**	92.5
	Rand	98160	30.1	206359	66.8	54243	90.7	153280	96.7
	SA	165386	58.5	217204	68.4	104995	95.2	145648	96.5
	SST	85243	19.5	212164	67.7	58757	91.4	150418	96.7
N_b_22_ot	EF	39825	12.6	81421	57.3	7300	75.1	43558	95.8
	ES	36864	5.6	82840	58.0	4683	61.2	45086	96.0
	LF	38157	8.8	**73406**	52.6	2912	37.5	**34229**	94.7
	LS	***34790***	0.0	80659	56.9	***1819***	0.0	40331	95.5
	Rand	46515	25.2	100857	65.5	11312	83.9	61892	97.1
	SA	50337	30.9	93941	63.0	17580	89.7	56525	96.8
	SST	39964	12.9	110237	68.4	7572	76.0	41201	95.6
S_21	EF	89238	7.7	167433	50.8	13921	34.5	82012	88.9
	ES	86963	5.3	167108	50.7	13127	30.5	81792	88.8
	LF	83294	1.1	**160465**	48.7	10230	10.9	84119	89.2
	LS	***82338***	0.0	165031	50.1	***9120***	0.0	**76098**	88.0
	Rand	93327	11.8	213250	61.4	27743	67.1	145037	93.7
	SA	127768	35.6	192355	57.2	50174	81.8	107149	91.5
	SST	107877	23.7	203907	59.6	26416	65.5	105539	91.4
S_21_1	EF	69506	10.1	124011	49.6	9096	54.2	55116	92.4
	ES	67375	7.2	125052	50.0	7843	46.9	55613	92.5
	LF	63843	2.1	119432	47.7	***4163***	0.0	**48969**	91.5
	LS	***62496***	0.0	**117164**	46.7	4275	2.6	51113	91.9
	Rand	69587	10.2	137197	54.4	13516	69.2	80692	94.8
	SA	87208	28.3	135671	53.9	25678	83.8	66254	93.7
	SST	76941	18.8	131811	52.6	13088	68.2	64853	93.6
S_21_2	EF	78431	9.8	**144104**	50.9	13003	31.4	70551	87.4
	ES	76733	7.9	152300	53.6	12483	28.6	78373	88.6
	LF	72360	2.3	147423	52.0	10463	14.8	**69799**	87.2
	LS	***70706***	0.0	152011	53.5	***8914***	0.0	70777	87.4
	Rand	85966	17.8	174695	59.5	29757	70.0	102784	91.3
	SA	108824	35.0	162605	56.5	41684	78.6	90456	90.1
	SST	93685	24.5	165950	57.4	23317	61.8	95750	90.7

Table 11 continued

		SWDP				SWTP			
Instance	PR	SSGS	%Gap	PSGS	%Gap	SSGS	%Gap	PSGS	%Gap
S_21_3	EF	84066	13.0	152216	52.0	14998	53.7	75037	90.7
	ES	79245	7.8	150620	51.5	11667	40.4	73898	90.6
	LF	76123	4.0	**140128**	47.8	7599	8.6	68363	89.8
	LS	***73100***	0.0	148065	50.6	***6949***	0.0	**66894**	89.6
	Rand	92343	20.8	188488	61.2	26453	73.7	109813	93.7
	SA	114608	36.2	175232	58.3	45185	84.6	97482	92.9
	SST	99000	26.2	172445	57.6	27352	74.6	84458	91.8
S_22	EF	76561	6.9	127928	44.3	8314	41.9	47882	89.9
	ES	77433	7.9	130142	45.2	9063	46.7	49808	90.3
	LF	***71306***	0.0	**123156**	42.1	5457	11.5	39936	87.9
	LS	71768	0.6	126609	43.7	***4827***	0.0	**39273**	87.7
	Rand	85810	16.9	143309	50.2	22345	78.4	61950	92.2
	SA	104575	31.8	143612	50.3	30676	84.3	62885	92.3
	SST	87057	18.1	148927	52.1	15418	68.7	61179	92.1
S_22_1	EF	62809	6.7	97144	39.7	7128	54.5	32863	90.1
	ES	59601	1.6	96475	39.2	6115	47.0	32817	90.1
	LF	59491	1.5	95853	38.8	***3240***	0.0	**27522**	88.2
	LS	***58620***	0.0	**90970**	35.6	3372	3.9	29296	88.9
	Rand	68470	14.4	112070	47.7	18493	82.5	43360	92.5
	SA	80005	26.7	119595	51.0	20690	84.3	53852	94.0
	SST	68537	14.5	111029	47.2	12813	74.7	39188	91.7
S_22_2	EF	70182	6.1	119330	44.8	7416	38.6	44734	89.8
	ES	71441	7.7	119243	44.7	8446	46.1	45661	90.0
	LF	***65921***	0.0	117830	44.1	***4552***	0.0	38350	88.1
	LS	66234	0.5	**115489**	42.9	4830	5.8	**37066**	87.7
	Rand	85191	22.6	141194	53.3	22127	79.4	59095	92.3
	SA	97044	32.1	130318	49.4	28297	83.9	54671	91.7
	SST	81911	19.5	129771	49.2	16108	71.7	43221	89.5
S_22_3	EF	65273	12.0	98578	41.7	6866	57.5	**33440**	91.3
	ES	62093	7.4	100065	42.6	6277	53.5	34712	91.6
	LF	60624	5.2	98526	41.7	3448	15.3	36539	92.0
	LS	***57469***	0.0	**95759**	40.0	***2920***	0.0	34624	91.6
	Rand	64855	11.4	110367	47.9	16186	82.0	53060	94.5
	SA	80555	28.7	112667	49.0	19709	85.2	46793	93.8
	SST	70155	18.1	113130	49.2	14717	80.2	46681	93.7

