Hindawi
Journal of Applied Mathematics
Volume 2020, Article ID 7132469, 11 pages
https://doi.org/10.1155/2020/7132469

Research Article

Solving Permutation Flow Shop Scheduling Problem with Sequence-Independent Setup Time

Jabrane Belabid, Said Aqil, and Karam Allali

Laboratory Mathematics and Applications, University Hassan II of Casablanca, FST, PO Box 146, Mohammedia, Morocco

Correspondence should be addressed to Jabrane Belabid; belabide@gmail.com

Received 28 August 2019; Revised 3 December 2019; Accepted 31 December 2019; Published 22 January 2020

Academic Editor: Kannan Krithivasan

In this paper, we study the resolution of a permutation flow shop problem with sequence-independent setup time. The objective is to minimize the maximum of job completion time, also called the makespan. In this contribution, we propose three methods of resolution, a mixed-integer linear programming (MILP) model; two heuristics, the first based on Johnson's rule and the second based on the NEH algorithm; and finally two metaheuristics, the iterative local search algorithm and the iterated greedy algorithm. A set of test problems is simulated numerically to validate the effectiveness of our resolution approaches. For relatively small-size problems, it has been revealed that the adapted NEH heuristic has the best performance than that of the Johnson-based heuristic. For the relatively medium and large problems, the comparative study between the two metaheuristics based on the exploration of the neighborhood shows that the iterated greedy algorithm records the best performances.

1. Introduction

The problem of scheduling in the manufacturing industry is characterized by the allocation of jobs on machines and establishing the correct sequence of those jobs in order to optimize an imposed criterion. Each job is identified by a set of tasks that form the path to be followed in the production cycle in the workshop. Every task has a running time on each machine when launching jobs in the industrial process. Several classifications are distinguished: one by machine grouping, one by job grouping, and one by mixed group. In this study, we are interested in a family where the set of jobs follows the same path on a set of machines arranged in series. This type of problem is recognized in the scheduling literature as the flow shop problem (see Figure 1). In this type of problem, several constraints may be involved, for example, in logistical means, one can find, transferring, assembly, and disassembly of parts between machines, as well as the adjustment of production tools. Solving the scheduling problem is considered as an NP-difficult issue given a certain size of the batch and number of machines. The approaches of resolution are diverse, and we cite exact methods, heuristics, metaheuristics, and other combined methods. In this paper, we propose these three classes for solving the flow shop with permutation problem (PFSP) under the constraint of sequence-independent setup time (SIST). In this case, the SIST constraint depends on the technology nature of the machine, as well as the means used to prepare it for the execution of a new job. This preparation time is considered unproductive, and its introduction into the scheduling is of great use for the control of the process. We note that the PFSPs are widely studied in the last 50 years under various constraints and objectives. Research is currently knowing significant advances by introducing new constraints and new trends of multiobjective optimization. The scheduling problem PFSP is known in the literature as $\mathcal{NP}$-complete optimization problem [1] for more than two machines. In general, since the number of jobs and the number of machines can be high, it is difficult to find the right solution with an exact method. Hence, metaheuristics are usually used to look for the right solution or at least approach it. The mainly used metaheuristics are based on an improvement of an initial solution by research in its neighborhood by one of the disruption procedures of the current solution. In the literature, we find

simulated annealing (SA) [2–4], the genetic algorithm (GA) [5–8], and the combined metaheuristics SA and GA [9, 10] that are used to solve flow shop scheduling problem. We find also other resolution methods such as tabu search [11–14] and the greedy randomized adaptive search procedure [15–18].

Other nature-inspired metaheuristics are applied for solving flow shop scheduling problem, the artificial bee colony algorithm [19–21], ant colony optimization algorithm [22, 23], and water wave optimization algorithm [24, 25]. The efficiency of these metaheuristics is observed during the resolutions of the flow shop scheduling problems with several jobs and machines. Sometimes, a machine has to be prepared in some way before it can process a certain job. It may need retooling, adjusting, and so on. The time required for such preparation is called setup time. The need to consider this new constraint, in simulation models, is pushing researchers to develop new models. Resolving flow shop problems with permutation and setup time is part of this new problem category. This topic has attracted a wide attention and has been subject of recent works for many researchers (see, for example, [26–29]).

Several works dealing with scheduling issues are presented in the optimization. Recently, a number of studies have been carried out on permutation flow shop scheduling problem under the constraint of setup time. We distinguish two types of setup time: the first depends on the sequence of the jobs on the same machine, called the sequence-dependent setup time (SDST), and the second is qualified as a setup time independent of the sequence of jobs in the same machine, called the sequence-independent setup time (SIST). For the first category of problem which is designated as the permutation flow shop scheduling problem with sequence-dependent setup time (PFSP-SDST), a large number of works have been carried out. This setup time depends on each job to be processed; for example, a big job requires more time of preparation than a small one. Often additional constraints are added such as no-waiting, blocking, and other new PFSP constraints that concertize industrial cases. The resolution of PFSP-SDST problem is presented in [30] using the migratory bird algorithm as well as a set of heuristics to minimize the makespan. In [31], the authors propose the water wave optimization algorithm to minimize the makespan with a blocking constraint for a PFSP-SDST problem. Similarly, [32] proposes an MILP model for the case of no-wait criterion for the same problem. In the same context, [33] proposes a case study of the PFSP-SDST problem under no-idle constraint to minimize the total flow time. Recently, PFSP-SDST problem is studied by using three metaheuristics with a local search neighborhood [29], and it was revealed that the iterated greedy gives good results comparing with the two other used methods. For the second category of problem, i.e., PFSP-SIST, we note that only few works has been tackled, which may be a way to be explored for future research. This second class of problem is very interesting to study when a given machine requires the same setup time for all jobs. For example, the machine can have an automatic fixed setup time to process every job; such a setup time can be interpreted in same industrial cases as cleaning, parts assembly-disassembly, or maintenance of a certain machine. We note that [34, 35] propose a set of MILP models and lower bounds for specific cases of the PFSP-SIST problem. In the same context, [36] propose an MILP model and a set of constructive heuristics to solve $F_m|\text{prmu, SIST}|C_{\max}$. We are continuing the investigation in PFSP-SIST issues by solving this problem for larger size of instances and testing two metaheuristics. To this end, we propose a study with three methods: MILP, heuristics, and metaheuristics. The two implemented metaheuristics for resolution are among the most powerful in the neighborhood exploration, the iterative local search (ILS) algorithm, and the iterated greedy (IG) algorithm. We note that our case study is particular because of the presence of the SIST constraint. In fact, this setup time may be the subject of several cases such as assembly and disassembly of parts or tools in the machine, cleaning, evacuation of production means, and so on. These activities can be carried out by robots or manipulators independent of the processed jobs in the machine. In this perspective, we present a study to contribute to the resolution of this problem which forms an essential manufacturing case to be encountered in the new industrial technology. We continue in this line of research and propose a model in MILP and two metaheuristics for solving the PFSP-SIST problem.

