Annals of Operations Research (2021) 296:615–637
https://doi.org/10.1007/s10479-019-03490-x

S.I.: MOPGP 2017

On a bi-criteria flow shop scheduling problem under constraints of blocking and sequence dependent setup time

Said Aqil[1] · **Karam Allali**[1]

Published online: 30 November 2019

Abstract
In this paper, we propose a bi-criteria optimization model for a flow shop scheduling problem with permutation, blocking and sequence dependent setup time. Indeed, these constraints are the most encountered in the industrial field, which demands high command flexibility. The objective is the minimization of two criteria, in our case the makespan and the total tardiness combined in a single objective function with a weighting coefficient for each criterion. To solve this problem, we propose a mixed integer linear programming method and a set of different metaheuristics. The suggested metaheuristics are; the genetic algorithm, the iterated greedy metaheuristic and the iterative local search algorithm. This last algorithm is proposed in two ways of exploration of the neighborhood. To verify the effectiveness of our resolution algorithms, a set of instances with n jobs and m machines is randomly generated from small instances to relatively large size ones. The analysis of the suggested simulation model allowed us to note that the iterative local search algorithm gives good results compared to the iterative greedy algorithm. Moreover, it was found that the weighting parameter plays an essential role in the problem decision making. However, it was established that it is difficult to find a good solution that minimizes both criteria at once, a suitable compromise will be necessary to be adopted using the weighting coefficient.

Keywords Flow shop · Sequence-dependent setup time · Blocking · Bi-criteria optimization · Mixed integer linear programming · Metaheuristic

1 Introduction

The blocking flow shop scheduling problem (BFSP) under constraint sequence-dependent setup time (SDST) is one of the most attractive problems in industrial optimization. The constraint of the blocking is often imposed when the stock is of null capacity between the machines. On the other hand, the setup time is important to be considered, for the execution

✉ Said Aqil
s_aqil@hotmail.fr

Karam Allali
allali@hotmail.com

[1] Laboratory Mathematics and Applications, University Hassan II of Casablanca, FST, P.O. Box 146, Mohammedia, Morocco

of a new job, when the machine requires a time of adjustment or cleaning etc. These two constraints are of great industrial importance that can influence the scheduling plan and therefore the proper decision-making. Hence, recent works on the study of single or multi-objective (MO) scheduling problems under one or both of these constraints have been fullfilled. For instance, Nouri and Ladhari (2018) presented a resolution approach using the non sorting genetic algorithm (NSGA) for solving a multi-objective BFSP. Also, we note that in Takano and Nagano (2017), we find a mixed integer linear programming (MILP) formulation and branch and bound (BB) procedures to minimize the makespan for solving BFSP problem under SDST constrainst. The consideration of the two both constraints of the blocking and SDST are recently studied. Indeed, the work Riahi et al. (2018) described an algorithm based on the variable neighborhood descent (VND) to minimize the mono-criterion makespan for the problem SDST/BFSP. Also, in Shao et al. (2018) the water wave optimization algorithm is used to solve the SDST/BFSP problem by minimizing the same criterion. All these last two recent works study the SDST/BFSP scheduling problem by optimizing only one criterion.

In our paper, we continue the investigation in SDST/BFSP scheduling problems by considering a MO optimization issue. The objective will be the minimization of two criteria, the makespan and the total tardiness of the jobs formulated by an objective function with a weighting coefficient for each criterion. We propose a MILP and four algorithms; the genetic algorithm (GA), the iterative greedy (IG) algorithm and the iterative local search (ILS) algorithm with two versions. The model represents therefore a bi-criteria optimization problem for a permutation flow shop scheduling problem (PFSP). This contribution is addressed to solve the MO scheduling problem with n jobs and m machines under both constraints of blocking and setup time. Indeed, these constraints are often encountered in several industrial processes, such as the food industry, the pharmaceutical industry and the petrochemical industry, etc. The review of the literature in Miyata and Nagano (2019) describes the different cases of BFSP scheduling problems. The MO optimization is becoming more and more a preference for current industrial case studies. In this perspective, MO optimization becomes a priority for most managers to solve among others the PFSP problem. We also recall that a review of multi-criteria optimization literature for the PFSP problem is described in (Sun et al. 2011; Yenisey and Yagmahan 2014). The work Xu et al. (2017) presents a MO optimization model for the SDST/PFSP problem minimizing the makespan and total weighted tardiness. Several resolution approaches are proposed to solve such problem, we distinguish three types of fundamental approaches of resolution, the exact methods, the heuristics and the metaheuristics.

Indeed, the exact methods representing the first class of resolutions attract the attention of researchers for its applied mathematical aspect. These models are intended to optimize one or more criteria defined by one or more objective functions. Yu and Seif (2016) provides MILP and a lower bound based on GA for solving a PFSP problem in order to minimize total tardiness and maintenance cost. A mathematical formulation by MILP for PFSP is described in Ta et al. (2018) to minimize the total tardiness. Also, Ronconi and Birgin (2012) presented several MILP models according to different PFSP configuration to minimize the total tardiness and precocity. Moreover, M'Hallah (2014) described a MILP model for the PFSP problem to minimize the same criterion. In (Trabelsi et al. 2012), different types of blocking are studied giving a mathematical model in the form of MILP. Besides, in (Moslehi and Khorasanian 2013) a MILP mathematical formulation and a resolution by the method of BB are proposed to solve BFSP in order to minimize the total completion time. Ronconi (2005) proposed two LB and BB algorithm for the resolution of the BFSP problem to minimize the makespan criterion. Likewise, Tellache and Boudhar (2018), Meziani et al. (2019)

presented a complexity study of the PFSP for two machines minimizing the makespan and also developing a set of LB resolution algorithms.

The second class of resolution approach is described by the heuristics based on the priority rules. We note in particular that the algorithm proposed in (Nawaz et al. 1983), is also designated in the scheduling literature by the algorithm (NEH). Later, several extented versions are derived for the PFSP multiple machines. In the work (Pan and Wang 2012), a set of heuristics based on the NEH algorithm is applied to solve the BFSP in order to minimize the makespan. As well as, in (Ribas and Companys 2015), several heuristics are proposed to solve the BFSP aiming to minimize the total flow time. We also note that in (Rossi et al. 2017) a constructive heuristics are described to minimize the total flow time for PFSP. In the recent work (Takano and Nagano 2019), a set of heuristics are also presented in order to solve SDST/BFSP problem.

The third class for PFSP resolution is based on metaheuristics whose objective is usually to improve the initial solutions already obtained by heuristics. Therefore these metaheuristics have a fundamental interest for solving problems with a larger size, especially when the two previous methods are limited. These metaheuristics are applied to solve PFSP scheduling problems under various constraints and optimizing various objectives. Applications are also extended for various case of these problems such as distributed permutation flow shop scheduling problem (DPFSP) as well as no wait permutation flowshop scheduling problem (NWPFSP). Ruiz et al. (2019) proposed IG algorithm minimizing the makespan for DPFSP, Li et al. (2018) proposed the same algorithm for solving SDST/NWPFSP with learning and forgetting effects to minimize the total flow time. Shoaardebili and Fattahi (2015) suggested algorithms based simulated annealing (SA) and NSGA algorithm for MO-PFSP in order to simultaneously minimize the total weighted flow time and the weighted sum of the earliness and tardiness under machine availability constraint. Li and Li (2015) used the ILS algorithm for MO-PFSP to minimize the makespan and the total flow time, as well as Li and Ma (2017) used the discreet artificial bee colony algorithm (ABC) for MO-SDST/PFSP to minimize the same goals. Jiang and Wang (2019) proposed the evolutionary decomposition algorithm (EDA) to minimize the makespan and the consumption energy for MO-SDST/PFSP. In the work Zhang et al. (2019), a hybrid version of the same EDA algorithm is used for solving MO-PFSP problem by minimizing the makespan and the maximum tardiness. Liu et al. (2017) developed the GA algorithm to minimize the energy consumption and tardiness penalty for MO-SDST/PFSP. Rifai et al. (2016) presented an adaptive large neighborhood search algorithm for solving the distributed and re-entrant PFSP to optimize the makespan, total cost and average tardiness. The whale optimization algorithm is used to solve the PFSP problem whose goal is to minimize the makespan (Abdel-Basset et al. 2018). Recently, Newton et al. (2019) used a new version of ILS algorithm to solve the SDST/BFSP with various constraints of blocking to minimize the makespan. In this present paper, we continue the investigations in SDST/BFSP problems by incorporating to the objective functional two criteria that are makespan and total tardiness. We develop a new approach solving problem based on a two-criteria optimization model. Our contribution is therefore to propose a MILP and a set of metaheuristics based on local research for a MO SDST/BFSP problem. To solve this kind of problem, we will develop a new implementation of the ILS algorithm. Indeed, we propose two versions of ILS with a vast exploitation of the neighborhood of the current solution. In the first version ILS$_1$, a set of solution is generated by the procedure of insertion and total shift of the elements of the solution to be explored. In the second version ILS$_2$, a set of neighboring solutions is tested using the procedure of total permutation of the elements of the current sequence. These two versions are compared with two other algorithms, in our case IG and GA. In addition, we propose a set of initialization heuristics based on the priority rules incorporated in our metaheuristics. In our resolution approach, we find that ILS algorithm in its second version gives the best results.

Our paper is organized in the following way. In Sect. 2, we present a description of the model retained for SDST/BFSP bi-objectives. Followed in Sect. 3 by a description of the proposed resolution algorithms. A comparative study is presented in Sect. 4. Section 5 concludes our contribution.