Table 11 continued

		SWDP				SWTP			
Instance	PR	SSGS	%Gap	PSGS	%Gap	SSGS	%Gap	PSGS	%Gap
S_b_21	EF	80651	8.4	**147562**	49.9	12279	29.8	70047	87.7
	ES	77511	4.7	161624	54.3	10185	15.4	84848	89.8
	LF	75901	2.7	150739	51.0	9691	11.0	**68344**	87.4
	LS	***73871***	0.0	161296	54.2	***8621***	0.0	74657	88.5
	Rand	93059	20.6	199969	63.1	22684	62.0	117886	92.7
	SA	114498	35.5	176376	58.1	45069	80.9	98926	91.3
	SST	98839	25.3	186481	60.4	24008	64.1	110365	92.2
S_b_22	EF	70824	9.7	109687	41.7	7633	60.3	37017	91.8
	ES	68564	6.7	115691	44.7	6923	56.2	42081	92.8
	LF	65123	1.8	**108011**	40.8	***3029***	0.0	**33367**	90.9
	LS	***63976***	0.0	110721	42.2	3304	8.3	36014	91.6
	Rand	85764	25.4	126822	49.6	17181	82.4	49149	93.8
	SA	103708	38.3	125138	48.9	36451	91.7	51517	94.1
	SST	77202	17.1	121254	47.2	15771	80.8	39088	92.3

Bold indicates which priority rule (PR) gives the best solution for a given heuristic (PSGS or SSGS)
Bolditalics indicates which combination of priority rule (PR) and heuristic (PSGS or SSGS) gives the best solution

Appendix B Metaheuristic comparison: detailed results

Table 12 Comparison of metaheuristics: SWDP objective and computational time limited to 600 s

Instance	Greedy SWDP	%Gap	SA SWDP	%Gap	GA SWDP	%Gap	hSGA SWDP	%Gap	MA SWDP	%Gap
B_21	24315	14.5	20958	0.9	21262	2.3	20837	0.3	**20778**	0.0
B_21_22	39648	15.6	34021	1.6	34929	4.1	33666	0.5	**33482**	0.0
B_22	13259	10.4	12137	2.2	12375	4.0	12057	1.5	**11876**	0.0
N_21	115060	6.2	110047	2.0	112748	4.3	108252	0.4	**107871**	0.0
N_21_1	110022	8.6	101667	1.1	106786	5.9	**100517**	0.0	101050	0.5
N_21_2	93778	6.5	89692	2.2	92432	5.1	88452	0.9	**87683**	0.0
N_21_3	118989	8.9	110971	2.3	114904	5.7	108886	0.5	**108386**	0.0
N_22	49078	6.9	45948	0.5	47578	3.9	45825	0.2	**45713**	0.0
N_22_ot	65088	5.7	62371	1.6	64325	4.6	**61391**	0.0	61418	0.0
N_22_1	55692	5.7	53085	1.1	54848	4.3	**52490**	0.0	52594	0.2
N_22_2	55253	5.8	52537	1.0	54529	4.6	**52030**	0.0	**52030**	0.0
N_22_3	62421	11.3	55757	0.7	59414	6.8	56041	1.2	**55388**	0.0
N_b_21	70676	12.3	64235	3.5	65947	6.0	62430	0.7	**62013**	0.0
N_b_21_1	63528	11.1	57317	1.4	59480	5.0	**56490**	0.0	56847	0.6
N_b_21_2	53717	9.3	49412	1.4	50922	4.3	49167	0.9	**48741**	0.0
N_b_21_3	68586	10.3	61903	0.6	64955	5.3	**61506**	0.0	61594	0.1
N_b_22_ot	34790	8.5	32313	1.4	32867	3.1	31916	0.2	**31849**	0.0
S_21	82338	7.3	78251	2.4	79497	3.9	**76364**	0.0	76969	0.8
S_21_1	62496	5.5	60507	2.4	61622	4.2	59110	0.1	**59044**	0.0
S_21_2	70706	6.0	68402	2.8	68532	3.0	66720	0.3	**66487**	0.0
S_21_3	73100	5.2	70737	2.1	72148	4.0	69645	0.5	**69283**	0.0
S_22	71768	6.2	68701	2.0	68418	1.6	**67293**	0.0	67749	0.7
S_22_1	58620	8.8	54675	2.2	54492	1.9	53820	0.7	**53450**	0.0
S_22_2	66234	7.3	62669	2.0	63096	2.7	61729	0.5	**61410**	0.0
S_22_3	57469	4.3	55392	0.7	55682	1.2	55056	0.1	**55002**	0.0
S_b_21	73871	5.5	71652	2.6	73053	4.4	70655	1.2	**69812**	0.0
S_b_22	63976	5.3	61019	0.7	61222	1.1	**60565**	0.0	60701	0.2

