European Journal of Operational Research 312 (2024) 1–21

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Invited Review

A review and classification on distributed permutation flowshop scheduling problems

Paz Perez-Gonzalez[a], Jose M. Framinan[a,b,*]

[a] Industrial Management, School of Engineering, University of Seville, Ave. Descubrimientos s/n, Seville E41092, Spain
[b] Laboratory of Engineering for Environmental Sustainability, University of Seville, Ave. Descubrimientos s/n, Seville E41092, Spain

ARTICLE INFO

Article history:
Received 10 August 2022
Accepted 1 February 2023
Available online 5 February 2023

Keywords:
Scheduling
Distributed permutation flowshop
Permutation flowshop scheduling

ABSTRACT

The Distributed Permutation Flowshop Scheduling (DPFS) problem is one of the fastest-growing topics in the scheduling literature, which in turn is among the most prolific fields in Operational Research (OR). Although the problem has been formally stated only twelve years ago, the number of papers on the topic is growing at a rapid pace, and the rising interest –both from academics and practitioners– on distributed manufacturing paradigms seems to indicate that this trend will continue to increase. Possibly as a side effect of this steady growth, the state-of-the-art on many decision problems within the field is far from being clear, with substantial overlaps in the solution procedures, lack of (fair) comparisons against existing methods, or the use of different denominations for the same problem, among other issues. In this paper, we carry out a review of the DPFS literature aimed at providing a classification and notation for DPFS problems under a common framework. Within this framework, contributions are exhaustively presented and discussed, together with the state-of-the-art of the problems and lines for future research.

1. Introduction

The Distributed Permutation Flowshop Scheduling (DPFS) problem is one of the fastest-growing topics in the scheduling literature, which in turn is among the most prolific fields in Operational Research (OR). Despite prior references addressing machine scheduling in distributed layouts, the first paper where the DPFS problem is formulated in its present state is Naderi & Ruiz (2010). Since then, the number of papers on the topic is growing at an exponential fashion, an interest that stems from the pre-eminence of distributed manufacturing in general as an avenue to reduce production costs and risks (see e.g., Kai Chan & Ho Chung, 2013). As a result, manufacturing companies with a single factory are less usual whereas supply chains with more than one plant become more common (see e.g., Fernandez-Viagas & Framinan, 2015; Naderi & Ruiz, 2010; Naderi & Ruiz, 2014). Furthermore, even for companies with a single factory, it is relatively common to have duplicate production lines for the same product (He et al., 1996), therefore the need to allocate the products to be manufactured among separated (distributed) resources arises. Apart from the specific (and, for many products, unrealistic) case where the production process solely consists of a single stage, classical parallel machine scheduling models cannot be employed to address such decisions, hence fuelling the research in this topic.

The DPFS can be seen as a generalisation of the Permutation Flowshop Scheduling (PFS) problem, where the jobs have to be processed sequentially in a set of machines or *flowshop* without changing its processing sequence from machine to machine (permutation constraint). In the DPFS, there are several flowshops where the jobs can be processed, so part of the decision problem is to allocate each job to one of the flowshops, and to sequence them in order to minimize some criteria. To the best of our knowledge, the first paper addressing the DPFS problem is Cao & Chen (2003), who named it the *parallel flowshop* scheduling problem. It should be noted that, before Cao & Chen (2003), some papers (e.g., Gooding et al., 1994; Musier & Evans, 1989; Sahinidis & Grossmann, 1991) have addressed scheduling problems related to parallel lines or flowshops, but in these contributions the total processing times of the jobs in each flowshop are assumed to be constant. Clearly, this is an oversimplification of the DPFS, which is then reduced to a parallel machine scheduling problem. Therefore, it can be considered that the first formalisation of the DPFS is given by Naderi & Ruiz (2010) and, since this seminal work, the number of contributions on the topic has increased steadily.

* Corresponding author at: Laboratory of Engineering for Environmental Sustainability, University of Seville, Ave. Descubrimientos s/n, Seville E41092, Spain.

E-mail addresses: pazperez@us.es (P. Perez-Gonzalez), framinan@us.es (J.M. Framinan).

https://doi.org/10.1016/j.ejor.2023.02.001

This rapid growth of the literature on the DPFS has led to a number of problems: i) It is relatively common that the authors omit existing methods when proposing new ones, which renders grasping the state of the art of some problems in the topic extremely hard; ii) There is substantial overlap in the proposed solution procedures. For example, as far as exact procedures are concerned, the same MILP model has been introduced as a new contribution in different papers and, for approximate procedures, the same metaheuristic –save small variations– has been proposed for the same or slightly different problems; iii) There is a lack of notation that makes it difficult to clearly understand the problem discussed in each paper, since they are usually denoted by acronyms that omit constraints and/or objectives. The problems inherent of using acronyms are exemplified by the fact that authors use different acronyms to denote the same problem; iv) In some cases, new sets of instances have been generated even when some testbeds were already available, thus making the outcome of the comparisons less generalisable; and v) The lack of a unifying classification further complicates the identification of related problems and the corresponding adaptation of their solution procedures.

The main goal of the paper is to address the aforementioned issues by conducting a review of the literature of the DPFS that includes a detailed problem description, a classification of the DPFS and related problems under a common framework, as well as an unified notation that also integrates related problems such as the distributed assembly scheduling problem, or the distributed hybrid flowshop. Nevertheless, in order to keep this review with a manageable number of references, the contributions on these problems will not be analysed. Therefore, the paper is structured as follows: In Section 2 a detailed description of the DPFS problem is given along with a general framework where related problems can be classified. An extension of the notation is introduced in Section 3 to encompass distributed scheduling problems. In Section 4, the methodology adopted is described, along with the main criteria to classify the references. For each criterion, the next sections present the state-of-the-art on the topic, including tables for each contribution and summarizing the most relevant information. More specifically, in Section 5 the canonical DPFS problems (i.e., single-objective, homogeneous DPFS problems without additional constraints) are reviewed; whereas single-objective, homogeneous DPFS problems with additional constraints are discussed in Section 6. Multi-objective DPFS problems are reviewed in Section 7, while non-deterministic DPFS problems are presented in Section 8. Moreover, heterogeneous DPFS problems (i.e., those addressing non-identical factories) are analysed in Section 9. Finally, in Section 10 the paper provides some conclusions and open research lines.

2. Problem description

In this section, the DPFS problem is formally described. Additionally, a general framework to denote these problems is provided, which allows introducing other well-known related problems, such as the Distributed Assembly Flowshop Scheduling problem and the Distributed Hybrid Flowshop Scheduling problem, among others.

2.1. Distributed permutation flowshop scheduling problem

In the (deterministic) Permutation Flowshop Scheduling (PFS) problem, a set of jobs $\mathcal{N} = \{1, \ldots, n\}$ are processed on a set of machines $\mathcal{M} = \{1, \ldots, m\}$ in series, and the route of all jobs is the same, i.e., the jobs visit all machines in the same order, being processed first on machine 1, then on machine 2 and so on. Therefore, each job has m operations, with processing times known in advance (denoted as p_{ij}, for $i \in \mathcal{M}$, $j \in \mathcal{N}$). In a schedule, jobs start as soon as possible (semi-active assumption, see e.g., Framinan et al., 2014), so a solution of the PFS problem is given by a sequence of jobs on each machine. Furthermore, the same sequence of jobs is kept throughout all machines under the permutation assumption, so in this case the problem has $n!$ possible solutions. The PFS problem is one of the most studied problems in OR (Fernandez-Viagas et al., 2020), and some papers reviewing the related literature or determining the state of the art methods to solve this problem for different objectives are, e.g., Fernandez-Viagas et al. (2017); Framinan et al. (2004); Hejazi et al. (2005); Minella et al. (2008); Pan & Ruiz (2013); Rossi et al. (2016); Ruiz & Maroto (2005); Vallada et al. (2008).

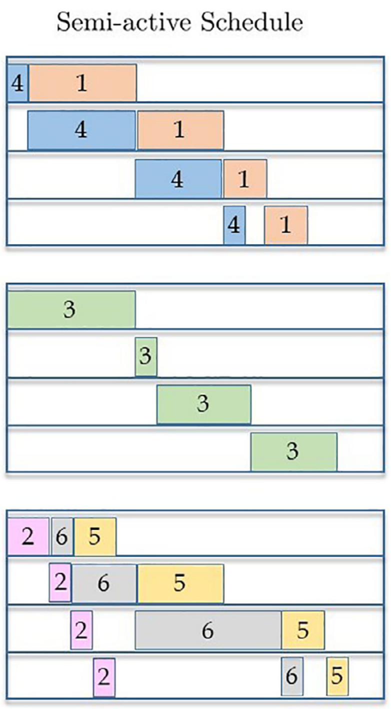
The layout of the Distributed Permutation Flowshop Scheduling (DPFS) problem is a generalization of that in the PFS, and consisting of a set of factories or shops $\mathcal{F} = \{1, \ldots, f\}$, each one being a flowshop. In their canonical definition (homogeneous DPFS), the flowshops are identical with m machines. Each job in the set $\mathcal{N}$ has to be assigned to one of the factories to be fully processed, i.e., once it has started the processing in a factory, it cannot be re-allocated to another one. When the flowshops are identical, the processing times do not depend on the factory, but only on the job and the machine (i.e., $p_{ijl} = p_{ij},\ \forall l \in \mathcal{F},\ \ i \in \mathcal{M},\ \ j \in \mathcal{N}$). The heterogeneous DPFS problem, where the number and/or types of machines in each factory are different is first addressed by Li et al. (2021a).

Since, for most of the criteria considered in scheduling problems, it can be proved that the optimal schedule is within the set of semi-active schedules, the scheduling decision problem is usually formulated as finding the best semi-active schedule (according to a given set of criteria) in a finite solution space. For the DPFS problem, it is clear that the size of the set of semi-active schedules is the same as for the parallel machine scheduling problem, which is known to be $\binom{n+m-1}{n} \cdot n!$ with m machines and n jobs. In this case, instead of assigning n jobs to m machines, n jobs are assigned to f factories, so the problem has $\binom{n+f-1}{n} \cdot n! = \frac{(n+f-1)!}{(f-1)!}$ semi-active schedules (taking into account the permutation constraint).

For the PFS problem, one semi-active schedule can be explicitly described by a permutation (sequence) of the jobs. However, in the DPFS problem different *solution representations* may be employed to describe a semi-active schedule. The solution representation is an important element for designing resolution methods (mainly heuristics and metaheuristics), as it determines the set of solutions that can be explored using this representation. There are three key features in a solution representation, i.e., *Completeness*, depending on whether a solution representation allows describing all semi-active schedules, or just a subset of the semi-active schedules, *Redundancy*, depending on whether different solutions within a solution representation describe the same semi-active schedule, and *Regularity*, depending on whether the size of the object(s) employed to describe a semi-active schedule is constant, or not.

Note that completeness is not an advantage *per se*, as an incomplete solution representation might be advantageously used to reduce the search space (provided that the schedules excluded are not good ones according to the objective function). In contrast, redundancy is always disadvantageous. Regularity allows an easier definition of neighbours of a solution. Two types of solution representations –adapted from the parallel machines literature– have been identified in the DPFS literature (see Fig. 1 for an illustrative example):

- Solution Representations Type 1: A semi-active schedule is defined by a (partial) sequence of jobs for each factory: $\Pi_l = (\pi_1^l, \ldots),\ \forall l \in \mathcal{F}$, with $\pi_j^l \in \mathcal{N}$, $\bigcup_{l=1}^{f} \Pi_l = \mathcal{N}$ and $\bigcap_{l=1}^{f} \Pi_l = \emptyset$. In this case, the allocation of jobs to the factories and their sequencing on each factory is given. This type of solution representation is complete, although in some cases re-



Solution Representations: Type 1

a) **Vector per factory**

$\Pi_1 = (4, 1)$
$\Pi_2 = (3)$
$\Pi_3 = (2, 6, 5)$

$$\begin{pmatrix} 4 & 1 & \\ 3 & & \\ 2 & 6 & 5 \end{pmatrix}$$

b) **Vector $n + f - 1$ size**

$$(4, 1, *, 3, *, 2, 6, 5)$$

c) **$2 \times n$-dimensional matrix (Job/Factory)**

$$\begin{array}{r} \text{Job} \\ \text{Factory} \end{array} \begin{pmatrix} 4 & 1 & 3 & 2 & 6 & 5 \\ 1 & 1 & 2 & 3 & 3 & 3 \end{pmatrix}$$

d) **$2 \times n$-dimensional matrix (Factory/Position in factory)**

$$\begin{array}{r} \text{Job} \\ \\ \text{Factory} \\ \text{Position in factory} \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{pmatrix} 1 & 3 & 2 & 1 & 3 & 3 \\ 2 & 1 & 1 & 1 & 3 & 2 \end{pmatrix} \end{array}$$

e) **Real numbers vector**

$$(1.33,\ 3.15,\ 2.25,\ 1.10,\ 3.81,\ 3.27)$$

Solution Representations: Type 2

$$(4, 3, 2, 6, 1, 5) + \text{FAM}$$
$$(4, 3, 2, 1, 6, 5) + \text{ECT}$$

Fig. 1. Example of a semi-active schedule in a DPFS problem with different solution representations.

dundancy and regularity are not avoided. This is the most extended type of solution representation used in the DPFS literature, which includes the following:

(a) A vector for each factory $\Pi_1, \ldots, \Pi_f$, being the most extended of this type even if it does not comply with regularity. Alternatively, it is represented using a jagged matrix with a row for each factory. This representation is employed e.g., by Naderi & Ruiz (2010, 2014).

(b) A vector with $n + f - 1$ components, including the n jobs and $f - 1$ separators (considering different characters) between factories. This representation avoids redundancy and it is regular, being employed by Lin & Ying (2016); Wang et al. (2017); Xiong et al. (2021); Ying & Lin (2017); Ying et al. (2017).

(c) A matrix formed by two rows and n columns: The first row represents a sequence of jobs $(\pi_1, \ldots, \pi_n)$, with $\pi_j \in \mathcal{N}$, and the second row provides the factory of each job $(f(\pi_1), \ldots, f(\pi_n))$, with $f(\pi_j) \in \mathcal{F}$. Although regular, this solution representation does not avoid redundancy, as e.g., swapping two columns in the matrix may not alter the semi-active schedule. This solution representation can be found e.g., in Cai et al. (2018); Li et al. (2019a); Wang et al. (2021, 2022a).

(d) A matrix formed by two rows and n columns: Each column belongs to a job, and the first row provides, for each job, the factory to be assigned, and the second row provides the position of the job in the factory. This regular solution representation is applied by Rifai et al. (2021, 2016), and it does not avoid redundancy as in the previous one.

(e) A vector containing n real numbers $(R_1, \ldots, R_n)$, with $R_j \in [1, f + 1)$: the integer part of R_j, $\lfloor R_j \rfloor \in \mathcal{F}$ indicates the factory, and the partial sequence of jobs in the factory is given by sorting the numbers R_j with the same integer part in non-decreasing order. This regular and redundant scheme is proposed by Zhang & Xing (2019).

- Solution Representations Type 2: A semi-active schedule is defined by a (complete) sequence of jobs $\Pi = (\pi_1, \ldots, \pi_n)$, with $\pi_j \in \mathcal{N}$, plus an assignment rule to allocate the jobs to the factories. In this case, the number of solutions is at maximum $n!$ (the same as that of the PFS problem), thus the representation is not complete (although it is regular and avoids redundancy). Regarding the assignment rules, they usually consist of iteratively allocating the jobs in the sequence either to the factory yielding the minimum value of a factory-specific indicator (denoted in the literature as the *local* indicator), or to the factory where a (*global*) indicator yields its minimum value. More specifically, the following indicators have been proposed:

  (a) For each job $\pi_j \in \Pi$, assign it to the factory with the current global lowest value of the objective function (i.e., before assigning job π_j). This rule has been first proposed by Naderi & Ruiz (2010) for the makespan and, in the parallel machines context, is similar to the First Available Machine (or FAM) rule. Note that this rule can be used only for min-max type criteria, as for min-sum criteria it is not possible to determine the factory with the current global lowest value of the objective function.

  (b) For each job $\pi_j \in \Pi$, assign it to the factory with the current local (factory-level) lowest value of the objective function (i.e., before assigning job π_j). This rule has been proposed for flowtime minimisation by Fernandez-Viagas et al. (2018b).

  (c) For each job $\pi_j \in \Pi$, assign it to the factory providing the lowest global value of the objective function

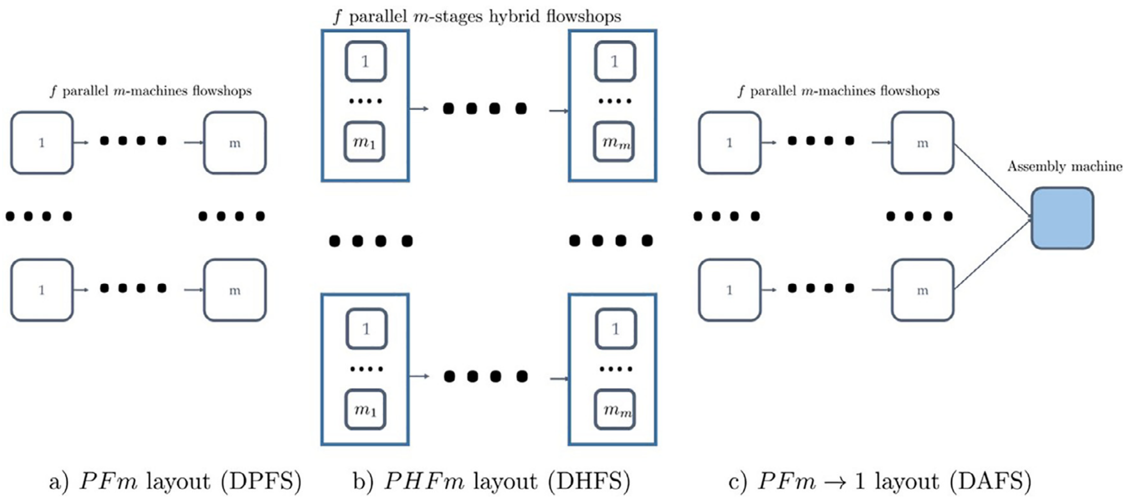

f parallel m-machines flowshops
f parallel m-stages hybrid flowshops
f parallel m-machines flowshops
Assembly machine
1
m
m_1
m_m
a) PFm layout (DPFS)
b) $PHFm$ layout (DHFS)
c) $PFm \rightarrow 1$ layout (DAFS)

Fig. 2. Distributed flowshop scheduling layouts and their corresponding notation as proposed in Section 3.