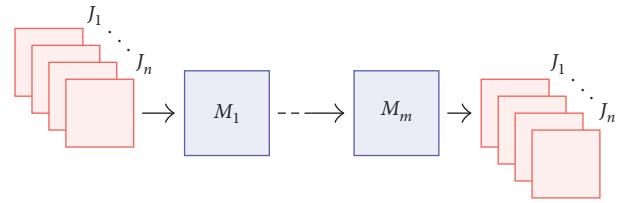

Figure 1: Flow shop problem with n jobs and m machines.

Our paper is organized as follows. In Section 2, a description of the problem is presented; in Section 3, we develop all the proposed approaches to resolution. In Section 4. a comparative study between the different methods of resolution is presented. Section 5 summarizes our resolution approaches.

2. Description of the PFSP-SIST Problem

In the PFSP-SIST scheduling problem, a set of jobs $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ constituting a production batch is launched in the production process consisting of a set of machines $\mathcal{M} = \{M_1, M_2, \ldots, M_m\}$ arranged in series. All jobs have the same path and m operations on all machines, and each job J_j starts with the first machine and ends its cycle with the last machine. We consider that $p_{i,j}$ the processing time of the job J_j on the machine M_i, and each machine has an extra time of preparation s_i when processing a new job. Denote a sequence $\pi = \{\pi_1, \pi_2, \ldots, \pi_n\}$ constituting a possible permutation when processed by the machines. We note C_{i,π_j} the completion time of the job π_j of the sequence on the M_i machine. The goal is to find the right sequence among the $n!$ possible sequence to minimize the production time of the batch of n jobs and determine the start and completion time

of each job on each machine. The assumptions used are those of the classic case, that is to say the interruption is not allowed, and each machine processes only one job at time and job can process only once by the same machines. All jobs and machines are available at the initial moment. The aim is to minimize the maximum completion time called makespan $C_{\max}$ under the selected assumptions as well as the availability constraints of the machines during scheduling in the real time of the entire batch. We give here a set of equations to determine the completion time of each job on each machine:

$$C_{1,\pi_1} = p_{1,\pi_1} + s_1, \tag{1}$$

$$C_{1,\pi_j} = C_{1,\pi_{j-1}} + p_{1,\pi_j} + s_1, \quad j = 2, \ldots, n, \tag{2}$$

$$C_{i,\pi_1} = \max\left\{s_i, C_{i-1,\pi_1}\right\} + p_{i,\pi_1}, \quad i = 2, \ldots, m, \tag{3}$$

$$C_{i,\pi_j} = \max\left\{C_{i,\pi_{j-1}} + s_i, C_{i-1,\pi_j}\right\} + p_{i,\pi_j}, \quad i = 2, \ldots, m; \quad j = 2, \ldots, n, \tag{4}$$

$$C_{\max} = \max_{1 \le j \le n}\left\{C_{m,\pi_j}\right\}. \tag{5}$$

Equations (1) and (2) allow to determine the end date of the first job and the other jobs on the first machine, respectively. Equations (3) and (4) allow to determine, respectively, the end date of the first job and the other jobs on all the other machines. Equation (5) calculates the makespan $C_{\max}$. In this model, the objective is to determine an optimal sequence π^* in the set permutation Π, such as $C_{\max}(\pi^*) \le C_{\max}(\pi)$, $\pi \in \Pi$.

2.1. Numerical Example. We give here an numerical illustration with small instance represented by three jobs and three machines. The processing time of all jobs and the setup time for all machines of are given in Table 1. We represent the Gantt chart for the sequence $\pi = (3, 1, 2)$, the makespan of this instance is 48 unite of time (see Figure 2). From this numerical example, it can be easily seen that jobs will be processed on the first machine continuously. On the contrary, when scheduling jobs in other machines, some machines remain on hold, which leads to a waste of time. In conclusion, the sequence in the first machine is a determining factor in the search for the right solution.

3. Resolution of PFSP-SIST Problem

The resolution of the scheduling problems during the last decades has undergone a great evolution and a great based on power of the height technology computers. To solve our problem $F_m|\text{prmu}, \text{SIST}|C_{\max}$, we propose three approaches, a MILP, two heuristics among the most powerful for the scheduling problem and finally two metaheuristics algorithm ILS and IG. In the last resolution category, we rely on the exploration structures of the neighboring of a sequence π based on the permutation or insertion of a job in the sequence to obtain a new sequence in the space search for solution.

Table 1: The processing time of 3 jobs on 3 machines.

Job	Job 1	Job 2	Job 3	s_i
M_1	9	5	9	3
M_2	8	8	8	2
M_3	7	6	6	3

3.1. Mixed-Integer Linear Programming. Modeling optimization problems in scheduling and in particular in the PFSP is one of the most important areas of research in operational research. In fact, these models propose a function to optimize under constraints solved by one of the solvers dedicated to this type of model. The models are numerous and diverse, often characterizing a practical case for industrial need or a theoretical case for academic research. Several models in MILP are proposed for different case study models of the PFSP; a comparative study is described in [37], and a new model is presented [38]. Furthermore, more models are presented in [39] for the PFSP-SDST problem. We present here a model in MILP for our problem, and we define a set of parameters and variables necessary for the realization of the proposed model.

$$x_{k,j} = \begin{cases} 1, & \text{if job } j \text{ is assigned to the position } k \text{ of the sequence,} \\ 0, & \text{otherwise,} \end{cases} \quad k, j = 1, \ldots, n,$$

$C_{i,k}$ = completion time of job at position k on machine i, $i = 1, \ldots, m$, $k = 1, \ldots, n$. (6)