Table 13 Comparison of metaheuristics: SWDP objective and computational time limited to 1800 s

	Greedy		SA		GA		hSGA		MA	
Instance	SWDP	%Gap	SWDP	%Gap	SWDP	%Gap	SWDP	%Gap	SWDP	%Gap
B_21	24315	14.8	20814	0.4	21181	2.1	20772	0.2	**20727**	0.0
B_21_22	39648	16.5	33938	2.5	34802	4.9	33438	1.0	**33106**	0.0
B_22	13259	10.6	12033	1.5	12354	4.1	12018	1.4	**11847**	0.0
N_21	115060	7.1	109306	2.2	112008	4.5	107534	0.5	**106947**	0.0
N_21_1	110022	8.9	100939	0.7	106175	5.6	**100221**	0.0	100622	0.4
N_21_2	93778	6.8	88949	1.8	91290	4.3	88118	0.9	**87358**	0.0
N_21_3	118989	9.4	109806	1.8	114305	5.6	108398	0.5	**107848**	0.0
N_22	49078	7.3	45760	0.6	47341	3.9	**45497**	0.0	45648	0.3
N_22_ot	65088	6.3	61867	1.4	63713	4.2	61114	0.2	**61014**	0.0
N_22_1	55692	6.3	52613	0.8	54293	3.9	**52194**	0.0	52548	0.7
N_22_2	55253	6.1	52313	0.8	54089	4.1	51942	0.1	**51884**	0.0
N_22_3	62421	11.6	55315	0.2	59074	6.5	55573	0.7	**55209**	0.0
N_b_21	70676	12.8	63195	2.5	65681	6.2	62097	0.8	**61610**	0.0
N_b_21_1	63528	11.3	56648	0.5	59427	5.2	**56360**	0.0	56409	0.1
N_b_21_2	53717	9.9	49064	1.3	50764	4.6	48857	0.9	**48422**	0.0
N_b_21_3	68586	10.7	**61264**	0.0	64620	5.2	61273	0.0	61346	0.1
N_b_22_ot	34790	9.2	32097	1.6	32817	3.7	**31594**	0.0	31635	0.1
S_21	82338	7.6	77539	1.9	79117	3.9	**76058**	0.0	76545	0.6
S_21_1	62496	6.1	59695	1.7	61496	4.6	58871	0.4	**58653**	0.0
S_21_2	70706	6.5	68008	2.8	68064	2.9	66185	0.2	**66084**	0.0
S_21_3	73100	5.8	70453	2.3	71943	4.3	69124	0.4	**68866**	0.0
S_22	71768	6.9	67945	1.6	68117	1.9	**66839**	0.0	67604	1.1
S_22_1	58620	9.1	54238	1.7	54405	2.0	53718	0.8	**53310**	0.0
S_22_2	66234	7.6	62272	1.7	62933	2.8	61572	0.6	**61188**	0.0
S_22_3	57469	4.7	55250	0.8	55462	1.2	54911	0.2	**54793**	0.0
S_b_21	73871	5.8	71320	2.4	72620	4.2	70231	0.9	**69582**	0.0
S_b_22	63976	5.9	60882	1.2	61017	1.4	**60173**	0.0	60179	0.0

Table 14 Comparison of metaheuristics: SWDP objective and computational time limited to 3600 s

