Computers & Industrial Engineering 169 (2022) 108276

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

A critical-path based iterated local search for the green permutation flowshop problem

Victor Fernandez-Viagas[a,*], Bruno de Athayde Prata[b], Jose M. Framinan[a,c]

[a] Industrial Management, School of Engineering, University of Seville, Camino de los Descubrimientos s/n, 41092 Seville, Spain
[b] Department of Industrial Engineering, Federal University of Ceara, Ceara, Brazil
[c] Laboratory of Engineering for Environmental Sustainability, University of Seville, Spain

ARTICLE INFO

Keywords:
Green scheduling
Bi-objective
Sustainable manufacturing
Flow shop
Carbon emissions reduction

ABSTRACT

The permutation flowshop scheduling problem is a widely studied combinatorial optimization problem with several real-world applications. In this paper we address a green variant of the problem with controllable processing times and two objective functions: one related to the service level of the factory (makespan) and another one related to the total cost or the total energy/carbon consumption. For this problem we propose a novel Critical-Path based Iterated Local Search. This metaheuristic incorporates several theoretical results to accelerate the search of solutions in the intensification phase. The proposed algorithm has been compared on an extensive benchmark with the most promising algorithms in the literature. The computational results show the excellent performance of the proposal.

1. Introduction

The flowshop scheduling problem is one of the most studied problems in the optimization literature (see e.g. the reviews by Fernandez-Viagas et al., 2017; Framinan et al., 2004; Reza and Saghafian, 2005; Ruiz and Maroto, 2005). This problem consists of obtaining the best sequence of jobs to be processed on a series of machines in order to optimize one or more criteria, where each job follows the same route of machines. Traditionally, this problem is addressed in the literature considering the same order of jobs in all machines and it is denoted as the Permutation Flowshop Scheduling Problem (PFSP). In the PFSP as well as in most scheduling literature, the processing times required for the jobs to be processed on the machines has been commonly considered as known and constant. However, in many real-world scenarios, these processing times may change if additional resources are assigned (Lu et al., 2014; Nowicki and Zdrzalka, 1990), a case that is denoted as Controllable Processing Times (CPT). Thereby, in a scheduling problem assuming CPT, the processing time of a job on a machine depends on one or several factors, such as the amount of resources assigned to the machine (which in this case usually models a manual operation), the skills of these resources, or the machine configuration applied. Reviews of manufacturing scheduling with CPT have been carried out by Nowicki and Zdrzalka (1990), Shabtay and Steiner (2007), Shioura et al. (2018), Fernandez-Viagas and Framinan (2015c).

Given the global pressure to reduce carbon emissions (Wang et al., 2011; Zhu et al., 2014), in the last years, a growing attention is being paid in the literature (see e.g. Amiri and Behnamian, 2020; Cota et al., 2021; Öztop et al., 2020) to a special case of scheduling with CPT –denoted as *green scheduling*– where the CPT are aimed to reduce the energy consumed by the machines during their processing. In this paper, we focus on green scheduling in a PFSP context with two criteria: one related to the service level of the factory (makespan) and another one related to the total cost or the total energy/carbon consumption. This problem can be denoted as $Fm|prmu, p_{ij}(u_{ij})|\#(C_{max}, Cost)$ according to Graham et al. (1979), T'Kindt and Billaut (2006) and it is NP-hard since the single-objective decision problem (i.e. the problem without CPT, $Fm|prmu|C_{max}$) is already NP-hard for more than two machines (Rinnooy Kan, 1976). In the literature, most researchers have addressed this problem by developing approximated procedures able to provide good solutions within a reasonable computational effort (see e.g. Mokhtari et al., 2011; Öztop et al., 2020). Despite these contributions, we are not aware that problem-specific properties have been studied to improve the effectiveness of the approximated procedures. In this regard, recent advances from related scheduling problems (see e.g. Benavides and Ritt, 2018; Fernandez-Viagas et al., 2020; Fernandez-Viagas and Framinan, 2015b; Fernandez-Viagas and Framinan, 2015a) show that much more

* Corresponding author.
E-mail address: vfernandezviagas@us.es (V. Fernandez-Viagas).

https://doi.org/10.1016/j.cie.2022.108276
Received 4 November 2021; Received in revised form 1 April 2022; Accepted 23 May 2022
Available online 28 May 2022

efficient solutions can be obtained by incorporating problem-specific properties in the algorithms rather than by using different meta-heuristics or more complex local search procedures. Therefore, we believe that there is room for improving the state-of-the-art of approximate solution procedures for the problem and present a novel iterated local search which is based on several theoretical results. To present this contribution, the rest of the paper is organised as follows: the problem is formally described and its related literature reviewed in Section 2; next, in Section 3 we develop some theoretical results which are incorporated in the proposed iterated local search detailed in Section 4. The efficiency of the proposal is analysed by comparison against the most effective metaheuristics in the literature in Section 5. Finally, the conclusions are discussed in Section 6.

2. Problem description and background

The problem under consideration can be defined as follows: a set $\mathcal{J}$ of n jobs ($\mathcal{J} := \{1,2,\ldots,n\}$) has to be scheduled in a set $\mathcal{M}$ of m machines. Each machine i can perform the processing of the jobs using K different speed levels or *speed modes*. Let s_{ik} denote the speed of machine i if speed mode k ($k \in \{1,\ldots,K\}$) is used. Then, the processing time of job j on machine i when speed mode k is employed is $p_{ijk} = \frac{\widehat{p}_{ij}}{s_{ik}}$, where $\widehat{p}_{ij}$ is a known reference processing time of job j on machine i. Each speed mode is associated to a time unit processing cost, i.e. processing each job on machine i at speed mode k incurs a cost per time unit of pc_{ik}, therefore the cost of processing job j on machine j at speed k would be $pc_{ik}{\cdot}p_{ijk}$. Furthermore, we assume that there is a time-unit cost ic_i due to having machine i idle (i.e. in stand-by mode). These two sources of costs would be discussed later in the context of energy consumption. Next, we briefly detail the rest of commonly assumed constraints in the problem under consideration:

- Each machine can process at most a job at the same time.
- Each job can be processed by at most one machine at the same time.
- To be processed, all jobs has to followed the same route of machines.
- Release times are set to 0.
- Data are assumed to be deterministic.
- No preemption is assumed.
- Setup times are considered as insignificant or non-anticipatory and sequence-independent (and therefore they could be added to the processing times).
- Transportation times can be considered either insignificant or constant for all jobs.

Given the problem above, since each job j can be processed on each machine using a different speed, let us u_{ij} denote the speed mode at which job j is to be processed on machine i. Furthermore, since the permutation constraint is considered in our study, $\mathcal{S} := \{\Pi, \mathcal{U}\}$ a feasible solution for the problem is univocally defined by giving a permutation $\Pi = (\pi_1,\ldots,\pi_n)$ where π_j indicates the job to be processed in order j in all the machines, and by giving $\mathcal{U} = (u_{11},\ldots,u_{ij},\ldots,u_{mn})$ the set of speed modes to be employed for each machine when processing each job. For the problem under study, we minimize a bi-objective function of the makespan (C_{max}) and total cost (TC). The makespan is the maximum completion time of any job on the last machine (i.e. $C_{max} = \max_j C_{m,j}$), where C_{ij} the completion time of job j on machine i is obtained using Eq. (1).

$$C_{i,\pi_j} = \max\left\{C_{i-1,\pi_j}, C_{i,\pi_{j-1}}\right\} + \frac{\widehat{p}_{i,\pi_j}}{s_{i,u_{i,\pi_j}}},\quad i = \left\{1,2,\ldots,m\right\},\quad j = \left\{1,2,\ldots,n\right\} \tag{1}$$

and $C_{i,\pi_0} = C_{0,\pi_j} = 0$.

Regarding the total cost TC, it can be computed using Eq. (2).

$$TC = \sum_{i=1}^{m}\sum_{j=1}^{n} pc_{i,u_{ij}}{\cdot}p_{ij,u_{ij}} + \sum_{i=1}^{m} ic_i{\cdot}IT_i \tag{2}$$

where IT_i is the total idle time on machine i, which can be defined according to Eq. (3).

$$IT_i = C_{i,\pi_n} - \sum_{j=1}^{n} p_{ij,u_{ij}} \tag{3}$$

The costs involved in the problem under consideration (processing costs and idle costs) can be linked to sustainable aspects, particularly if we take into account that, in many scenarios, most of these costs are due to state-dependent machine energy consumption (i.e. the energy consumption is different depending on if the machine is in stand-by, or processing a job using a certain speed mode). In addition, if such energy is obtained from the combustion of fossil fuels, then its consumption should be reduced as it is related to the emission of carbon dioxide into the atmosphere. Therefore, several authors have addressed the problem considering an energy-efficient approach. It is the case of Mansouri et al. (2016), Zhang et al. (2017), Zhang et al. (2019), Ramezanian et al. (2019), Öztop et al. (2020), Amiri and Behnamian (2020), which consider the processing cost as:

$$pc_{ik} = \frac{\lambda_k{\cdot}\delta_i}{60} \tag{4}$$

where λ_k and δ_i represent a conversion factor of the energy required to work at speed k and the power consumed by machine i, respectively.

Similarly, for the specific problem of carbon emissions minimisation, Foumani and Smith-Miles (2019) add a conversion factor to CO_2 (denoted as γ) and apply:

$$pc_{ik} = \frac{\lambda_k{\cdot}\gamma{\cdot}\delta_i}{60} \tag{5}$$

Regarding the idle cost, all previous authors assume for the green problem:

$$ic_i = \frac{\varphi_i{\cdot}\delta_i}{60} \tag{6}$$

where φ_i represent a conversion factor of the energy incurred by machine i when is in stand-by (or idle).

The problem under consideration is a multi-objective optimization problem. In this kind of problem, the goal is to obtain the Pareto-optimal solution set (denoted as $\mathcal{P}$) which is composed by all non-dominated solution (see e.g. Fathollahi-Fard et al., 2018). Note that, given two solutions $\mathcal{S}_1$ and $\mathcal{S}_2$, we say that solution $\mathcal{S}_1$ dominates solution $\mathcal{S}_2$ (or equivalently $\mathcal{S}_1 \prec \mathcal{S}_2$) when all objectives obtained by $\mathcal{S}_1$ are equal or better than the ones obtained by $\mathcal{S}_2$ and at least one is better (Minella et al., 2008; Zitzler et al., 2003). As the decision problem associated with at least one of our multi-objective functions corresponds to an NP-hard problem, the multi-objective optimization problem is NP-hard. Thus, heuristic algorithms are required to find good approximations of the Pareto front, taking into account proximity to the front as well as its spreading across the front.

In the literature, several researchers have addressed either the problem under consideration or related variants. Mixed integer linear programming models have been proposed by Fang et al. (2013), Mansouri et al. (2016), Foumani and Smith-Miles (2019). Using a single objective function, Fang et al. (2013) investigate a related flowshop scheduling problem taking into consideration the energy consumption as a constraint (i.e. using a peak-power consumption constraint) instead of as objective function. For this case, a maximal power consumption constraint limits the utilization of the resources in a given production period. Two mixed-integer linear programming formulations are proposed for the makespan minimisation. Furthermore, two fast heuristics are introduced to find feasible schedules as well as to evaluate the trade-off between makespan and peak-power consumption. Secondly, Mansouri et al. (2016) address the scheduling of a two-machine flowshop environment for the minimisation of makespan and energy consumption. A mixed-integer linear programming formulation is proposed to find the Pareto front to both objectives. A lower bound for both objectives are proposed, as well as a fast heuristic that finds good approximations for the Pareto front. Finally, Foumani and Smith-Miles (2019) address the flowshop scheduling problem with the bi-objective of makespan and total carbon emission minimisation. A mixed-integer linear programming formulation is proposed, and a weight aggregation approach converted the original problem into a single objective one. These authors employed random data and real-based data in computational experiments.

Regarding approximated algorithms proposed in the literature, the most related approaches are the proposals by Mokhtari et al. (2011), Öztop et al. (2020). The former considers a flowshop with CPT where a trade-off between makespan and the required amount of resources is sought to minimize. The authors decompose the problem into two subproblems (a sequencing problem and a resource allocation problem) and propose a solution procedure combining a discrete differential evolution algorithm (DDE) with a variable neighbourhood search (VNS). Regarding the latter, Öztop et al. (2020) address an energy-efficient flowshop scheduling problem for the minimisation of total flow time and total energy consumption. These authors present a bi-objective mixed-integer programming model formulation with a speed-scaling framework. An improved version of the well-known NEH algorithm (Nawaz et al., 1983) is proposed as initial solution. Furthermore, two variants of the iterated greedy metaheuristic and variable block-insertion algorithm are proposed. The iterated greedy algorithms presented better results in the computational experiments carried-out with randomly generated test instances. Finally, note that several interesting researches have also been proposed for related scheduling problem considering either no-wait constraints (see e.g. Gao et al., 2018; Gomes et al., 2020), either different setup times approaches (see e.g. Ramezanian et al., 2019), or hybrid flowshop layouts (see e.g. Behnamian and Ghomi, 2011, 2015, 2019, 2002).

Summarising, the present literature review highlights that several authors have addressed some related scheduling problems and therefore, although there is no optimization procedure for the problem under consideration, we found some metaheuristics dealing with related ones. These metaheuristics use novel mechanisms to conduct the local searches, yet they do not use problem-specific properties in their phases, a fact that can be exploited to substantially improve the performance of these methods. To cover this gap, in the following section we introduce some theoretical results, which will be incorporated in our proposal in Section 4.