(i.e., after assigning π_j). In the parallel machines context, a similar rule is known as ECT (earliest completion time). Note that, while for the (identical) parallel machines cases ECT and FAM are equivalent, this is not the case for the homogeneous DPFS problem. This rule has been originally proposed by Naderi & Ruiz (2010) and it is the most used one in the DPFS literature (see e.g., Khare & Agrawal, 2021; Lu et al., 2021; Wang et al., 2016; Wang et al., 2013; Xu et al., 2014).

(d) For each job $\pi_j \in \Pi$, assign it to the factory with the current local (factory-level) lowest value of the objective function (i.e., after assigning π_j). This rule has been proposed for total flowtime minimisation by Fernandez-Viagas et al. (2018b).

(e) For each job $\pi_j \in \Pi$, assign it to a factory at random. This rule has been used by Li et al. (2019a).

For DPFS problems with additional constraints, these solution representations may be enhanced. For example Li et al. (2021c); Lu et al. (2022); Schulz et al. (2022); Wang & Wang (2020) add the velocities of the machines, or Hou et al. (2022) add the allocation of the jobs to the vehicles and the assignment of the vehicles to the factories to transport the jobs before processing.

2.2. General framework for the distributed flowshop scheduling

The Hybrid Flowshop Scheduling (HFS) problem and the Assembly Flowshop Scheduling (AFS) problem are two widely-studied problems related to the PFS. Their extensions to the distributed case have also been addressed in the literature and denoted as the Distributed Hybrid Flowshop Scheduling (DHFS) problem and the Distributed Assembly Flowshop Scheduling (DAFS) problem, respectively. In Fig. 2 the three layouts corresponding to DPFS, DHFS and DAFS problems are represented.

Although our review focuses on the DPFS problem and does not include DHFS and DAFS problems, we next briefly describe them in order to provide in Section 3 a unified notation. The common factor among these problems is the fact that there are, at least, two decisions to make: i) the allocation of jobs to factories; ii) the sequencing of the jobs in the corresponding factory.

2.2.1. Distributed hybrid flowshop scheduling problem

The Hybrid (or Flexible) Flowshop Scheduling (HFS) problem is an extension of the flowshop where each operation of a job can be carried out by a set of parallel machines. Therefore, instead of machines on series, the hybrid flowshop is formed by stages. Each stage is formed by a single machine layout or a parallel (identical, uniform or unrelated) machines layout. At least one each stage contains a set of parallel machines. Each job has to be processed on each stage as in the flowshop layout, i.e., a job cannot start on one stage if it has not been finished in the previous one. If a stage is formed by parallel machines, the job is assigned to one of them to be processed. Some papers reviewing this layout or identifying the state of the art methods are, e.g., Çolak & Keskin (2022); Fernandez-Viagas et al. (2018a, 2019); Naderi et al. (2014); Ribas et al. (2010); Ruiz & Vázquez-Rodríguez (2010).

The Distributed Hybrid Flowshop Scheduling (DHFS) problem is a generalization of the DPFS problem. In the DHFS problem, there is a set of f factories with (usually identical) hybrid flowshops, each one with m stages. Each job $j \in \mathcal{N}$ is assigned to one factory. The DHFS problem was considered for the first time by Vairaktarakis & Elhafsi (2007), denoted as *parallel flowlines*, and to the best of our knowledge, Ying & Lin (2018) is the first paper addressing the DHFS problem.

2.2.2. Distributed assembly flowshop scheduling problem

The layout in the so-called Assembly Scheduling (AS) problem consists of two consecutive stages, i.e., production of the parts, and parts assembly. Each job is composed of a set of m_1 components processed in the first stage by m_1 dedicated parallel machines, and the components are assembled in the second stage formed by a single assembly machine or a more complex layout as a set of parallel assembly machines or an assembly flowshop. The AS problem is described, analysed and the related literature reviewed by Framinan et al. (2018).

The layout in the Distributed Assembly Flowshop Scheduling (DAFS) problem is formed by f_1 factories in the first stage (generally, with identical flowshops) where the parts of each job are produced to be assembled in the second stage formed by f_2 factories (with, usually identical, assembly layouts depending on the case, as, for example, one factory with a single assembly machine). When the permutation constraint is assumed, this problem is denoted Distributed Assembly Permutation Flowshop Scheduling. Each job $j \in \mathcal{N}$ is assigned to one factory in the first stage, and the job is processed in the corresponding flowshop of this factory. Then the jobs are transferred to the second stage, and the jobs needed to generate a product are assigned to a factory where the parts are assembled. Therefore, each product consists of a given number of the previously produced jobs assembled. The DAFS problem was proposed for the first time by Hatami et al. (2013).

3. Problem notation

As mentioned in Section 1, there is no available notation for the DPFS problem, and only some attempts to denote the most basic variants of the problem have been tried. In order to bridge this gap, an extension of the well-known Graham's notation (Graham et al., 1979) is provided in this section. This well-known notation consists of three fields $\alpha|\beta|\gamma$, where α represents the layout, β the additional constraints, and finally γ the objective function(s) to be minimized. The justification of this proposal is, on the one hand, to avoid the confusion with the dedicated machines problem (see Framinan et al., 2018), and, on the other hand, to extend the problem to the heterogeneous variant. Therefore, the proposed notation is based on the identical and unrelated parallel machines layouts. More specifically, the following notation is proposed:

- PFm indicates that there are f identical factories in parallel formed by m-machines flowshop layouts;
- RF indicates that there are f unrelated factories in parallel formed by different flowshop layouts. This case is the so-called *heterogeneous* distributed flowshop, and implies that the processing times depend on the machine, the job and the factory (p_{ijl}, $i \in \mathcal{M}$, $j \in \mathcal{N}$, $l \in \mathcal{F}$). If the number of machines is the same for all factories, it is denoted as RFm. In the general case, when there are f factories (each one with a flowshop with m_l machines, $l \in \mathcal{F}$), the notation is RF. For specific cases, similar to the notation for the hybrid floshop, where the layout of each stage is indicated in the α field, the layout of each factory is indicated as well for the RF layout. Therefore, for example, two factories with a two-machines flowshop and a single machine in each factory respectively is denoted as $RF, F2, 1$.

The notation can be easily extended for the DHFS problem as follows:

- $PHFm$ indicates that there are f identical factories in parallel formed by m-stages hybrid flowshop layouts;
- RHF indicates that there are f unrelated factories in parallel formed by different hybrid flowshop layouts. This case is called the heterogeneous distributed hybrid flowshop.

Finally, and following the notation $\alpha_1 \rightarrow \alpha_2$ proposed by Framinan et al. (2018) for the assembly problem, this notation can be extended to the DAFS problem. For example, according to this notation, the problem with a PFm in the first stage and where the second stage consists of only one factory with a single assembly machine is denoted $PFm \rightarrow 1$. With this notation, more general problems can be defined such as, for example, $RHF \rightarrow RHF$ where the manufacturing stage consists of different factories with hybrid flowshop layouts (they can include even factories with flowshops or single multi-purpose machines), and the assembly stage consists of different heterogeneous factories too. This notation can be used in the case that there are transportation tasks after the production stages since, for transportation, all jobs belonging to a batch to be transported should be completed (in the same way that the components to be assembled, or the parts of a customer order, see Framinan et al., 2018). In this case α_2 can be modelled as 1, Pm or Rm if the number of transportation vehicles is one, or m identical or heterogeneous vehicles respectively. As with the assembly of components, the jobs must be grouped in batches to be transported. Therefore, the batch condition is intrinsic in the $\alpha_1 \rightarrow \alpha_2$ notation so it does not need to be included in the β field.

Regarding the β field, the case $\beta = \varnothing$ implies some general conditions for the problem, most of them assumptions from the PFS problem:

- All jobs and machines are available during the scheduling horizon;
- Each machine can only process one job at a time, and each job can be processed only on one machine at a time;
- Preemption of jobs is not allowed, so once one operation is started, it should be finished until completion;
- Setup times are sequence-independent and included in the processing times, or ignored.
- Infinite capacity of buffers between machines is considered;
- Operations of a given job cannot overlap in the machines;
- Once a job is assigned to a factory, it cannot change to another one. In the case of a parallel flowshop or parallel lines, lines cannot be interconnected or are in geographically distributed plants.

The most common constraints in the DPFS problem literature are summarized as follows:

- *prmu*, is the permutation consideration in the flowshop layout. In the distributed context, it models the fact that all jobs assigned to a factory should be processed in the same order along all machines.
- *no-wait*, if job waiting times are not allowed from a machine to the following one.
- *no-idle*, if zero machine idle time is allowed between jobs for all machines.
- *blocking*, which is equivalent to buffers equal to zero between machines. Therefore, even if a job has completed its operation on the machine, it remains on the machine until the next machine is available.
- *mixed no-idle / mixed blocking*, similar to the previous corresponding case, but the constraint applies to a number of machines.
- $\text{buffer}_i = b$ indicates that the buffer where jobs wait to be processed on the machine i has a limited capacity of b jobs. If the buffer is full, a finished job in the previous machine cannot leave this machine.
- $[d_j^-, d_j^+]$, named as due windows, is an extension of the due dates. The due dates-related objectives are computed depending on whether the completion time of each job falls in the due interval, or not.
- M_j, named the eligibility constraint, which is common in parallel machines layouts. In the distributed context, this constraint indicates that job j has a set (M_j) of factories where it can be processed, since not all the factories are able to process all jobs.
- v_i denotes the velocity of machine i and it is usually associated to a flowshop with an objective related to energy consumption, where the consumption or cost of processing a job depends on the speed of the machine. The higher the velocity, the lower the processing times and the higher the costs/energy consumption. Note that this constraint implies an additional decision in the scheduling problem, since, generally, for each machine there are several processing speeds, and the solution should include the velocity selected for each machine (see Section 7 to check the related literature in this context for the DPFS problem).
- r_j, to denote that the jobs have release dates different than zero. The schedule is not feasible if the starting time of a job is previous to its release date.
- Setup times: As usually in the literature, the setup-times are denoted differently depending on the type. They are denoted s_{ijk} for the most general case, when job j is scheduled immediately before the job k in machine i. The corresponding subindex is missing for other specific cases (s_j, s_{ij}, s_{jk}).
- PM, to denote machine unavailability due to preventive maintenance. A particular case is pm, implying machine unavailability due to periodic maintenance.

Other less common constraints have been considered in the literature, and, due to space limitations, they are explained for each reference along with the paper. Additionally, taking into account that the distributed scheduling problem represents an abstraction of (at least) part of the supply chain, some other decisions in the supply chain are affected by scheduling decisions. Particularly, transportation is an important issue in supply chain management as the production process is greatly affected by the availability of materials, or by the capacity of the buffers. Therefore, an interesting issue in distributed problems is to consider decisions related to scheduling and transportation in an integrated way. Three types of transportation can be considered:

- tbp, transportation before processing: It refers to the transportation of raw materials from their source to the factories. This translates into release dates of the jobs to be scheduled (this case is denoted r_j as usual), or into a factory-dependent transportation time that can be seen as release dates depending on the job and the factory (see e.g., Cai et al., 2018).
- tdp, transportation during processing: It refers to the transportation of semi-finished products between factories, so a job that has started its processing in one factory may be completed in another factory.
- tap, transportation after processing: It refers to the transportation of the finished product to its final destination (e.g., a distribution centre/retailer/customer). In this case, the jobs are grouped in batches (trucks/vehicles) to be delivered, so the decision problem includes the allocation of the jobs to the batches (depending on the truck/vehicle size). Regarding the transportation times, different approaches have been identified: Li et al. (2021b) assume that the transportation times depend on the specific batch, whereas Fu et al. (2022) and Hou et al. (2022) consider a vehicle routing problem after processing, so the transportation times between two customers is given (in their approach each job belongs to a different customer with a different location) and the departure of the vehicles from each factory starts once all jobs assigned to them are finished.

The field γ contains the objective function(s). Usually, the objectives depend on the completion times of each job, denoted as C_j. The due dates are denoted d_j, in case that the objective is due date- related. In the reviewed literature, the following objective functions have been identified:

- $C_{\max}$: Makespan or maximum completion time $C_{\max} = \max_{1 \le j \le n} C_j$
- $\sum C_j$: Total completion time $\sum C_j = \sum_{j=1}^{n} C_j$
- $\sum F_j$: Total flowtime $\sum F_j = \sum_{j=1}^{n} C_j - r_j$
- $\sum T_j$: Total tardiness $\sum T_j = \sum_{j=1}^{n} \max\{C_j - d_j, 0\}$
- $\sum E_j$: Total earliness $\sum E_j = \sum_{j=1}^{n} \max\{d_j - C_j, 0\}$
- $\sum U_j$: Number of tardy jobs, with $U_j = 1$ if $C_j > d_j$, 0 in the opposite case
- $\max L_j$: Maximum lateness $\max L_j = \max_{1 \le j \le n}(C_j - d_j)$
- $\max T_j$: Maximum tardiness $\max T_j = \max_{1 \le j \le n} \max\{C_j - d_j, 0\}$
- TC is in general to denote any function described as a Total Cost. There is not a standard expression for this type of cost, so the reader should consult the cited paper where TC is used as the objective function.

Due to space limitations, other less common objectives are explained for each reference along with the paper. In the case of weighted objectives, the weight of each job is denoted w_j, and for example, total weighted completion time is denoted $\sum w_j C_j$ and computed as $\sum w_j C_j = \sum_{j=1}^{n} w_j C_j$.

Regarding multiobjective approaches, only the approach to determine the Pareto set –denoted $\#(A, B)$–, and the linear convex combination approach –denoted $F_l(A, B)$– have been used in the reviewed literature. The notation has been taken from T'kindt & Billaut (2002), and the detailed definition of these approaches can be found in the same reference. Furthermore, multiobjective DPFS problems are usually discussed either in the context of energy-efficiency, or sustainability. These scenarios can be described as follows:

- Energy-efficiency scenarios consider objective functions (denoted in general TEC) where the energy consumption usually depends on the state and speed of the machines. As there is not a standard expression for the TEC, and it is difficult to include the variety of formulae used in the reviewed literature, the reader should check the specific paper to know exactly the expression of each TEC objective.
- Sustainable scenarios include economic, environmental and social aspects in the objective function. The economic aspects are usually reflected using classic objectives, being the most extended the makespan. Additional costs (modelled in general as TC) are: worker's salaries Fathollahi-Fard et al. (2021); inventory costs Alaghebandha et al. (2019); and cost of a plant's utilization loss, start-up or overhead Rifai et al. (2016). Environmental aspects basically refer to waste reduction (see, for example, Fathollahi-Fard et al., 2021). Note that this aspect can be considered an economic one if the waste consists of defective products due to different machine faulty ratios. Finally, the consideration of social aspects is less extended. The objective called Negative-Social-Impact (NSI) is modelled by Lu et al. (2021) as a cost depending on the processing times. Other considerations are the number of operators working on a machine (using a particular technology) (see, for example, Fathollahi-Fard et al., 2021).

Note that previous objectives can be employed for non-distributed scheduling problems. However, for the DPFS problem, more sophisticated objective functions (denoted factory-related) can be considered. To the best of our knowledge, only Cao & Chen (2003) and Zhao et al. (2021) consider factory-related objectives. Cao & Chen (2003) compute a factory-related objective function, called the total factory-weighted makespan, denoted $\sum w_l C_{\max}^l$ that can be computed as $\sum_{l=1}^{f} w_l C_{\max}^l$ where $C_{\max}^l = \max_{j \in \mathcal{F}_l} C_j$ is the makespan of the jobs assigned to the factory $l \in \mathcal{F}$, and w_l is the weight of the factory l. Zhao et al. (2021) consider the objective that they name Factory Load Based (FLB), which is an energy efficiency objective computed as follows: $FLB = Std(FL^1, \ldots, FL^f)$ with $FL^l = F_l(C_{\max}^l, TEC^l), \forall l \in \mathcal{F}$, and Std the standard deviation.

4. Review methodology

To conduct the review a bibliographic search with the words "distributed flowshop scheduling" and "distributed flow shop scheduling" in the Abstract/Title/Keywords has been carried out using SCOPUS, excluding Conference Papers, or papers not written in English. The snowballing technique has been applied, so each one of the references has been analysed and both the cited papers and papers citing the reference have been considered as well. The selected papers have been denoted according to the notation introduced in Section 3 and have been classified along three axes: 1) the type of decision problem addressed, 2) the type of solution procedure(s) proposed to solve the problem, and 3) the testbed(s) employed to test the efficiency of the solution procedures proposed (if any). This classification may allow a more exhaustive discussion and would result in a clear state-of-the-art picture for each decision problem and/or solution procedure. The resulting groups are discussed in the next subsections.

4.1. Types of decision problems

The papers reviewed have been grouped into five categories (Classic, Constrained, Multi-objective, Non-deterministic and Heterogeneous DPFS problems) as follows:

- Classic DPFS problems include contributions on the DPFS problem without additional constraints (except the *prmu* constraint) and a single objective. Most of these papers deal with makespan or the total completion time, with only one reference addressing the total tardiness. These problems are reviewed in Section 5;
- Constrained DPFS problems are single-objective problems that include at least one additional constraint. The most addressed constraints are (mixed) blocking/buffer- related, no-wait, (mixed) no-idle and setup times. The most studied objective is the makespan, although the total completion time has been considered for blocking and no-idle, and total tardiness for blocking. Moreover, some contributions consider additional constraints, being the most extended those related to preventive maintenance. The papers in this group are reviewed in Section 6;
- Multi-objective DPFS problems are classified depending on the multi-objective approach adopted, i.e.: Pareto or Linear combination of objectives. Given the pre-eminence of papers addressing energy considerations with a Pareto approach, a specific category has been considered. Again, the makespan is the most used objective along with other cost measures. The majority of references include constraints taken from real-life cases, being the velocity of the machines the most extended constraint in the energy efficiency approach. These works are reviewed in Section 7;
- Non-deterministic DPFS problems are reviewed in Section 8, where the references with fuzzy, stochastic and uncertain problems are included. There are not many papers in this category, and all of them consider objectives related to makespan;
- Heterogeneous DPFS problems. The number of references in this group is low due to the complexity of this problem with non-identical factories, considering the cases with different numbers of machines, layouts or only different processing times among factories. The related references are analysed in Section 9.

Figure 3 provides a scheme of the classification indicating the number of references reviewed within each group.