Instance	Greedy SWDP	%Gap	SA SWDP	%Gap	GA SWDP	%Gap	hSGA SWDP	%Gap	MA SWDP	%Gap
B_21	24315	15.0	20770	0.4	21178	2.4	20701	0.1	**20677**	0.0
B_21_22	39648	16.6	33597	1.6	34751	4.9	33142	0.3	**33054**	0.0
B_22	13259	11.2	11909	1.1	12328	4.5	11978	1.7	**11778**	0.0
N_21	115060	7.2	109023	2.0	111611	4.3	107184	0.4	**106800**	0.0
N_21_1	110022	9.0	100588	0.5	105966	5.5	**100103**	0.0	100421	0.3
N_21_2	93778	7.0	88671	1.7	91069	4.2	87933	0.8	**87199**	0.0
N_21_3	118989	9.7	108457	0.9	114171	5.9	107749	0.3	**107449**	0.0
N_22	49078	7.4	45672	0.5	47237	3.8	**45441**	0.0	45509	0.1
N_22_ot	65088	6.5	61329	0.8	63474	4.1	61048	0.3	**60865**	0.0
N_22_1	55692	6.3	52184	0.0	54274	3.9	**52182**	0.0	52309	0.2
N_22_2	55253	6.3	52001	0.4	53781	3.7	51847	0.1	**51793**	0.0
N_22_3	62421	12.0	**54929**	0.0	58659	6.4	55256	0.6	54972	0.1
N_b_21	70676	13.2	62645	2.0	65679	6.6	61937	0.9	**61374**	0.0
N_b_21_1	63528	11.8	56366	0.6	58998	5.1	56353	0.6	**56004**	0.0
N_b_21_2	53717	10.0	48936	1.2	50537	4.4	48723	0.8	**48333**	0.0
N_b_21_3	68586	11.1	61145	0.3	64481	5.4	61057	0.1	**60992**	0.0
N_b_22_ot	34790	9.4	31987	1.5	32789	3.9	31524	0.1	**31508**	0.0
S_21	82338	7.9	77305	1.9	79044	4.1	**75814**	0.0	76234	0.6
S_21_1	62496	6.4	59526	1.7	61376	4.6	58713	0.3	**58527**	0.0
S_21_2	70706	7.0	67700	2.8	67922	3.2	**65774**	0.0	65910	0.2
S_21_3	73100	6.1	70096	2.0	71839	4.4	68864	0.3	**68663**	0.0
S_22	71768	6.9	67597	1.2	67906	1.6	**66795**	0.0	67506	1.1
S_22_1	58620	9.2	54102	1.6	54193	1.7	53642	0.7	**53247**	0.0
S_22_2	66234	7.8	62215	1.8	62899	2.9	61438	0.6	**61065**	0.0
S_22_3	57469	4.8	55057	0.6	55277	1.0	**54727**	0.0	54764	0.1
S_b_21	73871	6.1	71018	2.3	72467	4.3	69931	0.8	**69359**	0.0
S_b_22	63976	6.0	60530	0.7	60808	1.1	**60109**	0.0	60173	0.1

Table 15 Comparison of metaheuristics: SWTP objective and computational time limited to 600 s

Instance	Greedy SWDP	%Gap	SA SWDP	%Gap	GA SWDP	%Gap	hSGA SWDP	%Gap	MA SWDP	%Gap
B_21	2357	67.8	849	10.7	788	3.8	**758**	0.0	**758**	0.0
B_21_22	4214	64.4	1908	21.4	1594	5.9	1554	3.5	**1500**	0.0
B_22	1038	64.4	493	25.2	431	14.4	390	5.4	**369**	0.0
N_21	6150	61.7	2889	18.5	3035	22.4	**2355**	0.0	2475	4.8
N_21_1	12791	68.0	5779	29.1	6143	33.3	5200	21.2	**4099**	0.0
N_21_2	5621	62.7	2636	20.4	2473	15.2	**2098**	0.0	2104	0.3
N_21_3	11657	56.7	6951	27.4	6334	20.3	**5048**	0.0	5592	9.7
N_22	1063	44.3	707	16.3	636	6.9	646	8.4	**592**	0.0
N_22_ot	1007	73.5	365	26.8	293	8.9	**267**	0.0	302	11.6
N_22_1	1454	38.6	1079	17.3	946	5.7	**892**	0.0	**892**	0.0
N_22_2	1914	45.7	1133	8.3	1114	6.7	**1039**	0.0	1080	3.8
N_22_3	2796	59.1	1296	11.7	1336	14.4	**1144**	0.0	1180	3.0
N_b_21	5433	58.7	2330	3.7	2643	15.1	2500	10.3	**2243**	0.0
N_b_21_1	5545	65.4	2588	26.0	2872	33.3	2259	15.2	**1916**	0.0
N_b_21_2	2726	62.4	1113	7.9	1290	20.5	1111	7.7	**1025**	0.0
N_b_21_3	5037	54.8	2454	7.3	2978	23.6	**2274**	0.0	2491	8.7
N_b_22_ot	1819	35.9	1311	11.1	1173	0.6	1260	7.5	**1166**	0.0
S_21	9120	43.5	5534	6.9	6272	17.9	5533	6.9	**5151**	0.0
S_21_1	4275	43.7	2588	7.0	2462	2.2	**2407**	0.0	2592	7.1
S_21_2	8914	42.4	5586	8.2	5172	0.8	**5130**	0.0	5186	1.1
S_21_3	6949	44.4	4556	15.2	4998	22.7	4131	6.4	**3865**	0.0
S_22	4827	40.2	3240	11.0	3016	4.3	**2885**	0.0	2923	1.3
S_22_1	3372	32.9	2555	11.5	2373	4.7	2348	3.7	**2262**	0.0
S_22_2	4830	40.4	2926	1.5	**2881**	0.0	2901	0.7	2908	0.9
S_22_3	2920	21.5	2404	4.6	2428	5.6	2362	2.9	**2293**	0.0
S_b_21	8621	51.6	5523	24.4	5164	19.2	**4173**	0.0	4440	6.0
S_b_22	3304	33.5	2539	13.4	2336	5.9	**2198**	0.0	2274	3.3

Table 16 Comparison of metaheuristics: SWTP objective and computational time limited to 1800 s