3. Theoretical results

In this section, we detail some theoretical results necessary to develop our proposal. Firstly, we apply some common definitions of the scheduling literature to the problem under consideration: critical path, forward and backward decoding procedures, and slack time. Using them, we can prove four theorems that are applied in the proposed iterated local search-based metaheuristic (see Section 4) to improve its efficiency.

Let us start by defining two different decoding procedures to obtain a schedule from a solution of the problem $\mathcal{S} = (\Pi, \mathcal{U})$. Firstly the *Forward Decoding* obtains a semi-active schedule (see definition in Pinedo (2012)) inserting jobs from the left to the right of Π, while the *Backward Decoding* considers the reverse sequence executed in the reverse order of machines (see e.g. Ribas et al., 2010; Ribas et al., 2013). A formal definition of both procedures, applied to the problem under consideration, is detailed next:

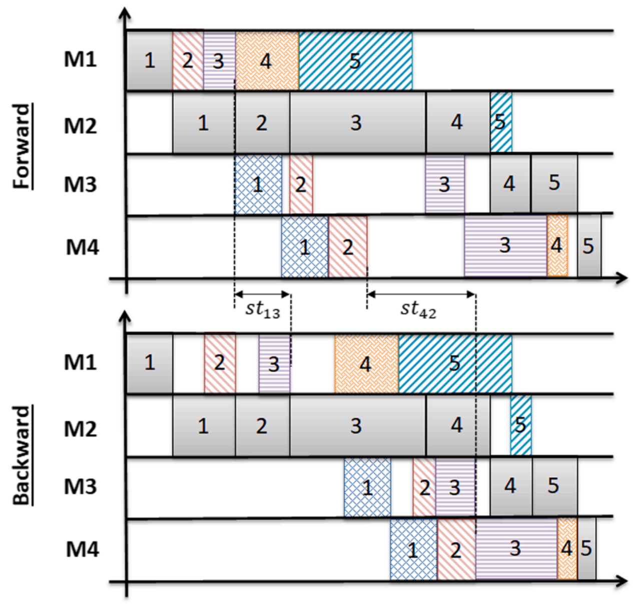

Fig. 1. Example for theoretical results.

Definition 3.1. (Forward Decoding) Let $\mathcal{S} = (\Pi, \mathcal{U})$, with $\Pi := (\pi_1, \ldots, \pi_n)$, be a solution for the $\mathcal{I}$ instance of the $Fm\left|prmu, p_{ij}(u_{ij})\right|\#(C_{max}, Cost)$ problem. Then, the forward decoding constructs a solution by processing each job π_j on each machine i (with $j \in \{1, \ldots, n\}$ and $i \in \{1, \ldots, m\}$) according to Eq. (1), with $C^{FD}_{i,\pi_0} = C_{0,\pi_j} = 0$ and $C^{FD}_{ij} = C_{ij}$ $(\forall j \in \{1, \ldots, n\}, \forall i \in \{1, \ldots, m\})$.

Definition 3.2. (Backward Decoding) Let $\mathcal{S} = (\Pi, \mathcal{U})$, with $\Pi := (\pi_1, \ldots, \pi_n)$ and $\mathcal{U} = (u_{11}, \ldots, u_{ij}, \ldots, u_{mn})$, be a solution for the $\mathcal{I}$ instance of the $Fm\left|prmu, p_{ij}(u_{ij})\right|\#(C_{max}, Cost)$ problem. Then, the backward decoding constructs a solution by processing each job π_j on each machine i following the reverse sequence of jobs and machines ($j \in \{n, n-1, \ldots, 1\}$ and $i \in \{m, m-1, \ldots, 1\}$), i.e. starting by processing job π_n on machine m. The completion time of operation $O_{i\pi_j}$ (denoted by C^{BD}_{i,π_j}) is defined by Eq. (7). Note that $\overline{C}_{ij}$ is the reverse completion time (i.e. starting from the final operation) defined in Eq. (8), where $\overline{C}_{i,\pi_{n+1}} = \overline{C}_{m+1,\pi_j} = 0$ $(\forall j \in \{n, n-1, \ldots, 1\}, \forall i \in \{m, m-1, \ldots, 1\})$.

$$C^{BD}_{i,\pi_j} = C_{max} - \overline{C}_{i,\pi_j} + p_{i,\pi_j,u_{i\pi_j}}, \forall j \in \{n, n-1, \ldots, 1\}, \forall i \in \{m, m-1, \ldots, 1\} \quad (7)$$

$$\overline{C}_{i,\pi_j} = \max\left\{\overline{C}_{i+1,\pi_j}, \overline{C}_{i,\pi_{j+1}}\right\} + p_{i,\pi_j,u_{i\pi_j}}, \forall j \in \{n, n-1, \ldots, 1\}, \forall i \in \{m, m-1, \ldots, 1\} \quad (8)$$

An example of forward and backward codifications is shown in Fig. 1 for five jobs and four machines. At the top of the figure, a schedule is constructed from the sequence (1,2,3,4,5) using the forward decodification, while alternatively at the bottom of the figure its backward decoding is shown.

We can observe that the makespan obtained by a solution $\mathcal{S} = (\Pi, \mathcal{U})$ using forward or backward decoding procedure is the same, which is a known property of the PFSP, denoted as *Reversibility of the PFSP* (see e.g. Ribas et al., 2010 for more details). Note that, when the speed modes ($\mathcal{U}$) are fixed, the problem under consideration, $Fm\left|prmu, p_{ij}(u_{ij})\right|\#(C_{max}, Cost)$, is obviously equivalent to the well-known $Fm|prmu|C_{max}$ problem, and in consequence this property applies for our problem. The reversibility in the problem is based on the existence of the *Critical Path* of a solution $\mathcal{S}$, denoted by $\mathcal{R}(\mathcal{S})$, whose definition in the traditional PFSP can be found in Fernandez-Viagas et al. (2020). Nevertheless, for the sake of completeness, we give the definition for the problem under consideration:

Definition 3.3. **(Critical Path)** Let $\mathcal{S} = (\Pi,\mathcal{U})$, with $\Pi := (\pi_1,\ldots,\pi_n)$, be a solution of $\mathcal{I}$ an instance of the $Fm\Big|prmu, p_{ij}(u_{ij})\Big|(C_{max}, Cost)$ problem. Then, considering semi-active schedules, we define the critical path of $\mathcal{S}, \mathcal{C}(\mathcal{S})$, as a path composed of operations whose sum of processing times is equal to the makespan and it can go from the first operation O_{1,π_1} to the last one (O_{m,π_n}) moving always in direction of machines (from i to $i+1$) or jobs (from π_j to π_{j+1}).

In Fig. 1, using the same example, we can observe the critical path for the sequence (1,2,3,4,5) in grey at the top of the figure, formed by Operations $(O_{11}, O_{21}, O_{22}, O_{23}, O_{24}, O_{34}, O_{35}, O_{45})$.

These definitions are the basis of the accelerations proposed by Taillard (1990) (denoted as Taillard's accelerations) for the $Fm|prmu|C_{max}$ problem (see Corollay 3.3 in Fernandez-Viagas et al. (2020) for more details), which explains an efficient mechanism to perform insertion-based methods in the $Fm|prmu|C_{max}$ problem. These accelerations can be applied to the problem under consideration when the modes $\mathcal{U}$ to execute the operations are fixed (Theorem 3.1).

Theorem 3.1. *Let $\mathcal{S} = (\Pi,\mathcal{U})$, with $\Pi := (\pi_1,\ldots,\pi_{n-1})$, be a solution of $\mathcal{I}$ an instance of the $Fm\Big|prmu, p_{ij}(u_{ij})\Big|(C_{max}, Cost)$ problem. Furthermore, let $\overline{C}_{i,\pi_j}$ denote the reverse completion times of $\mathcal{S}$ and $C^{\sigma}_{i,j}$ the completion time of job σ on machine i if this job σ is inserted in position j of Π using the set of modes $u_{i\sigma}$ ($\forall i \in \{1,\ldots,m\}$). Then, the best makespan that can be obtained by inserting σ in any position of Π, maintaining the same modes $\mathcal{U}$ can be computed as*

$$C_{max} = \min_{j=1,\ldots,n}\left\{\max_{i=1,\ldots,m}\left\{C^{\sigma}_{ij} + \overline{C}_{i,\pi_j}\right\}\right\} \tag{9}$$

The proof of this theorem is trivial in view of Corollary 3.3 by Fernandez-Viagas et al. (2020).

Next, let us define the final term *Slack Time* (st_{ij}) as:

Definition 3.4. **(Slack Time)** Let $\mathcal{S} = (\Pi,\mathcal{U})$, with $\Pi := (\pi_1,\ldots,\pi_n)$, be a solution of $\mathcal{I}$ an instance of the $Fm\Big|prmu, p_{ij}(u_{ij})\Big|\#(C_{max}, Cost)$ problem. Then, we define st_{i,π_j} the operation slack time O_{i,π_j} as the difference between the completion time of π_j obtained using forward and backward procedures, i.e. C^{FD}_{i,π_j}, and C^{BD}_{i,π_j} respectively. More specifically:

$$st_{i,\pi_j} = C^{BD}_{i,\pi_j} - C^{FD}_{i,\pi_j} \tag{10}$$

In Fig. 1, the slack times of operations O_{13} and O_{42} in the previous example are shown. Using this definition, we can present Theorems 3.2 and 3.4 that discuss which operations can be compressed or delayed in a schedule without increasing the makespan. Note that these theorems give conditions to reduce the speed modes of certain operations (and, therefore, reducing the total costs in TC) without worsening the makespan objective and will be used in the solution procedure presented in Section 4.

Theorem 3.2. *Let $\mathcal{F}$ be the forward schedule obtained from solution $\mathcal{S} = (\Pi,\mathcal{U})$, with $\Pi := (\pi_1,\ldots,\pi_n)$, using instance $\mathcal{I}$. Then, without altering solution $\mathcal{S}$, the operation O_{i,π_j} can be delayed for a maximum of st_{i,π_j} time units without increasing the makespan.*

Proof. According to the reversibility property, the makespan obtained using the backward decoding is the same as using the forward schedule. Then, by Definition 3.4, the backward schedule represents a feasible schedule where operation O_{i,π_j} finishes st_{i,π_j} time units later than in the forward schedule, but with the same final makespan. Therefore, operation O_{i,π_j} can be delayed for a maximum of st_{i,π_j} time units without increasing the makespan. Now, let us analyze the possibility to delay operation O_{i,π_j} more time units than st_{i,π_j} without increasing the makespan. According to Definition 3.2, the backward decoding generates a semi-active schedule from the right to the left (see Eq. 7). Hence, any delay of an operation in the backward schedule (without altering either the orders of jobs in the machines or the modes applied) must necessary delay that time the final makespan. As a consequence, an operation O_{i,π_j} can be delayed for at maximum st_{i,π_j} time units without increasing the makespan. square

Theorem 3.3. *Let $\mathcal{F}$ be the forward schedule obtained from solution $\mathcal{S} = (\Pi,\mathcal{U})$, with $\Pi := (\pi_1,\ldots,\pi_n)$, using instance $\mathcal{I}$ with critical path $\mathcal{R}(\mathcal{S})$. All operations in $\mathcal{R}(\mathcal{S})$ do not have slack time, i.e. $st_{O_{ij}} = 0, \forall O_{ij} \in \mathcal{R}$.*

Proof. By the definition of the critical path, there can be no idle or waiting times between operations belonging to the critical path, i.e. they cannot be contracted or expanded without altering solution $\mathcal{S}$. As a consequence, the completion times of these operations must remain unaltered if either forward or backward decoding is applied and, consequently their slack time is 0. square

Corollary 3.1. Let $\mathcal{F}$ be a forward schedule for instance $\mathcal{I}$ obtained from solution $\mathcal{S}$ and $\mathcal{R}(\mathcal{S})$ be the corresponding critical path. Then, a delay of δ time units in any operation in $\mathcal{R}(\mathcal{S})$ increases the makespan by exactly δ units.

The proof of the corollary is trivial according to Theorem 3.3 and Definition 3.3.

Theorem 3.4. *Let $\mathcal{F}$ be the forward schedule obtained from solution $\mathcal{S} = (\Pi,\mathcal{U})$, with $\Pi := (\pi_1,\ldots,\pi_n)$, using instance $\mathcal{I}$ with critical path $\mathcal{R}(\mathcal{S})$. Then, applying $\mathcal{F}$, any reduction in the completion time of an operation O_{i,π_j} not belonging the critical path cannot reduce the final makespan.*

The proof of this theorem is trivial by Definition 3.3 and using the same reasoning as in 3.3.

Theorem 3.5. *Let $\mathcal{F}$ be the forward schedule obtained from solution $\mathcal{S} = (\Pi,\mathcal{U})$, with $\Pi := (\pi_1,\ldots,\pi_n)$, using instance $\mathcal{I}$ with critical path $\mathcal{R}(\mathcal{S})$. Then, any reduction in the completion time of an operation O_{i,π_j} belonging the critical path could potentially reduce the final makespan C_{max}.*

Proof. By Definition 3.3, if some operation in the critical path is reduced, there are two possibilities:

1. Some idle time is inserted between operations in the critical path. In this case, there exists a new critical path with makespan less or equal than C_{max};
2. Otherwise, the critical path remains the same, and then, the makespan is reduced such amount of time. square

The previous theorems and corollaries establish a guideline to be followed if some operation is advanced or delayed in the schedule. Note that in the problem under consideration, this fact is very relevant as the processing times of the operations (and consequently their completion times) depend on the modes employed to execute them.