4.2. Types of solution procedures

The solution procedures proposed in the reviewed literature have been classified within one of the categories in Table 1, grouped into exact methods and approximate methods. Regarding exact methods, the most extended is the Mixed Integer Linear Programming (MILP) model formulation. For non-linear models, the method is noted as MIP. Some references use Constraint Programming (CP), or Benders Decomposition (BD). Regarding approximate methods, approximation schemes such as the Fully Polynomial Time Approximation Scheme (FPTA) and Polynomial-Time Approximation algorithm with a fixed worst-case ratio (PTA) have been applied in some contributions. The most common solution procedures are constructive heuristics and metaheuristics. Among the constructive heuristics, the most employed are sophisticated dispatching rules, adaptations of the well-known NEH heuristic for the PFS problem (Nawaz et al., 1983) or, in some cases, composite rules. Given the specifics of these methods, these are indicated as CH and not further described in detail in the subsections, so the interested reader must check the cited reference. Regarding metaheuristic approaches, the vast majority are variants of well-known metaheuristics, and due to space reasons, only the acronyms are given in Table 1.

4.3. Testbeds employed

Authors have used different sets of instances depending on the problem and the data needed to be solved. In some cases, the same set of instances has been used for several references, in order to provide a fair comparison among the methods. For this reason, the most used testbeds, or those available online in the reviewed literature are discussed in this section. In the contributions where the authors propose a new testbed not used in other papers, no information is given and the reader should check the original reference.

- Testbed TB1: This testbed proposed by Naderi & Ruiz (2010) has become the most used testbed for the DPFS problem. It is formed by two sets of instances: The first set is composed of small-size instances, $n \in \{4, 6, 8, 10, 12, 14, 16\}$, $m \in \{2, 3, 4, 5\}$ and $f \in \{2, 3, 4\}$, with five instances for each combination of the parameters, being a total of 420 instances, with the processing times uniformly distributed in the interval [1, 99]. The second set of large-size instances is formed by the well-known Taillard's instances (Taillard, 1993) for the PFS problem, with $n \times m \in \{20 \times 5, 20 \times 10, 20 \times 20, 50 \times 5, 50 \times 10, 50 \times 20, 100 \times 5, 100 \times 10, 100 \times 20, 200 \times 10, 200 \times 20, 500 \times 20\}$, with ten different replicates per size, being 120 instances in total. These instances are combined for seven values of $f \in \{2, 3, 4, 5, 6, 7\}$, thus providing a set of 720 instances. This test is available at http://soa.iti.es.
- Testbed TB2: The set of instances by Ying et al. (2017) is based on the well-known benchmark by Vallada et al. (2015) for the $Fm|prmu|C_{\max}$ problem, with two sets of instances (large and small). Ying et al. (2017) consider only the 240 large instances obtained by the combination of $n \in \{100, 200, 300, 400, 500, 600, 700, 800\}$ and $m \in \{20, 40, 60\}$, with 10 instances per size. The processing times are uniformly distributed between 1 and 99. The instances are combined for $f \in \{2, 3, 4, 5, 6, 7\}$, obtaining a total of 1440 instances. The original testbed by Vallada et al. (2015) is available at http://soa.iti.es
- Testbed TB3: This testbed is proposed by Khare & Agrawal (2021) as an extension of TB1 with due dates generated as in the procedure by Hasija & Rajendran (2004), and adapted for the DPFS problem as follows: $d_j = P_j(1 + \frac{rand}{2f})$ with $P_j = \sum_{i=1}^{m} p_{ij}$, and $rand \sim U[0, 1]$. The testbed is available on-line at https://doi.org/10.17632/4tkhdx4jjx
- Testbed TB4: This testbed consists of a due-dates based set of instances that is proposed by Ribas et al. (2019) for the DPFS problem with blocking constraints. Note that, although the blocking constraint does not influence, in principle, the testbed, the due-date generation procedure must take this fact into account in order not to generate instances with an excessive number of tardy jobs. In this case, the due dates are generated uniformly in the interval $[\alpha \cdot LB, \beta \cdot LB]$, with $\alpha = 1 - T - R/2$ and $\beta = -T + R/2$ as defined by Potts & Van Wassenhove (1982), with T the tardiness factor of jobs and R the dispersion range of due dates. Additionally, LB is the lower bound for the makespan proposed by Companys & Mateo (2007) for the $Fm|blocking|C_{\max}$ problem, multiplying it by $\frac{n/f+m-1}{n+m-1}$ for the distributed problem. The data can be found in http://hdl.handle.net/2117/123116.
- Testbed TB5: This is a set of instances for the DPFS problem with preventive maintenance. Mao et al. (2021) gener-

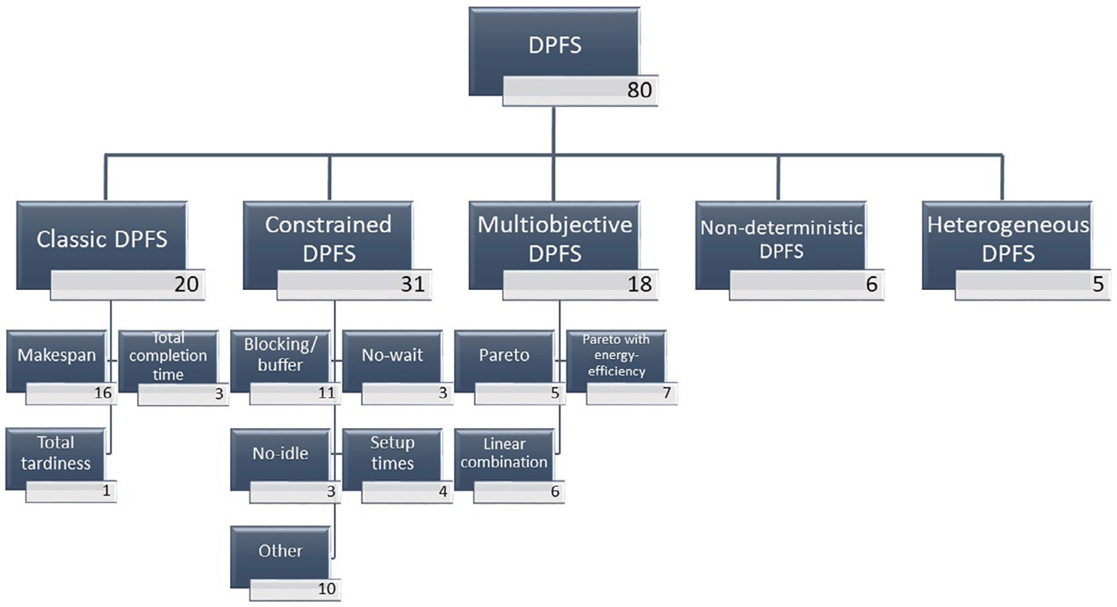


Fig. 3. Classification of the distributed permutation flowshop scheduling problems: References reviewed.

Table 1
Acronyms of the methods reviewed.

Name	Description	Name	Description
Exact methods			
BDA	Bender Decomposition Algorithm	MILP	Mixed Integer Linear Programming model
CP	Constraint Programming	MIP	Mixed Integer Programming model
Approximate methods: Constructive and Approximation Schemes			
CH	Constructive Heuristic	FPTAS	Fully Polynomial Time Approximation Scheme
		PTA	Polynomial-Time Approximation algorithm
Approximate methods: Metaheuristics			
ABC	Artificial Bee Colony	ILS	Iterated Local Search
BSO	Brain Storm Optimization	Jaya	Jaya Algorithm
BWO	Black Widow Optimization	KCA	Knowledge-based Cooperative Algorithm
COA	Collaborative Optimization Algorithm	LNS	Large Neighborhood Search
CRO	Chemical Reaction Optimization	MA	Memetic Algorithm
DEA	Differential Evolution Algorithm	MAA	Multi-Agent Approach
EA	Evolutionary Algorithm	MBO	Migratory Bird Optimization
EDA	Estimation of Distribution Algorithm	MBF	Monarch Butterfly Optimization
ES	Evolution Strategy	PSO	Particle Swarm Optimization
FFO	Fruit Fly Optimization	SS	Scatter Search
GA	Genetic Algorithm	TS	Tabu Search
HHO	Harris Hawks Optimisation	VNS	Variable Neighbourhood Search
HIA	Hybrid Immune Algorithm	WCA	Water Cycle Algorithm
HS	Harmony Search	WOA	Whale Optimization Algorithm
IG	Iterated Greedy Algorithm		

ates a benchmark based on a set of instances generated by Pan et al. (2019b) for a distributed assembly problem, and Ruiz et al. (2007) for the preventive maintenance assumption, with $f \in \{2, 4, 6\}$, $n \in \{100, 200, 300, 400, 500\}$, $m \in \{5, 8, 10\}$. The processing times are uniformly distributed in the interval [1,100], maintenance operation times uniformly distributed in [50,150], and the interval time of the maintenance operation (maximum utilization of the machine previous to the maintenance operation) uniformly distributed in $[25 \cdot n/f, 37.5 \cdot n/f]$ (i.e., it is set between the 50% and 75% of the processing time on each machine, so only one maintenance operation is carried out).

- Testbed TB6: This is a set of instances for the DPFS problem with energy efficiency considerations. Chen et al. (2019) generate a set of instances inspired by Ding et al. (2016) for the problem $Fm|prmu, v_i|\#(C_{\max}, TEC)$. The parameters are: $f \in \{2, 3, 4, 5\}$, $n \in \{20, 40, 60, 80, 100\}$, $m \in \{4, 8, 16\}$, $v_i \in \{1, 1.3, 1.55, 1.75, 2.1\}$. The processing times are uniformly distributed in the interval [5,50], and energy consumption depends on the velocity of the machine as $4 \times v_i^2$ (kiloWatt). The authors generate 10 instances per size, providing 600 instances in total (note that the velocity of each machine is part of the decision problem, so it has no influence in the number of instances).

5. Classic DPFS problems

In this section, the DPFS problem is reviewed for three classic objective functions from the scheduling literature: makespan, total completion time and total tardiness. Table 2 provides a summary of the reviewed contributions, which are discussed in the next subsections.

Table 2
Classic DPFS problems. [1] Processing times are multiplied by a constant.

Reference	Exact	Approximate	Testbed
$PFm\|prmu\|C_{\max}$			
Naderi & Ruiz (2010)	MILP	CH, VNS	TB1
Gao & Chen (2011a)		GA	TB1
Gao & Chen (2011b)		CH	TB1
Zhang & Van De Velde (2012)		PTA	TB1
Gao et al. (2012)		GA	TB1
Gao et al. (2013)		TS	TB1
Lin et al. (2013)		IG	TB1
Wang et al. (2013)		EDA	TB1
Naderi & Ruiz (2014)		SS	TB1
Xu et al. (2014)		HIA	TB1
Fernandez-Viagas & Framinan (2015)		IG	TB1
Dong et al. (2017)		FPTAS	TB1
Bargaoui et al. (2017)		CRO	TB1
Ruiz et al. (2019)		IG	TB1
Li et al. (2019a)		ABC	TB1[1]
Hamzadayı (2020)	MILP, BD		TB1
Ren et al. (2021)		MAA	
$PFm\|prmu\|\sum C_j$			
Fernandez-Viagas et al. (2018b)		CH, EA	TB1
Pan et al. (2019a)		CH, ABC, SS, ILS, IG	TB1
Ali et al. (2021)	MILP	TS	TB1
$PFm\|prmu\|\sum T_j$			
Khare & Agrawal (2021)	MILP	IG, HHO	TB3

5.1. Makespan objective

Since the preliminary paper of Cao & Chen (2003), no paper has been published on the DPFS until the best-known contribution by Naderi & Ruiz (2010). In their paper, the authors review the related literature, and describe and formalize the problem $PFm|prmu|C_{\max}$, providing the following contributions:

- A theoretical result on the reduction of the number of solutions for the makespan case due to the fact that the optimal solution includes at least one job on each factory (only if $n \geq f$). Therefore, for this objective, the authors prove that the number of solutions is $\binom{n-1}{f-1} \cdot n!$.
- Six MILP models, based on different ways to model the variables of the problem;
- Twelve heuristics based on existing heuristics from the PFS literature, resulting from the combination of six classic constructive heuristics. The best results are obtained by an adaptation of the NEH heuristic, with two allocation methods widely used in the later literature;
- One metaheuristic based on the Variable Neighbourhood Search (VNS);
- The two sets of instances forming the TB1 described in Section 4.3.

From this starting point, a number of papers proposing more efficient approximate methods for the problem were published. First, Gao & Chen (2011a) and Gao et al. (2012) employ different GA versions, providing statistically better results than the VNS method by Naderi & Ruiz (2010). The same authors in Gao & Chen (2011b) focus on constructive heuristics, developing an improved NEH algorithm that outperformed the heuristic methods in Naderi & Ruiz (2010). Later, Gao et al. (2013) propose a TS algorithm that improved the results provided by the GA. Lin et al. (2013) present for the first time some versions of the IG algorithm by Ruiz & Stützle (2007) for the distributed flowshop problem, widely used in the rest of literature presented in this paper. They compare their proposal to those by Gao & Chen (2011a); Gao et al. (2013); Naderi & Ruiz (2010). Wang et al. (2013) and Xu et al. (2014) generate variants of the EDA and HIA metaheuristics, respectively and compare their proposal only with the methods by Naderi & Ruiz (2010). A SS algorithm by Naderi & Ruiz (2014) outperforms all the methods presented in the previous papers, and in parallel, Fernandez-Viagas & Framinan (2015) do the same using an IG algorithm, providing some theoretical results embedded in different local search methods trying to reduce the space of solutions. At this point, the state-of-the-art is unclear due to the coincidence in time of both papers. Later, Bargaoui et al. (2017) outperform the methods by Naderi & Ruiz (2010) and Fernandez-Viagas & Framinan (2015) using a CRO algorithm. Independently, Ruiz et al. (2019) design three new versions of the IG algorithm, one of them with a local search embedded and based on the VNS algorithm previously tested in Naderi & Ruiz (2010), and the other one with an interesting structure, named two-stage IG, where the IG algorithm is applied to the complete layout in the first stage, and to the factory providing the makespan in the second stage. They compare their three IG versions with the IG algorithm by Fernandez-Viagas & Framinan (2015), the HIA by Xu et al. (2014) and the SS algorithm by Naderi & Ruiz (2014) and outperform them. Again, the state-of-the-art is unclear due to the coincidence in time with Bargaoui et al. (2017). Ren et al. (2021) propose a multi-agent approach using a Nash Q-learning algorithm. The method is compared to the three proposed by Ruiz et al. (2019) for three stopping criteria based on time. However, they do not use the complete TB1 for the comparison. Furthermore, their method is not compared to Bargaoui et al. (2017). Furthermore, in an independent work, Li et al. (2019a) present a problem that is modelled as a $PFm|prmu|C_{\max}$ problem. For this problem, they propose two metaheuristics (ABC and IG), which are compared with the VNS algorithm by Naderi & Ruiz (2010) as well as with other methods for related problems (i.e., the MA by Deng & Wang (2017) for the $PFm|prmu|\#(C_{\max}, \sum T_j)$ problem, and the BSA by Lin et al. (2017) for the distributed assembly problem). Note that, as a consequence of the independent works and the lack of comparison among the most recent existing methods, the state-of-the-art on approximate solution procedures for the $PFm|prmu|C_{\max}$ remains unclear.

Regarding exact procedures for the problem, it has been already mentioned that Naderi & Ruiz (2010) present six MILP formulations. Out of these six formulations, the two best-performing ones are compared by Hamzadayı (2020) with two new MILP formulations and their corresponding Benders decomposition algorithms. The computational results show that the Benders decompositions outperform the rest of the exact methods, obtaining new best-known solutions for some instances in TB1 that were too large to be solved to optimality.

Finally, the specific case $PF2|prmu|C_{\max}$ has been addressed using approximation algorithms. Since there are only two machines per factory, the main decision consists of assigning the jobs to the factories since, once assigned, their optimal sequence for each factory can be found using Johnson's rule. However, note that this does not reduce the problem to a parallel machines problem, since the jobs assigned to each factory influence the schedule given by Johnson's rule. Zhang & Van De Velde (2012) present a $\frac{3}{2}$-approximation algorithm for the $PF2|prmu|C_{\max}$ problem with two factories, and a $\frac{12}{7}$-approximation algorithm for the case $f = 3$, being that complexity of both algorithms is $\mathcal{O}(n \log n)$. Dong et al. (2017) develop a FPTAS for the same problem $PF2|prmu|C_{\max}$, based on a dynamic programming algorithm to solve exactly the problem for a fixed number of factories.

5.2. Total completion time

The first paper addressing the $PFm|prmu|\sum C_j$ problem is Fernandez-Viagas et al. (2018b), where new representations of the solutions different from those by Naderi & Ruiz (2010) are proposed (see Section 2), each one with a speed-up mechanism to compute the objective function of the partial solutions. The authors test the representation of the solutions embedded in a NEH procedure. An EA is proposed as approximate solution procedure, and this algorithm is favourably compared with the best-so-far method known for the $PFm|prmu|C_{\max}$ problem (i.e., Naderi & Ruiz, 2014 and Fernandez-Viagas & Framinan, 2015). Pan et al. (2019a) design some new neighbourhoods mixing sequence and assignment, and improve the speed-ups by Fernandez-Viagas et al. (2018b). Finally, they compare four metaheuristics (IG, ILS, SS and ABC) with the EA by Fernandez-Viagas et al. (2018b) for the same problem, and with two IG algorithms by Lin et al. (2013) and Fernandez-Viagas & Framinan (2015) respectively, and the SS algorithm by Naderi & Ruiz (2014) for the $PFm|prmu|C_{\max}$ problem. They found that the most efficient method is the ILS. Finally, Ali et al. (2021) propose a TS algorithm, but the authors do not compare their proposal with that by Pan et al. (2019a), and conduct a comparison with the results provided by Fernandez-Viagas et al. (2018b) but without re-coding and running the experiments in the same computer. As a result, the state-of-the-art regarding approximate procedures for the $PFm|prmu|\sum C_j$ problem is not clear.

Finally, regarding exact methods for the problem, the only proposal is Ali et al. (2021), who adapt one of the MILP models by Naderi & Ruiz (2010) for the makespan, simply modifying the objective function.

5.3. Total tardiness

Khare & Agrawal (2021) develop a MILP model for the $PFm|prmu|\sum T_j$ problem and propose two constructive heuristics and two metaheuristics: a version of the HHO algorithm, and an IG algorithm. They use TB3 (see Section 2) to compare their proposal to six metaheuristics for the $PFm|prmu|C_{\max}$ problem (i.e., those by Bargaoui et al., 2017; Gao & Chen, 2011a; Naderi & Ruiz, 2014; Xu et al., 2014), and an additional method for the $Fm|prmu|\sum T_j$ problem. This paper constitutes the only contribution for this problem, as well as for unconstrained DPFS problems with a different objective than makespan or flowtime.