Instance	Greedy SWDP	%Gap	SA SWDP	%Gap	GA SWDP	%Gap	hSGA SWDP	%Gap	MA SWDP	%Gap
B_21	2357	67.8	840	9.8	788	3.8	**758**	0.0	**758**	0.0
B_21_22	4214	64.5	1810	17.4	1568	4.6	1545	3.2	**1496**	0.0
B_22	1038	66.3	435	19.5	431	18.8	390	10.3	**350**	0.0
N_21	6150	64.8	2707	20.1	3011	28.2	**2163**	0.0	2433	11.1
N_21_1	12791	68.8	5597	28.7	5485	27.3	4912	18.8	**3989**	0.0
N_21_2	5621	65.3	2137	8.8	2354	17.2	1994	2.3	**1949**	0.0
N_21_3	11657	58.6	6640	27.2	6190	21.9	**4832**	0.0	5381	10.2

Table 16 continued

Instance	Greedy SWDP	%Gap	SA SWDP	%Gap	GA SWDP	%Gap	hSGA SWDP	%Gap	MA SWDP	%Gap
N_22	1063	44.9	702	16.5	636	7.9	635	7.7	**586**	0.0
N_22_ot	1007	73.5	365	26.8	293	8.9	**267**	0.0	292	8.6
N_22_1	1454	38.6	984	9.4	946	5.7	**892**	0.0	**892**	0.0
N_22_2	1914	45.7	1117	7.0	1109	6.3	**1039**	0.0	1056	1.6
N_22_3	2796	59.1	1296	11.7	1230	7.0	**1144**	0.0	1180	3.0
N_b_21	5433	59.7	2317	5.4	2584	15.2	2483	11.8	**2191**	0.0
N_b_21_1	5545	65.7	2544	25.2	2860	33.4	2232	14.7	**1904**	0.0
N_b_21_2	2726	62.4	1102	7.0	1290	20.5	1111	7.7	**1025**	0.0
N_b_21_3	5037	56.1	2394	7.6	2838	22.1	**2211**	0.0	2362	6.4
N_b_22_ot	1819	36.3	1263	8.2	1173	1.2	1260	8.0	**1159**	0.0
S_21	9120	46.3	5355	8.6	5778	15.2	5338	8.3	**4897**	0.0
S_21_1	4275	44.4	2561	7.3	2450	3.1	**2375**	0.0	2557	7.1
S_21_2	8914	45.4	5340	8.8	**4870**	0.0	4918	1.0	5136	5.2
S_21_3	6949	47.3	4281	14.5	4938	25.9	**3660**	0.0	3701	1.1
S_22	4827	41.4	3050	7.2	2949	4.0	**2831**	0.0	2880	1.7
S_22_1	3372	33.4	2457	8.6	2312	2.9	2342	4.1	**2245**	0.0
S_22_2	4830	40.6	2911	1.4	**2869**	0.0	2883	0.5	2886	0.6
S_22_3	2920	23.5	2348	4.8	2409	7.2	2339	4.4	**2235**	0.0
S_b_21	8621	52.7	5221	21.8	5052	19.2	**4081**	0.0	4380	6.8
S_b_22	3304	33.9	2483	12.0	2311	5.4	**2185**	0.0	2265	3.5

Table 17 Comparison of metaheuristics: SWTP objective and computational time limited to 3600 s

Instance	Greedy SWDP	%Gap	SA SWDP	%Gap	GA SWDP	%Gap	hSGA SWDP	%Gap	MA SWDP	%Gap
B_21	2357	67.8	816	7.1	788	3.8	**758**	0.0	**758**	0.0
B_21_22	4214	64.5	1733	13.7	1568	4.6	1507	0.7	**1496**	0.0
B_22	1038	66.3	388	9.8	431	18.8	374	6.4	**350**	0.0
N_21	6150	65.0	2628	18.2	3002	28.4	**2150**	0.0	2349	8.5
N_21_1	12791	69.2	5222	24.6	5398	27.1	4872	19.2	**3935**	0.0
N_21_2	5621	68.2	2083	14.1	2354	24.0	1967	9.0	**1789**	0.0
N_21_3	11657	59.2	6505	27.0	6188	23.2	**4751**	0.0	5208	8.8
N_22	1063	44.9	701	16.4	636	7.9	634	7.6	**586**	0.0
N_22_ot	1007	73.5	365	26.8	293	8.9	**267**	0.0	292	8.6
N_22_1	1454	38.7	984	9.3	946	5.7	**892**	0.0	**892**	0.0
N_22_2	1914	46.1	1079	4.4	1109	6.9	1039	0.7	**1032**	0.0
N_22_3	2796	59.2	1287	11.3	1204	5.1	**1142**	0.0	1167	2.1
N_b_21	5433	59.7	2294	4.5	2584	15.2	2476	11.5	**2191**	0.0
N_b_21_1	5545	65.7	2333	18.4	2797	31.9	2135	10.8	**1904**	0.0

