Computers & Operations Research 145 (2022) 105880

Contents lists available at ScienceDirect

Computers and Operations Research

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

Matheuristics for the flowshop scheduling problem with controllable processing times and limited resource consumption to minimize total tardiness

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

[a] Department of Industrial Engineering, Federal University of Ceara, Ceara, Brazil
[b] Department of Industrial Organization and Business Management I, University of Sevile, Sevile, Spain
[c] Industrial Management/Laboratory of Engineering for Environmental Sustainability, University of Seville, Seville, Spain
[d] Department of Statistics and Applied Mathematics, Federal University of Ceara, Ceara, Brazil

ARTICLE INFO

Keywords:
Production sequencing
Sustainable manufacturing
Flowshop scheduling
Mixed-integer linear programming
Heuristic cutting planes

ABSTRACT

This paper addresses the problem of scheduling jobs in a flowshop layout where the machines can operate at different speeds and require an amount of a resource that is a non-decreasing function of its speed. Furthermore, there is a limit in the maximum amount of the resource that can be consumed, so the problem serves to model either a constraint on the energy or raw material employed in the process, or on the maximum amount of pollutant that can be emitted. In contrast to the classical permutation flowshop scheduling problem, in the variant under study the processing times are controllable, there is an specific resource consumption, and each job has a due date. The objective function considered is the minimization of the total tardiness of the jobs. Given the NP-hard nature of this problem, we focus onto developing approximate solution procedures that can yield high-quality solutions with a reasonable CPU effort. More specifically, we develop a fast solution procedure to build an initial solution and two hybrid matheuristics based on mixed-integer linear programming formulations of the problem. The first one combines a relax-and-fix approach with the generation of heuristic cutting planes, while the second one relaxes some decision variables in the first stage and fixes other decision variables in the final stage. The computational experience carried out show that our proposals outperform the mathematical models as well as an approximate procedure proposed for a related problem.

1. Introduction

Permutation flowshop scheduling is a hard combinatorial optimization problem widely studied in the last decades. Several performance measures have been studied for the researchers of the production scheduling area, with an emphasis on the makespan minimization (Fernandez-Viagas et al., 2017). However, in several real-world applications, the permutation flowshop production environment presents additional features not considered in the classical formulation. In this regard, we can observe the possibility of controlling the processing times as an important characteristic since the machines can present speed scalability. Despite the prevalence of manufacturing scenarios where the machines can have different speeds, the explicit consideration of controllable processing times for the flowshop scheduling problem is rather scarce in the available literature (Shabtay and Steiner, 2007; Fernandez-Viagas and Framinan, 2015a).

Furthermore, higher machine speeds can lead to a reduction of the makespan or total tardiness. However, a fast speed consumes a large number of resources (i.e. energy, fuel,..), possibly with a higher emission of pollutants. Regarding such emission of pollutants or energy consumption, the term *green manufacturing* has been coined to encompass approaches that consider sustainability aspects in the productive processes (Dornfeld, 2012; Deif, 2011). In the last years, these problems are attracting the interest of researchers due to the improvements in the resolution methods and computational resources. Viewed in the context of the production scheduling domain, the terms energy-efficient scheduling (Albers, 2010; Gahm et al., 2016) and green scheduling (Bampis et al., 2015) have been applied to classify scheduling approaches that include aspects related to energy consumption, energy tariff, demand or power constraints, and pollutants emissions.

* Corresponding author.
E-mail addresses: baprata@ufc.br (B.A. Prata), vfernandezviagas@us.er (V. Fernandez-Viagas), framina@us.es (J.M. Framinan), diego@lia.ufc.br (C.D. Rodrigues).

https://doi.org/10.1016/j.cor.2022.105880
Received 24 December 2021; Received in revised form 24 March 2022; Accepted 9 May 2022
Available online 23 May 2022

In this paper we address a flowshop scheduling problem with controllable processing times, a constraint imposed on a resource consumption (typically power consumption) and the objective of minimizing the total tardiness. In our study, the speeds assume a discrete set of alternatives. Also, we consider that there is a linear relationship between speed and consumption. This variant arises in several real-world applications, such as paper manufacture, steel processing and finishing, and bread making. This problem is clearly NP-hard since the classical permutation flowshop scheduling problem with total tardiness minimization (without controllable processing times or a limit in the total resource consumption) is already NP-hard. We first present two mixed-integer linear programming (MILP) formulations for the problem. In addition, we develop two hybrid matheuristics (each one based in the MILP models developed) to find high-quality solutions within feasible computational times. Both matheuristics require an initial feasible solution, therefore we propose a method based on a knapsack-type linear programming model that provides such solution with negligible computational effort. Finally, we report an extensive computational experimentation to assess the performance of our proposals, also by comparing them with an state-of-the-art method from a related problem.

The remainder of this paper is organized as follows: in Section 2, we present the related approaches. In Section 3, we describe the problem under study. Besides, two MILP models for the problem are presented. These models with provide the basis for developing the proposed algorithms in Section 4. In Section 5, we explain the experimental design. In Section 6, we discuss some results from computational experiments; finally, in Section 7 we draw some conclusions and suggestions for future works.

2. Background

Since, to the best of our knowledge, the flowshop scheduling problem with controllable processing times and a limit in the total resource consumption with the objective of total tardiness minimization has not been addressed so far, here we review some related problems. More specifically, we have reviewed papers addressing either flowshop scheduling with controllable processing times, or green scheduling in the flowshop layout, where resource consumption (i.e. energy, fuel) is usually taken into account. Note that, in the literature of controllable processing times, one stream of research considers the machine speeds to take a discrete set of values (as in this study). However, another stream of research considers the speeds to take continuous values within a range. Some studies assume a linear relationship between machine speed and resource consumption, while others assume a non-linear relationship. We have not addressed this second line in our study.

Regarding the flowshop with controllable processing times, Janiak (1987) addresses a flowshop scheduling problem where the controllable processing times are expressed as decision variables, and the performance measure is the minimization of makespan. For each available machine, the processing times are resource-dependent. A branch-and-bound algorithm that uses some dominance rule is the proposed solution approach. Nowicki and Zdrzałka (1988) study a two-machine flowshop scheduling problem with controllable job processing times. The minimization of the total processing cost plus maximum completion time cost is considered as a performance measure. Two heuristics are developed: the first one presents a worst-case performance bound equal to m, and the second one yields high-quality results. Nowicki (1993) studies the m-machine permutation flowshop variant with controllable processing times. An extension of the algorithm proposed by Nowicki and Zdrzałka (1988) to the m-dimensional variant is addressed. Edwin Cheng and Shakhlevich (1999) address the proportionate flowshop with controllable processing times where each machine processes the jobs in the same order with equal processing times. An aggregation of makespan minimization and compression cost function is the performance measure considered. The corresponding bicriteria problem is also studied, and constructive algorithms are presented in both cases. Mokhtari et al. (2011) consider a flowshop with controllable processing time with a trade-off between makespan and the required amount of resources. The original problem is decomposed into two sub-problems: a sequencing problem and a resource allocation problem. As a solution procedure, a Discrete Differential Evolution algorithm (DDE) is combined with a Variable Neighborhood Search (VNS). Uruk et al. (2013) consider a non-preemptive two-machine flowshop scheduling with flexible operations and controllable processing times. The jobs are identical and must be processed three times: the first operation occurs in the first machine, the second operation occurs in the second machine, and the third operation occurs in one of the available machines. Two bi-criteria mixed-integer nonlinear models are presented for the minimization of the total manufacturing cost and makespan. A heuristic is proposed for finding high-quality solutions within admissible computational times.