6. Constrained DPFS problems

In this section, the DPFS problems with additional constraints are reviewed. First, in Section 6.1, the case with limited buffer is reviewed, including the blocking case (where the buffer capacity between machines in the flowshop of each factory is zero). Then, no-wait and no-idle constraints are considered in Sections 6.2 and 6.3 respectively. Setup time constraints are reviewed in Section 6.4, and finally, more complex constraints, or problems jointly considering different constraints are analysed in Section 6.5. In most cases, the objectives are the classical (makespan and total completion time), but other (single) objectives have been considered in the literature reviewed (recall that the multi-objective literature is reviewed separately in Section 7). Table 3 summarises the contributions, indicating the specific problem addressed, the solution procedures applied, and the testbeds used for the comparisons.

6.1. Blocking and buffer-related problems

Ribas et al. (2017) is the first reference to address the $PFm|prmu, blocking|C_{\max}$ problem. The authors provide a MILP model and compare several constructive heuristics, i.e., the adaptation of the 12 heuristics proposed for the $PFm|prmu|C_{\max}$ problem by Naderi & Ruiz (2010), 8 heuristics that combine the two allocation methods by Naderi & Ruiz (2010) with four existing constructive heuristics for the $Fm|prmu, blocking|C_{\max}$ problem, and 13 new heuristics proposed in the paper. Finally, two metaheuristics based on ILS and IG are proposed, both of them using an embedded VNS as the local search. These metaheuristics are compared with the best methods so far for the case without a blocking constraint, i.e., the IG algorithm by Fernandez-Viagas & Framinan (2015), and the SS algorithm by Naderi & Ruiz (2014), using the best constructive heuristic as the initial sequence. The IG algorithm proposed provides the best results. Independently to the work by Ribas et al. (2017); Ying & Lin (2017) consider the same problem and provide the same MILP model. Three versions of an IG metaheuristic are proposed and compared with methods for the $Fm|prmu, blocking|C_{\max}$ problem, so they complement the paper by Ribas et al. (2017) as they adapt single-factory methods to the distributed problem. One of these versions yields the best results among the existing methods for the $Fm|prmu, blocking|C_{\max}$ and the $PFm|prmu, blocking|C_{\max}$ problems. Zhang et al. (2018) propose two different ways for computing the makespan for the $Fm|prmu, blocking|C_{\max}$ problem and develop a DEA that is compared with the IG algorithm by Fernandez-Viagas & Framinan (2015), the SS algorithm by Naderi & Ruiz (2014) for the classical problem, and the IG algorithm by Ying & Lin (2017) for the blocking problem, missing the work by Ribas et al. (2017), possibly because the latter authors denoted it as a parallel flowshop problem instead of a distributed flowshop. Shao et al. (2020a) propose a FFO algorithm and compare it with the methods by Ying & Lin (2017); Zhang et al. (2018), and by Ribas et al. (2017). They improve 516 solutions of the 720 instances of TB1. In parallel to Shao et al. (2020a); Zhao et al. (2020) include the same MILP model as Ribas et al. (2017), and propose a metaheuristic based on DEA, and compare it to two methods from the distributed literature, i.e., Ruiz et al. (2019) for the $PFm|prmu|C_{\max}$ problem and Zhang et al. (2018) for the problem under consideration, as well as two more methods from the PFPS literature with blocking constraints. They do not compare their proposal with the method by Ribas et al. (2017) or the FFO algorithm by Shao et al. (2020a). Later, Chen et al. (2021a) develop six constructive heuristics and compare them with the adaptations of the NEH-based algorithms by Naderi & Ruiz (2010); Ruiz et al. (2019) for the $PFm|prmu|C_{\max}$ problem, and implement two of the constructive heuristics by Ribas et al. (2017) and one from

Table 3
Constrained DPFS problems.[1] The same model as the proposed by Ribas et al. (2017). [2] Specific machines with the blocking constraints are added. [3] The same model as the proposed by Pan et al. (2021).

Problem	Reference	Exact	Approximate	Testbed
Blocking and buffer related problems				
$PFm\|prmu, blocking\|C_{\max}$	Ribas et al. (2017)	MILP	CH,IG, ILS	TB1
	Ying & Lin (2017)	MILP[1]	IG	TB1
	Zhang et al. (2018)		DEA	TB1
	Zhao et al. (2020)	MILP[1]	DEA	TB1
	Shao et al. (2020a)		FFO	TB1
	Chen et al. (2021a)		CH,IG	TB1
	Karabulut et al. (2022a)	MIP, CP	ES	TB1
$PFm\|prmu, blocking\|\sum C_j$	Chen et al. (2021b)		IG	TB1
$PFm\|prmu, blocking\|\sum T_j$	Ribas et al. (2019)	MILP	CH,IG	TB4
$PFm\|prmu, mixed\ blocking\|C_{\max}$	Shao et al. (2021)		CH,IG	TB1,TB2[2]
$PFm\|prmu, \text{buffer}_i\|C_{\max}$	Zhang & Xing (2019)		DEA	
No-wait problems				
$PFm\|prmu, no\text{-}wait\|C_{\max}$	Lin & Ying (2016)	MILP	IG	TB1
	Komaki & Malakooti (2017)	MILP	CH,VNS	TB1
	Shao et al. (2017)	MILP	IG	TB1
No-idle problems				
$PFm\|prmu, no\text{-}idle\|C_{\max}$	Ying et al. (2017)	MILP	IG	TB1,TB2
$PFm\|prmu, mixed\ no\text{-}idle\|C_{\max}$	Cheng et al. (2019)	MILP	IG	
$PFm\|prmu, mixed\ no\text{-}idle\|\sum C_j$	Li et al. (2021d)	MILP	IG	
Setup times constraints				
$PFm\|prmu, s_{ij}\|TC$	Alaghebandha et al. (2019)	MILP	GA, WCA, MBO	
$PFm\|prmu, s_{ijk}\|C_{\max}$	Huang et al. (2020)	MILP	IG	
	Karabulut et al. (2022b)	MILP, CP	ES	
$PFm\|prmu, s_{jk}\|C_{\max}$	Huang et al. (2021a)		CH,ABC	
	Guo et al. (2022)		FFO	
Other constraints				
$PFm\|prmu, blocking, robot, deteriorating\text{-}jobs\|C_{\max}$	Li et al. (2019b)	MIP	IG	
$PFm\|prmu, s_{jk}, mixed\ no\text{-}idle\|C_{\max}$	Rossi & Nagano (2021)	MILP	IG	
$PFm\|prmu, no\text{-}wait, s_{ijk}, PM\|C_{\max}$	Miyata & Nagano (2021)	MILP	IG	
$PFm\|prmu, job\text{-}family, s_{ill'}\|C_{\max}$	Pan et al. (2021)	MILP	EA	
$PFm\|prmu, PM\|C_{\max}$	Mao et al. (2021)		CH,IG	TB5
$PFm\|prmu, PM\|\sum C_j$	Mao et al. (2022)	MIP	EA	TB5
$PFm\|prmu, r_j\|\sum PF(C_j)$	Li et al. (2022)	MILP	CH,IG	
$PFm\|prmu, job\text{-}family\|C_{\max}$	Pan et al. (2022)	MILP	GA,PSO,HS,ABC,Jaya	
$PFm\|prmu, tap\|C_{\max}$	Fu et al. (2022)	MILP	BWO	
$PFm\|prmu, job\text{-}family, s_{ill'}\|\sum T_l$	Wang et al. (2022b)	MILP[3]	CH,IG	

Shao et al. (2020a). Additionally they propose six composite heuristics, combining the constructive heuristics with a local search. Finally, they propose an IG algorithm, and compare it with the metaheuristics for the problem $PFm|prmu|C_{\max}$ by Fernandez-Viagas & Framinan (2015); Lin et al. (2013); Ruiz et al. (2019), and with the metaheuristics by Ribas et al. (2017); Shao et al. (2020a); Ying & Lin (2017); Zhang et al. (2018). The IG algorithm proposed by the authors statistically outperforms all the methods from the related literature. However, the state-of-the-art for the problem is not closed so far, since the authors missed the paper by Zhao et al. (2020). Karabulut et al. (2022a) and Karabulut et al. (2022b) propose a MIP model and a CP as exact methods, and an ES metaheuristic to solve the same problem with sequence-dependent-setup-times instead of the blocking constraint. The authors do not reimplement the previous proposed methods in the literature, but compare their results to the heuristic published by Chen et al. (2021a). As it can be seen from the above summary, the state-of-the-art of approximate solution procedures for this problem is far from being clear.

Shao et al. (2021) develop constructive heuristics and an IG algorithm for the $PFm|prmu, mixed\ blocking|C_{\max}$ problem. They compare their proposal using extended versions of TB1 and TB2 adding the machines where blocking is considered as in Riahi et al. (2017) for the PFS problem. The constructive heuristics are compared with other methods from the related literature, i.e., those by Naderi & Ruiz (2010); Ruiz et al. (2019) for the classic problem and the methods by Ribas et al. (2017); Shao et al. (2020a) for the blocking case. Additionally, they compare their IG algorithm with the metaheuristics proposed by Ribas et al. (2017); Shao et al. (2020a); Ying & Lin (2017); Zhang et al. (2018) for the $PFm|prmu, blocking|C_{\max}$ problem, and with an adaptation made by the authors of the metaheuristics by Fernandez-Viagas & Framinan (2015); Lin et al. (2013); Naderi & Ruiz (2014); Ruiz et al. (2019) for the $PFm|prmu|C_{\max}$ problem, Lin & Ying (2016) for the $PFm|prmu, no\text{-}wait|C_{\max}$ problem, Ying et al. (2017) for the $PFm|prmu, no\text{-}idle|C_{\max}$ problem, and Huang et al. (2020) for the $PFm|prmu, s_{ijk}|C_{\max}$ problem. In total, the authors reimplement 12 methods, all of them were outperformed by their proposal.

With respect to other objectives different than the makespan, Ribas et al. (2019) develop a MILP model for the $PFm|prmu, blocking|\sum T_j$ problem. A comparison of several constructive heuristics is provided, and an IG algorithm with a VNS algorithm embedded is developed. The concept of a critical line/flowshop/factory is introduced for the local search in two strategies, the one for the highest total tardiness, and the other one with the job for which the highest tardiness is found. They compare their proposal with the different constructive heuristics combined with the two strategies. An interesting result is obtained, since the best variant of the IG algorithm is not the one with the best constructive heuristic due to the effect of the computational times. In this case, the allocation mechanism consisting of selecting the flowshop with the highest total tardiness is the best option. They compare their proposal with adapted versions of the best-known methods from the $PFm|prmu|C_{\max}$ problem, i.e., the IG algorithm by Fernandez-Viagas & Framinan (2015) and the SS algorithm by Naderi & Ruiz (2014).

Chen et al. (2021b) propose an IG based metaheuristic for the $PFm|prmu, blocking|\sum C_j$ problem. Using TB1, the authors compare their proposal with the methods by Ribas et al. (2017); Shao et al. (2020a); Ying & Lin (2017) for the problem with the makespan objective, Fernandez-Viagas et al. (2018b); Pan et al. (2019a) for the $PFm|prmu|\sum C_j$ problem, and Ruiz et al. (2019) for the $PFm|prmu|C_{\max}$ problem.

Regarding buffer-constrained problems (which may be considered a general version of the problem with blocking constraints), Zhang & Xing (2019) consider the problem with limited buffers between machines $i-1$ and i ($i>1$) or $PFm|prmu, \text{buffer}_i|C_{\max}$ problem. They propose a differential DEA algorithm, implemented in a combinatorial search space after running the continuous version and compare their method with the IG algorithm by Ying & Lin (2017) for the blocking problem, and with the SS algorithm by Naderi & Ruiz (2014) for the unconstrained corresponding problem. These methods were selected as Ying & Lin (2017) claim that these were the best metaheuristics for the $PFm|prmu, blocking|C_{\max}$ problem, but since these methods were not compared with those proposed by Ribas et al. (2017) (see above), it is not clear if the IG algorithm by Ribas et al. (2017) or other subsequent blocking-related methods can outperform the current procedures for the problem with limited buffers.

6.2. No-wait problems

Lin & Ying (2016) is the first paper addressing the $PFm|prmu, no\text{-}wait|C_{\max}$ problem, i.e., jobs are processed without interruption either on or between any two consecutive machines in an assigned factory. The authors present a MILP model and propose a metaheuristic named Iterated Cocktail Greedy, which is a version of the IG algorithm with a more sophisticated destruction phase. They propose different variants of this metaheuristic, but do not compare their proposal with existing methods for the related $PFm|prmu|C_{\max}$ problem. Komaki & Malakooti (2017) propose a new MILP model based on the completion time distance matrix, related to the delay introduced in the first machine to obtain the no-wait schedule for a given sequence. The authors present some results related to speed-up methods to reduce the computational times. A number of constructive heuristics from the $Fm|prmu, no\text{-}wait|C_{\max}$ literature are adapted and tested to determine an initial solution for the metaheuristic approach. Several variants of the VNS algorithm are compared among them, but other methods from the related literature are not tested. Shao et al. (2017) include a different MILP model, similar to the one by Lin & Ying (2016) and a new constructive heuristic based on the NEH algorithm. Furthermore, three IG variants using different local search methods are proposed. An extensive computational comparison is carried out to compare their proposal to the eight existing metaheuristics from the $PFm|prmu|C_{\max}$ problem, the IG algorithm by Lin & Ying (2016) and the VNS algorithm by Komaki & Malakooti (2017) for the no-wait case. This work allows determining the state-of-the-art method for the no-wait variant of the problem, which is their IG algorithm combined with a variable neighbourhood descent. In their results, the authors show that, on the one hand, the IG algorithm by Lin & Ying (2016) is better than the later VNS algorithm by Komaki & Malakooti (2017), and on the other hand, that the IG version proposed by Fernandez-Viagas & Framinan (2015) for the $PFm|prmu|C_{\max}$ problem outperforms the IG and VNS algorithms developed specifically for the no-wait case by Lin & Ying (2016) and Komaki & Malakooti (2017) respectively, thus streamlining the importance of adapting existing methods from related problems. Regarding the state-of-the-art of exact methods for the problem, the different MILP models discussed above have not been compared among them.

6.3. No-idle problems

The first contribution to no-idle DPFS problems is Ying et al. (2017), who develop a MILP model and propose an IG-based metaheuristic for the $PFm|prmu, no\text{-}idle|C_{\max}$ problem. Additionally, they extend the testbed by Naderi & Ruiz (2010) with a new set of instances to test the methods for the distributed problem based on the well-known benchmark by Vallada et al. (2015) for the $Fm|prmu|C_{\max}$ problem, denoted TB2 in this paper (see Section 4.3). They compare their proposal with the state-of-the-art method by Pan & Ruiz (2014) for the $Fm|prmu, no\text{-}idle|C_{\max}$ problem, but they do not adapt any existing method from the distributed literature.

Cheng et al. (2019) consider the $PFm|prmu, mixed\ no\text{-}idle|C_{\max}$ problem, proposing a MILP model based on the models by Naderi & Ruiz (2010) and Pan & Ruiz (2014) for the $PFm|prmu|C_{\max}$ and $Fm|prmu, mixed\ no\text{-}idle|C_{\max}$ problems, respectively. An IG-based metaheuristic is proposed and compared to an adapted version of the IG algorithm by Pan & Ruiz (2014) for the $Fm|prmu, mixed\ no\text{-}idle|C_{\max}$ problem.

Regarding total completion time as the objective, the $PFm|prmu, mixed\ no\text{-}idle|\sum C_j$ problem has been addressed by Li et al. (2021d), presenting a MILP model and an IG algorithm. A speed-up procedure is employed to compute the objective function. The authors generate their own set of instances with only four instances per size, even if, at the time of the publication, already published testbeds could be used (e.g., the one used by Cheng et al., 2019). The proposed IG algorithm is compared with the methods by Rossi & Nagano (2021) for $PFm|prmu, s_{ijk}, mixed\ no\text{-}idle|C_{\max}$, Lin et al. (2013); Naderi & Ruiz (2014) for $PFm|prmu|C_{\max}$, and Cheng et al. (2019) for $PFm|prmu, mixed\ no\text{-}idle|C_{\max}$. However, no method from the distributed literature with the same objective has been adapted.

6.4. Setup times

In this section, the contributions are reviewed depending on the specific setup times considered. Regarding setups depending on the job and machine (s_{ij} constraint), the only contribution is Alaghebandha et al. (2019) for the $PFm|prmu, s_{ij}|TC$ problem, where the total costs include the setup costs, inventory holding cost for work-in-process-inventory (machine- and job-dependent costs), and inventory holding cost for finished products (job-dependent). These costs are computed using the EOQ model. They propose two mathematical (non-linear) models and their corresponding linearisation, and three metaheuristics based on WCA, MBF and GA and generate their own set of instances to compare the models against them, and the best-performing one with the metaheuristics.

Regarding the case of sequence-dependent setup times (s_{jk} constraint), the first reference is Huang et al. (2021a) for the $PFm|prmu, s_{jk}|C_{\max}$ problem. The authors propose a MILP model and develop constructive heuristics and an ABC-based algorithm. They compare their methods with those from the related literature, more specifically the five metaheuristics by Bargaoui et al. (2017); Fernandez-Viagas et al. (2018b); Gao & Chen (2011b); Naderi & Ruiz (2014); Ruiz et al. (2019) from the DPFS problem, and three additional methods from other related problems. Later, Guo et al. (2022) consider the same problem, proposing a MILP model and a FFO algorithm comparing the method to six metaheuristics: the method by Huang et al. (2021a) for the same problem, by Fernandez-Viagas et al. (2018b) for the $PFm|prmu|\sum C_j$ problem, Huang et al. (2020) for the $PFm|prmu, s_{ijk}|C_{\max}$ problem (see below), by Huang et al. (2021b) for a distributed assembly flowshop scheduling problem, and two additional methods from other related literature. The authors do not use the same set of instances as in the previous contribution, therefore it could be interesting to test the methods to check if the results depend on the testbed.