Table 17 continued

Instance	Greedy SWDP	%Gap	SA SWDP	%Gap	GA SWDP	%Gap	hSGA SWDP	%Gap	MA SWDP	%Gap
N_b_21_2	2726	62.4	1091	6.0	1130	9.3	1088	5.8	**1025**	0.0
N_b_21_3	5037	57.4	2391	10.2	2823	23.9	**2147**	0.0	2311	7.1
N_b_22_ot	1819	36.3	1238	6.4	1173	1.2	1260	8.0	**1159**	0.0
S_21	9120	46.3	5288	7.4	5761	15.0	5321	8.0	**4897**	0.0
S_21_1	4275	45.8	2489	7.0	2450	5.5	**2316**	0.0	2532	8.5
S_21_2	8914	47.1	5171	8.7	4865	3.0	**4719**	0.0	4930	4.3
S_21_3	6949	47.9	4126	12.3	4883	25.9	**3617**	0.0	3681	1.7
S_22	4827	41.6	3027	6.9	2949	4.4	**2819**	0.0	2867	1.7
S_22_1	3372	33.4	2426	7.5	2298	2.3	2280	1.5	**2245**	0.0
S_22_2	4830	43.3	2907	5.8	2859	4.2	**2739**	0.0	2885	5.1
S_22_3	2920	23.8	2336	4.8	2383	6.6	2329	4.5	**2225**	0.0
S_b_21	8621	53.3	5147	21.8	5044	20.2	**4026**	0.0	4310	6.6
S_b_22	3304	34.0	2447	10.8	2310	5.5	**2182**	0.0	2265	3.7

References

Afshar-Nadjafi, B. (2021). Multi-skilling in scheduling problems: A review on models, methods and applications. *Computers & Industrial Engineering, 151*, 107004.

Almeida, B. F., Correia, I., & Saldanha-da Gama, F. (2016). Priority-based heuristics for the multi-skill resource constrained project scheduling problem. *Expert Systems with Applications, 57*, 91–103.

Almeida, B. F., Correia, I., & Saldanha-da Gama, F. (2019). Modeling frameworks for the multi-skill resource-constrained project scheduling problem: A theoretical and empirical comparison. *International Transactions in Operational Research, 26*(3), 946–967.

Asta, S., Karapetyan, D., Kheiri, A., Özcan, E., & Parkes, A. J. (2016). Combining Monte-Carlo and hyper-heuristic methods for the multi-mode resource-constrained multi-project scheduling problem. *Information Sciences, 373*, 476–498.

Bellenguez, O., & Néron, E. (2004). Lower bounds for the multi-skill project scheduling problem with hierarchical levels of skills. In *International conference on the practice and theory of automated timetabling*, pp. 229–243. Springer.

Bellenguez-Morineau, O., & Néron, E. (2007). A branch-and-bound method for solving multi-skill project scheduling problem. *RAIRO-operations Research, 41*(2), 155–170.

Blazewicz, J., Lenstra, J. K., & Kan, A. R. (1983). Scheduling subject to resource constraints: Classification and complexity. *Discrete applied Mathematics, 5*(1), 11–24.

Bouleimen, K., & Lecocq, H. (2003). A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version. *European Journal of Operational Research, 149*(2), 268–281.

Bredael, D., & Vanhoucke, M. (2022). Multi-project scheduling: A benchmark analysis of metaheuristic algorithms on various optimisation criteria and due dates. *European Journal of Operational Research, 308*, 54.

Browning, T. R., & Yassine, A. A. (2010). Resource-constrained multi-project scheduling: Priority rule performance revisited. *International Journal of Production Economics, 126*(2), 212–228.

Brucker, P., Drexl, A., Möhring, R., Neumann, K., & Pesch, E. (1999). Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research, 112*(1), 3–41.

Chen, J. C., Chen, Y. Y., Chen, T. L., & Lin, Y. H. (2022). Multi-project scheduling with multi-skilled workforce assignment considering uncertainty and learning effect for large-scale equipment manufacturer. *Computers & Industrial Engineering, 169*, 108240.

Chen, J. C., Lee, H. Y., Hsieh, W. H., & Chen, T. L. (2022). Applying hybrid genetic algorithm to multi-mode resource constrained multi-project scheduling problems. *Journal of the Chinese Institute of Engineers, 45*(1), 42–53.

Chen, P. H., & Shahandashti, S. M. (2009). Hybrid of genetic algorithm and simulated annealing for multiple project scheduling with multiple resource constraints. *Automation in Construction, 18*(4), 434–443.

Christofides, N., Alvarez-Valdés, R., & Tamarit, J. M. (1987). Project scheduling with resource constraints: A branch and bound approach. *European Journal of Operational Research, 29*(3), 262–273.

Confessore, G., Giordani, S., & Rismondo, S. (2007). A market-based multi-agent system model for decentralized multi-project scheduling. *Annals of Operations Research, 150*(1), 115–135.

Correia, I., & Saldanha-da Gama, F. (2014). The impact of fixed and variable costs in a multi-skill project scheduling problem: An empirical study. *Computers & Industrial Engineering, 72*, 230–238.