2 Problem statement

2.1 Description of the model SDST/BFSP

The flow shop scheduling problem with permutation, under sequence-dependent setup time and blocking jobs in the machine queue constraints is designated by SDST/BFSP. The complex nature of this problem with the presence of such constraints, pushes business managers to look for mathematical models of simulation for resolution. Indeed, this problem is characterized by a set of jobs $J = \{J_1, J_2, \ldots, J_n\}$ whose sequence is launched in a cycle of production constituted by a set of serially implanted machines $M = \{M_1, M_2, \ldots, M_m\}$ forming the processes of jobs production. A job J_j admits a processing time p_{jl} on the machine l, the due time d_j, this latter represents the delay time fixed by the customer for each job; any job delivered beyond this date is considered late. The setup time of the job J_k on the machine l which has just processed the job J_j is denoted by s_{jkl}, when a job is not preceded by any job on the machine l, the setup time is noted s_{jjl}. This constraint is referred in the scheduling problem literature as the sequence dependent setup time (SDST). Another constraint is present in our study, it deals with the blocking of a J_j during its treatment on all concerned machines, knowing that it will be blocked in its machine until the next machine will be available. The departure date of the job J_j on the machine l is designated by D_{jl}. This date represents also the start date of processing on the next machine $l + 1$. A sequence $\pi = (\pi_1, \ldots, \pi_n)$ of jobs is processed in the set of machines in the same order, the retained assumptions are those of the classic case of the PFSP problems. Knowing that a machine processes only one job at a given time, the interruption is not allowed, a job J_j has more than one successor and a single predecessor, all the jobs and all the machines are available at the initial moment. Blocking jobs on machines and their setup times will be considered in the used model of this study. The objective is the minimization of two criteria in this case the makesapn and the total tardiness.

We propose a model formed by a combination of two criteria in a single objective function with a weighting coefficient for each criterion. It is worthy to notice that the adopted notation for the scheduling problem is usually represented by the three fields $\alpha|\beta|\gamma$, where the first field α designates the type of problem, the second one β stands for the different constraints imposed to the scheduling problem and the third field γ represents the nature of the objective function. In our study, the scheduling problem will be noted by $Fm|prmu, SDST, blocking|\mu C_{max} + (1-\mu)\sum_{j=1}^{n} T_j$, such that the parameter $\mu \in [0, 1]$ allows to attribute weight to each criterion. Another virtual machine is added at the beginning to all the machines. Similarly, a noted virtual job π_0 is inserted at the beginning of the sequence, whose processing time on the all machine is zero and the setup times with the other jobs in each machine is equal to the setup times s_{jjl} of the job J_j when it is not preceded by any job. Here, we give a set of equations to determine the parameters needed for the departure date of each job on each machine.

$$s_{\pi_0\pi_j l} = s_{\pi_j\pi_j l}, \quad l = 1, \ldots, m, \quad j = 1, \ldots, n \tag{1}$$

$$D_{\pi_0 l} = 0, \quad l = 1, \ldots, m \tag{2}$$

$$D_{\pi_j 0} = D_{\pi_{(j-1)}1} + s_{\pi_{(j-1)}\pi_j 1}, \quad j = 1, \dots, n \tag{3}$$

$$D_{\pi_j l} = max\{D_{\pi_{(j-1)}(l+1)} + s_{\pi_{(j-1)}\pi_j (l+1)}, D_{\pi_j (l-1)} + p_{\pi_j l}\}$$
$$j = 1, \dots, n, \quad l = 1, \dots, m-1 \tag{4}$$

$$D_{\pi_j m} = D_{\pi_j (m-1)} + p_{\pi_j m}, \quad j = 1, \dots, n \tag{5}$$

The first Eq. (1) makes it possible to identify the characteristics of the virtual job π_0 by assigning it the duration of dedicated setup times. The other Eqs. (2)–(5) allow to determine the departure dates of each job on each machine including the added virtual machine at the beginning of the scheduling. Our scheduling model consists of combining two criteria into a single objective function to minimize. The first goal is to minimize the makespan, knowing that the end date of a job on the last machine m is equal to its departure date on this machine in this case $D_{\pi_j m}$, the makespan will be expressed by:

$$C_{max} = max\{D_{\pi_j m}\}, \quad j = 1, \dots, n \tag{6}$$

The second considered objective in this study is to minimize the total tardiness of all jobs. For that, we calculate the tardiness of each job by $T_{\pi_j} = max(0, D_{\pi_j m} - d_{\pi_j})$, the total tardiness will be expressed by:

$$TT = \sum_{j=1}^{n} T_{\pi_j}, \quad j = 1, \dots, n \tag{7}$$

The main objective functional is therefore written in the form of a combination of these two criteria, the first is the makespan (C_{max}) and the second is the total tardiness (TT)

$$min \quad f = \mu C_{max} + (1-\mu)TT \tag{8}$$

Thus, we give a set of equations allowing the determination of the necessary parameters to evaluate the values of the two criteria fixed for the bi-objective function. Our goal is then to determine an optimal sequence that minimizes the objective function as much as possible across all permutations Π during the exploration of the entire neighborhood such as

$$f(\pi_*) \le f(\pi), \quad \pi \in \Pi \tag{9}$$

2.2 Numerical illustration

In order to highlight the proposed model, we present an illustrative example which consider an instance consisting of three jobs and three machines forming our workshop. We represent the processing times of the jobs on the machines by the matrix $[p_{jl}]$ and the inter-job setup times in the different machines by the matrix $[s_{jk1}]$, $[s_{jk2}]$ and $[s_{jk3}]$. The due dates will be expressed for each job such that $d_1 = 17, \ d_2 = 14, \ d_3 = 10$.

$$[p_{jl}] = \begin{bmatrix} 1 & 3 & 3 \\ 1 & 4 & 2 \\ 3 & 3 & 4 \end{bmatrix}, \quad [s_{jk1}] = \begin{bmatrix} 1 & 1 & 2 \\ 4 & 3 & 1 \\ 2 & 4 & 1 \end{bmatrix}, \quad [s_{jk2}] = \begin{bmatrix} 2 & 3 & 2 \\ 2 & 2 & 3 \\ 2 & 3 & 3 \end{bmatrix}, \quad [s_{jk3}] = \begin{bmatrix} 1 & 2 & 2 \\ 2 & 2 & 1 \\ 1 & 2 & 2 \end{bmatrix}$$

We want to schedule the sequence $\pi = (3 \ 2 \ 1)$, the first step is to determine the setup times of the virtual job π_0 on the different machines, knowing that this job will precede all the other jobs in the built sequence, according to Eq. (1), knowing that $\pi_1 = 3, \ \pi_2 = 2, \ \pi_3 = 1$, we can write: $s_{\pi_0\pi_1 1} = 1, \ s_{\pi_0\pi_2 1} = 3, \ s_{\pi_0\pi_3 1} = 1, \ s_{\pi_0\pi_1 2} = 3, \ s_{\pi_0\pi_2 2} = 2, \ s_{\pi_0\pi_3 2} = 2, \ s_{\pi_0\pi_1 3} = 2, \ s_{\pi_0\pi_2 3} = 2, \ s_{\pi_0\pi_3 3} = 1$. We note that Eq. (2) allows the initialization of computation parameters, so we can write: $D_{\pi_0 1} = 0, \ D_{\pi_0 2} = 0, \ D_{\pi_0 3} = 0$.

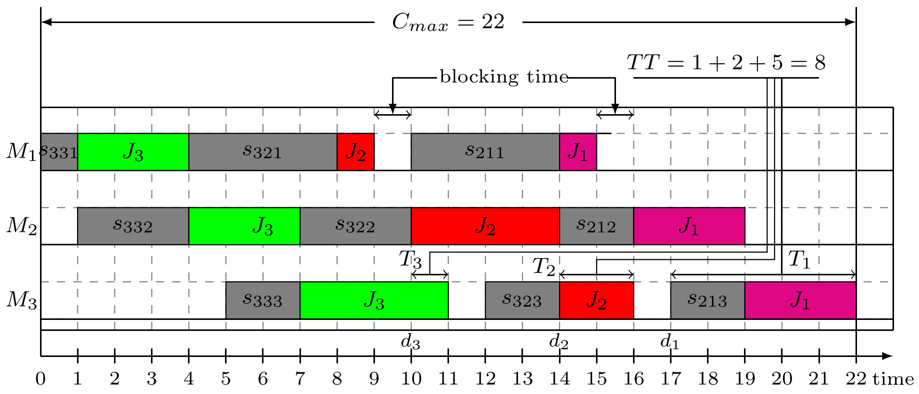

Fig. 1 Gantt chart of three jobs on three machines

With the other Eqs. (3)–(5), we give in detail the departure date of each job of all the machines of our sequence by the following expressions: $D_{\pi_1 0} = D_{\pi_0 1} + s_{\pi_0\pi_1 1} = 0+1 = 1$, $D_{\pi_1 1} = max\{D_{\pi_0 2} + s_{\pi_0\pi_1 2}, D_{\pi_1 0} + p_{\pi_1 1}\} = max\{0+3, 1+3\} = 4$, $D_{\pi_1 2} = max\{D_{\pi_0 3} + s_{\pi_0\pi_1 3}, D_{\pi_1 1} + p_{\pi_1 2}\} = max\{0+2, 4+3\} = 7$, $D_{\pi_1 3} = D_{\pi_1 2} + p_{\pi_1 3} = 7+4 = 11$; $D_{\pi_2 0} = D_{\pi_1 1} + s_{\pi_1\pi_2 1} = 4+4 = 8$, $D_{\pi_2 1} = max\{D_{\pi_1 2} + s_{\pi_1\pi_2 2}, D_{\pi_2 0} + p_{\pi_2 1}\} = max\{7+3, 8+1\} = 10$, $D_{\pi_2 2} = max\{D_{\pi_1 3} + s_{\pi_1\pi_2 3}, D_{\pi_2 1} + p_{\pi_2 2}\} = max\{11+2, 10+4\} = 14$, $D_{\pi_2 3} = D_{\pi_2 2} + p_{\pi_2 3} = 14+2 = 16$; $D_{\pi_3 0} = D_{\pi_2 1} + s_{\pi_2\pi_3 1} = 10+4 = 14$, $D_{\pi_3 1} = max\{D_{\pi_2 2} + s_{\pi_2\pi_3 2}, D_{\pi_3 0} + p_{\pi_3 1}\} = max\{14+2, 14+1\} = 16$, $D_{\pi_3 2} = max\{D_{\pi_2 3} + s_{\pi_2\pi_3 3}, D_{\pi_3 1} + p_{\pi_3 2}\} = max\{16+2, 16+3\} = 19$, $D_{\pi_3 3} = D_{\pi_3 2} + p_{\pi_3 3} = 19+3 = 22$.