The objective is to

$$\text{minimize } C_{\max} = C_{m,n}, \tag{7}$$

subject to

$$\sum_{j=1}^{n} x_{k,j} = 1, \quad k = 1, \ldots, n, \tag{8}$$

$$\sum_{k=1}^{n} x_{k,j} = 1, \quad j = 1, \ldots, n, \tag{9}$$

$$C_{1,1} \ge s_1 + \sum_{j=1}^{n} x_{1,j} \times p_{1,\pi_1}; \quad j = 1, \ldots, n, \tag{10}$$

$$C_{1,k} \ge s_1 + C_{1,k-1} + \sum_{j=1}^{n} x_{k,j} \times p_{1,\pi_j}; \quad k = 2, \ldots, n, \tag{11}$$

$$C_{i,k} \ge C_{i-1,k} + \sum_{j=1}^{n} x_{k,j} \times p_{i,\pi_j}, \quad i = 2, \ldots, m; \quad k = 1, \ldots, n, \tag{12}$$

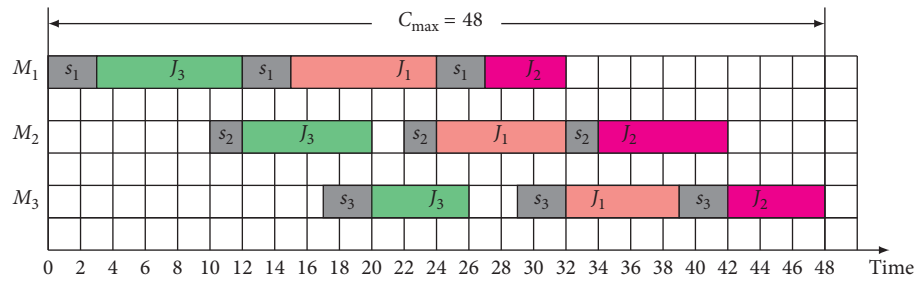

Figure 2: Gantt chart for three jobs in three machines.

$$C_{i,k} \geq s_i + C_{i,k-1} + \sum_{j=1}^{n} x_{k,j} \times p_{i,\pi_j}, \quad i = 1, \ldots, m, \quad k = 1, \ldots, n, \tag{13}$$

$$C_{i,k} \geq 0, \quad i = 1, \ldots, m;\ k = 1, \ldots, n, \tag{14}$$

$$x_{k,j} \in \{0, 1\}, \quad j, k = 1, \ldots, n. \tag{15}$$

Knowing that (7) defines the objective function to minimize. Constraints (8) and (9) make it possible to guarantee that each position in the sequence will be occupied by a single job, and each job can have only one position. Constraint (10) is used to determine the completion time of the job occupying the first position in the first machine. Constraint (11) calculates the completion time of the job at the position $k, k \geq 2$ in the first machine. Constraint (12) guarantees that for a job at the k position, the completion time in the machine i is greater than or equal to the completion time in the previous machine $i-1$ increasing its processing time on the i machine. Constraint (13) ensures that, for two jobs of successive positions, the completion time in the machine of the one in position k is greater than or equal to that of the one in position $k-1$ increasing in processing and setup times. Constraints (14) and (15) ensure that the completion time is positive and the assignment variables $x_{k,j}$ are binary, respectively. To solve our mixed-integer linear programming model dealing with PFSP-SIST problem, we will use LINGO 17.3 software.

3.2. Heuristics. The PFSP scheduling problems under various constraints can be solved through different heuristics. In fact, for the industrialists, the search of the solutions by heuristics constitutes simple and less expensive issue to ensure the production and to satisfy their customers. The heuristics are usually based on priority rules based on job characteristics such as processing time, setup time, and other priorities related to the problem being addressed. In this work, we present two heuristics, the heuristic Johnson's rule [40] and the one based on the NEH [41] algorithm. These two basic heuristics are among the most promising approaches in finding the right solution for most PFSP scheduling problems.

3.2.1. An Heuristic Based on Johnson's Rule. The PFSP problem with two machines is the basic problem of any flow shop scheduling problems. The studies that have followed since the first study refer to this problem. The extensions of these problems are numerous, and the cases studied often fall under the particular constraints like the preparation of the machines or the preparation of the jobs to schedule all the jobs in the workshop. The main objective of optimization is to minimize the makespan from which we can deduce many other intrinsic objectives. We were interested in this approach by proposing a heuristic based on Johnson's rule, which we adopted to solve our case study of $F_m|\text{prmu, SIST}|C_{\max}$. The following algorithm presents our heuristic of resolution by highlighting the characteristics our problem.

In this investigation, the Johnson algorithm is adapted to fit more than two machines flow shop problems. Algorithm 1 shows the template of an heuristic based on Johnson's rule (HBJR) with construction of two virtual machines by assembling the first $1 < k < m-1$ machines into one virtual machine and the $k-m$ remaining machines into the second virtual one. The processing time of a job π_j on a virtual machine (p'_{i,π_j}) is the sum of the processing times p_{i,π_j} of job π_j on the machines that compose the virtual machine.

3.2.2. NEH Heuristic. The NEH algorithm constitutes a heuristic pillar for solving the PFSP problem with n jobs and m machines. More and more researchers are developing more adopted versions of the case and following the constraints of each problem. We do several research work as in [42, 43] and more recently in [44]. Indeed, Ruiz and Maroto compared NEH with more modern and more complex heuristics, such as those of [45, 46] showing NEH algorithm realized the better performance. Here, we propose a version of the NEH algorithm for the resolution of PFSP-SIST, the Algorithm 2 presents the necessary steps to establish the sequence of NEH.

In order to adapt the NEH algorithm to the studied problem, the setup time is added to the sum of processing times of each job in the first step of the algorithm $T_j = \sum_{i=1}^{m} p_{i,\pi_j} + s_i$.

3.3. Metaheuristic Algorithms. The resolution of PFSP by metaheuristics has been very successful in recent years. Indeed, the instance size of the problems has become larger

Input: $\pi_0 = \{\pi_1, \ldots, \pi_n\}$ % The initial sequence
$\pi \longleftarrow \pi_0$
% Construction of π'
$p'_{1,\pi_j} \longleftarrow p_{1,\pi_j}$, $j = 1, \ldots, n$, $p'_{2,\pi_j} \longleftarrow \sum_{k=2}^{m} p_{k,\pi_j}$, $j = 1, \ldots, n$
$\pi^* \longleftarrow \pi'$ % Sequence π on two virtual machines with new processing time p'_{i,π_j}
for $i = 2$ to $m - 1$ **do**
% Reconstruction of new two machines
$p'_{1,\pi_j} \longleftarrow \sum_{k=1}^{i} p_{k,\pi_j}$, $j = 1, \ldots, n$, $p'_{2,\pi_j} \longleftarrow \sum_{k=i+1}^{m} p_{k,\pi_j}$, $j = 1, \ldots, n$
$\pi'' \longleftarrow$ Johnson's rule (π')
if $(C_{\max}(\pi'') < C_{\max}(\pi'))$ **then**
$\pi \longleftarrow \pi''$
if $(C_{\max}(\pi) < C_{\max}(\pi^*))$ **then**
$\pi^* \longleftarrow \pi$
end if
end if
end for
makespan $\longleftarrow C_{\max}(\pi^*)$
Output: π^*, makespan

Algorithm 1: An heuristic based on Johnson's rule.