Finally, the problem with machine and sequence-dependent setup times (s_{ijk} constraint) is first addressed by Huang et al. (2020), who propose a MILP model and an IG algorithm for the $PFm|prmu, s_{ijk}|C_{\max}$ problem. They generate their own set of instances, with the parameters based on the related literature, and compare their IG algorithm with nine metaheuristics, some of them DPFS-related, i.e., the methods by Bargaoui et al. (2017); Gao & Chen (2011a); Ruiz et al. (2019) for the $PFm|prmu|C_{\max}$ problem, Zhang et al. (2018) for the $PFm|prmu, blocking|C_{\max}$ problem, Fernandez-Viagas et al. (2018b) for the $PFm|prmu|\sum C_j$ problem, as well as others for the PFS and the HFS problems. The problem is also considered by Karabulut et al. (2022b) where two exact methods are proposed: a new MILP model, based on that by Naderi & Ruiz (2010), and a CP formulation. They compare both methods but, since they do not include the model by Huang et al. (2020) in their comparison, it is not possible to discern whether their proposal is the most efficient MILP model for the problem, or not. These authors also propose an ES-based method and compare it with the methods by Huang et al. (2020) for the same problem, by Huang et al. (2021a) for the $PFm|prmu, s_{jk}|C_{\max}$ problem, and by Ruiz et al. (2019) for the $PFm|prmu|C_{\max}$ problem.

6.5. Other constraints

Several papers address very specific constraints for the DPFS problem. Li et al. (2019b) consider the problem $PFm|prmu, blocking, robot, deteriorating\text{-}jobs|C_{\max}$, where blocking is generated by a robot which moves the jobs from machine to machine in each flowshop. The authors provide a MIP, and propose an IG algorithm which is compared to other methods, one of them the ABC algorithm from Li et al. (2019a) for the $PFm|prmu|C_{\max}$ problem, and two other methods not specified.

Rossi & Nagano (2021) propose a MILP model for the $PFm|prmu, s_{ijk}, mixed\ no\text{-}idle|C_{\max}$ problem, a constructive heuristic based on the NEH algorithm, and an IG-based metaheuristic. These are compared with the heuristic by Ruiz et al. (2019) for the $PFm|prmu|C_{\max}$ problem, and the metaheuristics by Ruiz et al. (2019) for the $PFm|prmu|C_{\max}$ problem, by Cheng et al. (2019) for the $PFm|prmu, mixed\ no\text{-}idle|C_{\max}$ problem, and Huang et al. (2020, 2021a) for the problem $PFm|prmu, s_{ijk}|C_{\max}$ problem. In their adaptation, the speed-up methods by Rossi & Seido Nagano (2019) and Fernandez-Viagas & Framinan (2015) are included whenever possible.

Mao et al. (2021) include preventive maintenance in the DPFS problem, i.e., the $PFm|prmu, PM|C_{\max}$ problem, where a preventive maintenance activity of a machine-dependent duration is scheduled after the machine has processed a number of jobs with accumulated total processing times equal or greater than a specific duration. After the preventive maintenance, the accumulated total processing time is reset to zero. A constructive heuristic and an IG-based metaheuristic are proposed. The constructive heuristic is compared to those by Naderi & Ruiz (2010) and Ruiz et al. (2019) for the $PFm|prmu|C_{\max}$ problem. The IG algorithm is compared to six methods: Zhang et al. (2018) for the $PFm|prmu, blocking|C_{\max}$ problem, Ruiz et al. (2019) for the $PFm|prmu|C_{\max}$ problem, Lin & Ying (2016) for the $PFm|prmu, no\text{-}wait|C_{\max}$ problem, and three additional methods from other related problems. Mao et al. (2022) consider again the preventive maintenance approach, this time with the total completion time as the objective, $PFm|prmu, PM|\sum C_j$. They propose a mathematical model and an EA metaheuristic. The metaheuristic is compared to five methods adapted from the distributed literature: their previous proposal for the $PFm|prmu, PM|C_{\max}$ problem (Mao et al., 2021), two methods from Pan et al. (2019a) and one from Fernandez-Viagas et al. (2018b) for the problem $PFm|prmu|\sum C_j$, the metaheuristic by Ruiz et al. (2019) for the $PFm|prmu|C_{\max}$ problem; and five methods from related problems. Miyata & Nagano (2021) address the $PFm|prmu, no\text{-}wait, s_{ijk}, PM|C_{\max}$ problem where the duration of the maintenance operations depends on the maintenance level, the machine, and the sequence. A MILP model is presented for the problem, and an IG algorithm is developed and compared with the methods by Ruiz et al. (2019) for the $PFm|prmu|C_{\max}$ problem, and with those by Shao et al. (2020a) for the $PFm|prmu, blocking|C_{\max}$ problem. The authors do not include any methods from the no-wait or sequence-dependent-setup-times literature in their comparison.

Pan et al. (2021) focus on the cellular manufacturing case to model a group scheduling problem consisting of processing jobs in a single manufacturing cell. In their problem, each factory is a cell with identical flowshops, and there are different families with different numbers of jobs to be assigned to the cells (the complete family), and processed consecutively without interruption by jobs from other families. The problem is similar to the one proposed by Meng & Pan (2021) in the heterogeneous version (see Section 9), where, instead of job families, the jobs are composed by sublots. The problem can be denoted as $PFm|prmu, job\text{-}family, s_{ill'}|C_{\max}$, with $s_{ill'}$ machine and family sequence dependent setup time, and different decisions have to be made: the allocation of families to cells, sequencing the families in each flowshop, and sequencing the jobs within the families. A MILP model is proposed and tested for small-size instances. An EA-based metaheuristic is developed and compared with the methods by Fernandez-Viagas et al. (2018b) for the $PFm|prmu|\sum C_j$ problem, Lin et al. (2013) for the $PFm|prmu|C_{\max}$ problem, some methods from the distributed assembly literature by Pan et al. (2019b); Sang et al. (2019) and an additional method from another related problem. Later, a similar problem but considering the total tardiness as the objective function is solved by Wang et al. (2022b) ($PFm|prmu, job\text{-}family, s_{ill'}, |\sum T_l$), where $\sum T_l$ is the total tardiness of families, instead of the jobs. The same MILP as the previous paper by Pan et al. (2021) is included, only changing the objective function, which is not used in the rest of the paper. An IG-based metaheuristic is proposed, and compared to the IG algorithms: by Fernandez-Viagas & Framinan (2015); Lin et al. (2013); Ruiz et al. (2019) for the $PFm|prmu|C_{\max}$ problem, by Huang et al. (2020) for the $PFm|prmu, s_{ijk}|C_{\max}$ problem, by Rossi & Nagano (2021) for the $PFm|prmu, s_{jk}, mixed\ no-idle|C_{\max}$ problem; and as previously commented the EA-based metaheuristic by Pan et al. (2021) for the $PFm|prmu, job\text{-}family, s_{ill'}, |C_{\max}$ problem.

Li et al. (2022) address the $PFm|prmu, r_j|\sum PF(C_j)$ problem where $PF(x)$ is an increasing stepwise linear function for given intervals $(0, D_1]$,...,$[D_K, \infty)$, where D_K are defined as delivery dates, modelling the called cumulative pay-offs to maximize the total revenue. The authors present a MILP model for the problem, some theoretical properties, nine constructive heuristics, and an IG-based metaheuristic. The IG algorithm is compared with the methods by Fernandez-Viagas & Framinan (2015); Lin et al. (2013); Naderi & Ruiz (2014) for the $PFm|prmu|C_{\max}$ problem and with the four methods by Pan et al. (2019a) for the $PFm|prmu|\sum C_j$ problem.

Pan et al. (2022) develop a MILP model for the $PFm|prmu, job\text{-}family|C_{\max}$ problem, and propose five metaheuristics based on the GA, PSO, HS, ABC and Jaya algorithm that are compared to the methods by Fernandez-Viagas et al. (2018b) for the $PFm|prmu|\sum C_j$ problem, and those by Meng & Pan (2021) for the $RFm|prmu, job\text{-}family, s_{ijk}|C_{\max}$ problem. The EA method in Pan et al. (2021) for the $PFm|prmu, job\text{-}family, s_{ijk}|C_{\max}$ problem is not included in the comparisons, which leaves the state-of-the-art on this problem unclear.

Finally, Fu et al. (2022) address the $PFm|prmu, tap|C_{\max}$ problem, where transportation is modelled as a vehicle routing prob-

Table 4
Multi-objective DPFS problems. [1] The setups are dependent of the layer, related to the reentrant constraint. [2] The same model as the proposed by Wang et al. (2020). [3] The instances are generated using the same parameters than for TB6. [4] The instances are generated using the same parameters than TB6 and extended with setup times.

Problem	Reference	Exact	Approximate	Test
Pareto approach				
$PFm\|prmu, reeentrant, s_{ij}^1\|\#(C_{\max}, TC, \sum Tj)$	Rifai et al. (2016)	MIP	LNS	
$PFm\|prmu\|\#(C_{\max}, \sum T_j)$	Deng & Wang (2017)		MA	
$PFm\|prmu, M_j, tbp\|\#(C_{\max}, \max L_j, TC)$	Cai et al. (2018)	MIP	CH,NSGA-II	
$PFm\|prmu, no\text{-}wait, s_{ijk}\|\#(C_{\max}, \sum w_jT_j)$	Shao et al. (2019)	MILP	EA	
$PFm\|prmu, reeentrant, s_{ijk}\|\#(C_{\max}, TC, \sum Tj)$	Rifai et al. (2021)		LNS	
Pareto with energy-efficiency considerations				
$PFm\|prmu, no\text{-}idle\|\#(C_{\max}, TEC)$	Chen et al. (2019)	MIP	COA	TB6
$PFm\|prmu, v_i\|\#(C_{\max}, TEC)$	Wang & Wang (2020)	MILP	KCA	TB6
$PFm\|prmu, v_i, s_{ij}\|\#(C_{\max}, TEC)$	Wang et al. (2020)	MIP	WOA	
	Wang et al. (2022a)	MIP[2]	WOA	TB6[3]
$PFm\|prmu, v_i\|\#(\sum C_j, TEC)$	Li et al. (2021c)	MILP	NSGA-II	TB6[4]
$PFm\|prmu, no\text{-}idle, v_i\|\#(\sum T_j, TEC, FLB)$	Zhao et al. (2021)		Jaya	
$PFm\|prmu, \text{buffer}_i, v_i\|\#(C_{\max}, TEC)$	Lu et al. (2022)		KCA	
Linear combination approach				
$PF2\|prmu, dj = d\|Fl(\sum w_l C_{\max}^l, \max T_j)$	Cao & Chen (2003)	MIP	TS	
$PFm\|prmu, [d_j^-, d_j^+]\|Fl(\sum w_jTj, \sum w_j'E_j)$	Jing et al. (2020)		IG	
$PF6\|prmu, p\text{-}batch(\infty), prmtn, pm\|Fl(\sum w_jTj, \sum w_j'E_j)$	Xiong et al. (2021)	MILP	IG,TS,VNS,GA	
$PFm\|prmu, v_i, tap\|Fl(C_{\max}, TEC)$	Li et al. (2021b)	MIP	WOA	
$PFm\|prmu, [d_j^-, d_j^+], tap\|Fl(\sum w_jT_j, \sum w_j'E_j)$	Hou et al. (2022)	MILP	BSO	
$PFm\|prmu, no\text{-}wait, s_{ijk}\|F_l(C_{\max}, \max T_j)$	Allali et al. (2022)	MILP	CH,GA, ABC, MBO	

lem after processing the jobs (belonging to a customer) in a factory. All jobs transported in a given vehicle should be finished, and the objective is to minimize the maximum delivery time. The authors propose a MILP and a BWO metaheuristic. The metaheuristic is compared to three methods from the related literature.

7. Multiobjective DPFS problems

Among multiobjective DPFS problems, Pareto is the most extended approach, although some papers consider the linear combination. Furthermore, in view of the remarkable number of papers combining objectives related to energy-efficiency with classic objectives, we have classified the references in three groups, i.e.: Pareto, Pareto with energy-efficiency considerations, and Linear combination. As in Section 3, the notation employed is from T'kindt & Billaut (2002), where a detailed definition of multiobjective approaches can be also found. Similarly to the tables in the previous sections, in Table 4 the papers are presented chronologically and classified according to the approach. These are further discussed in the next sections.

7.1. Pareto approaches

Rifai et al. (2016) consider the $PFm|prmu, reeentrant, s_{ij}|\#(C_{\max}, TC, \sum Tj)$ problem, in which the reentrant constraint consists of processing jobs consisting of a number of layers, which in turn consists of a set of operations. Both processing and setup times depend on the machine, the job and the layer. The authors present a MIP model and a LNS metaheuristic, which is compared to other versions of the LNS, and to the well-known NSGA-II (Deb et al., 2002) for multi-objective scheduling problems. Similar to Rifai et al. (2021, 2016) consider the $PFm|prmu, reeentrant, s_{ijk}|\#(C_{\max}, TC, \sum Tj)$ problem. Note that the setup times now are machine and sequence-dependent, while in the previous paper were machine-, job- and layer- dependent. The authors propose a new version of their LNS, which are compared also with the NSGA-II in the previous paper, and with another general method for multi-objective problems. Deng & Wang (2017) present a MA to solve the $PFm|prmu|\#(C_{\max}, \sum T_j)$ problem, which is compared with the NSGA-II and a random algorithm. Shao et al. (2019) present a MILP model for the $PFm|prmu, no\text{-}wait, s_{ijk}|\#(C_{\max}, \sum w_jT_j)$ problem and propose an EVA to provide approximate solutions. These methods are compared with nine methods from the literature, three of them from the distributed literature, even if some of the objectives and constraints are quite different from those in the problem under study.

To the best of our knowledge, the first paper considering transportation constraints is Cai et al. (2018). They consider a transportation operation previous to the processing of the jobs in a DPFS problem. Additionally, they assume eligibility constraints, which have not been previously considered. In their paper, a multi-objective function considering makespan, maximum lateness and the total cost related to the factory setup and transportation is optimized. The problem can be denoted as $PFm|prmu, M_j, tbp|\#(C_{\max}, \max L_j, TC)$. In this problem, tbp can be seen as a release time depending on the job and the factory. The authors apply some constructive heuristics from the literature and propose a new version of the NEH algorithm together with an improved and adapted NSGA-II metaheuristic. The authors generate a set of instances based on TB1 to test their proposal and compare it with the original NSGA-II and with another multi-objective metaheuristic.

7.2. Pareto with energy-efficiency considerations

Among the papers in the DPFS multi-objective literature, energy-efficiency approaches are widely extended, with the objective composed by makespan and total energy consumption ($\#(C_{\max}, TEC)$) standing out. Chen et al. (2019) tackle the $PFm|prmu, no\text{-}idle, v_i|\#(C_{\max}, TEC)$ problem and propose a MIP model together with some theoretical results for the $Fm|prmu, no\text{-}idle|C_{\max}$ problem that can be applied to the problem at hand in order to save energy without affecting the makespan. A COA is proposed and compared with NSGA-II and another metaheuristic proposed for a flowshop problem with different constraints but similar objective function. Testbed TB6 (see Section 4.3) is developed to compare their proposal with existing methods. A similar problem without the no-idle constraint (i.e., the $PFm|prmu, v_i|\#(C_{\max}, TEC)$ problem) is studied by Wang & Wang (2020), who propose a MILP model and a KCA metaheuristic, using TB6 in their experiments. Wang et al. (2020) address the $PFm|prmu, s_{ij}, v_i|\#(C_{\max}, TEC)$ problem with non-anticipatory setup times. Note that, although these setup times could be added to the processing times, they influence the value of the objective function as they are energy cost-dependent. For this problem, a WOA is proposed to provide approximate solutions. As they include setup times, the set of instances used in the computational experiments is generated for the specific problem based on the TB6 data. They compare their metaheuristic with other general metaheuristics for multi-objective problems, including NSGA-II. The same problem with setup times is considered again by Wang et al. (2022a), inspired on a welding manufacturing process where a flowshop is composed of stages containing different welding machines. This problem cannot be modelled as a hybrid flowshop since, when a job is scheduled in a stage with a given number of welding machines, the processing time is reduced proportionally, so the processing time is controllable depending on the assignment to the welding machine. The authors consider a distributed problem where each factory consists of a flowshop with a different number of stages, and where each stage has a maximum number of welding machines to be assigned to each job. As the authors consider a TEC objective function, there is a trade-off between the allocation of available welding machines to a job and their energy consumption. Therefore, for each factory the problem can be seen as a flowshop with machine velocities and the problem can be denoted as $PFm|prmu, s_{ij}, v_i|\#(C_{\max}, TEC)$, resulting in the same decision problem as in Wang et al. (2020). Note that another similar problem is presented in Wang et al. (2021) for the heterogeneous case (see Section 9), and to the already discussed Wang & Wang (2020) without the setup times constraint. In this occasion, the authors present a MIP model with minor changes with respect to the models presented in the other papers. As in the paper by Wang et al. (2020), the authors present a WOA, with the same initialization and minor changes and it is compared to the same methods. Furthermore, the authors present an application of the decision problem to a real manufacturing scenario. Finally, Lu et al. (2022) propose a KCA based method for the $PFm|prmu, \text{buffer}_i, v_i|\#(C_{\max}, TEC)$ problem, comparing it with two generic multi-objective methods, a method from the flowshop literature, and with the proposal by Wang & Wang (2020) for the $PFm|prmu, v_i|\#(C_{\max}, TEC)$ problem.

Apart from the combination of makespan and energy costs, other objectives have been addressed. Li et al. (2021c) propose a MILP model for the $PFm|prmu, v_i|\#(\sum C_j, TEC)$ problem together with an NSGA-II variant. Although the authors do not use TB6 in their experimentation, the instances employed are generated with the same pattern. Their proposal is compared with the metaheuristics by Deng & Wang (2017); Wang & Wang (2020) for other distributed problems with energy-efficiency objectives, and with two general methods for multi-objective problems, including NSGA-II. Zhao et al. (2021) address the $PFm|prmu, no\text{-}idle, v_i|\#(\sum T_j, TEC, FLB)$ problem with FLB (factory load balancing) defined as the standard deviation among the factories of a linear function depending on the makespan and the TEC of each factory. The authors proposed a Jaya algorithm, which is compared to the method by Chen et al. (2019) for the $PFm|prmu, no\text{-}idle|\#(C_{\max}, TEC)$ problem.

7.3. Linear combination approach

Regarding DPFS problems with a linear combination of objectives, Cao & Chen (2003) present a TS algorithm to solve a problem where each factory (line) has a two-machine permutation flowshop, and the objective is a linear combination of two functions: the total weighted makespan of each factory and –albeit expressed differently in the original paper– another criterion equivalent to the $\max T_j$ with common due dates. Therefore, their problem can be expressed as $PF2|prmu, d_j = d|Fl(\sum w_l C^l_{\max}, \max T_j)$. The authors propose a MIP model for the problem, state some theoretical properties, and a TS metaheuristic whose performance is tested on illustrative examples and compared to the results obtained by complete enumeration.