Regarding the green scheduling problem in a flowshop environment, Fang et al. (2013) investigate a flowshop environment with peak-power consumption constraints. The processing times are controllable since the machines present different speeds with associated power consumption. Besides, a maximal power consumption constraint limits the utilization of the resources in a given production period. Two MILP formulations are presented for the makespan minimization. Furthermore, two fast heuristics are proposed to find feasible schedules as well as to evaluate the trade-off between makespan and peak-power consumption. Mansouri et al. (2016) study the problem of scheduling a two-machine flowshop to minimize makespan and energy consumption. A MILP formulation is proposed to find the Pareto front for both objectives. A lower bound for both objectives are proposed, as well as a fast heuristic that finds good approximations for the Pareto front. Zhang et al. (2017) propose a MILP model to optimize the scheduling of a factory for minimal energy cost under real-time pricing of electricity. A case study is addressed taking several operational scenarios into account. The results found point to a substantial reduction in electricity costs. Gao et al. (2018) address the no-wait, two-machine permutation flowshop scheduling problem with learning effect, common due dates, and controllable processing times. The objective function aggregates earliness, tardiness, common due date cost, and the cost associated with the resource allocation per time unit. They propose a polynomial algorithm that optimally solves this problem. Ramezanian et al. (2019) introduce a green flowshop scheduling problem with sequence-dependent setup times. A MILP model is proposed for the minimization of makespan and energy consumption. A constructive heuristic is proposed to find good approximations to the Pareto front. Computational experiments with randomly generated test instances, as well as a real-world problem, are presented. Foumani and Smith-Miles (2019) address a bi-objective green flowshop scheduling problem for the minimization of makespan and total carbon emission. A MILP formulation is proposed, using a weight aggregation approach to transform the original problem into a single objective one. These authors employ random data and real-based data in computational experiments. Öztop et al. (2020) address an energy-efficient flowshop scheduling problem for the minimization of total flow time and total energy consumption. These authors present a bi-objective MILP formulation with a speed-scaling framework. An improved version of the well-known NEH algorithm (Nawaz et al., 1983) is proposed as an initial solution. Furthermore, two variants of the iterated greedy metaheuristic and variable block-insertion algorithm are proposed. The iterated greedy algorithms presents better results in the computational experiments carried-out with randomly generated test instances. Amiri and Behnamian (2020) study a multi-objective green flowshop to minimize makespan and energy consumption under uncertainty. A mathematical formulation is proposed, as well as an estimation distribution algorithm that presents high-quality approximations to the Pareto front, within admissible computational times. Gomes et al. (2021) address

a no-wait flowshop that arises in the steel industry. The bi-objective modeling considers the minimization of total tardiness and the Total Energy Cost (TEC). A MILP formulation provides initial solutions to a multi-objective variable neighborhood search algorithm into a matheuristic framework. The multi-objective matheuristic outperforms a metaheuristic approach in a set of 24 instances based on a real-world problem from a heat treatment line in steel production.

As it can be seen from the background of the problem presented above, in the previous contributions on the flowshop scheduling problem with total tardiness minimization, the processing times are not controllable and the machines cannot operate with distinct speeds. Furthermore, the solution procedures have been designed for multi-objective approaches that consider the generation of approximations to the Pareto front. In our view, the flowshop scheduling problem to minimize total tardiness needs also to be explored with more depth in a single-objective setting while also including power consumption considerations so these advances can be incorporated to the evaluation of multiple objectives. Hence, the main innovations of our work are the following. First, to the best of our knowledge, the permutation flowshop scheduling problem taking into account resource consumption and total tardiness minimization has not been previously addressed in the literature. Second, we are not aware of contributions on this problem in a single-criterion setting where the total resource consumption is considered a constraint. Finally, to the best of our knowledge, there are no other works on this problem providing a mathematical formulation that considers the possibility of distinct speeds in each position of the sequence in each available machine.

3. Problem statement

3.1. Problem description

The production environment under consideration is characterized by a flowshop layout where the m machines have different speeds and there is a resource consumption constraint, which serves to model a maximum feasible amount of energy consumption or carbon emissions. More specifically, we assume a linear relationship between machine speed and resource consumption. Furthermore, the machines can operate with three modes (slow, normal, and fast). In this manner, depending on the selected mode, the processing times, as well as the resource consumption, are different. In this environment, n jobs have to be scheduled over the machines in the same order (flowshop). We assume that the relative order of the jobs is the same across all machines (permutation hypothesis), therefore π a permutation of jobs plus their processing speed in the available machines represents a solution. Each job j requires a processing time p_{ijl} to be processed on machine i if the latter is operated at a speed l, and consumes an amount of resource q_{ijl}. For a given mode s_{il} used to process the job j on machine i, let p_{ij} and p_{ij} be the processing times and amount of resources for the normal mode, the processing time p_{ijl} is given by $p_{ijl} = p_{ij}/s_{il}$. Similarly, the amount of resource $q_{i,j,l}$ is given by $q_{ijl} = p_{ij} \times s_{il}$. Furthermore, there is a due date d_j for each job and the total resource consumption should not exceed an amount Q. In our problem, Q represents the total available resource to be used to process all jobs on all machines, which is non-renewable. A job scheduled in the position k of the sequence π has a completion time C_{kj} on machine i as well as a tardiness T_k which is computed using the following expression: $T_k = max\{C_{kj} - d_j, 0\}$. Given these definitions, the problem under study is to find a sequence that minimizes the sum of the tardiness of the jobs.

Consider an illustrative example with six jobs and two machines with three speeds (slow, normal and fast), adapted from Pan and Fan (1997). In this example, considering the non-controllable problem with machines operating on normal speed (Table 2), and with due dates $d_i = \{9, 14, 16, 17, 21, 23\}$, the sequence $\pi = \{1, 2, 3, 4, 6, 5\}$ presents the global optimal solution of 45 time units. Taking into account a resource consumption constraint $Q = 100$, machines with different speeds (as illustrated in Tables 1, 2, and 3), as well as resource consumptions in the available machines (as illustrated in Tables 4, 5, and 6), the sequence $\pi = \{1, 2, 3, 4, 6, 5\}$ presents a optimal solution of 23.5 time units. In this solution, all operations were performed with normal mode, except for jobs 1, 2, and 3 in machine 1, which were processed in the fast mode.

Table 1
Processing times for flowshop example (fast speed).

$i\backslash j$	J_1	J_2	J_3	J_4	J_5	J_6
M_1	2	2.5	2	3.5	4.5	3
M_2	2	4	1.5	1.5	4	3.5

Table 2
Processing times for flowshop example (normal speed).

$i\backslash j$	J_1	J_2	J_3	J_4	J_5	J_6
M_1	4	5	4	7	9	6
M_2	4	8	3	3	8	7

Table 3
Processing times for flowshop example (slow speed).

$i\backslash j$	J_1	J_2	J_3	J_4	J_5	J_6
M_1	8	10	8	14	18	12
M_2	8	10	6	6	16	14

Table 4
Resource consumption for flowshop example (fast speed).

$i\backslash j$	J_1	J_2	J_3	J_4	J_5	J_6
M_1	10	12	10	18	22	14
M_2	10	20	8	8	20	16

Table 5
Resource consumption for flowshop example (normal speed).

$i\backslash j$	J_1	J_2	J_3	J_4	J_5	J_6
M_1	5	6	5	9	11	7
M_2	5	10	4	4	10	8

Table 6
Resource consumption for flowshop example (slow speed).

$i\backslash j$	J_1	J_2	J_3	J_4	J_5	J_6
M_1	2.5	3	2.5	4.5	5.5	3.5
M_2	2.5	5	2	2	5	4

3.2. MILP models

In this section, we present two Wilson family models (Wilson, 1989) to the variant under study based on the existing classic mathematical formulations (Stafford et al., 2005; Ronconi and Birgin, 2012; Fang et al., 2013; Foumani and Smith-Miles, 2019). These models are used to develop the approximate procedures described in Section 4. In the proposed models, we consider binary positional decision variables to control the sequence of the jobs. In our modeling, a job must be processed with a given speed at each machine. Hereafter, the notation used for the first model proposed is presented.

Indices

i: index for machines {1,2, ..., m}.
j: index for jobs {1,2, ..., n}.
k: index for positions {1,2, ..., n}.
l: index for speed {1,2, ..., s}.

Parameters

p_{ijl}: processing time of job j on machine i with speed factor l.

q_{ijl}: amount of resource to process job j on machine i with speed factor l.
d_j: due date of job j.
Q: maximal amount of available non-renewable resources.