As a summary of the previous theoretical results, we can conclude that:

- Given a solution $\mathcal{S} = (\Pi,\mathcal{U})$, Taillard's accelerations represent an efficient method for reducing the complexity of computing the makespan if job σ is inserted in the best position of Π without changing the set of modes $\mathcal{U}$ (Theorem 3.1).
- Regarding the operations in the critical path:
  - A delay in the completion time in any of these operations causes an analogous increase in the final makespan.

```
Procedure CP_ILS($\Pi^{LPT}$)
    $\mathcal{P}$ := empty;
    // Initial Solution
    for $j = 1$ to $n$ do
        $u_j^i$ = Random speed mode (from 1 to $K$);
    end
    $\Pi^i = \text{NEH}(\Pi^{LPT}, u_j^i)$;
    $\Pi^{ii} = \text{LS\_S}(\Pi^i, u_j^i)$;
    Introduce $\mathcal{S} = \{\Pi^{ii}, u_j^i\}$ in $\mathcal{P}$;
    $(\mathcal{P}, \Pi^{ii}, u_j^{ii}) = \text{LS\_M}(\mathcal{S}, \mathcal{P})$;
    $\Pi^r := \Pi^{ii}$;
    $u_j^r := u_j^{ii}$;
    $C_{max}^r := C_{max}(\Pi, u_j)$;
    while stopping criterion is not met do
        //Perturbation Phase
        $\Pi$:=Permutation obtained after performing $d$ random adjacent interchanges on $\Pi^r$;
        //Type $\mathcal{T}^S$
        $\Pi^{'} = \text{LS\_S}(\Pi, u_j^r)$;
        Introduce $\mathcal{S} = \{\Pi^{'}, u_j^r\}$ in $\mathcal{P}$, if that solution is non-dominated;
        //Type $\mathcal{T}^U$
        $(\mathcal{P}, \Pi^{'}, u_j^{'}) = \text{LS\_M}(\mathcal{P}, \mathcal{S})$;
        for $i = 1$ to $m$ do
            $u_{ij} = u_j^{'}, \forall j \in \mathcal{J}$;
        end
        $(\mathcal{P}, \Pi^{'}, u_{ij}^{'}) = \text{LS\_NCP}(\mathcal{P}, \Pi^{'}, u_{ij})$;
        $(\mathcal{P}, \Pi^{'}, u_j^{''}) = \text{LS\_CP}(\mathcal{P}, \Pi^{'}, u_{ij}^{'})$;
        //Simple simulated annealing procedure
        if $C_{max}(\Pi^{'}, u_j^{'}) < C_{max}^r$ then
            $\Pi^r := \Pi^{'}$;
            $u_j^r := u_j^{'}$;
            $C_{max}^r := C_{max}(\Pi^{'}, u_j^{'})$;
        else if $random \leq exp\{-(C_{max}(\Pi^{'}, u_j^{'}) - C_{max}^r)/Temperature\}$ then
            $\Pi^r := \Pi^{'}$;
            $u_j^r := u_j^{'}$;
            $C_{max}^r := C_{max}(\Pi^{'}, u_j^{'})$;
        end
    end
end
```

Fig. 2. Critical-path based iterated local search.

- A reduction of the completion time in any of these operations (due, e.g., to reducing its processing time) can eventually reduce the final makespan.

- For any operation O_{ij} outside the critical path:
  - A delay δ (with $\delta < st_{ij}$) in the completion time of such an operation does not increase the final makespan.
  - Reducing the processing time of such an operation does not reduce the final makespan.

These theoretical results are embedded in a local search-based metaheuristic presented in the next section.

4. Critical-Path based Iterated Local Search, CP_ILS

In this section, we detail the proposed critical-path based iterated local search, denoted as CP_ILS, to obtain a Pareto set of solutions, denoted as $\mathcal{P}$. Starting from an initial solution, the traditional iterated local search iteratively perturbs a solution and searches in its neighbourhood for the best local solution. Despite being a basic procedure, this type of metaheuristic has found excellent results in solving permutation-based scheduling problems when a very efficient local search is applied, being state-of-the-art in several related scheduling problems (see e.g. Dong et al., 2013; M'Hallah, 2014; Subramanian et al., 2014). In each iteration of our proposal, after applying a simple perturbation mechanism (see Section 4.2 for more details), we carry out an intensive search using two different types of local search methods (types $\mathcal{T}^S$ and $\mathcal{T}^U$) that do not modify either the sequence of jobs or the speed modes. More specifically, the specific search of the proposal is composed of the following four different local search methods (one of type $\mathcal{T}^S$ and three of type $\mathcal{T}^U$) as follows:

- Type $\mathcal{T}^S$: An improvement in the current solution $\mathcal{S} := (\Pi, \mathcal{U})$ is sought by altering thesequence Π while maintaining $\mathcal{U}$.
  - The first proposed local search method performs insertion-based movements in Π without changing $\mathcal{U}$. Under these conditions, the requirements of Theorem 3.1 are fulfilled. In order to take advantage of it, we search for the best solution after inserting each job in each position. This local search is denoted LS_S. See Section 4.3 for more details.
- Type $\mathcal{T}^U$: An improvement in the current solution $\mathcal{S} := (\Pi, \mathcal{U})$ is sought by altering the modes configuration $\mathcal{U}$ while maintaining Π. For this type, the following local search methods are proposed:
  - LS_M: In this local search, we look for new solutions by changing the current speed mode of a job and assuming that this mode is used for all machines. The procedure is explained in detail in Section 4.4.

```
Procedure NEH($\Pi^{LPT}, u_j$)
    $\Pi^{LPT}$ := Jobs ordered by non-decreasing sum of the processing times, i.e. $p_j = \sum_{i=1}^{m}\{p_{ij}\}$, where $\Pi^{LPT} = (\pi_1^{LPT}, \ldots, \pi_n^{LPT})$;
    $\Pi^{NEH} := \{\pi_1^{LPT}\}$;
    for $l = 2$ to $n$ do
        Test job $\pi_l^{LPT}$ in any possible position of $\Pi^{NEH}$ and calculate its makespan considering $u_j$ as the modes configuration (and applying Theorem 3.1).
        $\Pi^{NEH}$ := permutation obtained by inserting $\pi_l^{LPT}$ in the position of $\Pi^{NEH}$ with less makespan;
    end
end
```

Fig. 3. NEH algorithm.

- LS_NCP: In this critical-path based local search, we use Theorem 3.2 and decrease the speed mode of the operations outside the critical path. The procedure is explained in detail in Section 4.5.
- LS_CP: The second critical-path based local search applies Theorem 3.5 and increases the speed mode of the operations in the critical path. The procedure is explained in detail in Section 4.6.

A global pseudo-code of CP_ILS is shown in Fig. 2, and in the next sections we describe its main parts.

4.1. Initial solution

To generate an initial solution, we first generate the mode of each operation O_{ij}. For simplicity, we assign a random mode for every job, and this mode is used for all machines, i.e. $u_{ij} = u_j, \forall i, j$. Once the set of nodes has been assigned, a simple NEH algorithm is applied to construct the initial sequence. This well-known algorithm (originally proposed by Nawaz et al. (1983) for the $Fm|prmu|C_{max}$ problem) obtains a sequence by sorting the jobs in Longest Processing Times order (denoted by $\Pi^{LPT} = (\pi_1^{LPT}, \ldots, \pi_n^{LPT})$). The initial sequence, denoted by Π^{NEH}, is then constructed by iteratively inserting job π_j^{LPT} ($j \in \{2, \ldots, n\}$) in Π^{NEH}, in the position that yields the minimal (partial) makespan. In the first iteration, Π^{NEH} is composed of a single job, i.e. $\Pi^{NEH} = (\pi_1^{LPT})$. The pseudo-code to construct the initial solution is shown in Fig. 3. Note that, as stated in Theorem 3.1, Taillard's accelerations can be applied to compute such a minimal makespan. Hence, a calculation of the total cost is not performed in this phase.

After a sequence is obtained using the NEH algorithm, the local search method LS_S is applied and the so-obtained solution is appended to $\mathcal{P}$ the Pareto set of solutions found by the algorithm, initially empty. Then, the local search method LS_M is applied to improve the solution. These two methods are discussed in detail in Sections 4.3 and 4.4, respectively. Once the two methods are applied, the algorithm performs a series of iterations until the stopping criterion (i.e. running time) is met. The first step in the iterations consists of the so-called perturbation phase, which is described in the next section.

4.2. Perturbation phase

In the perturbation phase, a different element from the space of solutions is intensively explored using advanced local search mechanisms. To ensure the algorithm performance, this new element should satisfy the following issues: firstly, it should have some similarity to a previous element to avoid a completely new restart in each iteration, which would make it difficult to find a high-quality local optimum; secondly, the new element should be different enough from the previous one so that the new local optimum could potentially be different. In this regard, Fernandez-Viagas et al. (2018) explored this trade-off by comparing eight different perturbation mechanisms. Among them, the best result was obtained by a simple perturbation composed of several random adjacent swaps. This perturbation is also considered in the proposed metaheuristic. More specifically, in this phase, a job is selected randomly and interchanged with the next job of the sequence. This procedure is repeated d times, where d is a parameter of the algorithm to be calibrated.

4.3. Local search: LS_S

In this section, we detail the local search method denoted by LS_S. This search starts with a sequence of jobs (Π) and a speed mode for each job that is maintained for all machines ($u_{ij} = u_j, \forall i$). Then, a local search is carried out by changing the sequence of jobs, without altering the set of speed modes (type $\mathcal{T}^S$ of local search methods). Thereby, each job of the sequence is removed from the sequence and reinserted in the position that minimises the makespan. Note that, as the modes configuration is the same for all sequences tested, Theorem 3.1 can be applied to accelerate the computation. This procedure is repeated until no improvement is found. A detailed pseudo-code is shown in Fig. 4.

4.4. Local search: LS_M

The *LS_M* method is our first proposed local search of type $\mathcal{T}^U$. This simple search receives a sequence of jobs, Π, and a mode configuration

```
Procedure LS_S($\Pi, u_j$)
    $\Pi^{ini} := \Pi$;
    $C_{max}$ := Makespan of solution $(\Pi, u_j)$;
    $flag$ := true;
    while $flag = true$ do
        $flag$ = false;
        for $l = 1$ to $n$ do
            Test job $\pi_k^{ini}$ in any possible position of $\Pi$ and calculate its makespan considering $u_j$ as the configuration of modes (applying Theorem 3.1).
            $\Pi$ := permutation obtained by inserting $\pi_k^{ini}$ in the position of $\Pi$ with less makespan (denote as $C_{max}^l$;
            if $C_{max}^l < C_{max}$ then
                $flag$ := true;
                $C_{max} := C_{max}^l$;
            end
        end
    end
end
```

Fig. 4. LS_S function.

```
Procedure $LS\_M(\mathcal{S},\mathcal{P})$
    $\mathcal{S}^{ite} := \{\Pi, u_j\}$;
    $C_{max}$ := Makespan of solution $(\Pi, u_j)$;
    $TC$ := Total cost of of solution $(\Pi, u_j)$;
    for $j = 1$ to $n$ do
        $u_j = u_j + 1$;
        if $u_j > K$ then
            $u_j = 1$;
        end
        $\mathcal{S} := \{\Pi, u_j\}$;
        if $\mathcal{S} \prec \mathcal{S}^{ite}$ then
            $\mathcal{S}^{ite} = \mathcal{S}$;
        else
            $u_j = u_j - 1$;
        end
        if $\mathcal{S}$ is non-dominated by a solution in $\mathcal{P}$ then
            Introduce $\mathcal{S}$ in $\mathcal{P}$;
        end
    end
    return $\{\mathcal{S}^{ite}, \mathcal{P}\}$
end
```

Fig. 5. LS_M function.

```
Procedure $LS\_NCP(\Pi, u_{ij}, \mathcal{P})$
    $C_{ij}^{FD}$ := Completion times of each job $j$ on machine $i$ using the forward decoding (see Definition 3.1);
    $C_{ij}^{BD}$ := Completion times of each job $j$ on machine $i$ using the backward decoding (see Definition 3.2);
    $st_{ij} := C_{ij}^{BD} - C_{ij}^{FD}, \forall i, j$;
    for $i = 1$ to $m$ do
        for $j = 1$ to $n$ do
            if $st_{ij} > 0$ then
                $p^{ref} = p_{ij}/s_{i,u_{ij}}$;
                for $k = u_{ij} + 1$ to $K$ do
                    $p^{aux} = p_{ij}/s_{i,k}$;
                    if $p^{aux} - p^{ref} <= st_{ij}$ then
                        $u_{ij} = k$;
                        Introduce $\mathcal{S} = \{\Pi, u_{ij}\}$ in $\mathcal{P}$, if that solution is non-dominated;
                    end
                end
            end
        end
    end
    return $\{\mathcal{P}, \Pi, u_{ij}\}$
end
```

Fig. 6. LS_NCP function.

for each job j, u_j (with $j \in \{1,\ldots,n\}$), which is used to obtain the speed to be set in every machine. Iteratively, the procedure selects a job j (with $j \in \{1,\ldots,n\}$) and increases its mode one unit. If the resulting mode is greater than K, it is set to the first mode. Then, the procedure firstly checks if the new solution belongs to the set of non-dominated solutions and, second, it replaces the current solution if it has been dominated. A detailed pseudo-code is shown in Fig. 5.