Jing et al. (2020) develop an IG algorithm for the $PFm|prmu, [d^-_j, d^+_j]|Fl(\sum w_j T_j, \sum w'_j E_j)$ problem. The authors extend TB1 by adding due windows, and compare their proposal with five methods from the DPFS literature, i.e., that by Fernandez-Viagas et al. (2018b); Pan et al. (2019a) for the $PFm|prmu|\sum C_j$ problem, and by Fernandez-Viagas & Framinan (2015); Lin et al. (2013); Naderi & Ruiz (2014) for the $PFm|prmu|C_{\max}$ problem.

A complex problem is considered in Xiong et al. (2021) where jobs in a precast concrete process have to be processed in 6-stage factories. The stages are not always available due to working hours/non-working hours, a case similar to the periodic maintenance constraint pm, (see e.g., Perez-Gonzalez et al., 2020). Due to the non-availability of the stages, job preemption is allowed in some stages. Additionally, one of the stages can process simultaneously any number of jobs. The problem can be approximately expressed as $PF6|prmu, p-batch(\infty), prmtn, pm|Fl(\sum w_j T_j, \sum w'_j E_j)$. For this problem, a MILP model, and IG, TS, VNS and GA- based metaheuristics are developed. The proposals are compared to the exact results obtained by the MILP model for small-size instances, and among them for large-size instances.

The only problem including the TEC objective in the linear convex combination approach is Li et al. (2021b), where the $PFm|prmu, v_i, tap|Fl(C_{\max}, TEC)$ problem is addressed, with transportation after processing that involves allocating the jobs to the batches to be delivered to the final customer. Both objective functions depend on the completion times (in this case the delivery time) of the batches. The authors propose a MIP and a WSO, being the latter compared with two methods from other scheduling problems. Finally, Hou et al. (2022) address the $PFm|prmu, [d^-_j, d^+_j], tap|Fl(\sum w_j T_j, \sum w'_j E_j)$. In this case, the tap constraint is similar to that by Fu et al. (2022) discussed in Section 6.5, adding the decision about the vehicle routing problem to deliver the jobs to the customers. Note that each job belongs to a customer, and that the completion time of the jobs is given by the delivery time. The authors propose a MILP and a BSO-based metaheuristic. The BSO algorithm is compared with five methods, three of them the same methods used by Fu et al. (2022) in their comparison from related literature, and two from the DPFS literature by Mao et al. (2021) and Pan et al. (2019a).

Finally, the $PFm|prmu, no\text{-}wait, s_{ijk}|F_l(C_{\max}, \max T_j)$ problem is addressed by Allali et al. (2022). The authors propose a MILP model, two constructive heuristics, and three metaheuristics (GA, ABC and MBO). The authors compare the approximate methods with the solutions provided by a solver for the MILP model on small-size instances. Additionally, the performance of the metaheuristics is compared for large instances.

8. Non-deterministic DPFS problems

In this section, DPFS contributions including non-deterministic considerations (stochastic/fuzzy/uncertain) are reviewed. The references analysed are presented in Table 5, indicating the problem considered, and the solution procedure(s) applied. The testbeds used have not been included since, as the problems are very specific, in each contribution the authors have generated their own instance set.

To the best of our knowledge, the first paper addressing a non-deterministic version of a DPFS problem is Wang et al. (2016), who address the $PFm|prmu, machine-breakdown|\widetilde{C}_{\max}$ problem, where $\widetilde{C}_{\max}$ is the fuzzy makespan, using a fuzzy variant of the EDA

Table 5
Non-deterministic DPFS problems.

Reference	Problem	Exact	Approximate
Wang et al. (2016)	$PFm\|prmu, machine-breakdown\|\widetilde{C}_{\max}$		EDA
Hatami et al. (2018)	$PFm\|prmu, \overline{d}\|E[C_{\max}]$		Sim-heuristic
Fu et al. (2019)	$PFm\|prmu, v_i, P[\sum Tj \leq T_e] \geq \alpha\|\#(\max E[C_j], TEC)$	MIP	BSO
Shao et al. (2020b)	$PFm\|prmu, blocking\|\widetilde{C}_{\max}$		CH,IG
Jing et al. (2021a)	$PFm\|prmu\|F_l(E[C^{\mu}_{\max}(\Psi)], Std[C^{\mu}_{\max}(\Psi)])$		IG, ILS
Jing et al. (2021b)	$PFm\|prmu, s_{ij}\|Fl(E[C_{\max}], \sum w_h\|C^h_{\max} - E[C_{\max}]\|)$		IG

method, and comparing it with methods from the deterministic PFS problem for different objectives.

Hatami et al. (2018) consider stochastic processing times, and study the $PFm|prmu, \overline{d}|E[C_{\max}]$ problem, with $\overline{d}$ meaning job deadlines. Three additional variants of the problem are considered: the first one relaxing the deadline; the second one with an objective named percentile makespan, where the products are finished on time with a given probability; and finally, the third one is the deterministic case, using the average processing times. They propose a sim-heuristic based on the ILS metaheuristic, tested in some of the Taillard's instances (specifically 27 of them), extended to their specific problem.

Fu et al. (2019) study the $PFm|prmu, v_i, P[\sum T_j \leq T_e] \geq \alpha|\#(\max E[C_j], TEC)$ problem where the speed selected for a given job is the same for all machines, $P[\sum Tj \leq T_e] \geq \alpha$ indicates that the total tardiness should not exceed a given expected total tardiness T_e with a given probability α, and TEC is a function of the processing and idle times. A BSO-based metaheuristic is proposed and compared with NSGA-II and a general EA.

Shao et al. (2020b) address the $PFm|prmu, blocking|\widetilde{C}_{\max}$ problem with fuzzy processing times (three possible values between the best/most likely/worst scenarios) and fuzzy makespan. They propose some constructive heuristics based on the NEH algorithm, and two variants of the IG algorithm. They adapt testbed TB1 by Naderi & Ruiz (2010) to their problem and compare the proposed constructive heuristics with the adaptation to their problem of the CH proposed by Naderi & Ruiz (2010) and Ruiz et al. (2019). Additionally, they compare their IG algorithms with metaheuristics from the DPFS literature, adapting 10 methods from the $PFm|prmu|C_{\max}$ problem (those by Fernandez-Viagas & Framinan, 2015; Lin et al., 2013; Naderi & Ruiz, 2014; Ruiz et al., 2019), the $PFm|prmu, blocking|C_{\max}$ problem (those by Ribas et al., 2017; Shao et al., 2020a; Ying & Lin, 2017; Zhang et al., 2018), the $PFm|prmu, no\text{-}idle|C_{\max}$ problem (Ying et al., 2017) and the $PFm|prmu, no\text{-}wait|C_{\max}$ problem (Lin & Zhang, 2016). An extensive computational analysis is carried out to compare the methods to prove the effectiveness of their approach.

Jing et al. (2021a) address a DPFS problem with uncertain processing times, generating a given number of scenarios μ randomly selected from a set of deterministic scenarios Ψ controlled by two parameters determining the upper and lower bounds of the interval for the generation of the processing times. In their approach, the robustness of the makespan is measured using a linear combination of the mean and the standard deviation of the makespan of each scenario. Therefore, the problem can be expressed as $PFm|prmu|F_l(E[C^{\mu}_{\max}(\Psi)], Std[C^{\mu}_{\max}(\Psi)])$. IG- and ILS-based metaheuristics are proposed. An extensive set of instances are generated, i.e., 100 scenarios of processing times with 900 instances per scenario. The methods are compared with the metaheuristics by Fernandez-Viagas & Framinan (2015); Lin et al. (2013); Naderi & Ruiz (2014) for the $PFm|prmu|C_{\max}$ problem, and by Fernandez-Viagas et al. (2018b); Pan et al. (2019a) for the $PFm|prmu|\sum C_j$ problem.

Jing et al. (2021b) propose an uncertain approach similar to that by Shao et al. (2020b) with three possible values for the processing times for the $PFm|prmu, s_{ij}|Fl(E[C_{\max}], \sum w_h|C^h_{\max} - E[C_{\max}]|)$ problem, where $C^h_{\max}$ is the makespan obtained for scenario $h \in$\{best/most-likely/worst processing times\}. Similar to Meng & Pan (2021), a testbed based on a printed circuit board manufacturing process is proposed. In the testbed used in the computational experimentation, the setup times are only machine-dependent. An IG-based metaheuristic is compared with the same six methods as in Jing et al. (2021a) for most general problems.

9. Heterogeneous DPFS problems

In this section, heterogeneous (i.e., non-identical factories) DPFS problems are reviewed. In Section 1, the notation is introduced to denote the variety of problems with this consideration. Only five references have been identified (see Table 6). To the best of our knowledge, Wang et al. (2017) is the first paper addressing a heterogeneous DPFS problem. The authors analyse a problem –based on a real-life scenario–with two different layouts in two factories, which can be denoted as $RF, F2, 1|prmu, r_j|\#(\sum F_j, \sum U_j)$. In this problem, the jobs do not have due dates but maximum waiting times $w^{\max}_j$ and, for each job j assigned to the flowshop (note that the other factory consists of a single machine), if the waiting time between the two machines is greater than $w^{\max}_j$, the job is labelled as tardy, and additional processing time on the second machine applies. The authors provide a MIP model of the problem and a NSGA-II -based EA. A set of 100 instances is generated to compare their proposal with variants of the NSGA-II.

Transportation constraints appear in the paper by Li et al. (2021a), where the $RF|prmu, tbp, no\text{-}wait|C_{\max}$ problem is studied. The authors propose a MILP model based on the one by Naderi & Ruiz (2010) and ABC- based metaheuristics, including different local search methods, one of them based on the VNS algorithm. They compare the proposed methods with the metaheuristics by Komaki & Malakooti (2017); Lin & Ying (2016) for the $PFm|prmu, no\text{-}wait|C_{\max}$ problem and Bargaoui et al. (2017) for the $PFm|prmu|C_{\max}$ problem.

Lu et al. (2021) present a new objective related to the so-called Negative-Social-Impact, which is modelled as a cost associated to the processing times. This $RFm|prmu|\#(C_{\max}, TEC, NSI)$ problem is modelled using a MILP. Note that, although TEC are included in their paper, the velocity of the machines is not, so the energy consumption depends on the state of the machine (idle/running), and on the factory where the machine belongs. A MA-based metaheuristic is proposed and compared with five general metaheuristics for multi-objective problems.

Fathollahi-Fard et al. (2021) address a DPFS problem including a decision about the technology to be used for each machine in each factory. The resulting problem can be modelled as $RFm|prmu|\#(C_{\max}, TEC, SC)$, where SC are the social costs. Their solution method is based on the called Social Engineering Optimizer (SEO), which is compared with 6 other general methods from the multi-objective literature.

Meng & Pan (2021) study the $RFm|prmu, job-family, s_{ijk}, |C_{\max}$ problem where each job is formed by sub-lots of equal size, and once a sublot is completed on a machine, it can be immediately

Table 6
Heterogeneous DPFS problems.

Reference	Problem	Exact	Approximate
Wang et al. (2017)	$RF, F2, 1\|prmu, r_j\|\#(\sum F_j, \sum U_j)$	MIP	EA
Li et al. (2021a)	$RF\|prmu, tbp, no\text{-}wait\|C_{\max}$	MILP	ABC
Wang et al. (2021)	$RF\|prmu, s_{ij}, v_i\|\#(C_{\max}, TEC)$	MIP	EA
Lu et al. (2021)	$RFm\|prmu\|\#(C_{\max}, TEC, NSI)$	MILP	MA
Meng & Pan (2021)	$RFm\|prmu, job-family, s_{ijk}\|C_{\max}$	MILP	CH,ABC
Schulz et al. (2022)	$RFm\|prmu, tap\|\#(C_{\max}, TEC)$	MILP	IGA

moved to the next machine. The processing times of each sublot depend on the job, the machine and the factory. The setup time is machine- and sequence- dependent, being applied to the first sublot of a scheduled job. The problem is similar to that proposed by Pan et al. (2021) in the homogeneous version where families are composed of jobs, instead of jobs formed by sublots (implying the $job-family$ constraint). Similarly, the three decisions described previously should be made, namely the allocation of jobs to factories, the sequencing of jobs in each factory, and the sequencing of the sublots within the jobs. The authors provide a MILP model, some constructive heuristics based on the NEH algorithm, and an ABC based metaheuristic. In their experimental results, the authors use a testbed based on a printed circuit board manufacturing process. The constructive heuristic is compared to two NEH versions proposed by Cai et al. (2018) for the $PFm|prmu, M_j, tbp|\#(C_{\max}, maxL_j, TC)$ problem. The metaheuristic is compared with the proposals by Bargaoui et al. (2017); Naderi & Ruiz (2014) for the $PFm|prmu|C_{\max}$ problem, Meng et al. (2019) for the distributed assembly flowshop problem, and a method from another related problem.

Wang et al. (2021) address the same welding manufacturing problem as in Wang et al. (2022a) (described previously in Section 7), but in this case the authors consider a heterogeneous distributed problem that can be denoted as $RF|prmu, s_{ij}, v_i|\#(C_{\max}, TEC)$. A multi-objective EA based metaheuristic is proposed, which is compared to five methods: two of them general multi-objective methods, the proposal by Wang & Wang (2020) for the similar $PFm|prmu, v_i|\#(C_{\max}, TEC)$ problem, and the rest from the energy-efficiency PFS problem literature.

Finally, Schulz et al. (2022) present a MILP model, and an IGA for the $RFm|prmu, v_i, tap|\#(C_{\max}, TEC)$ problem. They compare their proposal with the NSGA-II and perform a sensitivity analysis if the factories are identical.

10. Conclusions

Distributed permutation flowshop scheduling problems constitute a thriving area in scheduling research, and the rapid growth of the distributed manufacturing paradigm indicates that it will continue to be so. In this regard, the review reveals that there are many available tools to address a wide range of distributed scheduling NP-hard decision problems. Nevertheless, a number of issues worth of future research have been identified regarding three aspects: 1) The scope of the problems addressed, 2) The solution procedures proposed for the problems, and 3) The predominant research approach in this area.

Regarding the scope of the problems we have the following comments:

- Despite the copious literature on the topic, due-date related objectives have been scarcely considered, particularly for the classic DPFS problem, where objectives such as maximum lateness, total lateness, maximum tardiness, or number of tardy jobs have not been addressed so far. Additionally, the weighted cases have not been considered either. Given the importance of due-date fulfilment in many real-life environments, strengthening the research on due-date related objectives may enhance the applicability of this area.
- Factory-related objectives –such as maximum among the total completion times of each factory ($\max_l \sum_{j\in\mathcal{F}_l} C_j$), or average makespan of each factory ($\sum_l C^l_{\max}$)– have been scarcely addressed. Such objectives may make sense in a distributed environment, as quite often enterprises seek to balance the workload among factories. However, up to now, most objectives refer to classic, non-distributed, scheduling.
- The heterogeneous DPFS problem has only received a small fraction of the attention devoted to the homogeneous case. Whereas this focus on modelling a more stylised problem is understandable for a relatively new topic, it seems unrealistic that all factories in an enterprise possess exactly the same characteristics.
- It seems quite sensible that, in a distributed environment, transportation issues play an important role. However, only a limited number of references address transportation issues.
- In a distributed setting, it seems plausible that there are different sets of jobs, each one with its own (possibly conflicting) objectives. This may speak for the interest of investigating the suitability of multi-agent scheduling approaches.
- Stochastic/dynamic considerations as new job arrivals or machine failure have been scarcely addressed.

We believe that including some/all of the aforementioned aspects will produce models closer to real-world distributed scheduling, thus bridging the gap between theory and practice.

Regarding solution procedures, the vast majority of the contributions do not address exact solutions for the problems studied. The predominant approach is to propose a MILP (or MIP) model but, in some cases, several MILP models have been proposed independently and not been compared among them. In NP-hard problems such as the ones at hand, the role of exact solution procedures is confined usually to test the performance of approximate methods on small-size instances. However, some contributions simply formulate the models and do not report their performance in small instances, thus limiting the added-value of these models. With respect to approximate solution procedures, it is to note that many sophisticated metaheuristics have been proposed, in some cases without previously testing simpler methods which may have provided competitive results. Additionally, despite (or maybe because of) the high number of methods proposed, the state-of-the-art on the most efficient procedures for some problems is inconclusive. Perhaps due to the fast growth of the field, there is no clear conclusion on the most efficient solution procedures for many DPFS problems, including the most paradigmatic ones, due either to the omission of existing solution procedures in the comparison, or to the fact that these methods are not re-coded and run in the same computer. Moreover, some results have been obtained in specific, non-comparable testbeds, which further complicates determining the best available procedure. Some of these gaps appear in the classic $PFm|prmu|C_{\max}$ problem, where methods by Ren et al. (2021); Ruiz et al. (2019) and Bargaoui et al. (2017) have not been compared among them in the same condi-

Table 7
Suggestions for future research lines. [1], see the notation by Perez-Gonzalez & Framinan (2014).

Basic problems not-studied so far	Layout	Heterogeneous problems	$RFm\vert prmu\vert C_{\max}$
			$RFm\vert prmu\vert \sum C_j$
			$RF\vert prmu\vert C_{\max}$
			$RF\vert prmu\vert \sum C_j$
	Objective	Due dates related	$PFm\vert prmu\vert \max T_j$
			$PFm\vert prmu\vert \max E_j$
			$PFm\vert prmu\vert \sum E_j$
			$PFm\vert prmu\vert \sum U_j$
		Factory-related	$PFm\vert prmu\vert \max_l \sum_{j\in\mathcal{F}_l} C_j$
			$PFm\vert prmu\vert \sum_l C^l_{\max}$
		Multi-objective	$PFm\vert prmu\vert \#(C_{\max}, \sum C_j)$
			$PFm\vert prmu\vert F_l(C_{\max}, \sum C_j)$
		Multi-agent related	$PFm\vert prmu\vert \#(C^A_{\max}, C^B_{\max})^1$
		Stochastic	$PFm\vert prmu\vert E[C_{\max}]$
	Constraint	Transportation constraints	$PFm\vert prmu, tbp\vert C_{\max}$
			$PFm\vert prmu, tdp\vert C_{\max}$
Unclear state-of-the-art			$PFm\vert prmu\vert C_{\max}$
			$PFm\vert prmu\vert \sum C_j$
			$PFm\vert prmu, blocking\vert C_{\max}$
Recommended approaches	Analysis/properties/understanding/theoretical results		
	MILP/MIP models comparison/experimentation/analysis of solvers limit		
	Simpler approximate methods to analyse the effect of most sophisticated methods		

tions. For the $PFm|prmu|\sum C_j$ problem, the methods by Ali et al. (2021); Fernandez-Viagas et al. (2018b) and Pan et al. (2019a) have not been compared among them in the same conditions. This also happens with the methods by Chen et al. (2021a); Karabulut et al. (2022a); Shao et al. (2020a) and Zhao et al. (2020) for the $PFm|prmu, blocking|C_{\max}$ problem. Finally, it is quite surprising that no paper has so far addressed matheuristics methods, thus hybridizing exact and approximate procedures.