Decision variables

T_k: tardiness of job in position k.
S_{ik}: starting time of job in position k on machine i.

$$x_{ijkl} = \begin{cases} 1, & \text{if job } j \text{ is scheduled on machine } i \\ & \text{at position } k \text{ with speed } l. \\ 0, & \text{otherwise} \end{cases}$$

The resulting MILP model can be stated as follows.

$(MILP1)$ minimize

$$\sum_{k=1}^{n} T_k \tag{1}$$

subject to

$$T_k \geq S_{mk} + \sum_{j=1}^{n}\sum_{l=1}^{s} p_{mjl}x_{mjkl} - \sum_{j=1}^{n}\sum_{l=1}^{s} d_j x_{mjkl}, \qquad \forall_k \tag{2}$$

$$S_{i,k+1} \geq S_{ik} + \sum_{j=1}^{n}\sum_{l=1}^{s} p_{ijl}x_{ijkl}, \qquad \forall_{i,k\in\{1,2,\ldots,n-1\}} \tag{3}$$

$$S_{i+1,k} \geq S_{ik} + \sum_{j=1}^{n}\sum_{l=1}^{s} p_{ijl}x_{ijkl}, \qquad \forall_{i\in\{1,2,\ldots,m-1\},k} \tag{4}$$

$$\sum_{j=1}^{n}\sum_{l=1}^{s} x_{ijkl} = 1, \qquad \forall_{i,k} \tag{5}$$

$$\sum_{k=1}^{n}\sum_{l=1}^{s} x_{ijkl} = 1, \qquad \forall_{i,j} \tag{6}$$

$$\sum_{l=1}^{s} x_{ijkl} = \sum_{l=1}^{s} x_{i+1,jkl}, \qquad \forall_{i\in\{1,2,\ldots,m-1\},j,k} \tag{7}$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{s} q_{ijl}x_{ijkl} \leq Q \tag{8}$$

$$\sum_{l=1}^{s} x_{ijkl} \leq 1, \qquad \forall_{i,j,k} \tag{9}$$

$$T_k \geq 0, \qquad \forall_k \tag{10}$$

$$S_{ik} \geq 0, \qquad \forall_{i,k} \tag{11}$$

$$x_{ijkl} \in \{0,1\}, \qquad \forall_{j,k,l} \tag{12}$$

The objective function (1) is the total tardiness minimization. Constraint set (2) calculates the tardiness for each job processed in position k. Constraints (3) and (4) ensure that the completion and starting times are consistent with a flowshop. Constraint set (5) guarantees that a given position in the sequence is allocated only to one job. Constraint set (6) forces that a given job is processed in a single position of the sequence. Constraint set (7) guarantees a permutation flowshop. Constraint (8) enforces that the amount of resource used is less than or equal to the maximal amount of resource available. Since the jobs are processed in each machine with one speed, we can also define the valid inequalities (9). Finally, constraint sets (10), (11), and (12) establish the domain of the decision variables.

In our second MILP formulation, we consider two types of binary decision variables, as described below. The first decision variable z_{ijl} models the speed configuration of the machines to process the jobs, and the second decision variable y_{jk} is a standard positional variable for the jobs.

$$y_{jk} = \begin{cases} 1, & \text{if job } j \text{ is scheduled on position } k. \\ 0, & \text{otherwise} \end{cases}$$

$$z_{ijl} = \begin{cases} 1, & \text{if job } j \text{ is scheduled on machine } i \text{ with speed } l. \\ 0, & \text{otherwise} \end{cases}$$

Thus, the second proposed model can be stated as follows.

$(MILP2)$ $\min \sum_{k+1}^{n} T_k$ (13)

subject to:

$$T_k \geq S_{mk} + \sum_{l=1}^{s} p_{mjl}\cdot(z_{mjl} + y_{jk} - 1) - \sum_{j=1}^{n} d_j y_{jk}, \qquad \forall j,k \tag{14}$$

$$S_{i,k+1} \geq S_{ik} + \sum_{l=1}^{s} p_{ijl}\cdot(z_{ijl} + y_{jk} - 1), \qquad \forall i,j,k < n \tag{15}$$

$$S_{i+1,k} \geq S_{ik} + \sum_{l=1}^{s} p_{ijl}\cdot(z_{ijl} + y_{jk} - 1), \qquad \forall i < m,j,k \tag{16}$$

$$\sum_{j=1}^{n} y_{jk} = 1, \qquad \forall k \tag{17}$$

$$\sum_{k=1}^{n} y_{jk} = 1, \qquad \forall j \tag{18}$$

$$\sum_{l=1}^{s} z_{ijl} = 1, \qquad \forall i,j \tag{19}$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{l=1}^{s} q_{ijl}z_{ijl} \leq Q \tag{20}$$

$$T_k \geq 0 \qquad \forall k \tag{21}$$

$$S_{ik} \geq 0 \qquad \forall i,k \tag{22}$$

$$z_{ijl} \in \{0,1\} \qquad \forall i,j,l \tag{23}$$

$$y_{jk} \in \{0,1\} \qquad \forall j,k \tag{24}$$

The objective function (13) is the total tardiness minimization. Constraint set (14) calculates the tardiness for each job processed in position k. Constraints (15) and (16) ensure that the completion and starting times are consistent with a flowshop. Constraint set (17) forces that a given job is processed in a single position of the sequence. Constraint set (18) guarantees that a given position in the sequence is allocated only to one job. Constraint set (19) imposes that the jobs are processed in each machine with one speed. Constraint (20) enforces that the used resources are less than or equal to the maximal amount of available resources. Finally, constraint sets (21), (22), (23), and (24) establish the domain of the decision variables.

4. Proposed solution approaches

Recently, a number of contributions have successfully addressed the hybridization of mathematical programming and heuristics to solve production scheduling problems (Della Croce et al., 2014; Lin and Ying,

2016; Framinan and Perez-Gonzalez, 2018; Pitombeira-Neto and Prata, 2020; Prata et al., 2020). These matheuristics incorporate some knowledge of the decision problem to guide the search of the solutions. Along this line, we propose for our problem two innovative matheuristics that explore distinct structures for fixing some variables in the model. Both solution approaches use the information included in the linear relaxation of the MILP models. More specifically, based on preliminary computational experiments with the proposed MILP models, it can be observed that, if a scheduling decision variable is zero in the linear relaxation, then it is likely to be zero in the integer solution. Additionally, preliminary experiments showed that the proposed solution procedures were not able to find feasible integer solutions for all the instances within a reasonable time limit, so we propose a fast method to obtain an initial feasible solution. This procedure is presented in Section 4.1.

Since we present two MILP formulations for the problem under study, we decide to explore the characteristics of both models to provide competitive matheuristics. Concerning the first model (MILP1), we could observe that with the utilization of positional decision variables incurred in a decision variable with four indices. In this data structure, we have a sparse matrix in which the vast majority of the values are equal to zero. Thus, the consideration of heuristic cutting planes could improve the performance of this formulation. This approach is discussed in Section 4.2. Regarding the second formulation (MILP2), the utilization of distinct decision variables to schedule the jobs and allocate speeds can be used to decompose the problem into two subproblems. In the first subproblem, we are looking for the generation of a permutation, and in the second subproblem, we are generating a complete solution, taking the speeds into account. This approach is presented in Section 4.3.

4.1. Initial Solution (IS)

The problem under study presents two main decisions: Scheduling the jobs in the production sequence and determining the speeds in each machine for each position of the production sequence. Thus, we face a hard combinatorial optimization problem with an extremely large search space. Aiming to obtain a feasible initial solution, we propose a fast procedure – denoted Initial Solution (IS) in the following – that combines exact and heuristic methods. In the first step, we employ a simplified MILP model to determine the maximal feasible speed in the available machines using a knapsack-based model. This MILP model (denoted by Maximal Fixed Speeds — MFS) can be stated as follows:

(MFS) max

$$\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{l=1}^{s} q_{ijl} z_{ijl} \tag{25}$$

subject to

$$\sum_{l=1}^{s} z_{ijl} = 1, \qquad \forall_{i,j} \tag{26}$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{l=1}^{s} q_{ijl} z_{ijl} \leq Q \tag{27}$$

$$z_{ijl} \in \{0,1\}, \qquad \forall_{i,j,l} \tag{28}$$

The objective function (25) maximizes the resource consumption in the available machines taking into account the resource consumption as the weight of each job. Constraint set (26) forces that a given job is processed on a machine at a single speed. Constraint set (27) imposes a limit in the maximum consumption of the resource. Finally, constraint set (28) defines the decision variables of the MILP model.

