# HEURISTICS FOR THE PERMUTATION FLOW SHOP SCHEDULING PROBLEM UNDER SEQUENCE INDEPENDENT SETUP TIME

**Hajar SADKI[1] and Karam ALLALI[2]**

[1] *University Hassan II of Casablanca, FST, Mohammedia, PO Box 146, Morocco*
*E-mail: sadkisadkihajar@gmail.com*
[2] *University Hassan II of Casablanca, FST, Mohammedia, PO Box 146, Morocco*
*E-mail: allali@hotmail.com*

***ABSTRACT:*** *In this paper, we study a multi-objective permutation flow shop scheduling problem with sequence independent setup time. Our optimization problem is minimizing the makespan and the total flow time in a single objective function where each criterion has a weighted factor. The principal goal is to find the optimal sequence of the jobs that minimizes the two criteria of makespan and total flow time. Our purpose is to solve this problem, with a mixed integer linear programming model (MILP) and a set of four heuristics for different size instances: the first heuristic is an algorithm with extended Johnson's rule (EJR) and then the Nawaz-Enscore-Ham (NEH) algorithm, greedy randomized adaptive search procedure algorithm (GRASP) and path-relinking (PR) algorithm. To test the efficacy of our resolution, a series of instances with n jobs and m machines are randomly produced ranging in size from small instances. The analysis of the proposed simulation allowed us to remark that, for problems of relatively small size, the NEH heuristic has better performance for the considered problem than that of the Johnson algorithm.*

***KEYWORDS****: Flow shop scheduling. Sequence independent setup time. Makespan and total flow time. Heuristic.*

## 1 INTRODUCTION

The industrial companies are confronted with new ecological and sociological constraints, alongside other considerations such as competitors, market dynamics, and technological advancements. In today's industrial market, businesses are facing mounting pressure due to the competitive demands imposed by their rivals. Companies have turned to advanced technologies and computer simulation tools, leveraging the power of artificial intelligence (AI) concepts. By harnessing the potential of AI, companies aim to optimize their production lines through the efficient planning of activities and effective management of machine scheduling within their workshops. In this context, numerical methods and optimization techniques play a vital role in addressing the complexities associated with flow shop scheduling. These methods allow companies to improve their production processes, and minimize makespan or total completion time. In recent years, there has been significant research activity in tackling the permutation flow shop scheduling problem (PFSSP). This problem is a crucial area of focus as it directly affects the productivity, efficiency, and profitability of manufacturing processes. It involves determining the best sequence of operations for a set of jobs, taking into account various constraints, sequence independent setup time (SIST).

The permutation flow shop scheduling problem with sequence independent setup time (PFSSP-SIST) is a well-known optimization problem in the field of operations research. It involves scheduling a set of jobs on a series of machines in a flow shop environment, where each job must go through all the machines in a predetermined order. This constraint is commonly observed in various types of production machines. The SIST constraint is determined by the technology nature of the machine and the methods employed to prepare it for executing a new job. In this study, we focus on addressing the PFSSP-SIST problem, aiming to minimize the makespan Cmax and total flow time TFT. This problem is denoted as $Fm|prmu,SIST|(1-\delta)\times TFT+\delta\times Cmax$ in the literature on scheduling optimization, the weighting coefficient of each criterion δ, denoted by $\delta \in [0, 1]$, represents the preference of the scheduler. This writing follows the notation commonly used in the scheduling literature, which consists of three distinct fields, denoted as $\alpha|\beta|\gamma$. The first field, α, represents the manufacturing flow shop with Fm serial machines. The second field, β, indicates the

constraints: prmu, SIST. The third field, γ, identifies the criterion that needs to be minimized. The MPFSSP Zobolas et al. (2009); Rossi et al. (2017) has been extensively studied, and various solution approaches based on approximate and exact methods have developed. Exact methods for solving the MFSS primarily involve mathematical formulations and the branch and bound technique Gmys et al. (2020); Kim and Lee (2019). In Sadki et al. (2023); Belabid et al. (2020), a mathematical formulation based on mixed-integer linear programming (MILP) is presented as an optimization solution for PFSSP-SIST. Similarly, approximate techniques employ heuristic and metaheuristic tools to address the problem effectively. The majority of contributions to solving PFSSP problems consist of the latter methods. We also note that a lot of research (Ruiz and Maroto, 2005) has been devoted to the Nawaz-Enscore-Ham (NEH) algorithm. In Taillard (1990), the suggested NEH and Taboo Search algorithm solves the PFSSP problem Fm|prmu|Cmax. In addition, in Rajendran (1993) the problem Fm||TFT is solved using a heuristic algorithm. In Saravanan et al. (2015), the authors propose greedy randomized adaptive search procedure algorithm (B-GRASP) to minimizing $Fm||\delta \times \sum_{i=1} C_i + (1-\delta) \times C_{max}$, PFSSP problem. In Shao et al. (2018), The path relinking algorithm is proposed to solve the blocking flow shop scheduling problem (BFSP) with the makespan criterion. Our objective is to utilize the extended Johnson's rule (EJR), NEH, GRASP, and path-relinking (PR) algorithms on various instances to improve the current solution. These algorithms allow us to modify the sequence simply, resulting the best possible result. To achieve the best possible result, we carefully select the variable settings for each algorithm. Building on these contributions, we further our research by applying nature-inspired metaheuristics to solve the PFSSP-SIST problem.