We can deduce the end dates of the three jobs on the last machine, knowing that $D_{\pi_1 3} = 11$, $D_{\pi_2 3} = 16$, $D_{\pi_3 3} = 22$ as well as the value of the makespan $C_{max} = 22$. For the tardiness of the jobs during the scheduling, it is calculated taking into account the due date imposed for each job, calculating the tardiness of each job, knowing that, $T_3 = 1$, $T_2 = 2$, $T_1 = 5$, the total tardiness is $TT = 1+2+5 = 8$, with a value of $\mu = 0.5$ our objective function is written so $f = 0.5 \times 22 + 0.5 \times 8 = 15$. The Gantt chart illustrated in Fig. 1 represents the result of the scheduling in the three machines of the studied example with the blocking and setup dependent time constraints.

3 Resolution methods

We propose two approaches of resolutions for the SDST/BFSP bi-objective scheduling problem. The first is based on a mathematical formulation by a MILP, the second is constituted by a set of heuristics and metaheuristics. For the MILP an objective function is given by a combination of the two criteria and a set of constraints must be satisfied. The second approach is composed of IG, GA and ILS with two versions, intended to solve the problems of relatively large sizes, all the metaheuristics will be initialized by a set of heuristics based on the NEH algorithm.

3.1 Mixed integer linear programming model

We propose here a mathematical formulation in MILP of our bi-objective problem. We give a set of parameters and variables necessary for our model.

$$z_{kq} = \begin{cases} 1, & \text{if job } k \text{ is at the } q\text{-th position in the sequence} \\ 0, & \text{otherwise} \end{cases}$$

C_{ql}: the completion time of the q-th position job at machine l.
D_{ql}: the departure date of the q-th position job at machine l.

$$x_{jkq} = \begin{cases} 1, & \text{if the job } j \text{ precedes the job } k \text{ which is at the } q\text{-th position} \\ & \text{ in the sequence} \\ 0, & \text{otherwise} \end{cases}$$

$Cmax$: Makespan
T_q: Tardiness of q-th position job

$$Minmize\ f = \mu C_{max} + (1-\mu)\sum_{q=1}^{n} T_q \tag{10}$$

Subject to

$$\sum_{k=1}^{n} z_{kq} = 1,\ q = 1,\ldots,n \tag{11}$$

$$\sum_{q=1}^{n} z_{kq} = 1,\ k = 1,\ldots,n \tag{12}$$

$$\sum_{j=1}^{n}\sum_{k=1}^{n} x_{jk1} = 0 \tag{13}$$

$$\sum_{j=1}^{n}\sum_{k=1}^{n} x_{jkq} = 1,\ q = 2,\ldots,n \tag{14}$$

$$x_{jkq} \geq z_{kq} + z_{j(q-1)} - 1,\ j = 1,\ldots,n,\ k = 1,\ldots,n,\ q = 2,\ldots,n, (k \neq j) \tag{15}$$

$$C_{1l} = \sum_{k=1}^{n} s_{kkl} \times z_{k1},\ l = 1,\ldots,m \tag{16}$$

$$C_{ql} = D_{(q-1)l} + \sum_{j=1}^{n}\sum_{k=1}^{n} s_{jkl} \times x_{jkq},\ q = 2,\ldots,n,\ l = 1,\ldots,m \tag{17}$$

$$D_{ql} \geq C_{q(l+1)},\ q = 1,\ldots,n,\ l = 1,\ldots,m-1 \tag{18}$$

$$D_{q1} \geq C_{q1} + \sum_{j=1}^{n} p_{j1} \times z_{jq},\ q = 1,\ldots,n \tag{19}$$

$$D_{ql} \geq C_{q(l-1)} + \sum_{k=1}^{n} p_{kl} \times z_{kq},\ q = 1,\ldots,n,\ l = 2,\ldots,m \tag{20}$$

$$T_q \geq D_{qm} - \sum_{k=1}^{n} z_{kq} \times d_k,\ k = 1,\ldots,n \tag{21}$$

$$C_{max} \geq D_{nm} \tag{22}$$

$$T_q \geq 0,\ q = 1,\ldots,n \tag{23}$$

$$C_{ql},\ D_{ql} \geq 0,\ q = 1, \ldots, n,\ l = 1, \ldots, m \tag{24}$$

$$z_{kq} \in \{0, 1\},\ x_{jkq} \in \{0, 1\},\ j = 1, \ldots, n,\ k = 1, \ldots, n,\ q = 1, \ldots, n \tag{25}$$

Equation (10) expresses the two-criteria objective function the makespan and the total tardiness, each having a weighting coefficient. The constraints (11) and (12) make it possible to express that each job occupies a single position and each position is occupied by a single job in the sequence. Constraint (13) expresses that no job precedes the job of the first position in the sequence. The constraint (14) guarantees that only one job will be assigned to the position q in the sequence with a single predecessor except that of the first position. In the constraint (15), for each job k occupying the position q in the sequence is sequenced after the job j occupying the position $q - 1$ except for that of the first position. The constraint (16) relates to the first job of the sequence where its completion time of setup depends essentially on setup time s_{kkl} on the machine l. The constraint (17) makes it possible to calculate the completion time of setup of the jobs in the other machines. In addition this is expressed by the summation of the job departure date of the $(q - 1)$ position and the setup time with the successor job. The constraints of (18)–(20) make it possible to calculate the departures date of the jobs in the machines by considering the constraint of the blocking. The constraint (18) is applied for all the jobs in the first machine, the constraint (19) is applied for all the jobs in the other machines except the last, the constraint (20) guarantees that the departure date of the jobs of the machine l is greater than or equal to the starting date of the machine $(l - 1)$ plus its processing time at the machine l. The constraint (21) expresses that the tardiness of a job is greater than or equal to the difference of the departure date in the last machine and the imposed date due. The constraint (22) expresses that the makespan is greater than or equal to the departure date of the last job in the last machine. The constraints (23) and (24) make it possible to define the domain of variation of the parameters of the model.

3.2 Initialization heuristics

The most important phase in solving metaheuristic optimization problem is the determination of the initial solution. Indeed for our bi-objective minimization problem of SDST/BFSP, we have chosen six rules to build the initial solutions. We propose an adaptation of the NEH algorithm to improve the initial solutions by starting with the sequence obtained by each rule. Here, we give the characteristics of six rules by detailing the properties of each rule.

R_1 : The first rule is based on Johnson's algorithm (Johnson 1954), it consists in forming two virtual machines $M^{(1)}$ and $M^{(2)}$, dividing the number of machines into two sub assemblies whose respective number of machines is m_1 and $m - m_1$, knowing that $m_1 = \lfloor m/2 \rfloor$, where $\lfloor \cdot \rfloor$ denotes the floor function; the processing times on the two machines are calculated by the following expressions

$$\begin{cases} p_j^{(1)} = \sum_{l=1}^{m_1} (p_{jl} + s_{jjl})\ \ processing\ \ time\ \ in\ \ M^{(1)} \\ \\ p_j^{(2)} = \sum_{l=m_1+1}^{m} (p_{jl} + s_{jjl})\ \ processing\ \ time\ \ in\ \ M^{(2)} \end{cases} \tag{26}$$

To form the sequence of Johnson, one will determine the subsets $U = \{J_j,\ p_j^{(1)} < p_j^{(2)}\}$ and $V = \{Jj,\ p_j^{(1)} \geq p_j^{(2)}\}$, then will classify the jobs of U in increasing order of $p^{(1)}$

on $M^{(1)}$ and the jobs of V in decreasing order of $p^{(2)}$ on $M^{(2)}$. The Johnson sequence is formed by the concatenation of ordered sub sequences UV.

R_2 : The second rule is based on the notion of the longest processing time of job on all machines in flow shop, also called (LPT) by calculating the following expression

$$\zeta_j^{(1)} = \sum_{l=1}^{m}(p_{jl} + s_{jjl}), \quad j = 1, \dots, n \tag{27}$$

An initial sequence will be determined and the jobs ranked in descending order of $\zeta_j^{(1)}$.

R_3 : The third rule is based on the rule inspired by the model described in Rajendran and Ziegler (1997) that we adopt for our problem by calculating the following expression

$$\zeta_j^{(2)} = \sum_{l=1}^{m}(m - l + 1)(p_{jl} + s_{jjl}), \quad j = 1, \dots, n \tag{28}$$

An initial sequence is obtained by classifying the jobs in descending order of $\zeta_j^{(2)}$.

R_4 : The fourth rule is to determine the following expression

$$\zeta_j^{(3)} = \left(\frac{2}{m-1}\right) \times \sum_{l=1}^{m}(m - l)(p_{jl} + s_{jjl}) + \sum_{l=1}^{m}(p_{jl} + s_{jjl}) \\ j = 1, \dots, n \tag{29}$$

It is inspired by the model treated in (Tasgetiren et al. 2017), it also consists in also arranging the jobs in descending order of $\zeta_j^{(3)}$, to get the sequence for this rule.

R_5 : The fifth rule is based on the due date, it consists in classifying the jobs in ascending order of their due date d_j.

R_6 : The sixth rule and the last one consists in determining the expected advance by calculating for each job the following expression

$$\zeta_j^{(4)} = max(0, d_j - \sum_{l=1}^{m}(p_{jl} + s_{jjl})), \quad j = 1, \dots, n \tag{30}$$

An initial sequence will be obtained by classifying the jobs in ascending order of $\zeta_j^{(4)}$.

For the six rules mentioned above, we used the processing times, the setup times, the due dates as well as the number of jobs and the number of machines characterizing our problem. The improvement of these initial solutions is ensured by a adaptive heuristic based on the constructive algorithm NEH which we write *NEH_R*$_i$, relating respectively to the rules R_i, $i = 1, \dots, 6$. Our heuristic will be described by the Algorithm 1.