Input: processing time $p_{j1}, \ldots, p_{jm}$ and the setup time s_i in m machines.
Evaluate the total processing time and setup time of each job by: $T(j) = \sum_{i=1}^{m} p_{ji} + s_i$
Construct the sequence $\pi = (\pi_1, \ldots, \pi_n)$ by sorting the jobs in descending order of $T(j)$
Schedule the first two jobs π_1, π_2 and choose the best sequence π of the first two jobs
for $j = 3$ to n **do**
Insert π_j in the all positions of the current sequence built and select the best sequence with the minimum $C_{\max}$ in the last machine and update π_{NEH} of the optimal sequence.
end for
Output: π_{NEH}

Algorithm 2: NEH algorithm.

Input: $\pi_0 = \{\pi_1, \ldots, \pi_n\}$ the initial sequence
$\pi^{(\text{best})} \longleftarrow \pi_0$, $\pi \longleftarrow \pi_0$
while {the stopping criterion not satisfied} **do**
Choose π' from the neighborhood of $N_k(\pi)$
if $C_{\max}(\pi') < C_{\max}(\pi)$ **then**
$\pi \longleftarrow \pi'$
if $C_{\max}(\pi) < C_{\max}(\pi^{(\text{best})})$ **then**
$\pi^{(\text{best})} \longleftarrow \pi$
end if
else
if random $\leq \exp\{-(C_{\max}(\pi') - C_{\max}(\pi))/T_{\text{emp}}\}$ **then**
$\pi \longleftarrow \pi'$
end if
end if
end while
Output: $\pi^{(\text{best})}$

Algorithm 3: ILS algorithm.

and larger, exact methods such as MILP or the branch and bound procedure are limited to a small-size instance. The need to solve the real cases of problems induces researchers to develop metaheuristics to give the right solution in a reasonable time. In this paper, we propose two main metaheuristics, the iterative local search (ILS) algorithm and the iterated greedy (IG) algorithm. The two algorithms are proposed to solve the problem $F_m|\text{prmu}, \text{SIST}|C_{\max}$ with the

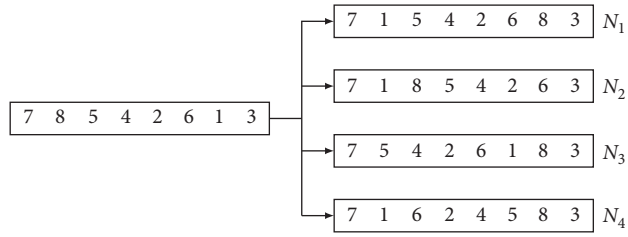

Figure 3: Neighborhood structure.

Input: π the initial sequence.
  $\pi^{(\text{best})} \longleftarrow \pi$
  while {unsatisfied stopping criterion} **do**
    $\pi\prime \longleftarrow \pi$
    for $u = 1$ to d **do**
      Extract a job π_u at random from the sequence π' and add the job π_u to the Φ subset.
    end for
    for $u = 1$ to d **do**
      Extract the π'_u job from Φ subset.
      Test the job on all positions in the current sequence π' and choose the best position giving the best completion time.
    end for
    Choose π'' from the neighborhood of $N_k(\pi')$
    if $C_{\max}(\pi'') < C_{\max}(\pi)$ **then**
      $\pi \longleftarrow \pi''$
      if $C_{\max}(\pi) < C_{\max}(\pi^{(\text{best})})$ **then**
        $\pi^{(\text{best})} \longleftarrow \pi$
      end if
    else
      if random $\leq \exp\{-(C_{\max}(\pi'') - C_{\max}(\pi))/T_{\text{emp}}\}$ **then**
        $\pi \longleftarrow \pi''$
      end if
    end if
  end while
Output: $\pi^{(\text{best})}$

Algorithm 4: Iterated greedy algorithm.

diversification of neighborhood exploration that will be given in detail as follows. For our considered instances, the number of machines $m \in 2, \ldots, 30$ and the number of jobs $n \in 5, \ldots, 600$; the processing times p_{ij} are defined in [1, 49] and the setup times s_i are defined in [1, 10].

3.3.1. ILS Algorithm. The ILS algorithm is considered one of the powerful optimization algorithms for solving scheduling problems. We find the latest applications of this algorithm in [47–49] for solving the PFSP problem and its extensions under various implementations. In Algorithm 3, we describe the steps of the development of this algorithm. In the first phase, an initial solution is given by one of the initialization heuristics. In the second, while the stopping criterion not satisfied a search procedure in the neighborhood of the current solution is triggered, each time, an update of the best solution is performed. In this implementation model, we choose an adaptation of the simulated annealing criterion to better explore the neighborhood.

The implementation of the ILS algorithm requires a set of parameter settings to better converge to the optimal solution depending on the size of the problem. The first parameter is the stop condition of the algorithm that we consider here in number of iterations, the second parameter is the acceptance condition of the simulated annealing model T_{emp}, We adopt the following model:

$$T_{\text{emp}} = T_0 \times \frac{\sum_{j=1}^{n}\sum_{i=1}^{m} p_{i,j} + s_i}{10 \times m \times n}. \tag{16}$$

This expression is given according to the characteristic of the PFSP-SIST problem. The parameter T_0 makes possible to control the speed of convergence of our algorithm, we choose in this study $T_0 \in \{0.5, \ldots, 0.98\}$. This model of temperature has been adopted in several research works concerning the PFSP problem (see, for example, [50]).

The local search metaheuristics requires the definition of the neighborhood structure exploration, we propose four structures (N_i, $i = 1, \ldots, 4$). Starting from the current sequence of Figure 3 where the jobs in

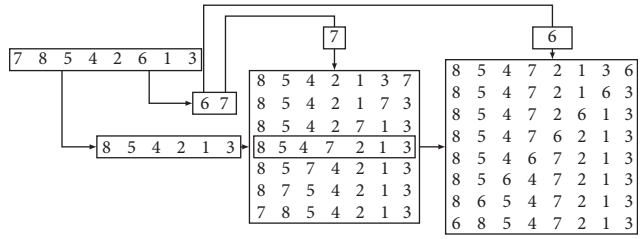

Figure 4: IG procedure.

positions 2 and 7, respectively, are used to generate the set of neighborhoods considered. Neighborhood N_1 is defined by the permutation of the position of two jobs. Neighborhood N_2 adopts the insertion and shift to the right, and Neighborhood N_3 considers the insertion and shift on the left. Finally, neighborhood N_4 is defined by job blocks inversion between two different positions in the sequence.