The paper's organization is as follows: Section 2 introduces the PFSSP-SIST model, while Section 3, discusses the adopted methods. In Section 4, we showcase the computational results and perform a comparative analysis of the developed methods; finally, we will conclude this work highlighting both the strengths of our work and potential avenues for future exploration.

## 2 DESCRIPTION OF THE PFSSP-SIST MODEL

In the permutation flow shop scheduling problem, there are a fixed number of machines, denoted by M, and a set of jobs, denoted by J. Each job consists of a sequence of operations that need to be processed on the machines. The goal is to find an optimal schedule that minimizes the total makespan $C_{max}$ and total flow time TFT, which is the time it takes to complete all the jobs. Which is formed by the function F and involves constraints expressed as functions of parameters and decision variables. We will also assume that no machine can take more than one job at a time and that no machine can do more than one job at a time. The following notations will be assumed in our study.

### 2.1 The notations

- n: The size of the input data of the jobs
- m: The size of the input data of the machines
- M = {1, 2 . . .m}: A set of machines
- J = {1, 2 . . . n}: A set of jobs
- pi,j ≥ 0: The processing time of the job
- Sti: The setup time of each machine
- β: All of the jobs are in sequence where β={β1,,βn}
- $C_{max}$: The makespan
- TFT: The total flow times
- $C_{t_i,\beta_j}$: The completion time of the job jth in the ith machines
- ηk,j: The boolean variable
- $C_{t_i,q}$: the Completion time of the job at position q on machine i, q = 1, . . . , n, i = 1, . . . ,m.
- Π: Probability of selecting job
- δ: Parameter and its value ranges from zero to one
- RCL: Restricted candidate list
- The objective is to determine the start and finish dates of each job on each machine to create a schedule for the batch that was introduced into the manufacturing process. Here, we give a series of equations above to determine the parameters of the start time of each job on the machine:

$$Ct_{1,\beta_1} = p_{1,\beta_1} + St_1 \ \#(1)$$

$$Ct_{1,\beta_j} = Ct_{1,\beta_{j-1}} + p_{1,\beta_j} + St_1, \qquad j = 2,\dots,n \#(2)$$

$$Ct_{i,\beta_j} = \max\{St_i, Ct_{i-1,\beta_1}\} + p_{i,\beta_1} \#(3)$$

$$Ct_{i,\beta_j} = \max\left\{Ct_{i,\beta_{j-1}} + St_i, Ct_{i-1,\beta_1}\right\} + p_{i,\beta_j} \#(4)$$

$$i = 2,\dots,m, j = 2,\dots,n$$

$$TFT = \sum_{j=1}^{n} Ct_{m,\beta_j} \#(5)$$

$$C_{max} = \max_{1\le j\le n}\left\{Ct_{m,\beta_j}\right\} \#(6)$$

$$\min F = (1-\delta) \times TFT + \delta \times C_{max} \#(7)$$

$$F(\beta^*) \le F(\beta), \beta \in \Pi \#(8)$$

In Eq.(1), the completion time of the first job, β1, on the first machine is determined. Eq.(2) are used to determine the completion time of job βj on the first machine. The completion time of the first job, β1, on the remaining machines is determined in Eq.(3). Eq.(4) defines the end date for job βj on the remaining machines. The objective is to minimize the total flow time, which is the sum of completion times for all jobs, as described in Eq.(5). Additionally, minimizing the makespan is achieved when the departure date from machine m is the same as the end date of a job on the last machine $Ct_{m,\beta_j}$, as shown in Eq.(6). Thus, the main objective function combines these two criteria: the makespan Cmax and the total flow time (TFT), as expressed in Eq.(6). Our goal in the flow shop scheduling problem is to find an optimal sequence that minimizes the permutation objective function Π by thoroughly exploring the entire neighborhood, as indicated in Eq.(7).

## 2.2 Illustration with numbers

To highlight the proposed model PFSSP-SIST, we present an illustrative example that considers an instance consisting of four jobs, three machines, and setup time forming our workshop. This problem amounts to finding a good permutation sequence to minimize the job production time and determine the start and end time of each job on each machine. We represent the processing times of the four jobs β = [j1, j2, j3, j4] on the three machines {1, 2, 3} in Table 1.

**Table 1: The processing time of four jobs on three machines**

| Job | : | J1 | J2 | J3 | J4 | Sti |
|---|---|---|---|---|---|---|
| Machine 1 | : | 4 | 5 | 3 | 6 | 1 |
| Machine 2 | : | 3 | 4 | 6 | 5 | 2 |
| Machine 3 | : | 6 | 6 | 4 | 3 | 1 |

.