Note that this local search is not very intensive. The idea here is to have a good initial configuration of modes to start the next local method. Hence, by considering both the same mode in all machines and a unique change in each mode, we can accelerate the search to set up a good initial configuration of modes. An excess of intensification in this phase could be counterproductive, since in the following phases (i.e. LS_NCP and LS_CP) the mode in each machine is going to be altered.

4.5. Critical-path based local search: LS_NCP

The second local search of type $\mathcal{T}^U$ is based on the concept of critical path (see Definition 3.3) and and more specifically, on Theorem 3.2. Note that this theorem states that an operation outside the critical path can be delayed by st_{ij} time units without increasing the makespan of the solution. Hence, the proposed method first calculates the slack time st_{ij} of every operation O_{ij} by performing the forward and backward decoding of the initial solution (see Eq. (10)). Then, these slack times are tried to be reduced by increasing the processing times of the corresponding operations. To do so, in each iteration, the speed of job j on machine is tried to be increased. If the increase in the processing time is lower than the corresponding slack time, the new speed replaces the last one and the values of the objective functions are computed. Finally, if the so-obtained solution is non-dominated, it is introduced in $\mathcal{P}$, see the pseudo-code in Fig. 6 for more details. Note that the slack times (st_{ij}) are not recomputed in each iteration in order to reduce the computational effort of the search.

4.6. Critical-path based local search: LS_CP

The last local search of type $\mathcal{T}^U$ is based on the operations in the critical path (see Definition 3.3). In this case, according to Theorem 3.5 only these operations could potentially decrease the makespan. Therefore, as in the previous local search method, the slack time of each operation is first calculated using Eq. (10). Once these are computed, the speed of job j on machine i is iteratively decreased and the corresponding values of the objective functions are computed. If the so-obtained solution is non-dominated, it is introduced in $\mathcal{P}$, and is set as the current solution. Then, the whole procedure is repeated while there are some improvements, as illustrated in Fig. 7. As in the previous case, the slack time (st_{ij}) is not recomputed once a new solution is found to reduce the computational effort of the method.

4.7. Simulated annealing procedure

Finally, in the last step of the metaheuristic, a simple simulated annealing procedure is applied to decide the current solution ($\mathcal{S} = \{\Pi, u_j\}$)[1]. To deal with that, a simulated annealing phase with a constant

[1] Note that the current solution includes only one mode for each job j, and is consequently assuming that the same mode is applied to all machines

```
Procedure $LS\_CP(\Pi, u_{ij}, \mathcal{P})$
    $C_{ij}^{FD}$ := Completion times of each job $j$ on machine $i$ using the forward decoding (see Definition 3.1);
    $C_{ij}^{BD}$ := Completion times of each job $j$ on machine $i$ using the backward decoding (see Definition 3.2);
    $st_{ij} := C_{ij}^{BD} - C_{ij}^{FD}, \forall i, j;$
    $u_{ij}^{'} = u_{ij};$
    $flag :=$ true;
    while $flag = true$ do
        $flag =$ false;
        for $i = 1$ to $m$ do
            for $j = 1$ to $n$ do
                if $st_{ij} = 0$ then
                    $p^{ref} = p_{ij}/s_{i,u_{ij}^{'}};$
                    for $k = u_{ij}^{'} - 1$ to $0$ do
                        $p^{aux} = p_{ij}/s_{i,k};$
                        if $p^{aux} - p^{ref} <= st_{ij}$ then
                            $u_{ij} = k;$
                            if $\mathcal{S} = \{\Pi, u_{ij}\}$ is non-dominated for any solution in $\mathcal{P}$ then
                                Introduce $\mathcal{S}$ in $\mathcal{P}$;
                                $flag :=$ true;
                            else
                                $u_{ij} = u_{ij}^{'};$
                            end
                        end
                    end
                end
            end
        end
    end
    return $\{\mathcal{P}, \Pi, u_{ij}\}$
end
```

Fig. 7. LS_CP function.

temperature is applied (see e.g. Fernandez-Viagas and Framinan, 2015b; Pan et al., 2008; Ruiz and Stützle, 2007). In this phase, a new solution $\mathcal{S}^{'}$, is kept in the next iteration if the makespan in the last iteration has been outperformed:

$$random \leqslant exp\{-(C_{max}(\mathcal{S}^{'}) - C_{max}(\mathcal{S}))/Temperature\} \tag{11}$$

where *Temperature* can be computed using parameter T as:

$$Temperature = T \cdot \frac{\sum_{\forall i}\sum_{\forall j} p_{ij}}{n \cdot m \cdot 10} \tag{12}$$

5. Computational evaluation

In this section, we check the effectiveness of the proposed metaheuristic by comparing it against the most relevant metaheuristics proposed in the related literature for the same problem. This comparison is performed using three sets of medium-large instances. Both sets are detailed in Section 5.1. The indicators chosen to compare the algorithms are shown in Section 5.2, while the algorithms implemented are detailed in Section 5.3. Finally, the computational results are presented in Section 5.4.

5.1. Benchmark

According to Section 2, the following parameters completely define an instance of the problem: n,m,S,p_{ijk},s_{ik},pc_{ik}, and ic_i. In this paper, these parameters have been chosen to cover a wide scope of the problem under consideration as well as its special case of energy consumption minimisation (green approach). Next, we explain the generation of the parameters:

- Number of machines: Analogously, we consider equidistant values ($m \in \{5, 10, 15, 20\}$) based on Ramezanian et al. (2019).
- Processing times: They are generated from a uniform distribution between 1 and 99 (see e.g. Foumani and Smith-Miles, 2019; Mansouri et al., 2016; Öztop et al., 2020).
- Number of modes: Three modes ($K = 3$) are generated to represent the different types of speeds of each machine, following the most common approach used in the literature (see e.g. Amiri and Behnamian, 2020; Foumani and Smith-Miles, 2019; Mansouri et al., 2016; Öztop et al., 2020; Ramezanian et al., 2019; Zhang et al., 2019)
- Type of benchmark: For these parameters, three types of instances are generated ($\mathcal{B}_1$, $\mathcal{B}_2$, and $\mathcal{B}_3$) representing different cases of the problem under consideration. The rest of the parameters (i.e. n, s_{ik}, pc_{ik}, and ip_i) are defined according to the following:
  - Set of instances $\mathcal{B}_1$. These instances are designed to test the algorithms on a generic random benchmark. For the number of jobs, we select equidistant values using bounds similar to Zhang et al. (2019), Mansouri et al. (2016), i.e., we select $n \in \{40, 80, 120, 160\}$. Then, 30 iterations are generated for each combination of n and m, resulting in a total of 480 instances. The speed parameter (s_{ik}) is generated using a uniform distribution $U[0.8, 1.2]$ (the same distribution as Öztop et al. (2020) but with bounds taken from Mansouri et al. (2016)). Three values are generated for each machine i and the highest value is assigned to the mode $k = 1$, the medium value to the mode $k = 2$, and the lowest value to the mode $k = 3$. Regarding the costs, three processing costs (pc_{ik}) are generated using a uniform distribution $U[0.5, 2.0]$ for each machine i. Analogously, the highest, medium and lowest values are assigned to modes 1, 2 and 3, respectively. Finally, the idle costs are generated using $U[0.025, 0.100]$.

Table 1
N_{NDS} of each metaheuristic grouped by n and m in benchmark $\mathcal{B}_1$. In bold, we represent the best found value for each instance and stopping criterion.

$n \times m$	$t = 60$						$t = 90$						$t = 120$					
	HDDE	IG	IG_{ALL}	VBIH	CP_ILS	Total	HDDE	IG	IG_{ALL}	VBIH	CP_ILS	Total	HDDE	IG	IG_{ALL}	VBIH	CP_ILS	Total
40 × 5	13.58	2.17	2.46	0.69	**179.75**	198.65	17.28	2.45	2.41	0.61	**177.07**	199.81	15.77	2.59	2.80	0.68	**178.63**	200.46
40 × 10	10.08	1.73	1.95	0.59	**255.25**	269.60	8.24	1.95	2.50	0.54	**250.18**	263.41	11.10	2.03	2.65	0.56	**256.24**	272.58
40 × 15	1.52	1.77	1.95	0.45	**403.08**	408.77	4.01	1.99	2.10	0.45	**400.40**	408.95	4.91	2.43	2.41	0.48	**398.79**	409.01
40 × 20	1.13	1.93	1.79	0.55	**476.93**	482.32	1.82	1.97	2.22	0.51	**480.65**	487.18	3.26	2.23	2.46	0.46	**473.49**	481.90
80 × 5	24.79	1.82	1.93	0.87	**275.81**	305.22	30.14	2.13	2.22	0.84	**288.14**	323.47	35.79	2.36	2.45	0.74	**280.70**	322.03
80 × 10	2.16	1.27	1.32	0.45	**517.72**	522.91	2.32	1.35	1.78	0.51	**520.51**	526.47	5.54	1.65	1.88	0.47	**522.13**	531.67
80 × 15	0.19	1.12	1.36	0.51	**696.96**	700.15	0.43	1.57	1.54	0.51	**691.22**	695.26	1.04	1.61	1.66	0.48	**703.53**	708.33
80 × 20	0.41	1.25	1.30	0.53	**949.59**	953.07	0.46	1.29	1.45	0.49	**948.30**	951.99	0.41	1.35	1.75	0.45	**958.71**	962.66
120 × 5	8.12	1.65	1.77	0.86	**454.79**	467.18	19.18	2.06	1.88	0.89	**447.92**	471.93	23.69	2.09	2.03	0.97	**444.23**	473.00
120 × 10	1.37	1.56	1.57	0.75	**741.49**	746.74	1.00	2.01	2.19	0.66	**746.77**	752.62	2.35	2.19	1.95	0.73	**749.01**	756.23
120 × 15	0.04	0.79	1.00	0.43	**1075.63**	1077.90	0.48	1.11	1.45	0.44	**1070.26**	1073.74	0.31	1.13	1.16	0.47	**1076.43**	1079.51
120 × 20	0.00	0.93	1.10	0.51	**1387.41**	1389.95	0.00	1.05	1.18	0.44	**1390.77**	1393.45	0.10	1.45	1.22	0.48	**1391.91**	1395.17
160 × 5	0.00	2.25	2.27	1.19	**675.84**	681.55	8.47	2.17	2.09	1.14	**671.61**	685.49	18.21	2.17	2.18	1.17	**651.23**	674.96
160 × 10	0.00	1.39	1.59	0.70	**1040.63**	1044.32	1.01	1.43	1.49	0.71	**1031.02**	1035.66	0.75	1.57	1.49	0.74	**1041.47**	1046.02
160 × 15	0.40	0.96	1.24	0.59	**1288.07**	1291.27	1.16	1.11	1.19	0.49	**1307.73**	1311.68	1.37	0.97	1.24	0.44	**1306.15**	1310.18
160 × 20	0.25	1.23	1.27	0.63	**1626.67**	1630.04	0.34	1.13	1.25	0.55	**1612.29**	1615.55	0.20	0.95	1.29	0.63	**1640.31**	1643.38
Average	4.00	1.49	1.62	0.64	752.85	**760.60**	6.02	1.67	1.81	0.61	752.18	**762.29**	7.80	1.80	1.91	0.62	754.56	766.69

– Set of instances $\mathcal{B}_2$. This set is designed to tackle the green special case of the problem and is composed of 480 instances varying the previous combinations of n and m ($n \in \{40, 80, 120, 160\}$ and $m \in \{5, 10, 15, 20\}$). To generate this set, we use the same approach as Öztop et al. (2020), Amiri and Behnamian (2020). More specifically, we use $\lambda_k \in \{1.5, 1.0, 0.6\}, \varphi = 0.05$, and $\delta_i = 60$. Replacing these values in Eqs. 4 and 6, we have $ic_i = 0.05$ and $pc_{ik} = pc_k = \{1.5, 1.0, 0.6\}$ for modes 1, 2, and 3, respectively. Regarding the speed parameter, s_{ik}, it is generated for three modes, regardless of the machine, as $s_{ik} = s_k \in \{1.2, 1.0, 0.8\}$.

– Set of instances $\mathcal{B}_3$. This set is designed to test the algorithms in big-size instances ($n \in \{300, 400, 500\}$). The rest of the parameters are generated using the same distributions as $\mathcal{B}_1$. This benchmark is composed of 30 instances for each combination of n and m, resulting in a total of 360 instances.