3.3.2. IG Algorithm. The second heuristic implemented to solve our problem is the IG algorithm (see Algorithm 4). This algorithm is based on three stages of resolution: the step of destruction of the current solution, the step of the reconstruction of the solution and the last step concerns the local search procedure. The first two stages constitute the first main phase of the principle of the algorithm IG, the second main phase is that of the neighborhood exploration. Recent work in applying the IG [51–53] algorithm shows the rising power of this algorithm in solving the optimization problem in the scheduling domain. The implementation of this algorithm requires also a set of setting parameters, the stop condition, the number d of jobs extracted in the sequence to build the new sequence, the parameter of the simulated annealing temperature, and finally the structure of the neighborhood used in the exploration phase. For the stop condition and the temperature parameter T_{emp}, we adopt the same condition as that adopted for the ILS algorithm. As for the neighborhood structure, we consider the same structure as ILS. For the d parameter of extracted jobs, we consider that it also depends on the size of the problem knowing that $d \in \{\lfloor n/4 \rfloor, \lfloor n/3 \rfloor\}$, for $n \in \{5, \ldots, 50\}$ and $d \in \{\lfloor n/3 \rfloor, \lfloor n/2 \rfloor\}$ for $n > 50$.

We present here an illustrative example of the procedure of destruction and reconstruction of the IG algorithm. Knowing that the current solution is $\pi = \{7, 8, 5, 4, 2, 6, 1, 3\}$, we randomly extract two jobs, for example, 7 and 6, the remaining sequence is $\pi_d = \{8, 5, 4, 2, 1, 3\}$. Subsequently, we insert the first 7 job in all positions, and we select the best constructed sequence giving the value of $C_{\max}$, for example $\pi = \{8, 5, 4, 7, 2, 1, 3\}$. We repeat the procedure for the new built sequence and insert job 6 in all possible positions, and at the end, we get the best value from the makespan. Figure 4 gives a representation of the procedure of destruction and reconstruction of the IG algorithm.

4. Computational Results

To validate our approaches of resolution for the studied problem, we propose two comparative studies between the different algorithms. The first comparative study is based on the calculation of the relative percentage deviation (RPD). The second concerns the convergence study of resolution algorithms for medium and large instances. For the first comparative study, we give the expression of RPD by the following formula:

$$\text{RPD}_{H_i} = \frac{C_{\max}^{H_i} - C_{\max}^{\text{best}}}{C_{\max}^{\text{best}}} \times 100; \quad i \in \{1, 2\}, \tag{17}$$

where H_1 stands for HBJR and H_2 represents NEH heuristic. In this expression, we measure the deviation of the solution found from the optimal solution given by the MILP, for relatively small instances, where $C_{\max}^{H_i}$ represents the solution found by the heuristic H_i and $C_{\max}^{\text{best}}$ represents the value found by the MILP. Table 2 shows the results found for some tested instances. We limit ourselves to such number of problems; our objective is to highlight the effectiveness of resolution heuristics and their relevance to solve this problem. From the results, we find that the heuristic NEH is better compared with the heuristic based on the rule of Johnson.

Table 2 shows the RPD_{H_i} obtained for different combinations $m \times n$, where m is the number of machines and n stands for the number of jobs. It is observed that for $m = 2$, the obtained results by the heuristics are the same and equal to those given by the MILP. It is also shown that for a number of jobs $n \geq 10$, the NEH heuristic gives better results than those calculated by the heuristic based on Johnson's rule.

For relatively large and medium instances, we measure the deviation from the best solution given by the ILS and IG algorithms. For this last class of problem, we vary the number of jobs and machines to reach high instances. We note that computation is given for an average of five instances for each algorithm. The result in this comparative study is given in Table 3; this table shows the clear superiority of the IG algorithm with respect to ILS for all tested instances. We report that the number of machines tested is $m \in \{5, 10, 20, 30\}$, and the number jobs varies in a way to cover different sizes and reaches a maximum of 600 jobs. We have tried to better concertize the real size of the jobs processed in industrial batches.

Table 2: Comparison between mixed-integer linear programming and heuristics.

job × machine	MILP	HBJR	NEH	RPD_{H_1}	RPD_{H_2}
5 × 2	177	177	177	0	0
10 × 2	324	324	324	0	0
15 × 2	495	495	495	0	0
20 × 2	658	658	658	0	0
30 × 2	928	928	928	0	0
5 × 5	316	332	316	5.06	0
10 × 5	449	459	453	2.22	0.89
15 × 5	598	660	602	10.36	0.66
20 × 5	763	826	767	8.25	0.52
5 × 10	460	476	493	3.47	7.17
10 × 10	599	652	616	8.84	2.83
15 × 10	764	841	780	10.07	2.09
5 × 15	599	599	605	0	1.0
10 × 15	727	765	739	5.22	1.65
15 × 15	882	994	919	12.69	4.19
5 × 20	748	757	759	1.2	1.47
10 × 20	900	958	933	6.44	3.66
15 × 20	1069	1166	1101	9.07	2.99

Table 3: Comparison between iterated local search and iterated greedy.

job × machine	RPD_{ILS}	RPD_{IG}
10 × 5	0.000	0.000
20 × 5	0.864	0.000
30 × 5	0.559	0.000
40 × 5	0.050	0.000
50 × 5	0.086	0.000
60 × 5	0.466	0.000
20 × 10	1.713	0.129
30 × 10	0.440	0.101
40 × 10	0.902	0.043
50 × 10	1.464	0.287
60 × 10	0.272	0.110
10 × 20	0.461	0.000
20 × 20	0.990	0.030
30 × 20	0.427	0.159
40 × 20	0.430	0.187
50 × 20	1.386	0.483
60 × 20	1.937	0.097
70 × 20	1.588	0.010
80 × 20	0.545	0.143
90 × 20	0.645	0.109
100 × 20	0.34	0.120
140 × 20	1.158	0.060
160 × 20	1.370	0.051
200 × 20	1.167	0.000
240 × 20	1.315	0.173
300 × 20	0.466	0.121
340 × 20	0.637	0.007
400 × 20	1.729	0.000
500 × 30	0.816	0.000
600 × 30	0.506	0.016

The analysis concerning the numerical results shows the efficiency of our suggested metaheuristics in term of convergence towards their good solutions.