The principle of this heuristic is based on two phases, the first phase consists in breaking down the sequence π into two sub-sequences $\pi = [\pi^{(1)}\ \pi^{(2)}]$, with $\pi^(1) = (\pi_1^{(1)}, \pi_2^{(1)}, \dots, \pi_m^{(1)})$ and $\pi^{(2)} = (\pi_{m+1}^{(2)}, \pi_{m+2}^{(2)}, \dots, \pi_n^{(2)})$ under the hypothesis $m < n$ that we consider satisfied for all our simulated instances. The second phase consists of extracting each job $\pi_k^{(2)}$ from the sub-sequence $\pi^{(2)}$ and reinserting it into the different positions of the constructed sub-sequence $\pi = [\pi^{(1)}\ \pi_k^{(2)}]$ and retaining the best sequence among all the sequences generated by insertion. This process will be repeated until all jobs are completed. This principle of decomposition and insertion will be repeated until satisfying a stop criterion defined as a limit iteration number (N_limite). This limit number will be determined according to the size of the studied instance. We choose $N\_limite = 30$ for small and medium size instances and $N\_limite = 50$ for relatively large size instances.

Algorithm 1 *NEH_R_i* algorithm.

```
Input: $\pi_{best} \leftarrow \pi = \{\pi_1, \ldots, \pi_n\}$ obtained by the rule $R_i$
  $iter = 1$
  while $\{iter < N\_limite\}$ do
    Extract randomly $m$ jobs and divide the $\pi$ into two sub-sequences $\pi = [\pi^{(1)}\ \pi^{(2)}]$
    for $k = 1$ to $n - m$ do
      Extract a job $\pi_k^{(2)}$ from the sub-sequence $\pi^{(2)}$ choose the best $\pi_{NEH}$ sequence from all obtained
      sub-sequences by evaluation its objective function value
      if $f(\pi_{NEH}) < f(\pi_{best})$ then
        $\pi_{best} \leftarrow \pi_{NEH}$
      end if
    end for
    $iter = iter + 1$
  end while
Output: $\pi_{best}$
```

For the bi-objective optimization, we note that for the choice of the right solution among all the solutions obtained by the two criteria, will be based on the combination of the two objectives in one that is written in the form of the Eq. (8). We note that in this model the coefficient $\mu \in [0, 1]$ is the weighting factor of the criterion chosen by the decision maker when establishing the scheduling strategy. It will be considered variable to give flexibility to the production manager at the level of each batch launched in the production cycle.

3.3 Iterative local search

The first algorithm proposed for solving our problem is the iterative local search algorithm for the bi-objective minimization of the SDST/BFSP. This type of algorithm has been adopted for the resolution of several types of scheduling problems, in particular for PFSP in (Dong et al. 2009, 2013; Xu et al. 2017). The principle of this algorithm is simple, it consists of defining a solution in the neighborhood of the current solution and comparing the objective function value with the current best value of this function. The principle is repeated until it satisfies a stop condition. In our approach, we consider a method which consists in the first phase of defining the set of neighboring solutions $\Gamma^{(i)},\ i \in \{1, 2\}$. Where $\Gamma^{(i)}$ denotes the set of solutions generated in the perturbation procedure ($Proced\_pert$), by total insertion ($i = 1$) or total permutation ($i = 2$), respectively. In the second phase, we choose among the generated solutions the best one, then comparing it with the best current solution. The process will be repeated until the imposed stop will be satisfied.

The description of the steps of this algorithm is described in the Algorithm 2. We consider two types of neighborhoods, the first one is based on the total insertion and the shift on the left and on the right, the second type of neighborhood is based on the total permutation of a job with all the other jobs without repetitions. In the last phase of the algorithm, we highlight the model simulated annealing to better explore the set of solutions. We distinguish in this algorithm two procedures for generating the set of neighborhoods called $Proced\_pert$. This procedure generates the neighborhood by involving the total exploration of the neighborhood of the current solution with two different ways ILS$_1$ and ILS$_2$:

We consider two types of neighborhoods, the first one is based on the total insertion and the shift on the left and on the right, the second type of neighborhood is based on the total permutation of a job with all the other jobs without repetitions. In the last phase of the algorithm, we highlight the model simulated annealing to better explore the set of solutions.

Algorithm 2 Iterative local search.

```
Input: $\pi = \{\pi_1, \dots, \pi_n\}$ the initial sequence
    $\pi^* \longleftarrow \pi.$
  while {the stopping criterion is not satisfied} do
    Generate a sequence set $\Gamma^{(i)}$ by $Proced\_pert(\pi^*)$
    $f(\pi_p^*) = \min_{\pi_p \in \Gamma^{(i)}} \{f(\pi_p)\}$
    if $f(\pi_p^*) < f(\pi^*)$ then
      $\pi^* \leftarrow \pi_p^*$
      $f(\pi^*) \leftarrow f(\pi_p^*)$
    else
      if $rand() \leq exp(-(f(\pi_p^*) - f(\pi^*))/\mathcal{T})$ then
        $\pi^* \leftarrow \pi_p^*$
      end if
    end if
  end while
Output: $\pi^*$, $f(\pi^*)$
```

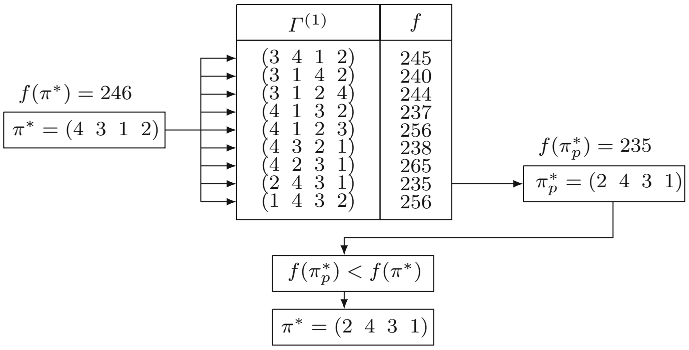

Fig. 2 An example illustrating the total insertion procedure $Proced\_pert$

We distinguish in this algorithm two procedures for generating the set of neighborhoods called $Proced\_pert$. This procedure generates the neighborhood by involving the total exploration of the neighborhood of the current solution with two different ways ILS$_1$ and ILS$_2$:

- Procedure $Proced\_pert$ for ILS$_1$: The total insertion disruption procedure is used in the first version of the iterative local search algorithm designated by ILS$_1$.
  Figure 2 shows an implementation example of the total insertion procedure $Proced\_pert$, here we give an example of 4 jobs, starting from a current best sequence $\pi^* = (\ 4\ 3\ 1\ 2)$ which $f(\pi^*) = 246$, knowing that we generating the entire neighborhood by total insertion, we obtain the best sequence $\pi_p^* = (\ 2\ 4\ 3\ 1)$ which $f(\pi_p^*) = 235$, we compare the objective function value of this sequence with the objective function value of the best current sequence. Finally, we get our new best sequence $\pi^* = (\ 2\ 4\ 3\ 1)$. We note that for a sequence of n jobs, we will have $(n-1)^2$ neighboring solutions representing the cardinal of $\Gamma^{(1)}$.
- Procedure $Proced\_pert$ for ILS$_2$: The total permutation perturbation procedure is used in the second version of the iterative local search algorithm noted ILS$_2$. In this procedure, the total number of permutations without repetition between the different jobs is generated to obtain the set of neighboring solutions noted $\Gamma^{(2)}$.

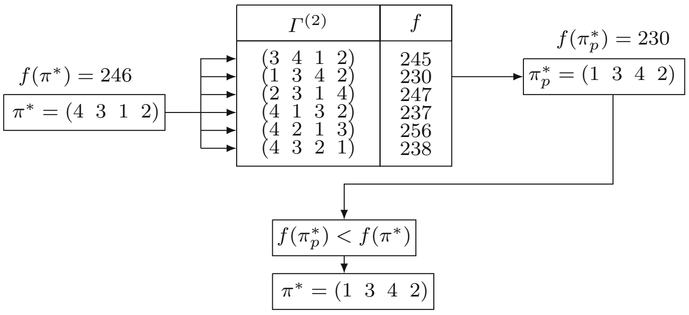

Fig. 3 An example illustrating the total permutation procedure *Proced_pert*

Figure 3 shows an implementation example of the total permutation procedure *Proced_pert*, here we give an example of 4 jobs, starting from the same best sequence $\pi^* = (4\ 3\ 1\ 2)$ which $f(\pi^*) = 244$ as before and by generating the whole of the total permutation neighborhood, we obtain the best sequence $\pi_p^* = (1\ 3\ 4\ 2)$, knowing that $f(\pi_p^*) = 230$. We compare the objective function value of this sequence with the best objective function value of the current sequence, that we find is lower. We obtain our new best sequence $\pi^* = (1\ 3\ 4\ 2)$ during the exploration of this neighborhood. Finally, we note that for a sequence of n jobs, we will have $\frac{n\times(n-1)}{2}$ neighboring solutions representing the cardinal of $\Gamma^{(2)}$.

In this model, the acceptance criterion of the simulated annealing with the parameter $\mathcal{T}$ is also integrated, which depends on the nature and the characteristics of the problem. We give an expression inspired by the model in (Ruiz and Stützle 2007) expressed for several types of flow shop scheduling problems; this expression is given by the following relation

$$\mathcal{T} = \lambda \times \frac{\sum_{j=1}^{n}\sum_{l=1}^{m}(p_{jl} + s_{jjl})}{10 \times m \times n} \tag{31}$$

We use the parameters of the problem, especially the processing times, the setup times, the number of jobs and the number of machines. In addition, for a right calibration of our algorithm, the coefficient λ is been varied adequately from 0.5 to 0.9.