5.2. Indicators

The literature is abundant in the use of indicators to analyse the performance of multi-objective approaches (for the interested reader, we refer to Zitzler et al. (2003)). In our case, in order to have a fair comparison with the algorithms implemented, we applied indicators similar to those of Öztop et al. (2020). Before introducing such indicators to evaluate the algorithms, let us assume that $\mathcal{P}^A$ is the Pareto set of solutions found by Algorithm A. In addition, let $\mathcal{P}^*$ be the reference Pareto set composed by the best non-dominated solutions found in an instance, formed by the solutions reached by any of the implemented algorithms. By doing so, based on the literature, we can measure the efficiency of the algorithms by using the following indicators to compare the Pareto set of each different algorithm. Note that these indicators try to measure the density, spread and quality of each set of solutions. These indicators are as follows:

- Number of non-dominated solutions (see e.g. Fathollahi-Fard et al., 2018; Kacem and Dammak, 2019; Pan et al., 2009), N_{NDS}. This indicator represents the number of global non-dominated solutions found by Algorithm A.
- I_R: Ratio of non-dominated solutions (see e.g. Pan et al., 2009). It represents the ratio of global non-dominated solutions found for Algorithm A with respect to the total solutions in $\mathcal{P}^*$, denoted as I_R(see Eq. 13).

$$I_R\left(A\right) = \frac{N_{NDS}(A)}{|\mathcal{P}^*|} \tag{13}$$

- I_A: Average distance to $\mathcal{P}^*$ (Pan et al., 2009; Zhang and Li, 2007). It is the mean of the normalized Euclidean distance between each element in $\mathcal{P}^*$ and the closest element in the Pareto set of algorithm A, i.e. $\mathcal{P}^A$. This normalized distance for each element $y \in \mathcal{P}^*$, denoted $d_y(\mathcal{P}^A)$, can be defined by Eq. (14), where $Cmax^{max}$ ($Cmax^{min}$) is the maximum (minimum) makespan value found in $\mathcal{P}^*$ and similarly, $Cost^{max}$ ($Cost^{min}$) is the maximum (minimum) cost in $\mathcal{P}^*$. Once $d_y(\mathcal{P}^A)$ is known, I_A is its mean value ($I_A = \overline{d}(\mathcal{P}^A)$) according to Eq. (15). A lower value of this indicator is desirable.

$$d_y\left(\mathcal{P}^A\right) = \min_{x \in \mathcal{P}^A}\left\{\left(\frac{Cmax(x) - Cmax(y)}{Cmax^{max} - Cmax^{min}}\right)^2 + \left(\frac{Cost(x) - Cost(y)}{Cost^{max} - Cost^{min}}\right)^2\right\} \tag{14}$$

$$I_A = \overline{d}\left(\mathcal{P}^A\right) = \frac{\sum_{y \in \mathcal{P}^*} d_y\left(\mathcal{P}^A\right)}{|\mathcal{P}^*|} \tag{15}$$

Table 3
I_R of each metaheuristic grouped by n and m in benchmark $\mathcal{B}_1$. In bold, we represent the best found value for each instance and stopping criterion.

$n \times m$	$t = 60$					$t = 90$					$t = 120$				
	HDDE	IG	IG_{ALL}	VBIH	CP_ILS	HDDE	IG	IG_{ALL}	VBIH	CP_ILS	HDDE	IG	IG_{ALL}	VBIH	CP_ILS
40 × 5	0.08	0.01	0.01	0.00	**0.90**	0.09	0.01	0.01	0.00	**0.88**	0.08	0.01	0.01	0.00	**0.89**
40 × 10	0.05	0.01	0.01	0.00	**0.94**	0.04	0.01	0.01	0.00	**0.94**	0.05	0.01	0.01	0.00	**0.93**
40 × 15	0.01	0.00	0.00	0.00	**0.98**	0.01	0.00	0.01	0.00	**0.98**	0.02	0.01	0.01	0.00	**0.97**
40 × 20	0.00	0.00	0.00	0.00	**0.99**	0.00	0.00	0.00	0.00	**0.99**	0.01	0.00	0.01	0.00	**0.98**
80 × 5	0.10	0.01	0.01	0.00	**0.89**	0.11	0.01	0.01	0.00	**0.87**	0.14	0.01	0.01	0.00	**0.84**
80 × 10	0.01	0.00	0.00	0.00	**0.98**	0.01	0.00	0.00	0.00	**0.98**	0.02	0.00	0.00	0.00	**0.97**
80 × 15	0.00	0.00	0.00	0.00	**1.00**	0.00	0.00	0.00	0.00	**0.99**	0.00	0.00	0.00	0.00	**0.99**
80 × 20	0.00	0.00	0.00	0.00	**1.00**	0.00	0.00	0.00	0.00	**1.00**	0.00	0.00	0.00	0.00	**1.00**
120 × 5	0.02	0.00	0.00	0.00	**0.97**	0.04	0.01	0.00	0.00	**0.94**	0.06	0.00	0.00	0.00	**0.93**
120 × 10	0.00	0.00	0.00	0.00	**0.99**	0.00	0.00	0.00	0.00	**0.99**	0.00	0.00	0.00	0.00	**0.99**
120 × 15	0.00	0.00	0.00	0.00	**1.00**	0.00	0.00	0.00	0.00	**1.00**	0.00	0.00	0.00	0.00	**1.00**
120 × 20	0.00	0.00	0.00	0.00	**1.00**	0.00	0.00	0.00	0.00	**1.00**	0.00	0.00	0.00	0.00	**1.00**
160 × 5	0.00	0.00	0.00	0.00	**0.99**	0.02	0.00	0.00	0.00	**0.97**	0.04	0.00	0.00	0.00	**0.96**
160 × 10	0.00	0.00	0.00	0.00	**1.00**	0.00	0.00	0.00	0.00	**0.99**	0.00	0.00	0.00	0.00	**1.00**
160 × 15	0.00	0.00	0.00	0.00	**0.99**	0.01	0.00	0.00	0.00	**0.99**	0.01	0.00	0.00	0.00	**0.99**
160 × 20	0.00	0.00	0.00	0.00	**0.99**	0.00	0.00	0.00	0.00	**0.99**	0.00	0.00	0.00	0.00	**1.00**
Average	0.02	0.00	0.00	0.00	**0.97**	0.02	0.00	0.00	0.00	**0.97**	0.03	0.00	0.00	0.00	**0.96**

- I_S: Spread in the Pareto set (see e.g. Zhang et al., 2019 for a related approach). It measures the diversity in the Pareto set of Algorithm A ($\mathcal{P}^A$) as compared to the Pareto set $\mathcal{P}^*$. Hence, a lower value is desirable and means a more uniform distribution in the distances $\mathcal{P}^A$ and $\mathcal{P}^*$, i.e. $d_y(\mathcal{P}^A)$. This spread is measured by the standard deviation of $d_y(\mathcal{P}^A)$, i.e. $\sigma(d_y(\mathcal{P}^A))$ defined by Eq. (16):

$$I_S = \sigma\left(d_y\left(\mathcal{P}^A\right)\right) = \frac{\sum_{y\in\mathcal{P}^*}\sqrt{\left(d_y(\mathcal{P}^A)-\bar{d}(\mathcal{P}^A)\right)^2}}{|\mathcal{P}^*|-1} \tag{16}$$

To increase the power of the experimentation, we carry out five runs per instance, and the average values for these indicators are stored.

5.3. Implemented algorithms

Based on the literature review carried out in Section 2, we adapt the following algorithms from related scheduling problems:

- HDDE: This metaheuristic is proposed by Mokhtari et al. (2011) for the $Fm|prmu, p_{ij}(u_j)|w_1\cdot C_{max} + w_2\cdot Cost$ problem. In order to adapt the algorithm to our problem, its initial set of assignments is replaced by a random initial assignment since it is dependent on the multi-objective function.
- IG: This metaheuristic is proposed by Öztop et al. (2020) for the $Fm|prmu, p_{ij}(u_j)|(\sum C_j, Cost)$ problem. In order to adapt the metaheuristic to the problem under consideration, the initial solution PFH_NEH is replaced by the NEH algorithm since the first part of the former heuristic (which is a profile-fitting constructive heuristic, PFH) cannot be efficiently adapted to the makespan minimisation.
- IG_{ALL}: This metaheuristic is proposed by Öztop et al. (2020) to solve the $Fm|prmu, p_{ij}(u_j)|(\sum C_j, Cost)$ problem. As in the previous case, we replace PFH_NEH by the NEH algorithm.
- VBIH: This metaheuristic is proposed by Öztop et al. (2020) to solve the $Fm|prmu, p_{ij}(u_j)|(\sum C_j, Cost)$ problem. Analogously, we replace PFH_NEH by the NEH algorithm.

These algorithms are compared with the proposed CP_ILS algorithm under the same condition, i.e. implementing the algorithms in the same programming language (C# under Visual Studio 2019) by the same

Table 5
I_A of each metaheuristic grouped by n and m in benchmark $\mathcal{B}_1$. In bold, we represent the best found value for each instance and stopping criterion.

$n \times m$	$t = 60$					$t = 90$					$t = 120$				
	HDDE	IG	IG_{ALL}	VBIH	CP_ILS	HDDE	IG	IG_{ALL}	VBIH	CP_ILS	HDDE	IG	IG_{ALL}	VBIH	CP_ILS
40 × 5	0.32	0.25	0.25	0.30	**0.01**	0.31	0.25	0.24	0.30	**0.01**	0.29	0.23	0.23	0.29	**0.01**
40 × 10	0.44	0.26	0.26	0.31	**0.01**	0.43	0.25	0.25	0.31	**0.01**	0.42	0.24	0.24	0.30	**0.01**
40 × 15	0.53	0.28	0.28	0.32	**0.00**	0.52	0.27	0.27	0.31	**0.01**	0.51	0.26	0.26	0.30	**0.01**
40 × 20	0.55	0.28	0.28	0.32	**0.00**	0.56	0.28	0.28	0.32	**0.00**	0.55	0.27	0.27	0.31	**0.00**
80 × 5	0.33	0.36	0.36	0.40	**0.02**	0.31	0.37	0.36	0.41	**0.02**	0.29	0.36	0.36	0.40	**0.02**
80 × 10	0.41	0.35	0.35	0.39	**0.00**	0.40	0.33	0.33	0.37	**0.00**	0.40	0.33	0.32	0.37	**0.00**
80 × 15	0.54	0.38	0.38	0.41	**0.00**	0.53	0.36	0.36	0.40	**0.00**	0.52	0.35	0.35	0.39	**0.00**
80 × 20	0.63	0.39	0.39	0.43	**0.00**	0.62	0.37	0.37	0.41	**0.00**	0.59	0.35	0.35	0.39	**0.00**
120 × 5	0.36	0.38	0.39	0.41	**0.01**	0.34	0.34	0.33	0.36	**0.01**	0.33	0.34	0.35	0.37	**0.01**
120 × 10	0.40	0.37	0.37	0.39	**0.00**	0.36	0.36	0.36	0.39	**0.00**	0.36	0.35	0.35	0.37	**0.00**
120 × 15	0.57	0.43	0.43	0.46	**0.00**	0.56	0.41	0.41	0.44	**0.00**	0.51	0.41	0.41	0.45	**0.00**
120 × 20	0.61	0.42	0.42	0.44	**0.00**	0.61	0.42	0.42	0.45	**0.00**	0.60	0.40	0.40	0.43	**0.00**
160 × 5	0.49	0.31	0.31	0.32	**0.00**	0.31	0.30	0.30	0.31	**0.00**	0.29	0.31	0.31	0.33	**0.01**
160 × 10	0.41	0.37	0.38	0.39	**0.00**	0.38	0.37	0.37	0.40	**0.00**	0.35	0.37	0.37	0.39	**0.00**
160 × 15	0.55	0.41	0.41	0.43	**0.00**	0.47	0.41	0.41	0.43	**0.00**	0.47	0.42	0.42	0.44	**0.00**
160 × 20	0.72	0.46	0.46	0.48	**0.00**	0.64	0.45	0.45	0.48	**0.00**	0.62	0.48	0.47	0.50	**0.00**
Average	0.49	0.36	0.36	0.39	**0.00**	0.46	0.35	0.35	0.38	**0.00**	0.45	0.34	0.34	0.38	**0.01**

Table 7
I_S of each metaheuristic grouped by n and m in benchmark $\mathcal{B}_1$. In bold, we represent the best found value for each instance and stopping criterion.