Cui, L., Liu, X., Lu, S., & Jia, Z. (2021). A variable neighborhood search approach for the resource-constrained multi-project collaborative scheduling problem. *Applied Soft Computing, 107*, 107480.

Dauzère-Pérès, S., P. Castagliola, & Lahlou, C. (2008). Service level in scheduling.

Dauzère-Pérès, S., & Paulli, J. (1997). An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search. *Annals of Operations Research, 70*, 281–306.

Deblaere, F., Demeulemeester, E., & Herroelen, W. (2011). Proactive policies for the stochastic resource-constrained project scheduling problem. *European Journal of Operational Research, 214*(2), 308–316.

Deckro, R. F., Winkofsky, E., Hebert, J. E., & Gagnon, R. (1991). A decomposition approach to multi-project scheduling. *European Journal of Operational Research, 51*(1), 110–118.

Drexl, A. (1991). Scheduling of project networks by job assignment. *Management Science, 37*(12), 1590–1602.

Elloumi, S., & Fortemps, P. (2010). A hybrid rank-based evolutionary algorithm applied to multi-mode resource-constrained project scheduling problem. *European Journal of Operational Research, 205*(1), 31–41.

Essafi, I., Mati, Y., & Dauzère-Pérès, S. (2008). A genetic local search algorithm for minimizing total weighted tardiness in the job-shop scheduling problem. *Computers & Operations Research, 35*(8), 2599–2616.

Felberbauer, T., Gutjahr, W. J., & Doerner, K. F. (2019). Stochastic project management: Multiple projects with multi-skilled human resources. *Journal of Scheduling, 22*(3), 271–288.

Gonçalves, J. F., de Magalhães Mendes, J. J., & Resende, M. G. (2005). A hybrid genetic algorithm for the job shop scheduling problem. *European Journal of Operational Research, 167*(1), 77–95.

Gonçalves, J. F., Mendes, J. J., & Resende, M. G. (2008). A genetic algorithm for the resource constrained multi-project scheduling problem. *European Journal of Operational Research, 189*(3), 1171–1190.

Habibi, F., Barzinpour, F., & Sadjadi, S. (2018). Resource-constrained project scheduling problem: Review of past and recent developments. *Journal of Project Management, 3*(2), 55–88.

Haroune, M., Dhib, C., Neron, E., Soukhal, A., Mohamed Babou, H., & Nanne, M. F. (2022). Multi-project scheduling problem under shared multi-skill resource constraints. *TOP, 31*, 194.

Hartmann, S. (1998). A competitive genetic algorithm for resource-constrained project scheduling. *Naval Research Logistics (NRL), 45*(7), 733–750.

Hartmann, S., & Briskorn, D. (2022). An updated survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research, 297*(1), 1–14.

Hartmann, S., & Kolisch, R. (2000). Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research, 127*(2), 394–407.

Heimerl, C., & Kolisch, R. (2010). Scheduling and staffing multiple projects with a multi-skilled workforce. *OR Spectrum, 32*(2), 343–368.

Javanmard, S., Afshar-Nadjafi, B., & Niaki, S. T. A. (2017). Preemptive multi-skilled resource investment project scheduling problem: Mathematical modelling and solution approaches. *Computers & Chemical Engineering, 96*, 55–68.

Knopp, S., Dauzère-Pérès, S., & Yugma, C. (2017). A batch-oblivious approach for complex job-shop scheduling problems. *European Journal of Operational Research, 263*(1), 50–61.

Kolisch, R. (1996). Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. *European Journal of Operational Research, 90*(2), 320–333.

Kolisch, R., & Hartmann, S. (2006). Experimental investigation of heuristics for resource-constrained project scheduling: An update. *European Journal of Operational Research, 174*(1), 23–37.

Kolisch, R., & Sprecher, A. (1997). Psplib-a project scheduling problem library: Or software-orsep operations research software exchange program. *European Journal of Operational Research, 96*(1), 205–216.

Koné, O., Artigues, C., Lopez, P., & Mongeau, M. (2011). Event-based MILP models for resource-constrained project scheduling problems. *Computers & Operations Research, 38*(1), 3–13.

Krüger, D., & Scholl, A. (2009). A heuristic solution framework for the resource constrained (multi-) project scheduling problem with sequence-dependent transfer times. *European Journal of Operational Research, 197*(2), 492–508.

Kurtulus, I., & Davis, E. (1982). Multi-project scheduling: Categorization of heuristic rules performance. *Management Science, 28*(2), 161–172.

Lancaster, J., & Ozbayrak, M. (2007). Evolutionary algorithms applied to project scheduling problems-a survey of the state-of-the-art. *International Journal of Production Research, 45*(2), 425–450.

Li, F., Xu, Z., & Li, H. (2021). A multi-agent based cooperative approach to decentralized multi-project scheduling and resource allocation. *Computers & Industrial Engineering, 151*, 106961.

Li, H., & Womer, K. (2009). Scheduling projects with multi-skilled personnel by a hybrid MILP/CP benders decomposition algorithm. *Journal of Scheduling, 12*(3), 281–298.