Since the consumption of the resource is a non-decreasing function of the speed of the machines, the MFS model actually determines the maximum speed of the machines to process the jobs that is compatible with the maximum amount of resource consumption. Therefore, with the speeds given by the MFS model we can compute the corresponding processing times for each job across the machines. Note, however, that this does not yield a solution of the problem under study, since this procedure does not take into account the production sequence. To determine such sequence, we use two well-known heuristics. i.e. the Earliest Due Date (EDD) rule and the Extensive Neighborhood Search (ENS) algorithm. In the EDD rule (first presented by Jackson (1955)), the jobs are sorted in a non-descending order of their due dates. The ENS algorithm by Kim (1993) starts from an initial sequence (usually generated by EDD), and attempts to improve it by exploring a neighborhood given by exchanging the positions of a pair of jobs in the current permutation. This improvement procedure is repeated until there is no better solution than the given one. Then, the solution with the minimum total tardiness is returned.

Using the processing times matrix given by the MFS model, we can employ the ENS algorithm to generate an initial sequence. This permutation can be used as a warm start x^{init}_{ijkl} for the MILP1 model. More specifically, the result of the MFS model provides a configuration of speeds to process the job j in the machine i. On the other hand, the ENS heuristic determines an initial allocation of each job j in the position k of the sequence. With both data, an initial configuration for the decision variables x^{init}_{ijl} can be built.

4.2. Relax-and-fix with heuristic cutting planes (RFHCP) matheuristic

Aiming to improve the solutions generated by the initial procedure, we first propose a hybrid matheuristic that considers a single-iteration relax-and-fix with heuristic cutting planes. Wolsey (1998) defined 0-1 knapsack inequalities called cover inequalities. If a given amount of resources exceeds the overall resource availability, an inequality could be inserted.

Definition 1. In the problem under consideration, a set C is a cover if

$$\sum_{i=1}^{m}\sum_{j\in C}\sum_{k=1}^{n}\sum_{l=1}^{s} q_{ijl} x_{ijkl} > Q \tag{29}$$

If C is a cover for the decision variable x, then

$$\sum_{i=1}^{m}\sum_{j\in C}\sum_{k=1}^{n}\sum_{l=1}^{s} x_{ijkl} \leq |C| - 1 \tag{30}$$

is valid for x.

We could add these cover inequalities in the first MILP model (MILP1), aiming to reduce the feasible solution space. Unfortunately, the number of possible cover inequalities is extremely high, and their insertion in the model is not possible, as the addition of cover inequalities in the branch-and-cut framework can be computationally prohibitive (Hunsaker and Tovey, 2005). Instead, we propose a heuristic approach to generate cutting planes. More specifically, we add heuristically cuts using cover inequalities determined for each machine. Therefore, we have not to guarantee that these are valid inequalities, since these cuts are treated as heuristic cutting planes or pseudo-cuts (Lasdon et al., 2010; Glover et al., 2011). Note that, although the inclusion of pseudo-cuts reduces the search space as compared to that of the original model, the new search space does not necessarily lead to the optimal solution of the original problem.

Another possible avenue to reduce the model in order to make it faster to solve would have been the exclusion of the decision variables as in Fanjul-Peyro and Ruiz (2011). Prata et al. (2020) proposed the Fixed Variable List Algorithm (FVLA), which incorporates the concepts of the well-known GRASP metaheuristic into a matheuristic procedure. Firstly, a list is constructed with the decision variables taken from the MILP model that could be set to zero as they probably would not be selected in an optimal or near-optimal solution. Subsequently, the decision variables in the list are fixed as zero, and a reduced model is solved. Nevertheless, excluding decision variables can deteriorate

the value of the objective function if some excluded variables would appear in high-quality solutions. Therefore, we propose a heuristic generation of cutting planes which is a more general approach than a variable-fixing heuristic: Since in the proposed MILP, we are working with positional decision variables, and with a total tardiness objective, we could generate cutting planes based on two distinct situations: (i) the jobs with the largest due dates values should not be scheduled in the first positions of the sequence; and (ii) the jobs with the lowest due dates values should not be scheduled in the last positions of the sequence. Based on these intuitive assumptions, we could establish pseudo-cuts or heuristic cutting planes aiming to reduce the search space.

To deal with this issue, we consider four parameters in our heuristic cutting plane generation:

- α: control the number of positions at the beginning of the sequence ($\alpha \in [0, 1]$).
- β: control the number of positions at the ending of the sequence ($\beta \in [0, 1]$).
- b_1 and b_2: right-hand side values that control the number of decision variables that can be selected in a given feasible integer solution.

Sorting the jobs considering the well-known earliest due date (EDD) dispatch rule, we can define the sets J_1, J_2, P_1, and P_2, as follows:

$$J_1 \leftarrow \{\pi_j^{EDD} | j \leq \lceil \alpha n \rceil\} \tag{31}$$

$$J_2 \leftarrow \{\pi_j^{EDD} | j \geq \lceil (1-\beta) n \rceil\} \tag{32}$$

$$P_1 \leftarrow \{j | j \leq \lceil \alpha n \rceil\} \tag{33}$$

$$P_2 \leftarrow \{j | j \geq \lceil (1-\beta) n \rceil\} \tag{34}$$

Thus, we can insert the following heuristic cutting planes in the proposed matheuristic:

$$\sum_{i=1}^{m} x_{ijkl} \leq b_1, \forall_{j \in J_1, k \in P_1, l} \tag{35}$$

$$\sum_{i=1}^{m} x_{ijkl} \leq b_2, \forall_{j \in J_2, k \in P_2, l} \tag{36}$$

Taking the illustrative example described in Section 3 into account, assuming $\alpha = 0.1$, $\beta = 0.5$, $b_1 = \lceil 0.5n \rceil$, and $b_2 = \lceil 0.3n \rceil$, we can calculate the following values: $J_1 = \{1\}$, $J_2 = \{4, 5, 6\}$, $b_1 = 3$, and $b_2 = 2$.

Algorithm 1 describes the proposed matheuristic, named Relax-and-Fix with Heuristic Cutting Planes (RFHCP). Firstly, we determine the fixed speeds using the knapsack-based formulation MFS. Using these fixed speeds, we determine an initial solution with the ENS heuristic. Taking the fixed speeds as well as the initial sequence, we can set initial values of the decision variables x_{ijkl}. After that, we perform a single iteration of the relax-and-fix algorithm (Wolsey, 1998), in which the integrality of the positional decision variables is relaxed. If $x_{ijkl}^{init} = x_{ijkl}^{relax} = 0, \forall i, j, k, l$, then we set x_{ijkl} as equal to zero. If a relaxed decision variable takes a zero value in the relaxed solution, there is a high possibility that this variable also is zero in the integer solution. We generate the heuristic cutting planes, as described in Section 4.2. Finally, the resulting MILP model is then solved, with the fixed decision variables and heuristic cutting planes, as previously described.

4.3. Relax-first fix-second (RFFS) matheuristic

In our second proposed solution approach, we address a heuristic procedure based on the relaxation and fixation of decision variable. We consider our second MILP model (MILP2) in this strategy. Let us consider that the speed modes are in non-decreasing order, that is, resource consumption q_{ijl} for a fixed machine i and job j are non-decreasing for $l = 1, \ldots, s$. On the other hand, it seems reasonable to assume that the processing times p_{ijl} are non-increasing with the speed. In this case we propose a linear approximation model in which modal binary variables z_{ijl} are substituted by continuous variables $0 \leq z_{ijl} \leq 1$.