A final comment refers to the predominant approach to address DPFS problems, which up to now is bottom-up rather than top-down, i.e., focusing on *solving* the problem rather than in *understanding* it, a fact that is exemplified by the acute scarcity of papers studying theoretical properties of the problems addressed. Although this may respond to the need of providing a solution for a specific decision problem in a short time (which in turn might indicate the high applicability of the research), such approach also involves some risks:

- The literature on related problems is often overlooked, and in some cases the selection of related problems is not sufficiently justified, thus dismissing potentially efficient methods for the problem at hand. Furthermore, this may skip theoretical properties from these related problems that could be successfully adapted to the problem under study.
- In some cases, perhaps due to this lack of theoretical guidance, the contributions are primarily of experimental nature, i.e., an approximate procedure (a metaheuristic) is proposed and, via a computational experience that sometimes resembles a trial-and-error procedure, the different variants of the metaheuristic and their performance are reported. Such results are of undoubted value if they help to explain why some elements of the procedure make it work for this specific problem, but this is not always the case.

Table 7 provides a summary of future research lines extracted from the conclusions above presented, including basic problems not studied so far; problems with unclear state-of-the-art; and less-applied approaches.

Aside to providing a clear picture of the state-of-the-art in DPFS research, the review conducted in this paper is aimed at presenting a unified view of the topic (including a common notation and classification) that may help to reconcile this bottom-up approach with the top-down one that is prevalent in other scheduling fields and that has proved to greatly help in boosting the research. Along this line, it would be interesting to enhance the literature review to encompass other related problems, such as the DAFS and DHFS, as in Section 3 it is shown that these can be integrated in the proposed classification.

Acknowledgements

The authors wish to thank the referees for the comments and suggestions provided during the reviewing process. This study was financed in part by the Spanish Ministry of Science and Education (ASSORT project, Grant no. PID2019-108756RB-I00), and the Andalusian Government (project DEMAND Grant no. P18-FR-1149).

References

Alaghebandha, M., Naderi, B., & Mohammadi, M. (2019). Economic lot sizing and scheduling in distributed permutation flow shops. *Journal of Optimization in Industrial Engineering, 12*(1), 103–117. https://doi.org/10.22094/JOIE.2018.542997.1510.

Ali, A., Gajpal, Y., & Elmekkawy, T. Y. (2021). Distributed permutation flowshop scheduling problem with total completion time objective. *OPSEARCH, 58*(2), 425–447. https://doi.org/10.1007/s12597-020-00484-3.

Allali, K., Aqil, S., & Belabid, J. (2022). Distributed no-wait flow shop problem with sequence dependent setup time: Optimization of makespan and maximum tardiness. *Simulation Modelling Practice and Theory, 116*, 102455. https://doi.org/10.1016/j.simpat.2021.102455.

Bargaoui, H., Belkahla Driss, O., & Ghédira, K. (2017). A novel chemical reaction optimization for the distributed permutation flowshop scheduling problem with makespan criterion. *Computers and Industrial Engineering, 111*, 239–250. https://doi.org/10.1016/j.cie.2017.07.020.

Cai, S., Yang, K., & Liu, K. (2018). Multi-objective optimization of the distributed permutation flow shop scheduling problem with transportation and eligibility constraints. *Journal of the Operations Research Society of China, 6*(3), 391–416. https://doi.org/10.1007/s40305-017-0165-3.

Cao, D., Chen, M., et al., (2003). Parallel flowshop scheduling using Tabu search. *International Journal of Production Research, 41*(13), 3059–3073. https://doi.org/10.1080/0020754031000106443.

Çolak, M., & Keskin, G. A. (2022). An extensive and systematic literature review for hybrid flowshop scheduling problems. *International Journal of Industrial Engineering Computations, 13*(2), 185–222. https://doi.org/10.5267/J.IJIEC.2021.12.001.

Chen, J.-F., Wang, L., & Peng, Z.-P. (2019). A collaborative optimization algorithm for energy-efficient multi-objective distributed no-idle flow-shop scheduling. *Swarm and Evolutionary Computation, 50*, 100557. https://doi.org/10.1016/j.swevo.2019.100557.

Chen, S., Pan, Q.-K., & Gao, L. (2021a). Production scheduling for blocking flowshop in distributed environment using effective heuristics and iterated Greedy algorithm. *Robotics and Computer-Integrated Manufacturing, 71*, 102155. https://doi.org/10.1016/j.rcim.2021.102155.

Chen, S., Pan, Q.-K., Gao, L., & Sang, H.-y. (2021b). A population-based iterated Greedy algorithm to minimize total flowtime for the distributed blocking flowshop scheduling problem. *Engineering Applications of Artificial Intelligence, 104*, 104375. https://doi.org/10.1016/j.engappai.2021.104375.

Cheng, C.-Y., Ying, K.-C., Chen, H.-H., Lu, H.-S., et al., (2019). Minimising makespan in distributed mixed no-idle flowshops. *International Journal of Production Research, 57*(1), 48–60. https://doi.org/10.1080/00207543.2018.1457812.

Companys, R., & Mateo, M. (2007). Different behaviour of a double branch-and-bound algorithm on Fm|prmu|$C_{\max}$ and Fm|block|$C_{\max}$ problems. *Computers and Operations Research, 34*(4), 938–953. https://doi.org/10.1016/J.COR.2005.05.018.

Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., et al., (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation, 6*(2), 182–197. https://doi.org/10.1109/4235.996017.

Deng, J., & Wang, L. (2017). A competitive memetic algorithm for multi-objective distributed permutation flow shop scheduling problem. *Swarm and Evolutionary Computation, 32*, 121–131. https://doi.org/10.1016/j.swevo.2016.06.002.

Ding, J. Y., Song, S., & Wu, C. (2016). Carbon-efficient scheduling of flow shops by multi-objective optimization. *European Journal of Operational Research, 248*(3), 758–771. https://doi.org/10.1016/J.EJOR.2015.05.019.

Dong, J., Tong, W., Luo, T., Wang, X., Hu, J., Xu, Y., Lin, G., et al., (2017). An FPTAS for the parallel two-stage flowshop problem. *Theoretical Computer Science, 657*, 64–72. https://doi.org/10.1016/J.TCS.2016.04.046.

Fathollahi-Fard, A. M., Woodward, L., & Akhrif, O. (2021). Sustainable distributed permutation flow-shop scheduling model based on a triple bottom line concept. *Journal of Industrial Information Integration, 24*, 100233. https://doi.org/10.1016/j.jii.2021.100233.

Fernandez-Viagas, V., & Framinan, J. M. (2015). A bounded-search iterated Greedy algorithm for the distributed permutation flowshop scheduling problem. *International Journal of Production Research, 53*(4), 1111–1123. https://doi.org/10.1080/00207543.2014.948578.

Fernandez-Viagas, V., Molina-Pariente, J. M., & Framinan, J. M. (2018a). New efficient constructive heuristics for the hybrid flowshop to minimise makespan: A computational evaluation of heuristics. *Expert Systems with Applications, 114*, 345–356. https://doi.org/10.1016/j.eswa.2018.07.055.

Fernandez-Viagas, V., Molina-Pariente, J. M., & Framinan, J. M. (2020). Generalised accelerations for insertion-based heuristics in permutation flowshop scheduling. *European Journal of Operational Research, 282*(3), 858–872. https://doi.org/10.1016/j.ejor.2019.10.017.

Fernandez-Viagas, V., Perez-Gonzalez, P., & Framinan, J. M. (2018b). The distributed permutation flow shop to minimise the total flowtime. *Computers and Industrial Engineering, 118*, 464–477. https://doi.org/10.1016/j.cie.2018.03.014.

Fernandez-Viagas, V., Perez-Gonzalez, P., & Framinan, J. M. (2019). Efficiency of the solution representations for the hybrid flow shop scheduling problem with makespan objective. *Computers & Operations Research, 109*, 77–88. https://doi.org/10.1016/j.cor.2019.05.002.

Fernandez-Viagas, V., Ruiz, R., & Framinan, J. M. (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. https://doi.org/10.1016/J.EJOR.2016.09.055.

Framinan, J. M., Gupta, J. N. D., & 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. https://doi.org/10.1057/palgrave.jors.2601784.

Framinan, J. M., Leisten, R., & Ruiz, R. (2014). *Manufacturing scheduling systems: An integrated view on models, methods and tools*: vol. 9781447162. Springer-Verlag London Ltd. https://doi.org/10.1007/978-1-4471-6272-8.

Framinan, J. M., Perez-Gonzalez, P., & Fernandez-Viagas, V. (2018). Deterministic assembly scheduling problems: A review and classification of concurrent-type scheduling models and solution procedures. *European Journal of Operational Research, 273*(2), 401–417. https://doi.org/10.1016/J.EJOR.2018.04.033.

Fu, Y., Hou, Y., Chen, Z., Pu, X., Gao, K., & Sadollah, A. (2022). Modelling and scheduling integration of distributed production and distribution problems via black widow optimization. *Swarm and Evolutionary Computation, 68*, 101015. https://doi.org/10.1016/j.swevo.2021.101015.

Fu, Y., Tian, G., Fathollahi-Fard, A. M., Ahmadi, A., & Zhang, C. (2019). Stochastic multi-objective modelling and optimization of an energy-conscious distributed permutation flow shop scheduling problem with the total tardiness constraint. *Journal of Cleaner Production, 226*, 515–525. https://doi.org/10.1016/j.jclepro.2019.04.046.

Gao, J., & Chen, R. (2011a). A hybrid genetic algorithm for the distributed permutation flowshop scheduling problem. *International Journal of Computational Intelligence Systems, 4*(4), 497–508. https://doi.org/10.1080/18756891.2011.9727808.

Gao, J., & Chen, R. (2011b). An NEH-based heuristic algorithm for distributed permutation flowshop scheduling problems. *Scientific Research and Essays, 6*(14), 3094–3100.

Gao, J., Chen, R., & Deng, W. (2013). An efficient tabu search algorithm for the distributed permutation flowshop scheduling problem. *International Journal of Production Research, 51*(3), 641–651. https://doi.org/10.1080/00207543.2011.644819.

Gao, J., Chen, R., Deng, W., & Liu, Y. (2012). Solving multi-factory flowshop problems with a novel variable neighbourhood descent algorithm. *Journal of Computational Information Systems, 8*(5), 2025–2032.

Gooding, W. B., Pekny, J. F., & McCroskey, P. S. (1994). Enumerative approaches to parallel flowshop scheduling via problem transformation. *Computers and Chemical Engineering, 18*(10), 909–927. https://doi.org/10.1016/0098-1354(94)E0029-M.

Graham, R. L., Lawler, E. L., Lenstra, J. K., Kan, A. H. G. R., et al., (1979). Optimization and heuristic in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics, 5*, 287–326.

Guo, H.-w., Sang, H.-y., Zhang, B., Meng, L.-l., & Liu, L.-l. (2022). An effective metaheuristic with a differential flight strategy for the distributed permutation flowshop scheduling problem with sequence-dependent setup times. *Knowledge-Based Systems, 242*, 108328. https://doi.org/10.1016/j.knosys.2022.108328.

Hamzadayı, A. (2020). An effective benders decomposition algorithm for solving the distributed permutation flowshop scheduling problem. *Computers and Operations Research, 123*, 105006. https://doi.org/10.1016/j.cor.2020.105006.

Hasija, S., & Rajendran, C. (2004). Scheduling in flowshops to minimize total tardiness of jobs. *International Journal of Production Research, 42*(11), 2289–2301. https://doi.org/10.1080/00207540310001657595.

Hatami, S., Calvet, L., Fernández-Viagas, V., Framiñán, J. M., Juan, A. A., et al., (2018). A simheuristic algorithm to set up starting times in the stochastic parallel flowshop problem. *Simulation Modelling Practice and Theory, 86*, 55–71. https://doi.org/10.1016/J.SIMPAT.2018.04.005.

Hatami, S., Ruiz, R., & Andrés-Romano, C. (2013). The distributed assembly permutation flowshop scheduling problem. *International Journal of Production Research, 51*(17), 5292–5308. https://doi.org/10.1080/00207543.2013.807955.

He, D. W., Kusiak, A., & Artiba, A. (1996). A scheduling problem in glass manufacturing. *IIE Transactions, 28*(2), 129–139. https://doi.org/10.1080/07408179608966258.

Hejazi, S. R., Saghafian, S., & Saghafianz, S. (2005). Flowshop-scheduling problems with makespan criterion: A review. *International Journal of Production Research, 43*(14), 2895–2929. https://doi.org/10.1080/0020754050056417.

Hou, Y., Fu, Y., Gao, K., Zhang, H., & Sadollah, A. (2022). Modelling and optimization of integrated distributed flow shop scheduling and distribution problems with time windows. *Expert Systems with Applications, 187*, 115827. https://doi.org/10.1016/j.eswa.2021.115827.

Huang, J.-P., Pan, Q.-K., & Gao, L. (2020). An effective iterated Greedy method for the distributed permutation flowshop scheduling problem with sequence-dependent setup times. *Swarm and Evolutionary Computation, 59*, 100742. https://doi.org/10.1016/j.swevo.2020.100742.

Huang, J.-P., Pan, Q.-K., Miao, Z.-H., & Gao, L. (2021a). Effective constructive heuristics and discrete bee colony optimization for distributed flowshop with setup times. *Engineering Applications of Artificial Intelligence, 97*, 104016. https://doi.org/10.1016/j.engappai.2020.104016.

Huang, Y.-Y., Pan, Q.-K., Huang, J.-P., Suganthan, P. N., & Gao, L. (2021b). An improved iterated Greedy algorithm for the distributed assembly permutation flowshop scheduling problem. *Computers and Industrial Engineering, 152*, 107021. https://doi.org/10.1016/j.cie.2020.107021.

Jing, X.-L., Pan, Q.-K., & Gao, L. (2021a). Local search-based metaheuristics for the robust distributed permutation flowshop problem. *Applied Soft Computing, 105*, 107247. https://doi.org/10.1016/j.asoc.2021.107247.

Jing, X.-L., Pan, Q.-K., Gao, L., & Wang, L. (2021b). An effective iterated Greedy algorithm for a robust distributed permutation flowshop problem with carryover sequence-dependent setup time. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 1–12. https://doi.org/10.1109/TSMC.2021.3131849.

Jing, X.-L., Pan, Q.-K., Gao, L., & Wang, Y.-L. (2020). An effective iterated Greedy algorithm for the distributed permutation flowshop scheduling with due windows. *Applied Soft Computing, 96*, 106629. https://doi.org/10.1016/j.asoc.2020.106629.

Kai Chan, H., & Ho Chung, S. (2013). Optimisation approaches for distributed scheduling problems. *International Journal of Production Research, 51*(9), 2571–2577. https://doi.org/10.1080/00207543.2012.755345.

Karabulut, K., Kizilay, D., Tasgetiren, M. F., Gao, L., & Kandiller, L. (2022a). An evolution strategy approach for the distributed blocking flowshop scheduling problem. *Computers and Industrial Engineering, 163*, 107832. https://doi.org/10.1016/j.cie.2021.107832.

Karabulut, K., Öztop, H., Kizilay, D., Tasgetiren, M. F., & Kandiller, L. (2022b). An evolution strategy approach for the distributed permutation flowshop scheduling problem with sequence-dependent setup times. *Computers and Operations Research, 142*, 105733. https://doi.org/10.1016/j.cor.2022.105733.

Khare, A., & Agrawal, S. (2021). Effective heuristics and metaheuristics to minimise total tardiness for the distributed permutation flowshop scheduling problem. *International Journal of Production Research, 59*(23), 7266–7282. https://doi.org/10.1080/00207543.2020.1837982.

Komaki, M., & Malakooti, B. (2017). General variable neighborhood search algorithm to minimize makespan of the distributed no-wait flow shop scheduling problem. *Production Engineering, 11*(3), 315–329. https://doi.org/10.1007/s11740-017-0716-9.

Li, H., Li, X., & Gao, L. (2021a). A discrete artificial bee colony algorithm for the distributed heterogeneous no-wait flowshop scheduling problem. *Applied Soft Computing, 100*, 106946. https://doi.org/10.1016/j.asoc.2020.106946.

Li, J.-q., Bai, S.-C., Duan, P.-y., Sang, H.-y., Han, Y.-y., & Zheng, Z.-x. (2019a). An improved artificial bee colony algorithm for addressing distributed flow shop with distance coefficient in a prefabricated system. *International Journal of Production Research, 57*(22), 6922–6942. https://doi.org/10.1080/00207543.2019.1571687.

Li, Q., Li, J., Zhang, X., & Zhang, B. (2021b). A wale optimization algorithm for distributed flow shop with batch delivery. *Soft Computing, 25*(21), 13181–13194. https://doi.org/10.1007/s00500-021-06099-0.

Li, W., Li, J., Gao, K., Han, Y., Niu, B., Liu, Z., & Sun, Q. (2019b). Solving robotic distributed flowshop problem using an improved iterated Greedy algorithm. *International Journal of Advanced Robotic Systems, 16*(5). https://doi.org/10.1177/1729881419879819.

Li, Y.-Z., Pan, Q.-K., Gao, K.-Z., Tasgetiren, M. F., Zhang, B., & Li, J.-Q. (2021c). A green scheduling algorithm for the distributed flowshop problem. *Applied Soft Computing, 109*, 107526. https://doi.org/10.1016/j.asoc.2021.107526.

Li, Y.-Z., Pan, Q.-K., He, X., Sang, H.-Y., Gao, K.-Z., & Jing, X.-L. (2022). The distributed flowshop scheduling problem with delivery dates and cumulative payoffs. *Computers and Industrial Engineering, 165*, 107961. https://doi.org/10.1016/j.cie.2022.107961.

Li, Y.-Z., Pan, Q.-K., Li, J.-Q., Gao, L., & Tasgetiren, M. F. (2021d). An adaptive iterated Greedy algorithm for distributed mixed no-idle permutation flowshop scheduling problems. *Swarm and Evolutionary Computation, 63*, 100874. https://doi.org/10.1016/j.swevo.2021.100874.

Lin, J., Wang, Z.-J., & Li, X. (2017). A backtracking search hyper-heuristic for the distributed assembly flow-shop scheduling problem. *Swarm and Evolutionary Computation, 36*, 124–135. https://doi.org/10.1016/j.swevo.2017.04.007.