Lin, J., Zhu, L., & Gao, K. (2020). A genetic programming hyper-heuristic approach for the multi-skill resource constrained project scheduling problem. *Expert Systems with Applications, 140*, 112915.

Lova, A., Maroto, C., & Tormos, P. (2000). A multicriteria heuristic method to improve resource allocation in multiproject scheduling. *European Journal of Operational Research, 127*(2), 408–424.

Mati, Y., Dauzère-Pérès, S., & Lahlou, C. (2011). A general approach for optimizing regular criteria in the job-shop scheduling problem. *European Journal of Operational Research, 212*(1), 33–42.

Moeuf, A., Pellerin, R., Lamouri, S., Tamayo-Giraldo, S., & Barbaray, R. (2018). The industrial management of SMEs in the era of industry 4.0. *International Journal of Production Research, 56*(3), 1118–1136.

Montoya, C., Bellenguez-Morineau, O., Pinson, E., & Rivreau, D. (2014). Branch-and-price approach for the multi-skill project scheduling problem. *Optimization Letters, 8*(5), 1721–1734.

Moscato, P., & Cotta, C. (2003). *A gentle introduction to memetic algorithms, Handbook of metaheuristics,* (pp. 105–144). Springer.

Murata, T., Ishibuchi, H., & Tanaka, H. (1996). Genetic algorithms for flowshop scheduling problems. *Computers & Industrial Engineering, 30*(4), 1061–1071.

Myszkowski, P. B., Skowroński, M. E., & Sikora, K. (2015). A new benchmark dataset for multi-skill resource-constrained project scheduling problem. In *2015 federated conference on computer science and information systems (FedCSIS)*, pp. 129–138. IEEE.

Özdamar, L., & Ulusoy, G. (1995). A survey on the resource-constrained project scheduling problem. *IIE Transactions, 27*(5), 574–586.

Pellerin, R., Perrier, N., & Berthaut, F. (2020). A survey of hybrid metaheuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research, 280*(2), 395–416.

Polo-Mejía, O., Artigues, C., Lopez, P., Mönch, L., & Basini, V. (2021). Heuristic and metaheuristic methods for the multi-skill project scheduling problem with partial preemption. *International Transactions in Operational Research, 30*, 858.

Pritsker, A. A. B., Waiters, L. J., & Wolfe, P. M. (1969). Multiproject scheduling with limited resources: A zero-one programming approach. *Management Science, 16*(1), 93–108.

Rahman, H. F., Chakrabortty, R. K., & Ryan, M. J. (2020). Memetic algorithm for solving resource constrained project scheduling problems. *Automation in Construction, 111*, 103052.

Sánchez, M. G., Lalla-Ruiz, E., Gil, A. F., Castro, C., & Voß, S. (2022). Resource-constrained multi-project scheduling problem: A survey. *European Journal of Operational Research, 309*, 958.

Sevaux, M., & Dauzère-Pérès, S. (2003). Genetic algorithms to minimize the weighted number of late jobs on a single machine. *European Journal of Operational Research, 151*(2), 296–306.

Snauwaert, J., & Vanhoucke, M. (2023). A classification and new benchmark instances for the multi-skilled resource-constrained project scheduling problem. *European Journal of Operational Research, 307*(1), 1–19.

Sörensen, K., & Sevaux, M. (2006). Ma pm: memetic algorithms with population management. *Computers & Operations Research, 33*(5), 1214–1225.

Syswerda, G. (1991). A study of reproduction in generational and steady-state genetic algorithms. *Foundations of Genetic Algorithms, 1*, 94–101.

Tamssaouet, K., Dauzère-Pérès, S., Knopp, S., Bitar, A., & Yugma, C. (2022). Multiobjective optimization for complex flexible job-shop scheduling problems. *European Journal of Operational Research, 296*(1), 87–100.

Van Peteghem, V., & Vanhoucke, M. (2010). A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problem. *European Journal of Operational Research, 201*(2), 409–418.

Vanhoucke, M., Coelho, J., Debels, D., Maenhout, B., & Tavares, L. V. (2008). An evaluation of the adequacy of project network generators with systematically sampled networks. *European Journal of Operational Research, 187*(2), 511–524.

Vercellis, C. (1994). Constrained multi-project plannings problems: A Lagrangean decomposition approach. *European Journal of Operational Research, 78*(2), 267–275.

Wauters, T., Kinable, J., Smet, P., Vancroonenburg, W., Vanden Berghe, G., & Verstichel, J. (2016). The multi-mode resource-constrained multi-project scheduling problem. *Journal of Scheduling, 19*(3), 271–283.

Wauters, T., Verbeeck, K., De Causmaecker, P., & Vanden Berghe, G. (2015). A learning-based optimization approach to multi-project scheduling. *Journal of Scheduling, 18*, 61–74.

Yugma, C., Dauzère-Pérès, S., Artigues, C., Derreumaux, A., & Sibille, O. (2012). A batching and scheduling algorithm for the diffusion area in semiconductor manufacturing. *International Journal of Production Research, 50*(8), 2118–2132.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.