Algorithm 1: Relax-and-Fix with Heuristic Cutting Planes Algorithm
Step 1: Determine initial (maximum) speeds by solving MFS.
Step 2: Determine an initial sequence of jobs using the ENS algorithm.
Step 3: Set x_{ijl}^{init}, an initial solution for MILP1 model according to the initial speeds and initial job sequence determined in the previous steps.
Step 4: Determine x_{ijl}^{relax} by solving the linear relaxation of the MILP1 model.
Step 5: If $x_{ijl}^{init} = x_{ijl}^{relax} = 0 \forall i, j, l$; $x_{ijl} = 0$.
Step 6: Generate the heuristic cutting planes using Eqs. (35) and (36).
Step 7: Solve the resulting MILP model (i.e. MILP1 plus the heuristic cutting planes added in the previous step).

This resulting model is a relaxation of MILP2, where instead of forcing machines and jobs to commit to have a fixed speed, we use a linear relaxation to let them choose any value inside an interval. This model could be used to find the ideal ordering of jobs prior to choose the modal values. If we consider a fixed value y^* for the job ordering, then MILP2 can be simplified by replacing the constraint set (14) with the constraint set (37):

$$T_k \geq S_{mk} + \sum_{l=1}^{s} p_{mjl} \cdot (z_{mjl} + y_{jk} - 1) - \sum_{j=1}^{n} d_j y_{jk}, \qquad \forall j, k : y_{jk}^* = 1 \tag{37}$$

The solution of Model 2 (which computes y^*) followed by Model 3 (resulting x^*) gives a viable, but possibly not optimal solution of MILP2. In the relaxed model, instead of forcing machines to process jobs at a fixed speed, we use a linear value to let them choose any value inside an interval. Thereby, we can find an initial sequence of the jobs without considering discrete speeds. One possible way to strengthen this solution is to solve Model 2 and Model 3 iteratively, with a tabu list that forbids repeating past solutions. However, this iterative process requires high computational times and we would not evaluate this strategy.

Algorithm 2 illustrates the second proposed matheuristic, named Relax-First Fix-Second (RFFS). Firstly, we determine the initial values of z_{ijl} using the knapsack-based formulation MFS. After that, we determine the initial values of y_{jk} with the ENS heuristic. We use these values for a warm start in the MILP2 model, relaxing the integrality of z_{ijl}. We run this relaxed model within a given time limit, returning the values of y_{jk}^*. Finally, we solve the MILP2 with the fixed values of y_{jk}^* until the overall time limit is reached.

Algorithm 2: Relax-First Fix-Second Algorithm
Step 1: Solve the MFS model to determine the initial values of z_{ijl}.
Step 2: Apply the ENS algorithm to determine the initial values of y_{jk}.
Step 3: Taking into account the initial values of z_{ijl} and y_{jk}, run the MILP2 with the integrality of the decision variables z_{ijl} relaxed during a given partial time limit, determining y_{jk}^*.
Step 4: Solve the resulting MILP2 model, with the fixed values of y_{jk}^* until the overall time limit is reached.

5. Experimental design

In this section we describe the experimental design. In Section 5.1, we present the statistics used in the analysis of the computational

Table 7
Parameters used in the test instances.

Parameter	Levels	Based on
Number of machines	$m \in \{5, 10, 15\}$	Ramezanian et al. (2019)
Number of jobs	$n \in \{40, 80, 120\}$	Mansouri et al. (2016), Zhang et al. (2019), Fernandez-Viagas and Framinan (2020)
Processing time distribution	$U[1,99]$	Taillard (1993), Mansouri et al. (2016), Foumani and Smith-Miles (2019), Öztop et al. (2020)
Resource consumption distribution	$U[1,99]$	
Processing speed	1.2,1,0.8	Mansouri et al. (2016)
Resource consumption constraint	$\left\lfloor 0.9 \times \sum_{i=1}^{m} \sum_{j=1}^{n} q_{ij2} \right\rfloor$	Pisinger (2005)
Tardiness factor	$T \in \{0.2, 0.4, 0.6\}$	Potts and Van Wassenhove (1982), Vallada et al. (2008)
Due date ranges	$R \in \{0.2, 0.6, 1.0\}$	Potts and Van Wassenhove (1982), Vallada et al. (2008)

experiments, while in Section 5.2 the test instances employed are described. Finally, in Section 5.3 we show the Taguchi experimental design used to calibrate the proposed RFHCP matheuristic.

5.1. Statistics used in the analysis of the computational experiments

We use the Relative Deviation Index (RDI) indicator as a performance measure, which is a standard indicator for scheduling problems with due-date objectives (see e.g. Fernandez-Viagas and Framinan (2015b), Karabulut (2016)). Let H be a set of methods, the RDI obtained for the method $s \in H$ when applied to instance t is calculated as in Eq. (38).

$$RDI_{st} = \begin{cases} 0, & \text{if } \min_{h \in H} T_{ht} = \max_{h \in H} T_{ht}, \\ \frac{T_{st} - \min_{h \in H} T_{ht}}{\max_{h \in H} T_{ht} - \min_{h \in H} T_{ht}} \cdot 100, & \text{otherwise.} \end{cases} \tag{38}$$

where T_{st} is the tardiness value obtained by method s in instance t. In our case $\min_{h \in H} T_{ht}$ is the best solution found among the methods under comparison. To summarize the computational results, we calculate the average RDI (ARDI) for a given method by grouping the RDI obtained across a given set of instances.

We also use the Success Rate (SR) as another performance indicator. SR is calculated as the number of times that a given method finds the best solution (with or without a draw) divided by the number of test instances in a given instance set, as expressed as in Eq. (39):

$$\text{SR} = \frac{n_{\text{BEST}}}{n_{\text{INST}}} \times 100 \tag{39}$$

where n_{BEST} is the number of instances in which a given method achieved the best solution and n_{INST} is the number of instances in the given instance set.

5.2. Test instances

We use the following parameters for the computational experiments: number of machines (m), number of jobs (n), tardiness factor (T), and due date range (R). The values of these parameters are defined based on a literature review, as presented in Table 7. The total number of instance classes is given by 3 (m) × 3(n) × 3(T) × 3 (R) = 81. For each class, we randomly generate 5 test instances aiming at reducing the sampling error, resulting in a total of 405 test instances.

5.3. Taguchi experimental design

Since the proposed RFHCP presents distinct parameters, we evaluate their influence on its performance using a Design Of Experiments (DOE) approach. Since the number of test instances and parameters to be calibrated is high, a full factorial design could be prohibitive in terms of computational effort. Thereby, we consider a Taguchi DOE (Antony, 2014) which is a robust method that reduces the required number of trials. We can observe that the Taguchi method has been widely used to calibrate parameters in production scheduling solution approaches (Gholami et al., 2009; Abreu et al., 2020).

Table 8
All experiments for L9(3∧4).

Trial	α	β	b_1	b_2	ARDI
1	0.1	0.1	$\lceil 0.1n \rceil$	$\lceil 0.1n \rceil$	22.2
2	0.1	0.3	$\lceil 0.3n \rceil$	$\lceil 0.3n \rceil$	33.1
3	0.1	0.5	$\lceil 0.5n \rceil$	$\lceil 0.5n \rceil$	33.8
4	0.3	0.1	$\lceil 0.3n \rceil$	$\lceil 0.5n \rceil$	32.8
5	0.3	0.3	$\lceil 0.5n \rceil$	$\lceil 0.1n \rceil$	37.0
6	0.3	0.5	$\lceil 0.1n \rceil$	$\lceil 0.3n \rceil$	37.4
7	0.5	0.1	$\lceil 0.5n \rceil$	$\lceil 0.3n \rceil$	33.1
8	0.5	0.3	$\lceil 0.1n \rceil$	$\lceil 0.5n \rceil$	33.0
9	0.5	0.5	$\lceil 0.3n \rceil$	$\lceil 0.1n \rceil$	41.2

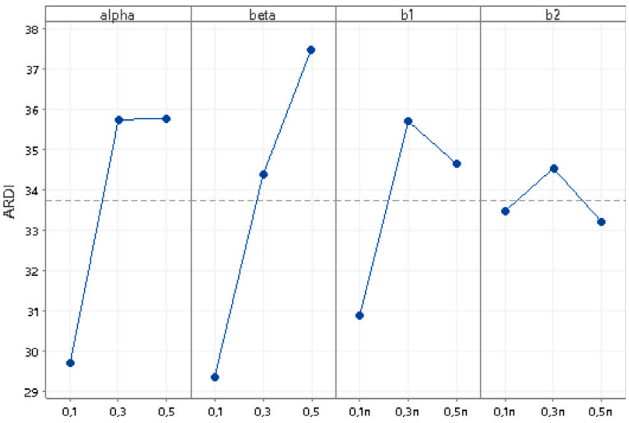

Fig. 1. Effects plots for difference levels of the controlled factors.