$n \times m$	$t = 60$					$t = 90$					$t = 120$				
	HDDE	IG	IG_{ALL}	VBIH	CP_ILS	HDDE	IG	IG_{ALL}	VBIH	CP_ILS	HDDE	IG	IG_{ALL}	VBIH	CP_ILS
40 × 5	0.16	0.06	0.06	0.06	**0.02**	0.16	0.06	0.06	0.06	**0.02**	0.16	0.06	0.06	0.06	**0.02**
40 × 10	0.20	0.05	0.05	0.05	**0.01**	0.20	0.05	0.05	0.05	**0.01**	0.19	0.05	0.05	0.05	**0.01**
40 × 15	0.18	0.03	0.03	0.03	**0.01**	0.18	0.03	0.03	0.03	**0.01**	0.19	0.03	0.03	0.03	**0.01**
40 × 20	0.17	0.03	0.03	0.03	**0.01**	0.18	0.04	0.03	0.03	**0.01**	0.18	0.03	0.03	0.03	**0.01**
80 × 5	0.17	0.07	0.07	0.07	**0.03**	0.17	0.07	0.07	0.07	**0.03**	0.16	0.07	0.07	0.07	**0.03**
80 × 10	0.14	0.06	0.06	0.06	**0.01**	0.14	0.05	0.06	0.05	**0.01**	0.15	0.06	0.06	0.06	**0.01**
80 × 15	0.15	0.05	0.05	0.05	**0.00**	0.16	0.05	0.05	0.05	**0.00**	0.17	0.05	0.05	0.05	**0.00**
80 × 20	0.15	0.05	0.05	0.05	**0.00**	0.16	0.05	0.05	0.05	**0.00**	0.16	0.05	0.05	0.05	**0.00**
120 × 5	0.14	0.07	0.07	0.07	**0.01**	0.15	0.07	0.07	0.07	**0.02**	0.15	0.07	0.07	0.07	**0.02**
120 × 10	0.12	0.06	0.06	0.06	**0.00**	0.13	0.06	0.06	0.05	**0.00**	0.13	0.06	0.06	0.05	**0.00**
120 × 15	0.13	0.07	0.07	0.07	**0.00**	0.15	0.07	0.07	0.07	**0.00**	0.15	0.07	0.07	0.07	**0.00**
120 × 20	0.11	0.05	0.06	0.05	**0.00**	0.14	0.05	0.05	0.05	**0.00**	0.15	0.05	0.05	0.05	**0.00**
160 × 5	0.12	0.07	0.07	0.07	**0.00**	0.13	0.07	0.07	0.07	**0.01**	0.14	0.07	0.07	0.07	**0.01**
160 × 10	0.10	0.07	0.07	0.06	**0.00**	0.11	0.07	0.07	0.06	**0.00**	0.12	0.07	0.07	0.06	**0.00**
160 × 15	0.10	0.06	0.06	0.06	**0.00**	0.13	0.06	0.06	0.06	**0.00**	0.14	0.07	0.07	0.07	**0.00**
160 × 20	0.08	0.08	0.08	0.08	**0.00**	0.10	0.08	0.08	0.08	**0.00**	0.12	0.08	0.09	0.08	**0.00**
Average	0.14	0.06	0.06	0.06	**0.01**	0.15	0.06	0.06	0.06	**0.01**	0.15	0.06	0.06	0.06	**0.01**

Table 9
Average I_S of each metaheuristic in both benchmarks

Benchmark	Parameter	$t = 60$					$t = 90$					$t = 120$				
		HDDE	IG	IG_{ALL}	VBIH	CP_ILS	HDDE	IG	IG_{ALL}	VBIH	CP_ILS	HDDE	IG	IG_{ALL}	VBIH	CP_ILS
β_1	$n = 40$	0.18	0.04	0.04	0.04	0.01	0.18	0.04	0.04	0.04	0.01	0.18	0.04	0.04	0.04	0.01
	$n = 80$	0.15	0.06	0.06	0.06	0.01	0.16	0.06	0.06	0.05	0.01	0.16	0.05	0.06	0.05	0.01
	$n = 120$	0.13	0.06	0.06	0.06	0.00	0.14	0.06	0.06	0.06	0.01	0.15	0.06	0.06	0.06	0.01
	$n = 160$	0.10	0.07	0.07	0.07	0.00	0.12	0.07	0.07	0.07	0.00	0.13	0.07	0.07	0.07	0.00
	$m = 5$	0.15	0.07	0.07	0.07	0.01	0.15	0.07	0.07	0.07	0.02	0.15	0.07	0.07	0.07	0.02
	$m = 10$	0.14	0.06	0.06	0.06	0.01	0.15	0.06	0.06	0.06	0.01	0.15	0.06	0.06	0.06	0.01
	$m = 15$	0.14	0.06	0.05	0.05	0.00	0.16	0.05	0.05	0.05	0.00	0.16	0.05	0.05	0.05	0.00
	$m = 20$	0.13	0.06	0.06	0.05	0.00	0.14	0.06	0.06	0.05	0.00	0.15	0.05	0.06	0.05	0.00
β_2	$n = 40$	0.10	0.04	0.04	0.04	0.01	0.10	0.04	0.04	0.04	0.02	0.11	0.04	0.04	0.04	0.02
	$n = 80$	0.09	0.06	0.06	0.06	0.01	0.09	0.06	0.06	0.06	0.01	0.09	0.06	0.06	0.06	0.01
	$n = 120$	0.09	0.06	0.06	0.07	0.01	0.09	0.06	0.06	0.06	0.01	0.09	0.06	0.06	0.06	0.01
	$n = 160$	0.10	0.06	0.06	0.07	0.01	0.10	0.07	0.07	0.07	0.01	0.10	0.06	0.06	0.06	0.01
	$m = 5$	0.13	0.06	0.06	0.07	0.01	0.13	0.06	0.06	0.07	0.01	0.13	0.06	0.06	0.07	0.01
	$m = 10$	0.10	0.06	0.06	0.06	0.01	0.10	0.06	0.06	0.06	0.01	0.10	0.06	0.06	0.06	0.01
	$m = 15$	0.08	0.06	0.06	0.06	0.01	0.08	0.06	0.05	0.06	0.01	0.08	0.05	0.05	0.06	0.01
	$m = 20$	0.07	0.05	0.05	0.05	0.01	0.07	0.05	0.05	0.05	0.02	0.07	0.05	0.05	0.05	0.02
	Average	0.12	0.06	0.06	0.06	0.01	0.12	0.06	0.06	0.06	0.01	0.13	0.06	0.06	0.06	0.01

person (using the same common functions and/or libraries), and executing them in the same computer (Intel Core i7-3770 with 3.4 GHz and 16 GB RAM). Regarding the parameters d and T of the proposed algorithm, to avoid an overcalibration, we generate a different benchmark following the same procedure as $\mathcal{B}_1$ and test our proposal for $d \in \{3,4,5\}$ and $T \in \{0.3, 0.4, 0.5\}$. After a full factorial design, the best combination of the parameters has been found for $d = 4$ and $T = 0.4$, regardless of the indicator considered. These values are used in the rest of the paper.

5.4. Computational results

In this section, we perform three computational evaluations on the benchmarks $\mathcal{B}_1, \mathcal{B}_2$ and $\mathcal{B}_3$ to analyse the performance of the implemented algorithms. All these algorithms have been run under the same stopping criterion $n{\cdot}m{\cdot}t/2$ milliseconds with $t = 60, 90, 120$, using the indicators of the previous section. Regarding the computational evaluation on the generic instances $\mathcal{B}_1$, the computational results found by the algorithms in terms of N_{NDS} and I_R are summarised in Tables 1 and 3, respectively. The number of non-dominated solutions of the proposed CP_ILS algorithm clearly outperforms every other algorithm regardless of the stopping criterion. For instance, the number of global non-dominated solutions found in average by CP_ILS is 752.85 (for $t = 60$) versus 4.00 found by HDD, 1.49 by IG, 1.62 by IG_{ALL} or 0.64 by VBIH, being 760.60 the total number of solutions in the Pareto set using $t = 60$. Similarly, the best ratio of non-dominated solutions is (for $t = 60$) 0.97, i.e. the algorithm finds around 97% of all non-dominated solutions found. This excellent result contrasts with the values found by the rest of the metaheuristics implemented, being 0.02 for HDDE, and 0.00 for the rest of the metaheuristics. All these results are equivalent regardless of the stopping criterion. This behaviour is analogous for the other stopping criteria ($t = 90$ and $t = 120$). Regarding I_A and I_S, detailed results are shown in Tables 5 and 7, respectively. Again, the best results are found by CP_ILS, where the average distance between CP_ILS and PR (i.e. I_A(CP_ILS)) is 0.00, 0.00 and 0.01 for $t = 60, t = 90$ and $t = 120$, respectively. However, the I_A indicator found by the other metaheuristics is between 0.34 and 0.48, being 0.34 only in $t = 120$ by the IG and IG_{ALL} algorithms. This is also the case of the spread of the Pareto set (I_S) found by each algorithm (see also Table 9 for average values of this indicator for each n and m). The best result is 0.01 found by CP_ILS

Table 2
N_{NDS} of each metaheuristic grouped by n and m in benchmark $\mathcal{B}_2$. In bold, we represent the best found value for each instance and stopping criterion.

$n \times m$	$t = 60$						$t = 90$						$t = 120$					
	HDDE	IG	IG_{ALL}	VBIH	CP_ILS	Total	HDDE	IG	IG_{ALL}	VBIH	CP_ILS	Total	HDDE	IG	IG_{ALL}	VBIH	CP_ILS	Total
40 × 5	1.95	6.24	6.45	1.56	**309.47**	325.66	0.95	6.72	7.18	1.70	**317.69**	334.25	1.63	7.54	7.69	1.68	**316.50**	335.04
40 × 10	0.00	6.36	6.80	1.40	**380.53**	395.09	0.00	6.90	7.42	1.30	**400.75**	416.37	0.21	8.23	8.39	1.26	**383.82**	401.91
40 × 15	0.05	8.17	8.01	1.22	**370.98**	388.43	0.00	10.39	8.79	1.29	**371.52**	391.99	0.15	10.84	10.87	1.31	**364.51**	387.69
40 × 20	0.43	9.37	9.63	1.60	**337.12**	358.15	0.40	11.57	10.78	1.35	**334.33**	358.43	0.21	12.28	12.00	1.22	**353.40**	379.11
80 × 5	4.54	4.82	4.71	2.01	**521.78**	537.85	3.51	5.37	5.31	1.87	**546.67**	562.74	2.81	6.17	6.07	1.91	**546.45**	563.42
80 × 10	3.19	6.29	7.03	2.42	**499.45**	518.37	2.54	7.21	7.96	2.21	**518.63**	538.55	1.58	8.26	9.41	2.04	**520.13**	541.43
80 × 15	3.95	8.71	9.27	3.62	**433.97**	459.52	3.31	9.79	11.64	3.25	**442.25**	470.23	3.30	11.06	12.69	3.03	**466.40**	496.48
80 × 20	3.33	10.87	10.07	3.95	**394.83**	423.06	5.25	12.07	12.07	2.92	**386.25**	418.57	5.37	14.05	12.14	3.35	**398.14**	433.05
120 × 5	3.96	4.27	4.33	2.43	**739.56**	754.55	4.31	5.48	5.47	2.25	**747.74**	765.24	6.14	5.78	5.70	2.35	**742.67**	762.65
120 × 10	8.15	5.76	6.37	3.58	**612.96**	636.82	9.23	7.13	7.94	2.93	**653.77**	681.00	10.13	7.55	9.03	3.29	**638.91**	668.92
120 × 15	7.11	7.65	9.29	4.15	**456.73**	484.92	11.68	9.15	10.17	4.63	**459.26**	494.89	14.06	9.54	10.63	3.87	**462.81**	500.91
120 × 20	3.48	9.09	7.99	4.47	**426.57**	451.60	6.04	10.40	10.18	4.27	**419.43**	450.32	11.58	10.83	9.99	4.87	**415.72**	452.99
160 × 5	0.00	5.41	5.41	2.88	**809.91**	823.61	6.42	5.16	5.04	2.81	**855.77**	875.19	9.59	5.29	5.16	2.67	**835.73**	858.44
160 × 10	9.39	6.11	7.75	3.85	**645.09**	672.19	14.97	6.37	7.31	3.49	**674.43**	706.56	16.02	5.88	7.36	3.43	**688.89**	721.59
160 × 15	3.98	6.57	7.96	4.47	**529.87**	552.85	10.08	6.14	8.36	4.67	**524.29**	553.55	16.49	6.97	8.85	4.63	**523.08**	560.02
160 × 20	0.14	7.79	7.92	4.84	**442.49**	463.17	6.01	7.99	8.66	4.71	**438.11**	465.47	8.45	8.12	9.23	4.34	**437.17**	467.31
Average	3.35	7.09	7.44	3.03	**494.46**	515.37	5.29	7.99	8.39	2.85	**505.68**	530.21	6.73	8.65	9.08	2.83	**505.90**	533.18

regardless of the stopping criterion, improving the results obtained by the other implemented algorithms (between 0.06 and 0.15). All these results show the effectiveness of the proposal which clearly outperforms every other metaheuristic in terms of all indicators, i.e. N_{NDS}, I_R, I_A and I_S. This hypothesis has been tested using a non-parametric Mann-Whitney test at the 95% confidence level. A p-value of 0.000 has been obtained in any comparison between our proposal and the second best metaheuristic for every indicator.

Regarding the computational evaluation performed on the green set of instances $\mathcal{B}_2$, we found very similar trends than in the previous case. The detailed computational results are shown in Tables 2, 4, 6 and 8, for indicators N_{NDS}, I_R, I_A and I_S, respectively. As concerns the N_{NDS} and I_R indicators, the best result among the algorithms implemented of the literature is found by IG_{ALL}. For this algorithm, N_{NDS} (I_R) takes 7.44 (0.02), 8.39 (0.02) and 9.08 (0.02) for t equals to 60, 90 and 120, respectively. These values are clearly outperformed by CP_ILS, which obtains in average 494.46, 505.68, and 505.90 non-dominated solutions for t equals to 60, 90 and 120, respectively. Regarding I_R, our proposal obtained 0.95 for $t = 30$ and 0.94 for the other criteria. Similarly, our proposal finds the best I_A value, 0.01 for every stopping criterion, which clearly outperforms the metaheuristics in the literature (in this regard, the I_A value obtained by IG_{ALL} is between 0.12 and 0.13). The worst result is found by HDDE with average values between 0.25 and 0.27. This is also the case of the spread in the Pareto set (I_S) which is 0.01 for CP_ILS and 0.06 for the second best metaheuristic (IG_{ALL}), and this difference remains fairly homogeneous regardless of the value of the parameter n or m (see in this regard Table 9). Four non-parametric Mann-Whitney tests have been performed to check the effectiveness of the proposal, one for each indicator. A p-value of 0.000 has been obtained in each comparison with the second best metaheuristic (IG_{ALL}), which statistically highlights the effectiveness of our proposal. Finally, a last experimentation has been performed checking the performance of the implemented algorithm in big-size instances (using $\mathcal{B}_3$). Computational results are shown in Tables 10 and 11 (for $t = 60$). We can see that the proposed CP_ILS metaheuristic even improves its performance when n increases, as compared as the other implemented metaheuristics from the literature (regardless of the indicator applied).