In Figure 5, we represent the evolution of the values of the objective function as a function of the number of iteration for two instances 40×10 and 40×20; the two algorithms converge to their stable values about after 80^{th} iteration, and IG saves an important gain of units time compared with ILS. For the second instances, the two ILS and IG algorithms converge to their stable bounds at about 280^{th} iterations and that IG is better compared to ILS.

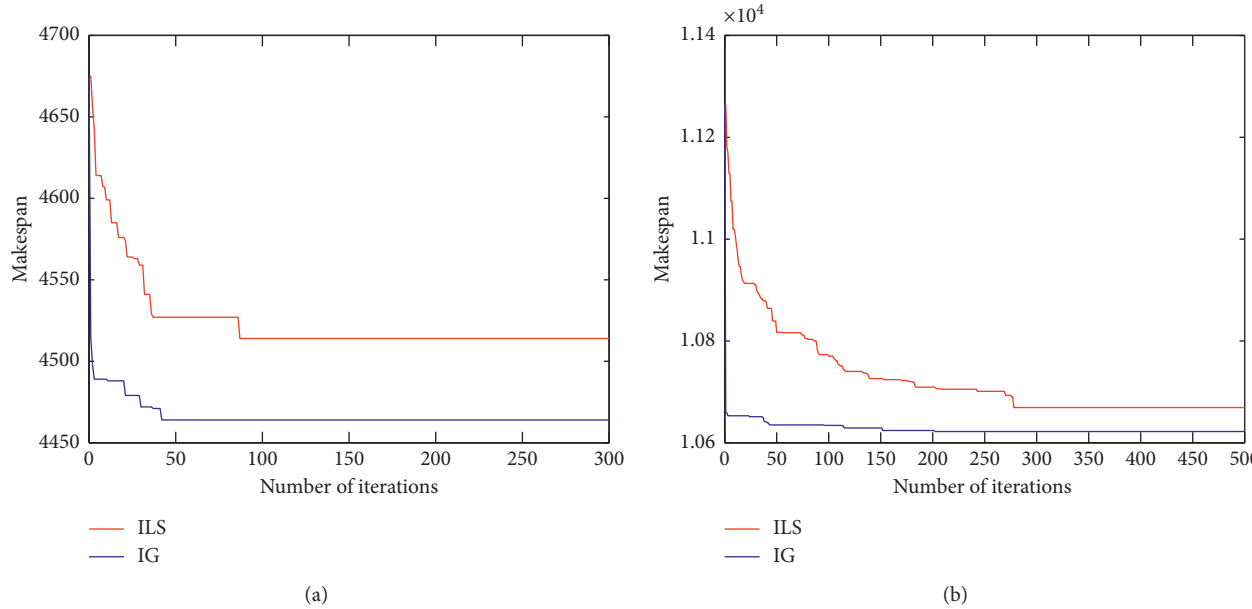

Figure 5: Makespan vs. number of iterations for $n = 40$ and $m = 10$ (a) and for $n = 40$ and $m = 20$ (b).

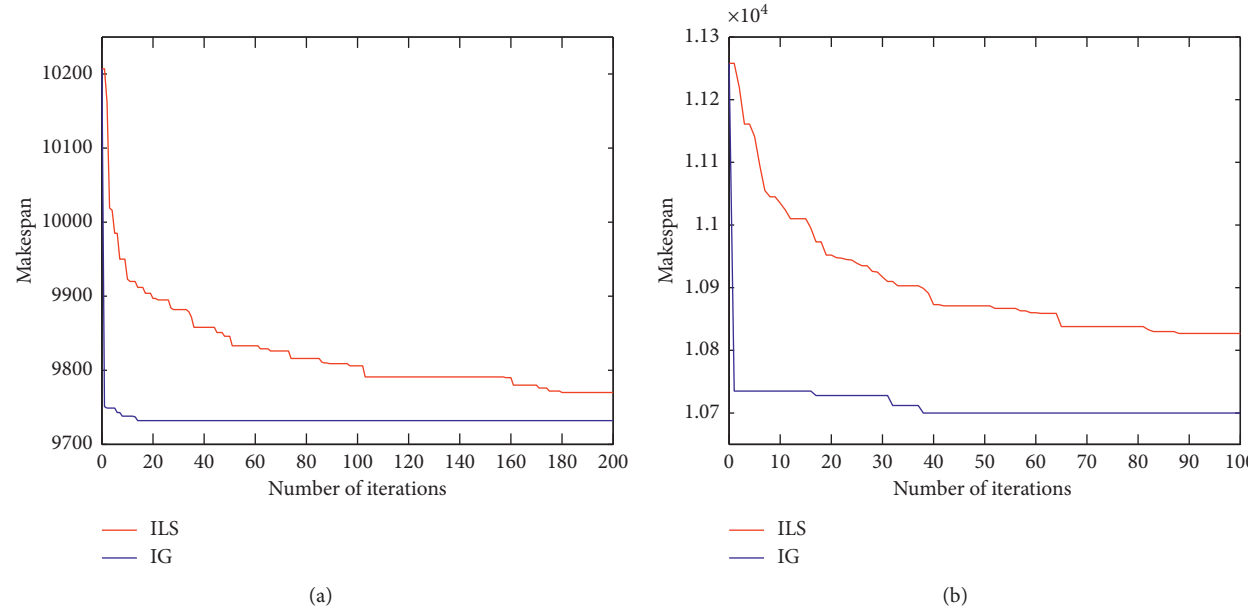

Figure 6: Makespan vs. number of iterations for $n = 100$ and $m = 10$ (a) and for $n = 100$ and $m = 20$ (b).

In Figure 6, we represent the evolution of the objective function for the two instances 100×10 and 100×20, respectively. These two instances can be considered relatively larger instances. This type of size is frequently encountered in modern industry, where customers ask for a fairly high batch. For the first instance, we can see that the two algorithms stabilize after the 180^{th} iteration; a significant gain in calendar scheduling time is recorded by IG compared with ILS algorithm. For the second instance, we can conclude that the IG gives best results comparing with the ILS algorithm after 90^{th} iteration.

The four tested instances, in the convergence comparison study, have been implemented for the NEH initialization heuristic. Indeed, this initialization algorithm gives us better results in terms of RPD for relatively small instances and relatively large and medium instances.