3.4 Iterative greedy algorithm

The second proposed algorithm in our approach of resolution is the iterative greedy algorithm (Ying 2008; Pan and Ruiz 2014; Fernandez-Viagas et al. 2018), which we apply with its traditional version. The principle of this algorithm is based on two main phases, the exploration phase of the current neighborhood solution and the local search phase. The exploration phase of the current neighborhood solution is constructed of two sub-phases, the destruction and the construction of the neighboring solution. In the sub-phase of destruction a subset of job is extracted, in the second sub-phase each job of the extracted set is inserted in the different positions of the partial built solution, and we retain the best solution among all the generated solutions. In the local search phase, we use the same model presented in the local search iterative algorithm for setting the temperature parameter of the simulated annealing model. This metaheuristic is one of the most used in several scheduling problems. We apply it to solve our bi-objective problem whose parameters are adopted for our case study. We present in the Algorithm 3 the detailed steps of determination of the solution obtained by the iterative greedy algorithm.

Algorithm 3 Iterated greedy algorithm with local search.

```
Input: A sequence $\pi$ obtained by an initialization heuristic
  $\pi^* \leftarrow \pi$
  while $\{unsatisfied\ stopping\ criterion\}$ do
    % Destruction phase%
    $\pi' \leftarrow \pi$
    for $h = 1$ to $d$ do
      Extract a random job $\pi_h$ from the sequence $\pi'$ and add the job $\pi_h$ to a subset $\Omega$
    end for
    % Construction phase%
    for $h = 1$ to $d$ do
      Extract the job $\pi'_h$ from the subset $\Omega$
      Test the job on the different positions in the current sequence $\pi'$ and choose the best position giving
      the smallest objective functional value
    end for
    % local search phase%
    Choose $\pi''$ from the neighborhood of $N_k(\pi')$
    if $f(\pi'') < f(\pi)$ then
      $\pi \leftarrow \pi''$
      if $f(\pi) < f(\pi^*)$ then
        $\pi^* \leftarrow \pi$
      end if
    else
      if $random \leq exp\{-(f(\pi'') - f(\pi))/\mathcal{T}\}$ then
        $\pi \leftarrow \pi''$
      end if
    end if
  end while
Output: $\pi^*$
```

The implementation of this metaheuristic requires two basic settings parameters, the parameter T and the parameter d. For the parameter T, we consider the same adopted model for the iterative local search algorithm which allows a good exploration of the neighborhood of the current solution. For the parameter d, which represents the number of jobs to be extracted in the destruction phase. We propose a simulation study by varying the number of jobs to be extracted in the destruction sub-phase by considering $d = m$ for small and medium instances whose number of jobs $n \in \{20, 30, \ldots, 90\}$ and the number of machines $m \in \{5, 10\}$, while for large size instances, we consider $n \in \{100, 150, \ldots, 400\}$ and the number of machines $m \in \{10, 20\}$, we assume that $d \in \{10, 20, \ldots, 50\}$.
In this case, for the chosen neighborhood system, we propose three types of neighborhoods

- The $N_1(\pi)$ random neighborhood permutation: A randomly chosen job swapping position with another randomly selected job in the current sequence to obtain a new job sequence.
- The $N_2(\pi)$ neighborhood by block inversion: A block of job is chosen in the current sequence and all the jobs of this block invert their positions to obtain the new sequence
- The $N_3(\pi)$ shift neighborhood with insertion: A job is selected at random from position p is inserted into a q position drawn randomly in the current sequence, all jobs p and q will be shifted to the left if $p < q$ or right otherwise.

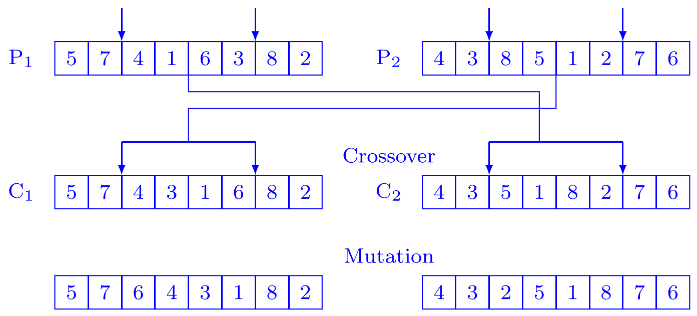

Fig. 4 Crossover and mutation example

One of the fundamental steps for metaheuristics is the initialization phase, a good starting solution that saves a lot of time in finding the right solution. Our approach first allows us to find an initial sequence using one of the proposed rules, then the second phase is to apply the NEH algorithm which will give the initial solution for our two metaheuristics of resolution which we compare in terms of quality of the founded solution and the speed of convergence to the good optimal solution.

3.5 Genetic algorithm

The metaheuristics based on the genetic algorithm is among the most implemented metaheuristics for scheduling optimization problems. Several versions are applied for mono objective or multi-objective optimization. The Algorithm 4 gives a description of the different stages of the genetic algorithm evolution.

Algorithm 4 Genetic algorithm.

```
Generate initial population P using the initialization heuristics
Evaluate the individuals of P to determine the best solution: $\pi_{best}$
while {$unsatisfied\ stopping\ criterion$} do
    Randomly select two parents $P_1$, $P_2$ from the population
    Generate two children $C_1$, $C_2$ by the crossing procedure
    Apply the mutation procedure to $C_1$ and $C_2$
    Evaluate the set of generated individuals
    Update the population P and deduce $\pi_{best}$
end while
```

The main phases of this metaheuristics are, selection, crossing and mutation. In this algorithm we simulate the behavior of evolution of the genetic code in the human body to arrive at the proper code to each individual. In the Fig. 4, from an initial population, in the selection phase two parents P_1, P_2 are selected according to a previously defined procedure, subsequently in the crossing phase two children C_1, C_2 will be generated and each will take part of the code of both parents form its own genetic code. In the mutation phase, a procedure based on local search insertion methods is applied to new individuals. At the end an evaluation of all the individuals is carried out by the objective function and an update of the population is carried out, the process is repeated until satisfying the stopping criterion.

4 Effectiveness of resolution algorithms

4.1 Simulation instances

The most important step in the simulation study by computer tools is the comparative study between different solving approaches. We highlight in our approach a set of heuristics and metaheuristics for the resolution of the bi-objective SDST/BFSP scheduling problem. For this we study, a set of instances that we have classified into two size families, the family of small/medium size and the family of the large one. The classification is based on the number of jobs and the number of machines for each family. For small and medium instances that we consider, the number of machines $m \in \{2, \ldots, 10\}$ and the number of jobs $n \in \{10, \ldots, 90\}$; the processing times p_{jl} are defined in $[1, 49]$ and the setup times s_{jkl} are defined in $[1, 10]$. For the family of the large size, the number of machines is $m \in \{10, \ldots, 20\}$ and the number of jobs $n \in \{100, 150, \ldots, 400\}$; we consider that the processing times p_{jl} belong to $[50, 99]$ and the setup times s_{jk1} belong to $[10, 20]$. We note that the setup times are much lower than the processing times by not exceeding 25%, which actually reflects industrial cases. The due dates for the two families of instances are expressed as function of p_{jl} and s_{jkl} defined by the expression

$$d_j = \delta \times \left[\sum_{l=1}^{m} (p_{jl} + s_{jjl})(1 + rand()) \right], \quad j = 1, \ldots, n \tag{32}$$

where δ depends on the studied instance family, such that $\delta \in \{2, \ldots, 5\}$. The studied cases cover the maximum of industrial flow shop production types ranging from small size instances to larger ones. A comparative study between the different resolution heuristics and metaheuristics makes it possible to conclude the right approach that we can apply for our problem. This study concerns the two families of instances by varying the necessary parameters for each resolution algorithm. To evaluate their efficiency, we calculate the relative percentage increase (RPI) by the following formula

$$RPI = \left(\frac{f - f_{min}}{f_{min}} \right) \times 100 \tag{33}$$

This type of analysis requires the evaluation of several instances, to be able to express a good judgment on the effectiveness of the resolution methods. A good calibration to measure the desired performances is necessary, one carries out a set of tests and for each test one retains the average of 10 instances for each test problem. A statistical study is essential tool in this kind of study to validate the correct approach of resolution among all the proposed approaches.

4.2 Comparative study between metaheuristics and MILP

We made a comparative study between the four metaheuristics improved by the NEH algorithm with the exact solution obtained by the MILP. The study is carried out for small instances by varying the weighting parameter μ for the two criteria of the objective function. We tested four instances, with eleven cases for each problem depending on the value of μ, with total of 44 problems. In the Table 1, we give the results of the simulation of the study in terms of the objective function value.

We can see that the ILS_2 algorithm records the best result with a success rate of 90.9% compared to the other three algorithms. In addition, the ILS algorithm with these two versions

Table 1 Objective function value of small instances

$n \times m$	μ										
	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
10×3											
MILP	221.3	236.5	252.3	267.5	283.6	298.5	314.4	329.5	342.3	351.8	356.8
ILS_1	238.4	240.5	260.5	280.3	300.4	315.6	323.4	340.2	352.2	360.3	380.4
ILS_2	233.5	240.5	252.8	267.5	300.5	298.5	320.5	329.5	342.3	358.8	370.4
IG	238.5	255.2	260.4	290.5	308.5	308.5	328.7	351.2	347.8	362.5	385.5
GA	238.7	255.6	260.5	295.3	308.5	317.5	323.4	336.6	347.8	367.5	380.3
10×5											
MILP	256.2	278.2	297.4	316.6	335.8	355.6	374.5	393.3	411.6	429.3	443.9
ILS_1	268.7	292.5	312.6	320.4	342.2	365.8	380.7	402.2	422.6	438.5	460.5
ILS_2	260.4	287.4	312.6	316.6	340.4	360.4	380.5	407.4	418.4	436.2	458.2
IG	280.4	290.3	325.2	324.8	352.2	370.5	390.8	412.2	422.6	442.8	470.7
GA	280.3	290.8	333.4	322.6	348.8	365.7	390.5	208.2	422.6	448.2	466.5
15×3											
MILP	128.2	190.3	268.6	291.9	327.2	355.5	415.6	452.9	468.4	512.2	552.3
ILS_1	147.6	206.4	270.3	295.7	331.8	360.3	422.3	467.7	482.3	520.4	560.2
ILS_2	138.3	208.5	268.6	295.7	329.5	358.2	420.5	465.2	480.2	516.8	556.4
IG	145.4	215.5	282.1	299.2	335.2	360.3	435.5	477.2	485.6	520.4	570.1
GA	151.5	217.2	288.3	299.2	335.2	360.3	436.2	480.3	490.3	520.4	570.8
15×5											
MILP	280.8	342.5	348.6	400.5	406.6	448.5	471.8	498.7	547.8	553.4	576.8
ILS_1	284.3	350.3	362.8	410.4	413.3	456.6	480.2	510.7	560.3	560.6	588.9
ILS_2	282.5	346.5	353.4	408.8	412.8	454.2	476.8	508.3	556.7	564.2	580.4
IG	290.8	356.4	358.5	412.3	422.2	460.2	482.4	510.4	561.5	568.3	592.5
GA	288.7	344.5	358.6	416.2	422.2	458.6	480.2	512.5	570.8	572.4	597.3

dominates the other two algorithms. We note that the exact solution has been implemented under the software LINGO 17.3 and that the result is given for the average five instance tested. We limit ourselves to representing the results of a set of small instances including $n \in \{10, 15\}$ and $m \in \{3, 5\}$. We note that the simulation algorithms are implemented using the Matlab 2014a language and realized on PC with the Intel (R) Core (TM) i3 CPU, M350, 2.27GHz, 2.26 GHz and a 6 GB RAM.