Lin, J., & Zhang, S. (2016). An effective hybrid biogeography-based optimization algorithm for the distributed assembly permutation flow-shop scheduling problem. *Computers and Industrial Engineering, 97*, 128–136. https://doi.org/10.1016/j.cie.2016.05.005.

Lin, S.-W., & Ying, K.-C. (2016). Minimizing makespan for solving the distributed no-wait flowshop scheduling problem. *Computers and Industrial Engineering, 99*, 202–209. https://doi.org/10.1016/j.cie.2016.07.027.

Lin, S.-W., Ying, K.-C., & Huang, C.-Y. (2013). Minimising makespan in distributed permutation flowshops using a modified iterated Greedy algorithm. *International Journal of Production Research, 51*(16), 5029–5038. https://doi.org/10.1080/00207543.2013.790571.

Lu, C., Gao, L., Gong, W., Hu, C., Yan, X., & Li, X. (2021). Sustainable scheduling of distributed permutation flow-shop with non-identical factory using a knowledge-based multi-objective memetic optimization algorithm. *Swarm and Evolutionary Computation, 60*, 100803. https://doi.org/10.1016/j.swevo.2020.100803.

Lu, C., Huang, Y., Meng, L., Gao, L., Zhang, B., & Zhou, J. (2022). A Pareto-based collaborative multi-objective optimization algorithm for energy-efficient scheduling of distributed permutation flow-shop with limited buffers. *Robotics and Computer-Integrated Manufacturing, 74*, 102277. https://doi.org/10.1016/j.rcim.2021.102277.

Mao, J.-y., Pan, Q.-k., Miao, Z.-h., & Gao, L. (2021). An effective multi-start iterated Greedy algorithm to minimize makespan for the distributed permutation flowshop scheduling problem with preventive maintenance. *Expert Systems with Applications, 169*, 114495. https://doi.org/10.1016/j.eswa.2020.114495.

Mao, J.-Y., Pan, Q.-K., Miao, Z.-H., Gao, L., & Chen, S. (2022). A hash map-based memetic algorithm for the distributed permutation flowshop scheduling problem with preventive maintenance to minimize total flowtime. *Knowledge-Based Systems, 242*, 108413. https://doi.org/10.1016/j.knosys.2022.108413.

Meng, T., & Pan, Q.-K. (2021). A distributed heterogeneous permutation flowshop scheduling problem with lot-streaming and carryover sequence-dependent setup time. *Swarm and Evolutionary Computation, 60*, 100804. https://doi.org/10.1016/j.swevo.2020.100804.

Meng, T., Pan, Q.-K., & Wang, L. (2019). A distributed permutation flowshop scheduling problem with the customer order constraint. *Knowledge-Based Systems, 184*, 104894. https://doi.org/10.1016/j.knosys.2019.104894.

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. https://doi.org/10.1287/ijoc.1070.0258.

Miyata, H. H., & Nagano, M. S. (2021). Optimizing distributed no-wait flow shop scheduling problem with setup times and maintenance operations via iterated Greedy algorithm. *Journal of Manufacturing Systems, 61*, 592–612. https://doi.org/10.1016/j.jmsy.2021.10.005.

Musier, R. F. H., & Evans, L. B. (1989). An approximate method for the production scheduling of industrial batch processes with parallel units. *Computers and Chemical Engineering, 13*(1–2), 229–238. https://doi.org/10.1016/0098-1354(89)89020-9.

Naderi, B., Gohari, S., & Yazdani, M. (2014). Hybrid flexible flowshop problems: Models and solution methods. *Applied Mathematical Modelling, 38*(24), 5767–5780. https://doi.org/10.1016/J.APM.2014.04.012.

Naderi, B., & Ruiz, R. (2010). The distributed permutation flowshop scheduling problem. *Computers and Operations Research, 37*(4), 754–768. https://doi.org/10.1016/j.cor.2009.06.019.

Naderi, B., & Ruiz, R. (2014). A scatter search algorithm for the distributed permutation flowshop scheduling problem. *European Journal of Operational Research, 239*(2), 323–334. https://doi.org/10.1016/j.ejor.2014.05.024.

Nawaz, M., Enscore, E. E., & Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega, 11*(1), 91–95. https://doi.org/10.1016/0305-0483(83)90088-9.

Pan, Q.-K., Gao, L., & Wang, L. (2021). An effective cooperative co-evolutionary algorithm for distributed flowshop group scheduling problems. *IEEE Transactions on Cybernetics*, 1–14. https://doi.org/10.1109/TCYB.2020.3041494.

Pan, Q.-K., Gao, L., Wang, L., Liang, J., & Li, X.-Y. (2019a). Effective heuristics and metaheuristics to minimize total flowtime for the distributed permutation flowshop problem. *Expert Systems with Applications, 124*, 309–324. https://doi.org/10.1016/j.eswa.2019.01.062.

Pan, Q.-K., Gao, L., Xin-Yu, L., & Jose, F. M. (2019b). Effective constructive heuristics and meta-heuristics for the distributed assembly permutation flowshop scheduling problem. *Applied Soft Computing, 81*, 105492. https://doi.org/10.1016/j.asoc.2019.105492.

Pan, Q.-K., & Ruiz, R. (2013). A comprehensive review and evaluation of permutation flowshop heuristics to minimize flowtime. *Computers and Operations Research, 40*(1), 117–128. https://doi.org/10.1016/J.COR.2012.05.018.

Pan, Q. K., & Ruiz, R. (2014). An effective iterated Greedy algorithm for the mixed no-idle permutation flowshop scheduling problem. *Omega, 44*, 41–50. https://doi.org/10.1016/J.OMEGA.2013.10.002.

Pan, Y., Gao, K., Li, Z., & Wu, N. (2022). Improved meta-heuristics for solving distributed lot-streaming permutation flow shop scheduling problems. *IEEE Transactions on Automation Science and Engineering*, 1–11. https://doi.org/10.1109/TASE.2022.3151648.

Perez-Gonzalez, P., Fernandez-Viagas, V., & Framinan, J. M. (2020). Permutation flowshop scheduling with periodic maintenance and makespan objective. *Computers and Industrial Engineering, 143*, 106369. https://doi.org/10.1016/j.cie.2020.106369.

Perez-Gonzalez, P., & Framinan, J. M. (2014). A common framework and taxonomy for multicriteria scheduling problems with interfering and competing jobs: Multi-agent scheduling problems. *European Journal of Operational Research, 235*(1), 1–16. https://doi.org/10.1016/j.ejor.2013.09.017.

Potts, C. N., & Van Wassenhove, L. N. (1982). A decomposition algorithm for the single machine total tardiness problem. *Operations Research Letters, 1*(5), 177–181. https://doi.org/10.1016/0167-6377(82)90035-9.

Ren, J. F., Ye, C. M., & Li, Y. (2021). A new solution to distributed permutation flow shop scheduling problem based on NASH Q-learning. *Advances in Production Engineering andManagement, 16*(3), 269–284. https://doi.org/10.14743/apem2021.3.399.

Riahi, V., Khorramizadeh, M., Hakim Newton, M. A., & Sattar, A. (2017). Scatter search for mixed blocking flowshop scheduling. *Expert Systems with Applications, 79*, 20–32. https://doi.org/10.1016/J.ESWA.2017.02.027.

Ribas, I., Companys, R., & Tort-Martorell, X. (2017). Efficient heuristics for the parallel blocking flow shop scheduling problem. *Expert Systems with Applications, 74*, 41–54. https://doi.org/10.1016/J.ESWA.2017.01.006.

Ribas, I., Companys, R., & Tort-Martorell, X. (2019). An iterated Greedy algorithm for solving the total tardiness parallel blocking flow shop scheduling problem. *Expert Systems with Applications, 121*, 347–361. https://doi.org/10.1016/J.ESWA.2018.12.039.

Ribas, I., Leisten, R., & Framinan, J. M. (2010). Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective. *Computers and Operations Research, 37*(8), 1439–1454. https://doi.org/10.1016/j.cor.2009.11.001.

Rifai, A. P., Mara, S. T. W., & Sudiarso, A. (2021). Multi-objective distributed reentrant permutation flow shop scheduling with sequence-dependent setup time. *Expert Systems with Applications, 183*, 115339. https://doi.org/10.1016/j.eswa.2021.115339.

Rifai, A. P., Nguyen, H.-T., & Dawal, S. Z. M. (2016). Multi-objective adaptive large neighborhood search for distributed reentrant permutation flow shop scheduling. *Applied Soft Computing, 40*, 42–57. https://doi.org/10.1016/j.asoc.2015.11.034.

Rossi, F. L., & Nagano, M. S. (2021). Heuristics and iterated Greedy algorithms for the distributed mixed no-idle flowshop with sequence-dependent setup times. *Computers and Industrial Engineering, 157*, 107337. https://doi.org/10.1016/j.cie.2021.107337.

Rossi, F. L., Nagano, M. S., & Neto, R. F. T. (2016). Evaluation of high performance constructive heuristics for the flow shop with makespan minimization. *International Journal of Advanced Manufacturing Technology*, 1–12. https://doi.org/10.1007/s00170-016-8484-9.

Rossi, F. L., & Seido Nagano, M. (2019). Heuristics for the mixed no-idle flowshop with sequence-dependent setup times. *Journal of the Operational Research Society, 72*(2), 417–443. https://doi.org/10.1080/01605682.2019.1671149.

Ruiz, R., García-Díaz, J. C., & Maroto, C. (2007). Considering scheduling and preventive maintenance in the flowshop sequencing problem. *Computers and Operations Research, 34*(11), 3314–3330. https://doi.org/10.1016/j.cor.2005.12.007.

Ruiz, R., & Maroto, C. (2005). A comprehensive review and evaluation of permutation flowshop heuristics. *European Journal of Operational Research, 165*(2), 479–494. https://doi.org/10.1016/j.ejor.2004.04.017.

Ruiz, R., Pan, Q.-K., & Naderi, B. (2019). Iterated Greedy methods for the distributed permutation flowshop scheduling problem. *Omega, 83*, 213–222. https://doi.org/10.1016/j.omega.2018.03.004.

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. https://doi.org/10.1016/j.ejor.2005.12.009.

Ruiz, R., & Vázquez-Rodríguez, J. A. (2010). The hybrid flow shop scheduling problem. *European Journal of Operational Research, 205*(1), 1–18. https://doi.org/10.1016/J.EJOR.2009.09.024.

Sahinidis, N. V., & Grossmann, I. E. (1991). MINLP model for cyclic multiproduct scheduling on continuous parallel lines. *Computers and Chemical Engineering, 15*(2), 85–103. https://doi.org/10.1016/0098-1354(91)87008-W.

Sang, H.-Y., Pan, Q.-K., Li, J.-Q., Wang, P., Han, Y.-Y., Gao, K.-Z., & Duan, P. (2019). Effective invasive weed optimization algorithms for distributed assembly permutation flowshop problem with total flowtime criterion. *Swarm and Evolutionary Computation, 44*, 64–73. https://doi.org/10.1016/j.swevo.2018.12.001.

Schulz, S., Schönheit, M., & Neufeld, J. S. (2022). Multi-objective carbon-efficient scheduling in distributed permutation flow shops under consideration of transportation efforts. *Journal of Cleaner Production, 365*, 132551. https://doi.org/10.1016/J.JCLEPRO.2022.132551.

Shao, W., Pi, D., & Shao, Z. (2017). Optimization of makespan for the distributed no-wait flow shop scheduling problem with iterated Greedy algorithms. *Knowledge-Based Systems, 137*, 163–181. https://doi.org/10.1016/j.knosys.2017.09.026.

Shao, W., Pi, D., & Shao, Z. (2019). A Pareto-based estimation of distribution algorithm for solving multiobjective distributed no-wait flow-shop scheduling problem with sequence-dependent setup time. *IEEE Transactions on Automation Science and Engineering, 16*(3), 1344–1360. https://doi.org/10.1109/TASE.2018.2886303.

Shao, Z., Pi, D., & Shao, W. (2020a). Hybrid enhanced discrete fruit fly optimization algorithm for scheduling blocking flow-shop in distributed environment. *Expert Systems with Applications, 145*, 113147. https://doi.org/10.1016/j.eswa.2019.113147.

Shao, Z., Shao, W., & Pi, D. (2020b). Effective heuristics and metaheuristics for the distributed fuzzy blocking flow-shop scheduling problem. *Swarm and Evolutionary Computation, 59*, 100747. https://doi.org/10.1016/j.swevo.2020.100747.

Shao, Z., Shao, W., & Pi, D. (2021). Effective constructive heuristic and iterated Greedy algorithm for distributed mixed blocking permutation flow-shop scheduling problem. *Knowledge-Based Systems, 221*, 106959. https://doi.org/10.1016/j.knosys.2021.106959.

Taillard, E. D. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research, 64*, 278–285.

T'kindt, V., & Billaut, J.-C. (2002). *Multicriteria scheduling: Theory, models and algorithms* (2nd ed.). Springer Berlin Heidelberg.

Vairaktarakis, G., Elhafsi, M., et al., (2007). The use of flowlines to simplify routing complexity in two-stage flowshops. *IIE Transactions (Institute of Industrial Engineers), 32*(8), 687–699. https://doi.org/10.1080/07408170008967427.

Vallada, E., Ruiz, R., & Framinan, J. M. (2015). New hard benchmark for flowshop scheduling problems minimising makespan. *European Journal of Operational Research, 240*(3), 666–677. https://doi.org/10.1016/j.ejor.2014.07.033.

Vallada, E., Ruiz, R., & Minella, G. (2008). Minimising total tardiness in the *m*-machine flowshop problem: A review and evaluation of heuristics and metaheuristics. *Computers and Operations Research, 35*(4), 1350–1373. https://doi.org/10.1016/j.cor.2006.08.016.

Wang, G., Gao, L., Li, X., Li, P., & Tasgetiren, M. F. (2020). Energy-efficient distributed permutation flow shop scheduling problem using a multi-objective whale swarm algorithm. *Swarm and Evolutionary Computation, 57*, 100716. https://doi.org/10.1016/j.swevo.2020.100716.

Wang, G., Li, X., Gao, L., & Li, P. (2021). Energy-efficient distributed heterogeneous welding flow shop scheduling problem using a modified MOEA/D. *Swarm and Evolutionary Computation, 62*, 100858. https://doi.org/10.1016/j.swevo.2021.100858.

Wang, G., Li, X., Gao, L., & Li, P. (2022a). An effective multi-objective whale swarm algorithm for energy-efficient scheduling of distributed welding flow shop. *Annals of Operations Research, 310*(1), 223–255. https://doi.org/10.1007/s10479-021-03952-1.

Wang, H., Fu, Y., Huang, M., Huang, G. Q., Wang, J., et al., (2017). A NSGA-II based memetic algorithm for multiobjective parallel flowshop scheduling problem. *Computers and Industrial Engineering, 113*, 185–194. https://doi.org/10.1016/J.CIE.2017.09.009.

Wang, J.-J., & Wang, L. (2020). A knowledge-based cooperative algorithm for energy-efficient scheduling of distributed flow-shop. *IEEE Transactions on Systems, Man, and Cybernetics: Systems, 50*(5), 1805–1819. https://doi.org/10.1109/TSMC.2017.2788879.

Wang, K., Huang, Y., & Qin, H. (2016). A fuzzy logic-based hybrid estimation of distribution algorithm for distributed permutation flowshop scheduling problems under machine breakdown. *Journal of the Operational Research Society, 67*(1), 68–82. https://doi.org/10.1057/jors.2015.50.

Wang, S. Y., Wang, L., Liu, M., Xu, Y., et al., (2013). An effective estimation of distribution algorithm for solving the distributed permutation flow-shop scheduling problem. *International Journal of Production Economics, 145*(1), 387–396. https://doi.org/10.1016/J.IJPE.2013.05.004.

Wang, Z. Y., Pan, Q. K., Gao, L., & Wang, Y. L. (2022b). An effective two-stage iterated Greedy algorithm to minimize total tardiness for the distributed flowshop group scheduling problem. *Swarm and Evolutionary Computation, 74*, 101143. https://doi.org/10.1016/J.SWEVO.2022.101143.

Xiong, F., Chu, M., Li, Z., Du, Y., & Wang, L. (2021). Just-in-time scheduling for a distributed concrete precast flow shop system. *Computers and Operations Research, 129*, 105204. https://doi.org/10.1016/j.cor.2020.105204.

Xu, Y., Wang, L., Wang, S., & Liu, M. (2014). An effective hybrid immune algorithm for solving the distributed permutation flow-shop scheduling problem. *Engineering Optimization, 46*(9), 1269–1283. https://doi.org/10.1080/0305215X.2013.827673.

Ying, K.-C., & Lin, S. W. (2017). Minimizing makespan in distributed blocking flowshops using hybrid iterated Greedy algorithms. *IEEE Access, 5*, 15694–15705. https://doi.org/10.1109/ACCESS.2017.2732738.

Ying, K.-C., & Lin, S.-W. (2018). Minimizing makespan for the distributed hybrid flowshop scheduling problem with multiprocessor tasks. *Expert Systems with Applications, 92*, 132–141. https://doi.org/10.1016/j.eswa.2017.09.032.

Ying, K.-C., Lin, S.-W., Cheng, C.-Y., & He, C.-D. (2017). Iterated reference Greedy algorithm for solving distributed no-idle permutation flowshop scheduling problems. *Computers and Industrial Engineering, 110*, 413–423. https://doi.org/10.1016/j.cie.2017.06.025.

Zhang, G., & Xing, K. (2019). Differential evolution metaheuristics for distributed limited-buffer flowshop scheduling with makespan criterion. *Computers and Operations Research, 108*, 33–43. https://doi.org/10.1016/j.cor.2019.04.002.

Zhang, G., Xing, K., & Cao, F. (2018). Discrete differential evolution algorithm for distributed blocking flowshop scheduling with makespan criterion. *Engineering Applications of Artificial Intelligence, 76*, 96–107. https://doi.org/10.1016/j.engappai.2018.09.005.

Zhang, X., & Van De Velde, S. (2012). Approximation algorithms for the parallel flow shop problem. *European Journal of Operational Research, 216*(3), 544–552. https://doi.org/10.1016/J.EJOR.2011.08.007.

Zhao, F., Ma, R., & Wang, L. (2021). A self-learning discrete jaya algorithm for multiobjective energy-efficient distributed no-idle flow-shop scheduling problem in heterogeneous factory system. *IEEE Transactions on Cybernetics*, 1–12. https://doi.org/10.1109/TCYB.2021.3086181.

Zhao, F., Zhao, L., Wang, L., & Song, H. (2020). An ensemble discrete differential evolution for the distributed blocking flowshop scheduling with minimizing makespan criterion. *Expert Systems with Applications, 160*, 113678. https://doi.org/10.1016/j.eswa.2020.113678.