5. Conclusion

In this paper, we have studied a flow shop problem with permutation and sequence-independent setup time in order to minimize the makespan. First, we have conducted a

comparative study between the exact mixed-integer linear programming method and two heuristics, one is based on Johnson's rule, and the other is based on NEH algorithm. The two heuristics were compared with the results of the MILP for all instances. Simulations are performed for relatively small instances by varying the number of jobs and machines. It was observed that the NEH heuristic gives better results than those given by the heuristic based on Johnson's rule. On the contrary, two metaheuristics are used for medium to relatively large instances. We have concluded that IG algorithm gives good results comparing with ILS metaheuristic. In this paper, we have studied PFSP-SIST problem by using MILP model and different metaheuristics. However, adding an other constraint to this problem like unavailability or no-idle machines can be considered as a potential future issue.

Data Availability

The data used the support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] M. R. Garey, D. S. Johnson, and R. Sethi, "The complexity of flowshop and jobshop scheduling," *Mathematics of Operations Research*, vol. 1, no. 2, pp. 117–129, 1976.

[2] I. Osman and C. Potts, "Simulated annealing for permutation flow-shop scheduling," *Omega*, vol. 17, no. 6, pp. 551–557, 1989.

[3] F. A. Ogbu and D. K. Smith, "The application of the simulated annealing algorithm to the solution of the $n/m/C_{\max}$ flowshop problem," *Computers & Operations Research*, vol. 17, no. 3, pp. 243–253, 1990.

[4] M. Henneberg and J. S. Neufeld, "A constructive algorithm and a simulated annealing approach for solving flowshop problems with missing operations," *International Journal of Production Research*, vol. 54, no. 12, pp. 3534–3550, 2016.

[5] C. R. Reeves, "A genetic algorithm for flowshop sequencing," *Computers & Operations Research*, vol. 22, no. 1, pp. 5–13, 1995.

[6] T. Murata, H. Ishibuchi, and H. Tanaka, "Genetic algorithms for flowshop scheduling problems," *Computers & Industrial Engineering*, vol. 30, no. 4, pp. 1061–1071, 1996.

[7] S. K. Iyer and B. Saxena, "Improved genetic algorithm for the permutation flowshop scheduling problem," *Computers & Operations Research*, vol. 31, no. 4, pp. 593–606, 2004.

[8] R. Ruiz, C. Maroto, and J. Alcaraz, "Two new robust genetic algorithms for the flowshop scheduling problem," *Omega*, vol. 34, no. 5, pp. 461–476, 2006.

[9] M. Andresen, H. BräSel, M. MöRig, J. Tusch, F. Werner, and P. Willenius, "Simulated annealing and genetic algorithms for minimizing mean flow time in an open shop," *Mathematical and Computer Modelling*, vol. 48, no. 7-8, pp. 1279–1293, 2008.

[10] H. Wei, S. Li, H. Jiang, J. Hu, and J. Hu, "Hybrid genetic simulated annealing algorithm for improved flow shop scheduling with makespan criterion," *Applied Sciences*, vol. 8, no. 12, p. 2621, 2018.

[11] E. Nowicki and C. Smutnicki, "A fast tabu search algorithm for the permutation flow-shop problem," *European Journal of Operational Research*, vol. 91, no. 1, pp. 160–175, 1996.

[12] J. Grabowski and M. Wodecki, "A very fast tabu search algorithm for the permutation flow shop problem with makespan criterion," *Computers & Operations Research*, vol. 31, no. 11, pp. 1891–1909, 2004.

[13] A. Ahmad and Q. Zhang, "MOEA/D with tabu search for multiobjective permutation flow shop scheduling problems," in *Proceedings of the 2014 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1155–1164, IEEE, Beijing, China, July 2014.

[14] H. Bargaoui and O. B. Driss, "Multi-agent model based on tabu search for the permutation flow shop scheduling problem," *Distributed Computing and Artificial Intelligence*, vol. 3, no. 8, p. 27, 2014.

[15] R. L. Daniels and R. J. Chambers, "Multiobjective flow-shop scheduling," *Naval Research Logistics*, vol. 37, no. 6, pp. 981–995, 1990.

[16] G. Prabhaharan, B. S. H. Khan, and L. Rakesh, "Implementation of grasp in flow shop scheduling," *The International Journal of Advanced Manufacturing Technology*, vol. 30, no. 11-12, pp. 1126–1131, 2006.

[17] L. P. Molina-Sánchez and E. M. González-Neira, "Grasp to minimize total weighted tardiness in a permutation flow shop environment," *International Journal of Industrial Engineering Computations*, vol. 7, no. 1, pp. 161–176, 2016.

[18] E. Alekseeva, M. Mezmaz, D. Tuyttens, and N. Melab, "Parallel multi-core hyper-heuristic grasp to solve permutation flow-shop problem," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 9, p. e3835, 2017.

[19] Y. Ince, K. Karabulut, M. F. Tasgetiren, and Q.-k. Pan, "A discrete artificial bee colony algorithm for the permutation flowshop scheduling problem with sequence-dependent setup times," in *Proceedings of the 2016 IEEE Congress on Evolutionary Computation (CEC)*, pp. 3401–3408, IEEE, Vancouver, Canada, July 2016.

[20] X. Li and S. Ma, "Multiobjective discrete artificial bee colony algorithm for multiobjective permutation flow shop scheduling problem with sequence dependent setup times," *IEEE Transactions on Engineering Management*, vol. 64, no. 2, pp. 149–165, 2017.

[21] D. Gong, Y. Han, and J. Sun, "A novel hybrid multi-objective artificial bee colony algorithm for blocking lot-streaming flow shop scheduling problems," *Knowledge-Based Systems*, vol. 148, pp. 115–130, 2018.

[22] V. Riahi and M. Kazemi, "A new hybrid ant colony algorithm for scheduling of no-wait flowshop," *Operational Research*, vol. 18, no. 1, pp. 55–74, 2018.

[23] O. Engin and A. Güçlü, "A new hybrid ant colony optimization algorithm for solving the no-wait flow shop scheduling problems," *Applied Soft Computing*, vol. 72, pp. 166–176, 2018.

[24] X. Yun, X. Feng, X. Lyu, S. Wang, and Bo Liu, "A novel water wave optimization based memetic algorithm for flow-shop scheduling," in *Proceedings of the 2016 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1971–1976, IEEE, Vancouver, Canada, July 2016.