4.3 Comparison of heuristics

We present here the second analysis concerning the comparative study between the different heuristics based on the NEH algorithm and on the rules mentioned in the initialization phase. This study shows the difference recorded between the different rules in order to verify the findings retained in the carried out analysis by the metaheuristics. The analysis will be done by calculating the RPI term according to the variation of the weighting criteria coefficient μ, while fixing the number of jobs and the number of machines for each studied problem. We present in Fig. 5 the results of the statistical study represented by the ANOVA diagram concerning the instance 30×10 .We consider 10 replications for each instance by varying

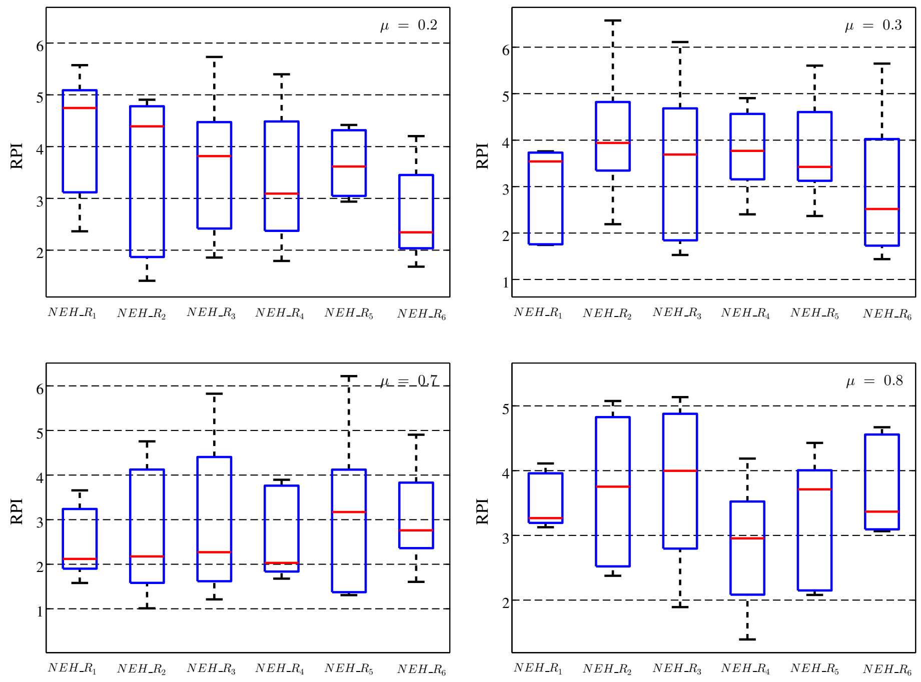

Fig. 5 Box plot of RPI variation for all used heuristics and for $\mu = 0.2$ (top left), $\mu = 0.3$ (top right), $\mu = 0.7$ (bottom left) and for $\mu = 0.8$ (bottom right)

the coefficient such that $\mu \in \{0.2, 0.3, 0.7, 0.8\}$, the analysis will be made on median value of RPI.

From the four box diagrams of ANOVA, we find that for the case of $\mu = 0.2$, the heuristic *NEH_R*$_6$ gives the best RPI value of 2.25% followed by the heuristic *NEH_R*$_4$ with a value of 3.11%; while the highest value of RPI is 4.77% recorded for *NEH_R*$_1$. In the case of $\mu = 0.3$ the heuristic *NEH_R*$_6$ gives the best of RPI 2.53% followed by *NEH_R*$_5$ with a value of RPI 3.25%, while the largest value is recorded by *NEH_R*$_2$. For the two previous cases when privileging the criterion of the total tardiness the heuristic *NEH_R*$_6$ gives good results compared to other heuristics.

In the case of $\mu = 0.7$ the heuristic NEH_R_4 gives the best RPI value of 2.05% followed by the heuristic *NEH_R*$_1$ with a value of 2.12%; while the highest value of RPI is 3.15% recorded for *NEH_R*$_5$. In the case of $\mu = 0.8$ the heuristic *NEH_R*$_4$ gives the best of RPI 2.95% followed by *NEH_R*$_1$ with a value of RPI 3.25%, while the largest value is recorded by *NEH_R*$_6$ with a value of 4.05%. For the last two treated cases, when privileging the criterion of makespan, the heuristic *NEH_R*$_4$ give the good results compared to other heuristics.

4.4 Comparison of the suggested metaheuristics

The third analysis concerns the convergence of the proposed metaheuristics in terms of the quality of the solution and the search computation time. A determining factor in the notion of convergence is the CPU running time. We consider the time limit $\tau = \rho \times n \times m$ in seconds where n and m respectively denote the number of jobs and the number of machines characterizing the size of the problem. The coefficient ρ is a correction coefficient that

Table 2 Values of the calibration parameter ρ for different instances

$n \in \{10, \dots, 90\}$		$n \in \{100, \dots, 400\}$	
$m = 5$	$m = 10$	$m = 10$	$m = 20$
$\rho = 0.3$	$\rho = 0.5$	$\rho = 1.2$	$\rho = 1.5$

makes it possible to calibrate the implemented algorithm for the resolution of our problem. We consider in the Table 2, the appropriate values of ρ corresponding to the size of the studied problem.

We note that these conditions on ρ make it possible to limit the computation time by supposing that the search for good solutions beyond this time becomes heavy and that our algorithms converge towards their good solutions when this time is reached.

To verify the effectiveness of the resolution metaheuristics, we propose a comparative study between the four algorithms of resolutions. The simulation concerns the two problem types according to the size of each category. The algorithms are tested by respecting a convergence time limit according to the size of the instance. We give the results of the value of RPI for an average of ten instances for each type of problem. In the Table 3, we represent the results of the relatively medium instances, we consider that $n \in \{20, 30, \dots, 60\}$ and $m \in \{5, 10\}$. From the results, we can see that the algorithm ILS$_2$ is the best algorithm by the four implemented, it records a success rate of 87.5%.

In the same way, we continue our analysis and our interpretation of the results concerning relatively large instances. For this category of problems, we study the case of $n \in \{100, 200, \dots, 400\}$ and $m \in \{10, 20\}$. We note that this type of instance is often the subject of production batches in flow shop workshops. Such a study is very useful for giving manufacturing managers a best decision-making tool in the efficient management of their machines. The results stored in Table 4 also show the dominance of the ILS$_2$ algorithm with a success rate of 88.63%.

The speed of convergence of the algorithms towards their good solutions is of fundamental utility in the analysis of the efficiency of the metaheuristiues. In Fig. 6, we represent the evolution of the objective function value for the four algorithms IG, ILS$_1$, ILS$_2$, and GA as function of the CPU time. In this illustration, we simulate the analysis of the instance 50×10 considered as a relatively medium instance. The plot of the convergence curves is given for a calculation time limit $\tau = 250$ s and $\mu \in \{0.2, 0.3, 0.7, 0.8\}$. We choose the four values of μ in such a way as to favor one of the two criteria, ie the total tardiness with $\mu \in \{0.2, 0.3\}$ or the makespan C_{max} with $\mu \in \{0.7, 0.8\}$. We note that the ILS$_2$ algorithm converges quickly towards its good solution and remains more efficient compared to other algorithms in the four studied cases.

We note that the plots of the objective function value are given for the four weighting coefficient values corresponding to the two criteria. We limit ourselves to these cases of simulations that we consider largely sufficient to concertize the comparative study of the convergence of the algorithms.