We adopt a Taguchi experimental design with 4 factors (α, β, b_1, and b_2) and 9 tests. The notation is L9($3 \wedge 4$), in which each factor presents three levels. Table 8 describes each test. For the Taguchi experiment, we select a random sample of 20%, corresponding to 81 test instances. Fig. 1 describes the main effects of the analyzed factors using the ARDI indicator. The trials 1, 5, 6, 8, and 9 resulted in 3, 17, 17, 33, and 33 instances without a feasible integer solution in the specified time limit, so we removed these instances to calculate the ARDI values. The rest of the trials returned integer solutions in all the sampled test instances. In view of the results of the Taguchi experimental design, the best parametrization for the proposed RFHCP matheuristic is $\alpha = 0.1$, $\beta = 0.1$, $b_1 = \lceil 0.1n \rceil$, and $b_2 = \lceil 0.1n \rceil$. However, this parametrization generated infeasible solutions in some test instances. Thereby, we use the parametrization that leads to the best results in the Taguchi experimental design: $\alpha = 0.3$, $\beta = 0.1$, $b_1 = \lceil 0.3n \rceil$, and $b_2 = \lceil 0.5n \rceil$.

Table 9
Average RDI and SR values for m and n.

		RDI					SR				
m	n	MILP1	MILP2	VNS	RFHCP	RFFS	MILP1	MILP2	VNS	RFHCP	RFFS
5	40	1.0	100.0	36.7	5.9	45.6	80.0	0.0	0.0	20.0	0.0
	80	7.3	89.5	43.2	0.0	27.5	2.2	0.0	0.0	100.0	0.0
	120	100.0	37.0	66.0	0.0	9.2	0.0	0.0	0.0	100.0	0.0
10	40	7.4	95.8	38.4	4.1	43.7	37.8	0.0	2.2	42.2	17.8
	80	100.0	30.8	61.5	0.1	5.1	0.0	0.0	0.0	88.9	11.1
	120	41.5	56.5	80.2	0.4	4.1	15.6	0.0	0.0	80.0	4.4
15	40	1.7	55.8	95.6	3.9	11.8	53.3	0.0	0.0	20.0	26.7
	80	100.0	30.7	65.5	1.7	3.4	0.0	0.0	0.0	37.8	62.2
	120	90.0	49.6	79.8	1.9	0.6	0.0	0.0	0.0	31.1	68.9
Average		49.9	60.6	63.0	2.0	16.8	21.0	0.0	0.2	55.6	21.2
Minimum		1.0	30.7	36.7	0.0	0.6	0.0	0.0	0.0	0.2	0.0
Maximum		100.0	100.0	95.6	5.9	45.6	80.0	0.0	2.2	100.0	68.9

6. Computational results

We implemented all the methods using Julia version 1.6 (https://julialang.org/) with Visual Studio Code IDE (https://code.visualstudio.com/). For the mathematical programming model, we used the commercial solver Gurobi (https://www.gurobi.com/) version 9.0.3 with JuMP library (https://www.juliaopt.org/JuMP.jl/stable/) (Lubin and Dunning, 2015). We performed the computational experience on a PC with Intel Core i7-8700 CPU 3.20 GHz and 32 GB memory, with the Ubuntu 20.04 LTS operating system.

In order to compare our proposals with existing related approaches, note that, regarding the green flowshop scheduling problem, the majority of solution procedures were developed for makespan related objectives (Mokhtari et al., 2011; Mansouri et al., 2016; Öztop et al., 2020). In our view, since such proposed solution approaches were not designed for the total tardiness performance measure, a comparison with the proposed matheuristic would not be fair. Therefore, we adapt the closest approach in the available literature and selected the matheuristic proposed by Gomes et al. (2021) for the no-wait flowshop to minimize total tardiness and total energy costs. In our adaptation, we consider a single objective (total tardiness minimization). This matheuristic is composed of two phases. First, an initial solution is generated by the MILP1 model, as in Section 3.2. After that, a Variable Neighborhood Search (VNS) improves the initial solution provided by the MILP1 model. We consider a fixed permutation in all machines since the computational cost for a local search procedures for each available machine is prohibitive for large-sized test instances. The first operator of our adaptation of the VNS is a shake procedure in the sequence. Considering a given solution π, this permutation is modified with two movements randomly selected: the exchange (swap) of two jobs and the insertion of a job in a new position of the sequence. Furthermore, we included an adaptation of this shake procedure to the speeds, as proposed by Mokhtari et al. (2011). Next, a VNS procedure is performed with a first improvement strategy. Two neighborhoods are evaluated: the set of all jobs given by the exchange and insert operations. The best total tardiness found is returned using a best improvement policy.

Therefore, in the computational experiments we considered the following solution approaches:

- MILP model MILP1.
- MILP model MILP2.
- An adaptation to the problem of the VNS proposed by Gomes et al. (2021), denoted VNS in the following.
- Relax-and-Fix with Heuristic Cutting Planes (RFHCP).
- Relax-First Fix-Second (RFFS).

For all these methods, we adopt a time limit $t_{limit} = mn \times 500/1{,}000$ s. Regarding the computational times, we note the following issues. Concerning the hybrid VNS proposed by Gomes et al. (2021), the original time limit used for the MILP warm start in their proposal is 3,600s, and there is no time limit for the VNS procedure. Here we adopt a time limit of $t_{limit}/2$ for both the MILP and VNS steps, respectively.

Concerning the proposed RFFS matheuristic, we consider a time limit of 80% of the total time limit in Steps 1, 2, and 3, and 20% of the global time limit for the remainder of the search process.

Table 9 illustrates the average and standard deviation of the RDI values for the distinct values of m and n. Based on such results, the following comments can be highlighted:

- The MILP1 model is the best solution approach for the test instances with 40 jobs. Considering problems with 80 and 120 jobs, the ARDI values for the MILP model tend to increase. In general terms, the MILP2 model returned better results than the MILP1 for test instances with 80 and 120 jobs.
- The VNS matheuristic returned the worst ARDI values among all the methods under comparison.
- The RFHCP matheuristic returned the best ARDI values for all the test instances except for the test instances with 15 machines and 120 jobs, where RFFS returned the best average results.
- The RFHCP matheuristic returned the best SR values for the test instances with 5 and 10 machines, while the RFFS matheuristic returned the best SR values for the test instances with 15 machines.
- The average RDI values of the MILP1, MILP2, VNS, RFHCP, and RFFS are 49.9%, 60.6%, 63.0%, 2.0%, and 16.8%, respectively. Concerning the average SR, the above-mentioned solution approaches returned the following values, respectively: 21.0%, 0.0%, 0.2%, 55,6%, and 21.2%. Thus, the proposed matheuristics presented better results than all the other evaluated solution procedures.

Table 10 describes the same results aggregated with respect to T and R parameters. From this table, we can highlight that the aggregation of the results concerning T and R values does not provide a significant difference in the results.

Aiming to achieve a pairwise comparison between the evaluated solution approaches, we performed an ANOVA procedure, followed by a Tukey's test (Montgomery, 2017). Figs. 2 and 3 illustrate the boxplot for the ARDI values and the Tukey multiple comparisons of means with 95% family-wise confidence level, respectively. Based on the achieved results, the following comments can be highlighted:

- The ANOVA test returned a statistics $F = 350.6$. Since this value is much higher than the critical value of 4.39, the difference among the solution procedures is statistically significant.
- The difference in the results returned by the MILP1 and MILP2 models is not statistically significant.

Table 10
Average RDI and SR values for T and R.