6. Conclusions

This paper tackles the bi-objective green permutation flow shop scheduling problem to minimise the makespan and the total cost or the total energy/carbon consumption, which is attracting the interest of many researchers and practitioners. The problem considers that each machine can process the jobs using different speeds, and consequently changing its normal processing times (typically denoted in the literature as controllable processing times). To analyse in detail the problem, we first propose several properties of the problem based on the concept of critical path and slack time. Equipped with these problem properties, we propose a critical-path based iterated local search (denoted CP_ILS). This metaheuristic performs several different specific local search methods after a simple perturbation mechanism. The main idea of the proposal is to calculate the critical path of each sequence and to construct methods based on operations inside and outside this path. To test the efficiency of the proposal, we have generated three extensive different data sets comparing the proposal with the most promising metaheuristics in the literature. In the first experimentation, we consider the general CPT problem, while using the second benchmark we address the green special case. The computational results obtained in both experimentations show the excellent performance found by CP_ILS, which outperforms every other metaheuristic proposed in the related literature regardless of the indicator used to test the proposals. Regarding future research lines, although we have addressed the traditional green permutation flowshop scheduling problem, the literature is still scarce in solving more constrained green scheduling variants. Thus, specific algorithms for related green scheduling problems are pertinent. In this regard, the algorithm

Table 4
I_R of each metaheuristic grouped by n and m in benchmark $\mathcal{B}_2$. In bold, we represent the best found value for each instance and stopping criterion.

$n \times m$	$t = 60$					$t = 90$					$t = 120$				
	HDDE	IG	IG_{ALL}	VBIH	CP_ILS	HDDE_VNS	IG	IG_{ALL}	VBIH	CP_ILS	HDDE_VNS	IG	IG_{ALL}	VBIH	CP_ILS
40 × 5	0.01	0.02	0.02	0.00	**0.95**	0.00	0.02	0.02	0.01	**0.95**	0.00	0.02	0.02	0.01	**0.94**
40 × 10	0.00	0.02	0.02	0.00	**0.96**	0.00	0.02	0.02	0.00	**0.96**	0.00	0.02	0.02	0.00	**0.95**
40 × 15	0.00	0.02	0.02	0.00	**0.95**	0.00	0.03	0.03	0.00	**0.94**	0.00	0.03	0.03	0.00	**0.93**
40 × 20	0.00	0.03	0.03	0.01	**0.93**	0.00	0.04	0.03	0.00	**0.92**	0.00	0.04	0.04	0.00	**0.92**
80 × 5	0.01	0.01	0.01	0.00	**0.97**	0.01	0.01	0.01	0.00	**0.97**	0.01	0.01	0.01	0.00	**0.97**
80 × 10	0.01	0.01	0.02	0.01	**0.96**	0.01	0.02	0.02	0.00	**0.96**	0.00	0.02	0.02	0.00	**0.95**
80 × 15	0.01	0.02	0.02	0.01	**0.93**	0.01	0.02	0.03	0.01	**0.93**	0.01	0.03	0.03	0.01	**0.93**
80 × 20	0.01	0.03	0.03	0.01	**0.92**	0.02	0.03	0.03	0.01	**0.91**	0.02	0.04	0.03	0.01	**0.91**
120 × 5	0.01	0.01	0.01	0.00	**0.98**	0.01	0.01	0.01	0.00	**0.97**	0.01	0.01	0.01	0.00	**0.97**
120 × 10	0.02	0.01	0.01	0.01	**0.95**	0.02	0.01	0.01	0.01	**0.95**	0.02	0.01	0.02	0.01	**0.95**
120 × 15	0.02	0.02	0.02	0.01	**0.93**	0.03	0.02	0.02	0.01	**0.92**	0.03	0.02	0.02	0.01	**0.91**
120 × 20	0.01	0.02	0.02	0.01	**0.94**	0.02	0.03	0.03	0.01	**0.92**	0.03	0.03	0.02	0.01	**0.91**
160 × 5	0.00	0.01	0.01	0.00	**0.98**	0.01	0.01	0.01	0.00	**0.97**	0.02	0.01	0.01	0.00	**0.97**
160 × 10	0.02	0.01	0.01	0.01	**0.95**	0.03	0.01	0.01	0.01	**0.95**	0.03	0.01	0.01	0.01	**0.95**
160 × 15	0.01	0.01	0.02	0.01	**0.95**	0.02	0.01	0.02	0.01	**0.94**	0.03	0.01	0.02	0.01	**0.93**
160 × 20	0.00	0.02	0.02	0.01	**0.95**	0.02	0.02	0.02	0.01	**0.93**	0.02	0.02	0.02	0.01	**0.93**
Average	0.01	0.02	0.02	0.01	**0.95**	0.01	0.02	0.02	0.01	**0.94**	0.01	0.02	0.02	0.01	**0.94**

Table 6
I_A of each metaheuristic grouped by n and m in benchmark $\mathcal{B}_2$. In bold, we represent the best found value for each instance and stopping criterion.

$n \times m$	$t = 60$					$t = 90$					$t = 120$				
	HDDE	IG	IG_{ALL}	VBIH	CP_ILS	HDDE_VNS	IG	IG_{ALL}	VBIH	CP_ILS	HDDE_VNS	IG	IG_{ALL}	VBIH	CP_ILS
40 × 5	0.26	0.09	0.09	0.13	**0.01**	0.26	0.08	0.08	0.12	**0.01**	0.26	0.08	0.08	0.12	**0.01**
40 × 10	0.26	0.10	0.10	0.14	**0.01**	0.26	0.10	0.10	0.14	**0.01**	0.25	0.10	0.10	0.13	**0.01**
40 × 15	0.23	0.11	0.11	0.14	**0.01**	0.23	0.10	0.10	0.14	**0.01**	0.25	0.10	0.10	0.14	**0.01**
40 × 20	0.22	0.12	0.11	0.15	**0.01**	0.22	0.11	0.11	0.15	**0.01**	0.22	0.11	0.11	0.15	**0.01**
80 × 5	0.24	0.13	0.13	0.14	**0.01**	0.24	0.12	0.12	0.14	**0.00**	0.25	0.12	0.12	0.14	**0.01**
80 × 10	0.23	0.13	0.13	0.15	**0.01**	0.22	0.13	0.12	0.14	**0.01**	0.23	0.12	0.12	0.14	**0.01**
80 × 15	0.24	0.14	0.14	0.16	**0.01**	0.23	0.14	0.13	0.16	**0.01**	0.23	0.13	0.13	0.16	**0.01**
80 × 20	0.25	0.14	0.14	0.16	**0.01**	0.24	0.13	0.13	0.16	**0.01**	0.23	0.13	0.13	0.15	**0.01**
120 × 5	0.25	0.13	0.13	0.14	**0.00**	0.24	0.13	0.13	0.14	**0.00**	0.23	0.13	0.13	0.14	**0.00**
120 × 10	0.26	0.15	0.15	0.16	**0.00**	0.24	0.14	0.14	0.15	**0.01**	0.24	0.14	0.14	0.15	**0.01**
120 × 15	0.27	0.14	0.14	0.15	**0.01**	0.28	0.14	0.13	0.15	**0.01**	0.26	0.13	0.13	0.14	**0.01**
120 × 20	0.30	0.13	0.13	0.14	**0.01**	0.27	0.12	0.12	0.13	**0.01**	0.26	0.12	0.12	0.13	**0.01**
160 × 5	0.30	0.14	0.14	0.15	**0.00**	0.25	0.14	0.14	0.14	**0.00**	0.25	0.13	0.13	0.14	**0.00**
160 × 10	0.29	0.16	0.16	0.17	**0.00**	0.28	0.16	0.16	0.17	**0.00**	0.27	0.16	0.16	0.16	**0.00**
160 × 15	0.33	0.14	0.13	0.14	**0.00**	0.29	0.13	0.13	0.14	**0.01**	0.29	0.13	0.13	0.13	**0.01**
160 × 20	0.37	0.11	0.11	0.12	**0.00**	0.31	0.11	0.11	0.12	**0.01**	0.30	0.11	0.11	0.11	**0.01**
Average	0.27	0.13	0.13	0.14	**0.01**	0.25	0.12	0.12	0.14	**0.01**	0.25	0.12	0.12	0.14	**0.01**

Table 8
I_S of each metaheuristic grouped by n and m in benchmark $\mathcal{B}_2$. In bold, we represent the best found value for each instance and stopping criterion.

$n \times m$	$t = 60$					$t = 90$					$t = 120$				
	HDDE	IG	IG_{ALL}	VBIH	CP_ILS	HDDE_VNS	IG	IG_{ALL}	VBIH	CP_ILS	HDDE_VNS	IG	IG_{ALL}	VBIH	CP_ILS
40 × 5	0.13	0.04	0.04	0.05	**0.01**	0.13	0.03	0.03	0.05	**0.01**	0.13	0.03	0.03	0.05	**0.01**
40 × 10	0.10	0.04	0.04	0.04	**0.01**	0.10	0.03	0.03	0.04	**0.01**	0.11	0.03	0.03	0.04	**0.02**
40 × 15	0.09	0.04	0.04	0.04	**0.02**	0.09	0.04	0.04	0.04	**0.02**	0.10	0.04	0.04	0.04	**0.02**
40 × 20	0.08	0.04	0.04	0.04	**0.02**	0.08	0.04	0.04	0.04	**0.02**	0.09	0.04	0.04	0.04	**0.02**
80 × 5	0.12	0.06	0.06	0.06	**0.01**	0.13	0.05	0.06	0.06	**0.01**	0.13	0.05	0.06	0.06	**0.01**
80 × 10	0.09	0.06	0.06	0.06	**0.01**	0.09	0.06	0.06	0.06	**0.01**	0.09	0.06	0.06	0.06	**0.01**
80 × 15	0.08	0.06	0.06	0.06	**0.01**	0.08	0.06	0.06	0.06	**0.01**	0.08	0.06	0.06	0.06	**0.01**
80 × 20	0.07	0.06	0.06	0.06	**0.01**	0.07	0.06	0.06	0.06	**0.02**	0.07	0.06	0.06	0.06	**0.02**
120 × 5	0.13	0.07	0.07	0.07	**0.01**	0.13	0.07	0.07	0.07	**0.01**	0.13	0.07	0.07	0.07	**0.01**
120 × 10	0.09	0.07	0.07	0.07	**0.01**	0.10	0.07	0.07	0.07	**0.01**	0.09	0.07	0.07	0.07	**0.01**
120 × 15	0.07	0.06	0.06	0.06	**0.01**	0.08	0.06	0.06	0.06	**0.01**	0.08	0.06	0.06	0.06	**0.01**
120 × 20	0.06	0.06	0.06	0.06	**0.01**	0.07	0.06	0.06	0.06	**0.01**	0.07	0.06	0.06	0.06	**0.01**
160 × 5	0.14	0.07	0.07	0.08	**0.01**	0.14	0.07	0.07	0.08	**0.01**	0.14	0.07	0.07	0.08	**0.01**
160 × 10	0.10	0.07	0.07	0.07	**0.01**	0.10	0.07	0.07	0.07	**0.01**	0.10	0.07	0.07	0.07	**0.01**
160 × 15	0.08	0.06	0.06	0.06	**0.01**	0.08	0.06	0.06	0.06	**0.01**	0.08	0.06	0.06	0.06	**0.01**
160 × 20	0.07	0.05	0.05	0.05	**0.01**	0.07	0.05	0.05	0.05	**0.01**	0.07	0.05	0.05	0.05	**0.01**
Average	0.09	0.06	0.06	0.06	**0.01**	0.10	0.06	0.06	0.06	**0.01**	0.10	0.06	0.06	0.06	**0.01**

Table 10
N_{NDS} and I_R of each metaheuristic (with $t = 60$) grouped by n and m in big-size instances.