In this analysis, we also highlight the comparison of the approaches proposed in the single-criteria case. Indeed, when $\mu = 0$ the problem studied will be equivalent to $F_m|prmu, SDST, blocking|\sum_{j=1}^{n} T_j$, in case $\mu = 1$ problem is equivalent to $F_m|prmu, SDST, blocking|C_{max}$. We notice that the for small instances, the optimal solution given by the MILP for the TT minimization case does not imply a good solution for C_{max} and vice versa. Similarly, when simulating the relatively large and medium sized instances, this property remains valid. We have recorded the clear dominance of the ILS$_2$ algorithm in the two extreme cases of single-criteria optimization. Despite the intrinsic link between the

Table 3 RPI value of relatively medium instances. Best values are in bold

$n \times m$	μ										
	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
20×5											
ILS_1	3.52	4.61	2.73	**2.81**	3.23	5.64	**3.41**	3.72	4.23	2.54	**3.52**
ILS_2	**2.23**	**3.55**	**2.11**	3.22	**2.83**	**4.53**	3.57	**2.52**	**2.82**	**2.16**	3.74
IG	4.51	4.23	3.43	4.23	3.53	6.24	3.67	4.28	4.83	3.52	3.82
GA	5.23	3.54	5.38	4.57	4.25	4.57	3.42	5.63	3.73	4.17	3.82
20×10											
ILS_1	1.38	3.52	2.94	4.31	2.17	3.23	5.27	4.22	3.87	5.25	1.52
ILS_2	**0.92**	**3.23**	**3.14**	**4.25**	**1.92**	**3.12**	**5.14**	**3.97**	**3.33**	**5.14**	**1.38**
IG	3.23	4.87	4.22	5.25	3.88	4.34	4.26	5.37	4.85	5.83	2.36
GA	4.25	3.34	4.26	5.23	3.87	4.23	5.86	4.65	3.84	5.47	2.42
30×5											
ILS_1	3.58	3.27	4.54	4.34	3.23	1.25	2.37	2.45	3.67	4.25	3.68
ILS_2	**3.25**	**3.17**	**4.25**	4.24	**3.13**	**0.82**	**2.27**	**2.15**	**3.35**	**4.48**	**3.56**
IG	4.27	4.56	5.28	**3.84**	4.23	2.53	3.58	4.22	4.15	5.23	4.72
GA	3.82	4.25	5.14	3.87	4.25	3.83	3.25	4.56	5.35	5.84	4.26
30×10											
ILS_1	1.83	2.25	1.82	2.77	**1.54**	2.81	2.23	2.94	2.85	**2.25**	2.35
ILS_2	**0.82**	**1.54**	**1.35**	**1.36**	2.84	**2.12**	**1.34**	**2.23**	**1.43**	2.76	**1.34**
IG	1.14	2.15	1.64	2.45	1.96	2.26	1.53	2.35	1.67	2.86	1.56
GA	1.53	2.54	3.25	1.86	2.78	3.25	2.64	2.82	2.27	2.64	2.15
40×5											
ILS_1	4.25	4.83	**5.24**	6.23	**5.34**	**4.26**	5.37	4.86	5.83	4.82	4.35
ILS_2	**4.12**	**4.35**	5.46	**5.34**	5.76	5.82	**4.45**	**4.61**	**5.22**	**4.35**	**3.84**
IG	5.23	5.84	4.83	5.83	6.24	6.39	6.16	5.82	6.33	5.75	5.28
GA	5.35	5.83	6.27	5.55	6.24	6.23	5.85	5.78	5.29	5.74	5.32
40×10											
ILS_1	3.26	**3.69**	4.93	3.85	3.774	3.62	3.25	3.64	**3.42**	3.87	4.23
ILS_2	**2.95**	3.84	**4.26**	**3.58**	**3.29**	3.83	**3.13**	**3.32**	3.85	**4.24**	**3.65**
IG	4.23	4.35	4.46	3.94	3.82	**3.55**	3.82	4.27	4.25	4.56	3.84
GA	4.13	4.53	4.15	3.84	4.86	3.92	4.25	4.46	4.61	4.88	5.23
50×10											
ILS_1	1.12	1.53	2.14	1.35	1.76	1.48	1.86	2.27	1.69	1.83	1.14
ILS_2	**0.72**	**1.14**	**1.53**	**0.83**	**1.28**	**0.95**	**1.47**	**2.86**	**1.84**	**1.98**	**2.23**
IG	1.53	2.36	2.23	2.55	2.63	2.24	2.34	1.85	2.26	2.57	1.68
GA	2.23	2.86	3.27	2.85	3.11	3.24	3.59	2.57	3.23	2.55	2.53
60×10											
ILS_1	0.85	1.14	0.92	1.42	0.67	1.23	1.35	1.72	0.93	1.14	0.85
ILS_2	**0.65**	**0.84**	**0.67**	**1.23**	**0.53**	**1.12**	**1. 25**	**1.24**	**0.86**	**0.83**	**0.76**
IG	1.53	2.15	1.34	1.72	1.85	1.97	1.25	1.46	1.83	1.35	2.17
GA	2.23	2.12	1.86	1.65	2.24	1.86	2.27	1.83	1.74	2.27	1.57

Table 4 RPI value of relatively large instances. Best values are in bold

$n \times m$	μ										
	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
100 × 10											
ILS$_1$	2.25	**2.66**	3.75	3.82	4.27	3.34	3.29	**2.48**	2.26	2.82	2.16
ILS$_2$	**1.65**	2.84	**3.46**	**3.12**	**3.25**	**2.23**	**4.52**	2.93	**1.83**	**3.17**	**1.96**
IG	3.52	5.22	2.45	3.16	4.25	5.28	4.46	3.28	3.74	4.22	4.16
GA	4.36	4.25	3.86	5.24	3.98	5.42	2.94	5.67	4.38	3.15	3.87
100 × 20											
ILS$_1$	1.37	1.45	1.16	2.18	1.82	1.65	1.78	1.53	1.45	2.27	1.66
ILS$_2$	**0.82**	**0.94**	**0.93**	**0.53**	0.97	**0.52**	0.99	**0.82**	**0.97**	**1.25**	**0.85**
IG	2.35	1.16	0.96	1.36	**0.92**	1.87	**0.93**	1.63	1.47	1.25	1.18
GA	1.36	1.25	1.34	1.46	0.92	0.85	1.42	0.98	1.44	1.36	1.15
200 × 10											
ILS$_1$	0.12	0.56	**0.24**	**0.35**	0.58	0.77	0.38	0.44	0.85	0.93	0.52
ILS$_2$	**0.08**	**0.25**	0.46	0.82	**0.08**	**0.52**	**0.27**	**0.53**	**0.46**	**0.58**	**0.83**
IG	0.85	1.12	0.95	0.76	1.28	0.83	0.75	0.85	1.13	0.96	1.12
GA	0.93	0.87	0.95	1.34	1.23	1.14	0.93	1.56	0.83	0.96	1.17
200 × 20											
ILS$_1$	1.25	1.32	1.66	1.63	1.48	1.19	1.25	1.76	1.23	1.35	1.24
ILS$_2$	**0.96**	**0.85**	**1.13**	**0.56**	**1.28**	**1.32**	**0.65**	**0.87**	**1.12**	**0.85**	**0.94**
IG	2.16	2.25	2.38	2.14	1.36	2.59	1.92	2.23	1.83	2.56	1.96
GA	2.14	2.26	2.49	2.14	2.85	2.73	2.68	1.52	1.83	1.76	1.55
300 × 10											
ILS$_1$	0.56	0.68	0.75	0.56	0.82	0.75	0.58	0.38	0.15	0.76	0.56
ILS$_2$	**0.06**	**0.07**	**0.35**	**0.12**	**0.26**	**0.37**	**0.33**	**0.28**	**0.08**	**0.34**	**0.28**
IG	0.85	0.92	0.86	0.52	0.73	0.66	0.84	0.98	0.76	0.82	0.83
GA	0.94	0.85	0.87	0.93	0.57	0.72	0.85	0.52	0.78	0.72	0.56
300 × 20											
ILS$_1$	0.75	0.08	0.25	0.56	0.15	0.52	0.86	0.38	0.25	0.62	0.53
ILS$_2$	**0.05**	**0.08**	**0.16**	**0.23**	**0.24**	**0.07**	**0.57**	**0.38**	**0.17**	**0.52**	**0.38**
IG	0.94	0.95	0.72	0.55	0.75	0.58	0.56	0.09	0.35	0.76	0.58
GA	0.82	0.93	1.16	0.65	0.78	0.82	0.95	0.75	0.82	0.83	0.93
400 × 10											
ILS$_1$	1.14	1.25	0.92	0.87	0.52	1.34	**0.35**	0.56	0.72	0.83	0.98
ILS$_2$	**0.93**	**0.85**	**0.76**	**0.42**	**0.25**	**0.83**	0.92	**0.52**	**0.44**	**0.77**	**0.65**
IG	1.23	1.35	1.46	1.58	1.25	1.34	1.26	1.47	1.58	0.92	1.96
GA	1.56	1.62	1.72	1.85	1.86	1.74	1.63	1.95	1.76	1.28	1.42
400 × 20											
ILS$_1$	0.93	0.56	0.72	0.45	**0.35**	0.67	0.87	**0.38**	0.41	**0.83**	0.92
ILS$_2$	**0.25**	**0.07**	**0.55**	**0.33**	0.75	**0.43**	**0.83**	0.54	**0.36**	0.96	**0.75**
IG	1.1	1.35	0.89	1.17	1.15	0.95	1.14	1.12	0.98	1.11	0.85
GA	1.23	1.32	1.18	1.15	1.25	1.16	1.26	1.24	1.32	1.35	0.96

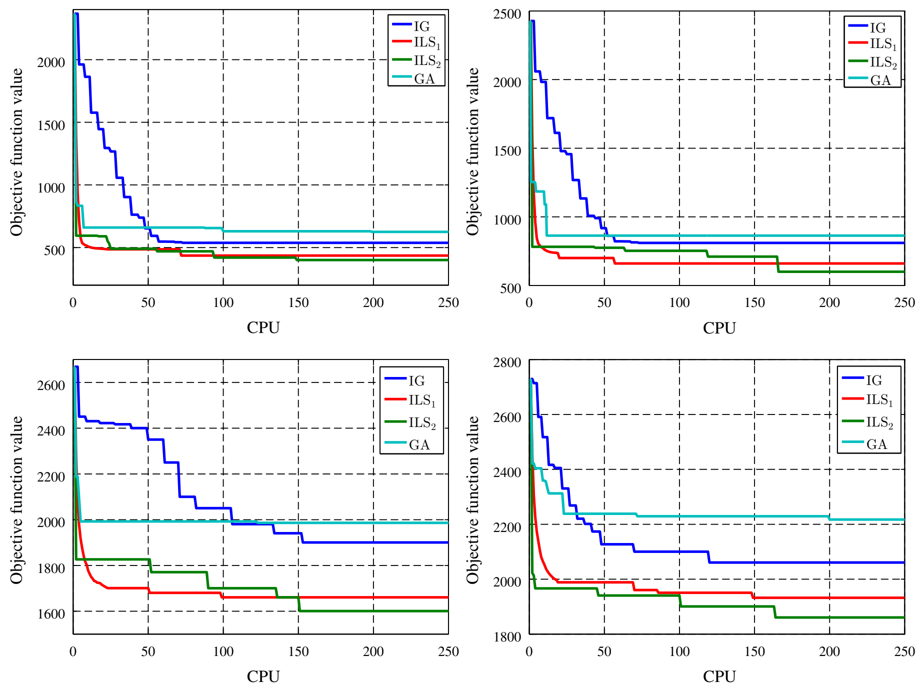

Fig. 6 Objective function value CPU time for $\mu = 0.2$ (top left), $\mu = 0.3$ (top right), $\mu = 0.7$ (bottom left) and $\mu = 0.8$ (bottom right)

two criteria in their expressions, a good TT solution does not imply a good C_{max} solution and vice versa. In final observation, we find it is difficult to find a solution that minimizes both criteria at once, a compromise is necessary to favor one of the two criteria using the weighting coefficient. In this vision, the production managers can consider the weighted multi-criteria optimization factor as a powerful tool to find a good compromise to their customers requirements.