		RDI					SR				
T	R	MILP1	MILP2	VNS	RFHCP	RFFS	MILP1	MILP2	VNS	RFHCP	RFFS
0.2	0.2	57.7	58.0	65.5	2.4	18.9	22.2	0.0	0.0	62.2	15.6
	0.6	56.0	60.2	64.9	2.3	18.8	20.0	0.0	0.0	53.3	26.7
	1.0	55.4	53.4	66.7	3.0	18.6	22.2	0.0	2.2	55.6	20.0
0.4	0.2	53.3	59.3	61.8	1.6	21.4	17.8	0.0	0.0	64.4	17.8
	0.6	49.0	64.7	55.6	1.4	17.7	20.0	0.0	0.0	53.3	26.7
	1.0	46.7	63.5	61.3	2.0	12.8	20.0	0.0	0.0	60.0	20.0
0.6	0.2	41.9	61.8	64.7	1.9	13.9	20.0	0.0	0.0	53.3	26.7
	0.6	46.7	61.7	61.6	1.7	15.1	26.7	0.0	0.0	53.3	20.0
	1.0	42.2	63.2	64.8	1.8	13.9	20.0	0.0	0.0	62.2	17.8
Average		49.9	60.6	63.0	2.0	16.8	21.0	0.0	0.2	57.5	21.2
Minimum		41.9	53.4	55.6	1.4	12.8	17.8	0.0	0.0	53.3	15.6
Maximum		57.7	64.7	66.7	3.0	21.4	26.7	0.0	2.2	64.4	26.7

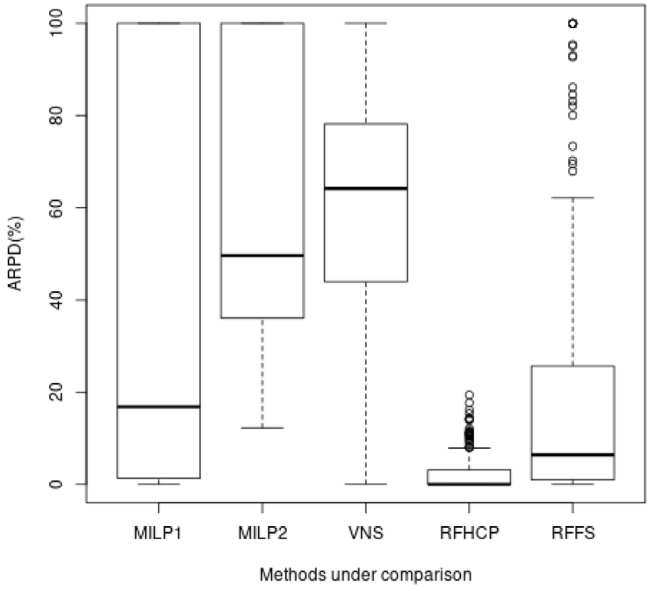

Fig. 2. Boxplot for ARDI values.

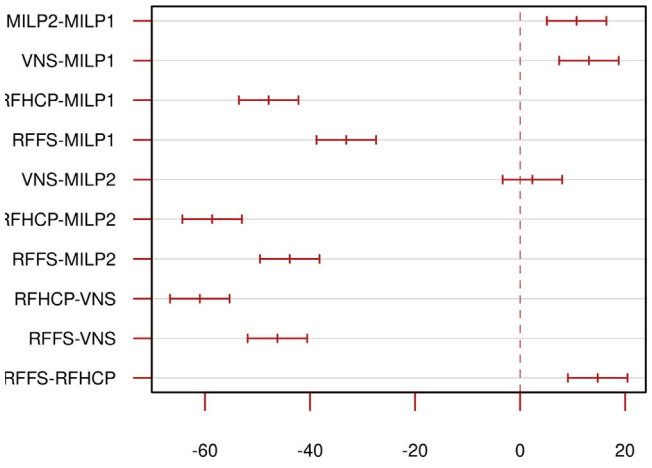

Fig. 3. Tukey confidence intervals for ARDI values.

- The VNS metaheuristic is outperformed by the proposed matheuristics.
- The difference in the results returned by the proposed RFHCP and RFFS matheuristics is not statistically significant.

7. Conclusions

In this paper, we have investigated a new variant of the permutation flowshop scheduling problem considering controllable processing times. The objective is to minimize the total tardiness, subject to maximal resource consumption. We presented two mixed-integer linear programming (MILP) formulations and two matheuristics based in the MILP models, together with a heuristic procedure to generate a feasible initial solution for the matheuristics. Computational experiments were carried out in order to evaluate the performance of the proposed solution approaches. We used the relative deviation index and the success rate as performance measures. In most tested problem instances, the proposed matheuristics showed the best performance, outperforming the MILP models and the hybrid VNS addressed by Gomes et al. (2021) with statistical significance.

Considering the average RDI indicator, the difference between RFHCP and RFFS matheuristics is not statistically significant. Nevertheless, considering the SR indicator, RFHCP returned better results than RFFS. However, it is to note that RFHCP requires the calibration of four parameters, while the RFFS presents a low dependency on parameters. As a summary, it can be stated that the proposed matheuristics are promising solutions approaches to solve the problem under study.

As extensions of this work, we recommend the use of metaheuristics in order to improve the solutions generated by the initial solution procedure. Future studies could also investigate the behavior of the proposed solution procedures considering other objective functions, such as in a Just-in-Time environment (Rolim and Nagano, 2020). Finally, it is worth studying a non-permutation encoding for the variant under study since the total tardiness can be improved with this assumption (Liao and Huang, 2010).

CRediT authorship contribution statement

Bruno de Athayde Prata: Conceptualization, Investigation, Software, Formal analysis, Writing – original draft, Visualization, Funding acquisition. **Victor Fernandez-Viagas:** Conceptualization, Writing – review & editing. **Jose M. Framinan:** Conceptualization, Writing – review & editing, Supervision, Project administration, Funding acquisition. **Carlos Diego Rodrigues:** Formal analysis, Methodology.

Acknowledgments

This study was financed in part by the National Council for Scientific and Technological Development (CNPq), through grants 303594/2018-7, 309755/2021-2, and 407151/2021-4, the Spanish Ministry of Science and Education (ASSORT project, Grant no. PID2019-108756RB-I00), and the Andalusian Government (projects DEMAND Grant no. P18-FR-1149 and EFECTOS, Grant no. US-1264511).

References

Abreu, L.R., Cunha, J.O., Prata, B.A., Framinan, J.M., 2020. A genetic algorithm for scheduling open shops with sequence-dependent setup times. Comput. Oper. Res. 113, 104793.

Albers, S., 2010. Energy-efficient algorithms. Commun. ACM 53 (5), 86–96.

Amiri, M.F., Behnamian, J., 2020. Multi-objective green flowshop scheduling problem under uncertainty: Estimation of distribution algorithm. J. Cleaner Prod. 251, 119734.

Antony, J., 2014. Design of Experiments for Engineers and Scientists. Elsevier.

Bampis, E., Letsios, D., Lucarelli, G., 2015. Green scheduling, flows and matchings. Theoret. Comput. Sci. 579, 126–136.

Deif, A.M., 2011. A system model for green manufacturing. J. Cleaner Prod. 19 (14), 1553–1559.

Della Croce, F., Salassa, F., T'kindt, V., 2014. A hybrid heuristic approach for single machine scheduling with release times. Comput. Oper. Res. 45, 7–11.

Dornfeld, D.A., 2012. Green Manufacturing: Fundamentals and Applications. Springer Science & Business Media.

Edwin Cheng, T., Shakhlevich, N., 1999. Proportionate flow shop with controllable processing times. J. Sched. 2 (6), 253–265.

Fang, K., Uhan, N.A., Zhao, F., Sutherland, J.W., 2013. Flow shop scheduling with peak power consumption constraints. Ann. Oper. Res. 206 (1), 115–145.

Fanjul-Peyro, L., Ruiz, R., 2011. Size-reduction heuristics for the unrelated parallel machines scheduling problem. Comput. Oper. Res. 38 (1), 301–309.

Fernandez-Viagas, V., Framinan, J.M., 2015a. Controllable processing times in project and production management: analysing the trade-off between processing times and the amount of resources. Math. Probl. Eng. 2015.