$n \times m$	N_{NDS}						I_R				
	HDD	IG	IG_{ALL}	VBIH	CP_ILS	Total	HDD	IG	IG_{ALL}	VBIH	CP_ILS
300 × 5	0.00	2.17	2.09	1.11	1050.85	1056.23	0.00	0.00	0.00	0.00	0.99
300 × 10	0.00	1.63	1.59	0.82	1600.23	1604.27	0.00	0.00	0.00	0.00	1.00
300 × 15	0.00	1.17	1.45	0.55	2403.59	2406.76	0.00	0.00	0.00	0.00	1.00
300 × 20	0.00	1.13	1.36	0.54	3277.27	3280.29	0.00	0.00	0.00	0.00	1.00
400 × 5	0.00	1.97	1.96	0.95	1351.64	1356.51	0.00	0.01	0.01	0.00	0.98
400 × 10	0.00	1.61	1.77	0.73	2325.38	2329.49	0.00	0.00	0.00	0.00	0.99
400 × 15	0.00	1.47	1.52	0.59	2981.11	2984.69	0.00	0.00	0.00	0.00	1.00
400 × 20	0.00	1.13	1.35	0.53	3784.39	3787.39	0.00	0.00	0.00	0.00	1.00
500 × 5	0.00	2.28	2.36	1.20	1720.81	1726.65	0.00	0.00	0.00	0.00	1.00
500 × 10	0.00	1.49	1.84	0.84	2718.79	2722.96	0.00	0.00	0.00	0.00	1.00
500 × 15	0.00	1.54	1.34	0.61	3818.02	3821.51	0.00	0.00	0.00	0.00	1.00
500 × 20	0.00	1.45	1.29	0.48	4492.87	4496.09	0.00	0.00	0.00	0.00	1.00
Average	0.00	1.59	1.66	0.75	2627.08	2631.07	0.00	0.00	0.00	0.00	1.00

Table 11
I_A and I_S of each metaheuristic (with $t = 60$) grouped by n and m in big-size instances.

$n \times m$	I_A					I_S				
	HDD	IG	IG_{ALL}	VBIH	CP_ILS	HDD	IG	IG_{ALL}	VBIH	CP_ILS
300 × 5	0.43	0.33	0.33	0.34	0.00	0.11	0.07	0.07	0.07	0.00
300 × 10	0.80	0.48	0.48	0.50	0.00	0.12	0.10	0.10	0.10	0.00
300 × 15	0.64	0.44	0.44	0.45	0.00	0.10	0.08	0.08	0.08	0.00
300 × 20	0.69	0.46	0.46	0.47	0.00	0.09	0.08	0.08	0.08	0.00
400 × 5	0.66	0.37	0.36	0.37	0.01	0.13	0.07	0.07	0.07	0.01
400 × 10	0.60	0.41	0.41	0.41	0.00	0.12	0.08	0.08	0.07	0.00
400 × 15	0.73	0.44	0.44	0.45	0.00	0.10	0.08	0.08	0.08	0.00
400 × 20	0.87	0.52	0.51	0.53	0.00	0.11	0.10	0.10	0.10	0.00
500 × 5	0.51	0.34	0.35	0.34	0.00	0.12	0.08	0.08	0.08	0.00
500 × 10	0.63	0.42	0.42	0.43	0.00	0.12	0.09	0.09	0.09	0.00
500 × 15	0.72	0.48	0.48	0.49	0.00	0.11	0.08	0.08	0.08	0.00
500 × 20	0.93	0.54	0.54	0.55	0.00	0.11	0.12	0.12	0.11	0.00
Average	0.68	0.44	0.44	0.44	0.00	0.11	0.09	0.09	0.08	0.00

proposed in this paper could be extended for solving related multi-objective problems in the search of near-optimal solutions. Furthermore, in view of the excellent results obtained by embedding the critical path into a local search algorithm, related problems addressing performance objectives different than the makespan could be tackled by applying the equivalent concepts such as the ones in Fernandez-Viagas et al. (2020).

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This research has been funded by the Spanish Ministry of Science and Innovation, under the project “ASSORT” with reference PID2019-108756RB-I00, by the Junta de Andalucia under the project “‘EFECTOS”, with reference US-1264511, and by the Coordination for the Improvement of Higher Education Personnel (CAPES) and the National Council for Scientific and Technological Development (CNPq), through grants 303594/2018-7, 309755/2021-2, and 407151/2021-4.

References

Amiri, M. F., & Behnamian, J. (2020). Multi-objective green flowshop scheduling problem under uncertainty: Estimation of distribution algorithm. *Journal of Cleaner Production, 251*, 119734.

Behnamian, J., & Ghomi, S. F. (2011). Hybrid flowshop scheduling with machine and resource-dependent processing times. *Applied Mathematical Modelling, 35*(3), 1107–1123.

Benavides, A., & Ritt, M. (2018). Fast heuristics for minimizing the makespan in non-permutation flow shops. *Computers and Operations Research, 100*, 230–243.

Cota, L., Coelho, V., Guimarẽs, F., & Souza, M. (2021). Bi-criteria formulation for green scheduling with unrelated parallel machines with sequence-dependent setup times. *International Transactions in Operational Research, 28*(2), 996–1017.

Dong, X., Chen, P., Huang, H., & Nowak, M. (2013). A multi-restart iterated local search algorithm for the permutation flow shop problem minimizing total flow time. *Computers & Operations Research, 40*(2), 627–632.

Fang, K., Uhan, N. A., Zhao, F., & Sutherland, J. W. (2013). Flow shop scheduling with peak power consumption constraints. *Annals of Operations Research, 206*(1), 115–145.

Fathollahi-Fard, A., Hajiaghaei-Keshteli, M., & Tavakkoli-Moghaddam, R. (2018). A bi-objective green home health care routing problem. *Journal of Cleaner Production, 200*, 423–443.

Fernandez-Viagas, V., & Framinan, J. (2015a). A bounded-search iterated greedy algorithm for the distributed permutation flowshop scheduling problem. *International Journal of Production Research, 53*(4), 1111–1123.

Fernandez-Viagas, V., & Framinan, J. (2015b). Efficient non-population-based algorithms for the permutation flowshop scheduling problem with makespan minimisation subject to a maximum tardiness. *Computers & Operations Research, 64*, 86–96.

Fernandez-Viagas, V., & Framinan, J. M. (2015c). Controllable processing times in project and production management: analysing the trade-off between processing times and the amount of resources. *Mathematical Problems in Engineering.*

Fernandez-Viagas, V., Molina-Pariente, J., & Framinan, J. (2020). Generalised accelerations for insertion-based heuristics in permutation flowshop scheduling. *European Journal of Operational Research, 282*(3), 858–872.

Fernandez-Viagas, V., Ruiz, R., & Framinan, J. (2017). A new vision of approximate methods for the permutation flowshop to minimise makespan: State-of-the-art and computational evaluation. *European Journal of Operational Research, 257*(3), 707–721.

Fernandez-Viagas, V., Valente, J., & Framinan, J. (2018). Iterated-greedy-based algorithms with beam search initialization for the permutation flowshop to minimise total tardiness. *Expert Systems with Applications, 94*, 58–69.

Foumani, M., & Smith-Miles, K. (2019). The impact of various carbon reduction policies on green flowshop scheduling. *Applied Energy, 249*, 300–315.

Framinan, J., Gupta, J., & Leisten, R. (2004). A review and classification of heuristics for permutation flow-shop scheduling with makespan objective. *Journal of the Operational Research Society, 55*(12), 1243–1255.

Gao, F., Liu, M., Wang, J.-J., & Lu, Y.-Y. (2018). No-wait two-machine permutation flow shop scheduling problem with learning effect, common due date and controllable job processing times. *International Journal of Production Research, 56*(6), 2361–2369.

Gomes, A. C. L., Ravetti, M. G., & Carrano, E. G. (2020). Multi-objective matheuristic for minimization of total tardiness and energy costs in a steel industry heat treatment line. *Computers & Industrial Engineering*, 106929.

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Rinnooy Kan, A. H. G. (1979). Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey. *Annals of Discrete Mathematics, 5*, 287–326.

Gupta, J. N., Krüger, K., Lauff, V., Werner, F., & Sotskov, Y. N. (2002). Heuristics for hybrid flow shops with controllable processing times and assignable due dates. *Computers & Operations Research, 29*(10), 1417–1439.

Jiang, S., Liu, M., Hao, J., & Qian, W. (2015). A bi-layer optimization approach for a hybrid flow shop scheduling problem involving controllable processing times in the steelmaking industry. *Computers & Industrial Engineering, 87*, 518–531.

Kacem, A., & Dammak, A. (2019). Bi-objective scheduling on two dedicated processors. *European Journal of Industrial Engineering, 13*(5), 681–700.

Lu, Y.-Y., Li, G., Wu, Y.-B., & Ji, P. (2014). Optimal due-date assignment problem with learning effect and resource-dependent processing times. *Optimization Letters, 8*(1), 113–127.

Mansouri, S. A., Aktas, E., & Besikci, U. (2016). Green scheduling of a two-machine flowshop: Trade-off between makespan and energy consumption. *European Journal of Operational Research, 248*(3), 772–788.

M'Hallah, R. (2014). An iterated local search variable neighborhood descent hybrid heuristic for the total earliness tardiness permutation flow shop. *International Journal of Production Research, 52*(13), 3802–3819.

Minella, G., Ruiz, R., & Ciavotta, M. (2008). A review and evaluation of multiobjective algorithms for the flowshop scheduling problem. *INFORMS Journal on Computing, 20*(3), 451–471.

Mokhtari, H., Abadi, I., & Cheraghalikhani, A. (2011). A multi-objective flow shop scheduling with resource-dependent processing times: Trade-off between makespan and cost of resources. *International Journal of Production Research, 49*(19), 5851–5875.

Nawaz, M., Enscore, J. E. E., & Ham, I. (1983). A Heuristic Algorithm for the m-Machine, n-Job Flow-shop Sequencing Problem. *OMEGA, The International Journal of Management Science, 11*(1), 91–95.

Nowicki, E., & Zdrzalka, S. (1990). A survey of results for sequencing problems with controllable processing times. *Discrete Applied Mathematics, 26*(2–3), 271–287.

Öztop, H., Tasgetiren, M. F., Eliiyi, D. T., Pan, Q.-K., & Kandiller, L. (2020). An energy-efficient permutation flowshop scheduling problem. *Expert Systems with Applications, 150*, 113279.

Pan, Q.-K., Tasgetiren, M., & Liang, Y.-C. (2008). A discrete differential evolution algorithm for the permutation flowshop scheduling problem. *Computers and Industrial Engineering, 55*(4), 795–816.

Pan, Q.-K., Wang, L., & Qian, B. (2009). A novel differential evolution algorithm for bi-criteria no-wait flow shop scheduling problems. *Computers and Operations Research, 36*(8), 2498–2511.

Pinedo, M. (2012). *Scheduling: Theory, Algorithms and Systems*. Springer.

Ramezanian, R., Vali-Siar, M. M., & Jalalian, M. (2019). Green permutation flowshop scheduling problem with sequence-dependent setup times: a case study. *International Journal of Production Research, 57*(10), 3311–3333.

Reza Hejazi, S., & Saghafian, S. (2005). Flowshop-scheduling problems with makespan criterion: A review. *International Journal of Production Research, 43*(14), 2895–2929.

Ribas, I., Companys, R., & Tort-Martorell, X. (2010). Comparing three-step heuristics for the permutation flow shop problem. *Computers & Operations Research, 37*(12), 2062–2070.

Ribas, I., Companys, R., & Tort-Martorell, X. (2013). A competitive variable neighbourhood search algorithm for the blocking flow shop problem. *European Journal of Industrial Engineering, 7*(6), 729–754.

Rinnooy Kan, A. H. G. (1976). *Machine Scheduling Problems: Classification, Complexity and Computations*. Martinus Nijhoff, The Hague.

Ruiz, R., & Maroto, C. (2005). A comprehensive review and evaluation of permutation flowshop heuristics. *European Journal of Operational Research, 165*(2), 479–494.

Ruiz, R., & Stützle, T. (2007). A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research, 177*(3), 2033–2049.

Shabtay, D., & Steiner, G. (2007). A survey of scheduling with controllable processing times. *Discrete Applied Mathematics, 155*(13), 1643–1666.

Shioura, A., Shakhlevich, N., & Strusevich, V. (2018). Preemptive models of scheduling with controllable processing times and of scheduling with imprecise computation: A review of solution approaches. *European Journal of Operational Research, 266*(3), 795–818.

Subramanian, A., Battarra, M., & Potts, C. (2014). An iterated local search heuristic for the single machine total weighted tardiness scheduling problem with sequence-dependent setup times. *International Journal of Production Research, 52*(9), 2729–2742.

Taillard, E. (1990). Some efficient heuristic methods for the flow shop sequencing problem. *European Journal of Operational Research, 47*(1), 65–74.

T'Kindt, V., & Billaut, J.-C. (2006). *Multicriteria Scheduling: Theory, Models and Algorithms* (2nd ed.). New York: Springer.

Wang, R., Liu, W., Xiao, L., Liu, J., & Kao, W. (2011). Path towards achieving of china's 2020 carbon emission reduction target-a discussion of low-carbon energy policies at province level. *Energy Policy, 39*(5), 2740–2747.

Zhang, B., Pan, Q.-K., Gao, L., Li, X.-Y., Meng, L.-L., & Peng, K.-K. (2019). A multiobjective evolutionary algorithm based on decomposition for hybrid flowshop green scheduling problem. *Computers & Industrial Engineering, 136*, 325–344.

Zhang, H., Zhao, F., & Sutherland, J. W. (2017). Scheduling of a single flow shop for minimal energy cost under real-time electricity pricing. *Journal of Manufacturing Science and Engineering, 139*(1).

Zhang, Q., & Li, H. (2007). Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation, 11*(6), 712–731.

Zhu, Z.-S., Liao, H., Cao, H.-S., Wang, L., Wei, Y.-M., & Yan, J. (2014). The differences of carbon intensity reduction rate across 89 countries in recent three decades. *Applied Energy, 113*, 808–815.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C., & Da Fonseca, V. (2003). Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation, 7*(2), 117–132.