5 Conclusion

In this paper, we have presented a mixed integer linear programming method and a set of metaheuristics to solve flow shop scheduling problem with permutation under constraints of blocking and sequence dependent setup time. The suggested problem consists in minimizing an objective functional by a combination of the two criteria: the makespan and the total tardiness each having a weighting coefficient. A comparative study between the various resolution methods is carried out on a set of instances covering small, average and relatively big size instances. To this end, we have varied the number of jobs and the number of machines; also the weighting parameter of our problem has been changed adequately in order to favor one criterion over another. We found in our simulation that the best initial solutions are given by heuristics based on NEH procedure. At the algorithm performance level, we found that the iterative local search algorithm in its second version gives better results compared to the other three algorithms in terms of the quality of the solution and the speed of convergence. Finally, it was established that the value of the objective function is strongly affected by varying the bi-criteria weighting parameter. In addition, we conclude that it is difficult to find a good solution that minimizes both criteria at the same time. Hence, a compromise will be necessary to favor one of the two optimization criteria using the weighting coefficient. In this paper, we have studied a bi-criteria optimization scheduling problem. It well be interesting to extend the work to three or more criteria in the objective function. Also, adding another constraint like the unavailability of the machines remains of a potential interest.

References

Abdel-Basset, M., Manogaran, G., El-Shahat, D., & Mirjalili, S. (2018). A hybrid whale optimization algorithm based on local search strategy for the permutation flow shop scheduling problem. *Future Generation Computer Systems*, *85*, 129–145.

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

Dong, X., Huang, H., & Chen, P. (2009). An iterated local search algorithm for the permutation flowshop problem with total flowtime criterion. *Computers & Operations Research*, *36*(5), 1664–1669.

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

Jiang, E., & Wang, L. (2019). An improved multi-objective evolutionary algorithm based on decomposition for energy-efficient permutation flow shop scheduling problem with sequence-dependent setup time. *International Journal of Production Research*, *57*(6), 1756–1771.

Johnson, S. M. (1954). Optimal two-and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly*, *1*(1), 61–68.

Li, X., & Li, M. (2015). Multiobjective local search algorithm-based decomposition for multiobjective permutation flow shop scheduling problem. *IEEE Transactions on Engineering Management*, *62*(4), 544–557.

Li, X., & Ma, S. (2017). Multiobjective discrete artificial bee colony algorithm for multiobjective permutation flow shop scheduling problem with sequence dependent setup times. *IEEE Transactions on Engineering Management*, *64*(2), 149–165.

Li, X., Yang, Z., Ruiz, R., Chen, T., & Sui, S. (2018). An iterated greedy heuristic for no-wait flow shops with sequence dependent setup times, learning and forgetting effects. *Information Sciences*, *453*, 408–425.

Liu, G. S., Zhou, Y., & Yang, H. D. (2017). Minimizing energy consumption and tardiness penalty for fuzzy flow shop scheduling with state-dependent setup time. *Journal of Cleaner Production*, *147*, 470–484.

Meziani, N., Oulamara, A., & Boudhar, M. (2019). Two-machine flowshop scheduling problem with coupled-operations. *Annals of Operations Research*, *275*(2), 511–530.

Miyata, H. H., & Nagano, M. S. (2019). The blocking flow shop scheduling problem: a comprehensive and conceptual review. *Expert Systems with Applications*, *137*, 130–156.

Moslehi, G., & Khorasanian, D. (2013). Optimizing blocking flow shop scheduling problem with total completion time criterion. *Computers & Operations Research*, *40*(7), 1874–1883.

M'Hallah, R. (2014). Minimizing total earliness and tardiness on a permutation flow shop using vns and mip. *Computers & Industrial Engineering*, *75*, 142–156.

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

Newton, M. H., Riahi, V., Su, K., & Sattar, A. (2019). Scheduling blocking flowshops with setup times via constraint guided and accelerated local search. *Computers & Operations Research*, *109*, 64–76.

Nouri, N., & Ladhari, T. (2018). Evolutionary multiobjective optimization for the multi-machine flow shop scheduling problem under blocking. *Annals of Operations Research*, *267*(1–2), 413–430.

Pan, Q. K., & Ruiz, R. (2014). An effective iterated greedy algorithm for the mixed no-idle permutation flowshop scheduling problem. *Omega*, *44*, 41–50.

Pan, Q. K., & Wang, L. (2012). Effective heuristics for the blocking flowshop scheduling problem with makespan minimization. *Omega*, *40*(2), 218–229.

Rajendran, C., & Ziegler, H. (1997). An efficient heuristic for scheduling in a flowshop to minimize total weighted flowtime of jobs. *European Journal of Operational Research*, *103*(1), 129–138.

Riahi V, Newton MH, Su K, Sattar A (2018) Local search for flowshops with setup times and blocking constraints. In *Twenty-eighth international conference on automated planning and scheduling*

Ribas, I., & Companys, R. (2015). Efficient heuristic algorithms for the blocking flow shop scheduling problem with total flow time minimization. *Computers & Industrial Engineering*, *87*, 30–39.

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.

Ronconi, D. P. (2005). A branch-and-bound algorithm to minimize the makespan in a flowshop with blocking. *Annals of Operations Research*, *138*(1), 53–65.

Ronconi DP, Birgin EG (2012) Mixed-integer programming models for flowshop scheduling problems minimizing the total earliness and tardiness. In R. Z. Rios-Mercado & Y. A. Ríos-Solís (Eds.), *Just-in-time systems, Springer Optimization and its Applications* (Vol. 60, pp. 91–105). New York, NY: Springer.

Rossi, F. L., Nagano, M. S., & Sagawa, J. K. (2017). An effective constructive heuristic for permutation flow shop scheduling problem with total flow time criterion. *The International Journal of Advanced Manufacturing Technology*, *90*(1–4), 93–107.

Ruiz, R., Pan, Q. K., & Naderi, B. (2019). Iterated greedy methods for the distributed permutation flowshop scheduling problem. *Omega*, *83*, 213–222.

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.

Shao, Z., Pi, D., & Shao, W. (2018). A novel discrete water wave optimization algorithm for blocking flowshop scheduling problem with sequence-dependent setup times. *Swarm and Evolutionary Computation*, *40*, 53–75.

Shoaardebili, N., & Fattahi, P. (2015). Multi-objective meta-heuristics to solve three-stage assembly flow shop scheduling problem with machine availability constraints. *International Journal of Production Research*, *53*(3), 944–968.

Sun, Y., Zhang, C., Gao, L., & Wang, X. (2011). Multi-objective optimization algorithms for flow shop scheduling problem: a review and prospects. *The International Journal of Advanced Manufacturing Technology*, *55*(5–8), 723–739.

Ta, Q. C., Billaut, J. C., & Bouquard, J. L. (2018). Matheuristic algorithms for minimizing total tardiness in the m-machine flow-shop scheduling problem. *Journal of Intelligent Manufacturing*, *29*(3), 617–628.

Takano, M. I., & Nagano, M. S. (2017). A branch-and-bound method to minimize the makespan in a permutation flow shop with blocking and setup times. *Cogent Engineering*, *4*(1), 1389638.

Takano, M., & Nagano, M. (2019). Evaluating the performance of constructive heuristics for the blocking flow shop scheduling problem with setup times. *International Journal of Industrial Engineering Computations*, *10*(1), 37–50.

Tasgetiren, M. F., Kizilay, D., Pan, Q. K., & Suganthan, P. N. (2017). Iterated greedy algorithms for the blocking flowshop scheduling problem with makespan criterion. *Computers & Operations Research*, *77*, 111–126.

Tellache, N. E. H., & Boudhar, M. (2018). Flow shop scheduling problem with conflict graphs. *Annals of Operations Research*, *261*(1–2), 339–363.

Trabelsi, W., Sauvey, C., & Sauer, N. (2012). Heuristics and metaheuristics for mixed blocking constraints flowshop scheduling problems. *Computers & Operations Research*, *39*(11), 2520–2527.

Xu, J., Wu, C. C., Yin, Y., & Lin, W. C. (2017). An iterated local search for the multi-objective permutation flowshop scheduling problem with sequence-dependent setup times. *Applied Soft Computing*, *52*, 39–47.

Yenisey, M. M., & Yagmahan, B. (2014). Multi-objective permutation flow shop scheduling problem: Literature review, classification and current trends. *Omega*, *45*, 119–135.

Ying, K. C. (2008). Solving non-permutation flowshop scheduling problems by an effective iterated greedy heuristic. *The International Journal of Advanced Manufacturing Technology*, *38*(3–4), 348.

Yu, A. J., & Seif, J. (2016). Minimizing tardiness and maintenance costs in flow shop scheduling by a lower-bound-based ga. *Computers & Industrial Engineering*, *97*, 26–40.

Zhang, W., Wang, Y., Yang, Y., & Gen, M. (2019). Hybrid multiobjective evolutionary algorithm based on differential evolution for flow shop scheduling problems. *Computers & Industrial Engineering*, *130*, 661–670.