[25] F. Zhao, H. Liu, Y. Zhang, W. Ma, and C. Zhang, "A discrete water wave optimization algorithm for no-wait flow shop scheduling problem," *Expert Systems with Applications*, vol. 91, pp. 347–363, 2018.

[26] J. Xu, C.-C. Wu, Y. Yin, and W.-C. Lin, "An iterated local search for the multi-objective permutation flowshop

scheduling problem with sequence-dependent setup times," *Applied Soft Computing*, vol. 52, pp. 39–47, 2017.

[27] A. Costa, F. A. Cappadonna, and S. Fichera, "A hybrid genetic algorithm for minimizing makespan in a flow-shop sequence-dependent group scheduling problem," *Journal of Intelligent Manufacturing*, vol. 28, no. 6, pp. 1269–1283, 2017.

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

[29] S. Aqil and A. Karam, "Three metaheuristics for solving the flow shop problem with permutation and sequence dependent setup time," in *Proceedings of the 2018 4th International Conference on Optimization and Applications (ICOA)*, pp. 1–6, IEEE, Mohammedia, Morocco, April 2018.

[30] A. Sioud and C. Gagné, "Enhanced migrating birds optimization algorithm for the permutation flow shop problem with sequence dependent setup times," *European Journal of Operational Research*, vol. 264, no. 1, pp. 66–73, 2018.

[31] Z. Shao, D. Pi, and W. Shao, "A novel discrete water wave optimization algorithm for blocking flow-shop scheduling problem with sequence-dependent setup times," *Swarm and Evolutionary Computation*, vol. 40, pp. 53–75, 2018.

[32] C.-Y. Cheng, K.-C. Ying, S.-F. Li, and Y.-C. Hsieh, "Minimizing makespan in mixed no-wait flowshops with sequence-dependent setup times," *Computers & Industrial Engineering*, vol. 130, pp. 338–347, 2019.

[33] F. L. Rossi and M. S. Nagano, "Heuristics for the mixed no-idle flowshop with sequence-dependent setup times and total flowtime criterion," *Expert Systems with Applications*, vol. 125, pp. 40–54, 2019.

[34] T. Ladhari, M. K. Msakni, and A. Allahverdi, "Minimizing the total completion time in a two-machine flowshop with sequence-independent setup times," *Journal of the Operational Research Society*, vol. 63, no. 4, pp. 445–459, 2012.

[35] A. Gharbi, T. Ladhari, M. K. Msakni, and M. Serairi, "The two-machine flowshop scheduling problem with sequence-independent setup times: new lower bounding strategies," *European Journal of Operational Research*, vol. 231, no. 1, pp. 69–78, 2013.

[36] J. Belabid, S. Aqil, and A. Karam, "Solving flow shop problem with permutation and sequence independent setup time," in *Proceedings of the 2019 5th International Conference on Optimization and Applications (ICOA)*, pp. 1–5, IEEE, Kenitra, Morocco, April 2019.

[37] E. F. Stafford Jr., F. T. Tseng, and J. N. D. Gupta, "Comparative evaluation of milp flowshop models," *Journal of the Operational Research Society*, vol. 56, no. 1, pp. 88–101, 2005.

[38] F. T. Tseng and E. F. Stafford Jr., "New MILP models for the permutation flowshop problem," *Journal of the Operational Research Society*, vol. 59, no. 10, pp. 1373–1386, 2008.

[39] F. T. Tseng and E. F. Stafford Jr., "Two MILP models for the n × m SDST flowshop sequencing problem," *International Journal of Production Research*, vol. 39, no. 8, pp. 1777–1809, 2001.

[40] S. M. Johnson, "Optimal two- and three-stage production schedules with setup times included," *Naval Research Logistics Quarterly*, vol. 1, no. 1, pp. 61–68, 1954.

[41] M. Nawaz, E. E. Enscore, and I. Ham, "A heuristic algorithm for the m-machine, *n*-job flow-shop sequencing problem," *Omega*, vol. 11, no. 1, pp. 91–95, 1983.

[42] S. Turner and D. Booth, "Comparison of heuristics for flow shop sequencing," *Omega*, vol. 15, no. 1, pp. 75–78, 1987.

[43] E. Taillard, "Some efficient heuristic methods for the flow shop sequencing problem," *European Journal of Operational Research*, vol. 47, no. 1, pp. 65–74, 1990.

[44] R. Ruiz and C. Maroto, "A comprehensive review and evaluation of permutation flowshop heuristics," *European Journal of Operational Research*, vol. 165, no. 2, pp. 479–494, 2005.

[45] C. Koulamas, "A new constructive heuristic for the flowshop scheduling problem," *European Journal of Operational Research*, vol. 105, no. 1, pp. 66–71, 1998.

[46] S. M. A. Suliman, "A two-phase heuristic approach to the permutation flow-shop scheduling problem," *International Journal of Production Economics*, vol. 64, no. 1–3, pp. 143–152, 2000.

[47] R. M'Hallah, "An iterated local search variable neighborhood descent hybrid heuristic for the total earliness tardiness permutation flow shop," *International Journal of Production Research*, vol. 52, no. 13, pp. 3802–3819, 2014.

[48] A. A. Juan, H. R. Lourenço, M. Mateo, R. Luo, and Q. Castella, "Using iterated local search for solving the flow-shop problem: parallelization, parametrization, and randomization issues," *International Transactions in Operational Research*, vol. 21, no. 1, pp. 103–126, 2014.

[49] X. Dong, M. Nowak, P. Chen, and Y. Lin, "Self-adaptive perturbation and multi-neighborhood search for iterated local search on the permutation flow shop problem," *Computers & Industrial Engineering*, vol. 87, pp. 176–185, 2015.

[50] R. Ruiz and T. Stützle, "An iterated greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives," *European Journal of Operational Research*, vol. 187, no. 3, pp. 1143–1159, 2008.

[51] J.-Y. Ding, S. Song, J. N. D. Gupta et al., "An improved iterated greedy algorithm with a tabu-based reconstruction strategy for the no-wait flowshop scheduling problem," *Applied Soft Computing*, vol. 30, pp. 604–613, 2015.

[52] M. F. Tasgetiren, D. Kizilay, Q.-K. Pan, and P. N. Suganthan, "Iterated greedy algorithms for the blocking flowshop scheduling problem with makespan criterion," *Computers & Operations Research*, vol. 77, pp. 111–126, 2017.

[53] R. Ruiz, Q.-K. Pan, and B. Naderi, "Iterated greedy methods for the distributed permutation flowshop scheduling problem," *Omega*, vol. 83, pp. 213–222, 2019.