Fernandez-Viagas, V., Framinan, J.M., 2015b. NEH-based heuristics for the permutation flowshop scheduling problem to minimise total tardiness. Comput. Oper. Res. 60, 27–36.

Fernandez-Viagas, V., Framinan, J., 2020. Design of a testbed for hybrid flow shop scheduling with identical machines. Comput. Ind. Eng. 141.

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 J. Oper. Res. 257 (3), 707–721.

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

Framinan, J.M., Perez-Gonzalez, P., 2018. Order scheduling with tardiness objective: Improved approximate solutions. European J. Oper. Res. 266 (3), 840–850.

Gahm, C., Denz, F., Dirr, M., Tuma, A., 2016. Energy-efficient scheduling in manufacturing companies: A review and research framework. European J. Oper. Res. 248 (3), 744–757.

Gao, F., Liu, M., Wang, J.-J., Lu, Y.-Y., 2018. No-wait two-machine permutation flow shop scheduling problem with learning effect, common due date and controllable job processing times. Int. J. Prod. Res. 56 (6), 2361–2369.

Gholami, M., Zandieh, M., Alem-Tabriz, A., 2009. Scheduling hybrid flow shop with sequence-dependent setup times and machines with random breakdowns. Int. J. Adv. Manuf. Technol. 42 (1–2), 189–201.

Glover, F., Lasdon, L., Plummer, J., Duarte, A., Marti, R., Laguna, M., Rego, C., 2011. Pseudo-cut strategies for global optimization. Int. J. Appl. Metaheuristic Comput. (IJAMC) 2 (4), 1–12.

Gomes, A.C.L., Ravetti, M.G., Carrano, E.G., 2021. Multi-objective matheuristic for minimization of total tardiness and energy costs in a steel industry heat treatment line. Comput. Ind. Eng. 106929. http://dx.doi.org/10.1016/j.cie.2020.106929, URL http://www.sciencedirect.com/science/article/pii/S0360835220306112.

Hunsaker, B., Tovey, C.A., 2005. Simple lifted cover inequalities and hard knapsack problems. Discrete Optim. 2 (3), 219–228.

Jackson, J.R., 1955. Scheduling a production line to minimize maximum tardiness. In: Management Science Research Project. University of California.

Janiak, A., 1987. Flow shop scheduling with controllable operation processing times. IFAC Proc. Vol. 20 (9), 533–536, 4th IFAC/IFORS Symposium on Large Scale Systems: Theory and Applications 1986, Zurich, Switzerland, 26-29 August 1986.

Karabulut, K., 2016. A hybrid iterated greedy algorithm for total tardiness minimization in permutation flowshops. Comput. Ind. Eng. 98, 300–307.

Kim, Y.-D., 1993. Heuristics for flowshop scheduling problems minimizing mean tardiness. J. Oper. Res. Soc. 44 (1), 19–28.

Lasdon, L., Duarte, A., Glover, F., Laguna, M., Martí, R., 2010. Adaptive memory programming for constrained global optimization. Comput. Oper. Res. 37 (8), 1500–1509.

Liao, L.-M., Huang, C.-J., 2010. Tabu search for non-permutation flowshop scheduling problem with minimizing total tardiness. Appl. Math. Comput. 217 (2), 557–567.

Lin, S.-W., Ying, K.-C., 2016. Optimization of makespan for no-wait flowshop scheduling problems using efficient matheuristics. Omega 64, 115–125.

Lubin, M., Dunning, I., 2015. Computing in operations research using julia. INFORMS J. Comput. 27 (2), 238–248.

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

Mokhtari, H., Abadi, I.N.K., Cheraghalikhani, A., 2011. A multi-objective flow shop scheduling with resource-dependent processing times: trade-off between makespan and cost of resources. Int. J. Prod. Res. 49 (19), 5851–5875.

Montgomery, D.C., 2017. Design and Analysis of Experiments. John wiley & sons.

Nawaz, M., Enscore, Jr., E.E., Ham, I., 1983. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. Omega 11 (1), 91–95.

Nowicki, E., 1993. An approximation algorithm for the m-machine permutation flow shop scheduling problem with controllable processing times. European J. Oper. Res. 70 (3), 342–349.

Nowicki, E., Zdrzałka, S., 1988. A two-machine flow shop scheduling problem with controllable job processing times. European J. Oper. Res. 34 (2), 208–220.

Öztop, H., Tasgetiren, M.F., Eliiyi, D.T., Pan, Q.-K., Kandiller, L., 2020. An energy-efficient permutation flowshop scheduling problem. Expert Syst. Appl. 150, 113279.

Pan, J.C.-H., Fan, E.-T., 1997. Two-machine flowshop scheduling to minimize total tardiness. Internat. J. Systems Sci. 28 (4), 405–414.

Pisinger, D., 2005. Where are the hard knapsack problems? Comput. Oper. Res. 32 (9), 2271–2284. http://dx.doi.org/10.1016/j.cor.2004.03.002, URL http://www.sciencedirect.com/science/article/pii/S030505480400036X.

Pitombeira-Neto, A.R., Prata, B.A., 2020. A matheuristic algorithm for the one-dimensional cutting stock and scheduling problem with heterogeneous orders. Top 28 (1), 178–192.

Potts, C.N., Van Wassenhove, L.N., 1982. A decomposition algorithm for the single machine total tardiness problem. Oper. Res. Lett. 1 (5), 177–181.

Prata, B.A., Rodrigues, C.D., Framinan, J.M., 2020. Customer order scheduling problem to minimize makespan with sequence-dependent setup times. Comput. Ind. Eng. 106962. http://dx.doi.org/10.1016/j.cie.2020.106962, URL http://www.sciencedirect.com/science/article/pii/S0360835220306355.

Ramezanian, R., Vali-Siar, M.M., Jalalian, M., 2019. Green permutation flowshop scheduling problem with sequence-dependent setup times: a case study. Int. J. Prod. Res. 57 (10), 3311–3333.

Rolim, G.A., Nagano, M.S., 2020. Structural properties and algorithms for earliness and tardiness scheduling against common due dates and windows: A review. Comput. Ind. Eng. 149, 106803.

Ronconi, D.P., Birgin, E.G., 2012. Mixed-integer programming models for flowshop scheduling problems minimizing the total earliness and tardiness. In: Just-in-Time Systems. Springer, pp. 91–105.

Shabtay, D., Steiner, G., 2007. A survey of scheduling with controllable processing times. Discrete Appl. Math. 155 (13), 1643–1666.

Stafford, E., Tseng, F.T., Gupta, J.N., 2005. Comparative evaluation of MILP flowshop models. J. Oper. Res. Soc. 56 (1), 88–101.

Taillard, E., 1993. Benchmarks for basic scheduling problems. European J. Oper. Res. 64 (2), 278–285.

Uruk, Z., Gultekin, H., Akturk, M.S., 2013. Two-machine flowshop scheduling with flexible operations and controllable processing times. Comput. Oper. Res. 40 (2), 639–653.

Vallada, E., Ruiz, R., Minella, G., 2008. Minimising total tardiness in the m-machine flowshop problem: A review and evaluation of heuristics and metaheuristics. Comput. Oper. Res. 35 (4), 1350–1373. http://dx.doi.org/10.1016/j.cor.2006.08.016, URL http://www.sciencedirect.com/science/article/pii/S0305054806002048.

Wilson, J., 1989. Alternative formulations of a flow-shop scheduling problem. J. Oper. Res. Soc. 40 (4), 395–399.

Wolsey, L.A., 1998. Integer Programming. Vol. 52. John Wiley & Sons.

Zhang, B., Pan, Q.-k., Gao, L., Li, X.-y., Meng, L.-l., Peng, K.-k., 2019. A multiobjective evolutionary algorithm based on decomposition for hybrid flowshop green scheduling problem. Comput. Ind. Eng. 136, 325–344.

Zhang, H., Zhao, F., Sutherland, J.W., 2017. Scheduling of a single flow shop for minimal energy cost under real-time electricity pricing. J. Manuf. Sci. Eng. 139 (1).