We need to schedule the sequence $\beta^* = [j_3, j_1, j_2, j_4]$, the Eqs. (1)- (4) the following expressions provide a detailed breakdown of the departure dates for each job performed by each machine in our sequence: $Ct_{1,j3} = p_{1,j3} + St_1 = 3 + 1 = 4$, $Ct_{1,j1} = Ct_{1,j3} + p_{1,j1} + St_1 = 4 + 4 + 1 = 9$, $Ct_{1,j2} = Ct_{1,j1} + p_{1,j2} + St_1 = 9 + 5 + 1 = 15$, $Ct_{1,j4} = Ct_{1,j2} + p_{1,j4} + St_1 = 15 + 6 + 1 = 22$, $Ct_{2,j3} = \max\{St_2, Ct_{1,j3}\} + p_{2,j3} = \max\{2, 4\} + 6 = 10$, $Ct_{3,j3} = \max\{St_3, Ct_{2,j3}\} + p_{3,j3} = \max\{1, 10\} + 4 = 14$, $Ct_{2,j1} = \max\{Ct_{2,j3} + St_2, Ct_{1,j1}\} + p_{2,j1} = \max\{10 + 2, 9\} + 3 = 15$, $Ct_{2,j2} = \max\{Ct_{2,j1} + St_2, Ct_{1,j2}\} + p_{2,j2} = \max\{15 + 2, 15\} + 4 = 21$, $Ct_{2,j4} = \max\{Ct_{2,j2} + St_2, Ct_{1,j4}\} + p_{2,j4} = \max\{21 + 2, 22\} + 5 = 28$, $Ct_{3,j1} = \max\{Ct_{3,j3} + St_1, Ct_{2,j2}\} + p_{3,j1} = \max\{14 + 1, 15\} + 3 = 18$, $Ct_{3,j2} = \max\{Ct_{3,j1} + St_1, Ct_{2,j2}\} + p_{3,j2} = \max\{18 + 1, 21\} + 6 = 27$, $Ct_{3,j4} = \max\{Ct_{3,j2} + St_1, Ct_{2,j4}\} + p_{3,j4} = \max\{17 + 1, 28\} + 3 = 31$. Knowing that the three jobs were completed on the last machine, we can determine when they ended, $Ct_{3,j3} = 14$, $Ct_{3,j1} = 18$, $Ct_{3,j2} = 27$, $Ct_{3,j4} = 31$ as well as the makespan value $C_{max} = 31$ time units. When calculating the total flow time for the jobs during scheduling, the due dates for each job are added together, and the result is TFT = 14+18+27+31 = 90 time units, with a value of δ = 0.75 our goal function is constructed in such a way: F = (1−0.75)×90+0.75×31 = 45.75. The Gantt chart in Fig. 1 shows the result of scheduling the three machines in the studied example under the constraints of sequence independent setup time.

$$F = (1 - 0.75) \times 90 + 0.75 \times 31 = 45.75$$

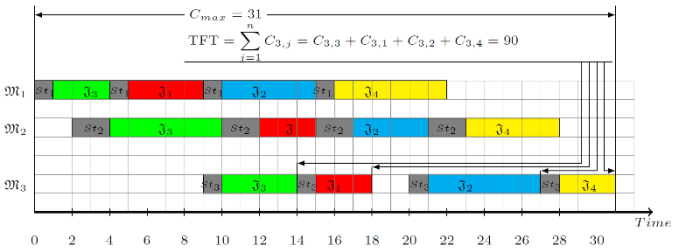


**Figure 1:Gantt chart for four jobs in three machine**

# 3 THE RESOLUTION APPROACHES

We suggest two approaches to resolving the PFSSP-SIST bi-objective scheduling problem. The first is based on a MILP mathematical formulation, whereas the second approach consists of affection heuristics.

## 3.1 The mixed integer linear programming model approach

Mixed Integer Linear Programming (MILP) is a mathematical optimization technique that can be used to solve flow shop scheduling problems. In the flow shop scheduling problem, a set of jobs needs to be processed on a set of machines in a specific order, to minimize the function objective. With our approach, we define a set of variables and parameters that are necessary for modeling the problem to be solved. The model can be formulated as follows:

$$\eta_{q,j} = \begin{cases} 1, \textit{if the job } j \textit{ is scheduled at position } q^{th} \textit{ in the sequence} \\ 0, \textit{otherwise} \end{cases}$$
$$j, q = 1, \dots, n$$

The objective is:

$$Minimize\ F = (1 - \delta) \times \mathrm{TFT} + \delta \times C_{max} \#(9)$$

The subject is:

$$\sum_{j=1}^{n} \eta_{q,j} = 1\ ,\ q = 1, \dots, n\ \#(10)$$

$$\sum_{q=1}^{n} \eta_{q,j} = 1\ ,\ j = 1, \dots, n\ \#(11)$$

$$\mathrm{Ct}_{1,1} \geq \mathrm{St}_1 + \sum_{j=1}^{n} \eta_{1,j} \times \mathrm{p}_{1,\beta_1} \#(12)$$

$$\mathrm{Ct}_{1,q} \geq \mathrm{St}_1 + \mathrm{Ct}_{1,(q-1)} + \sum_{j=1}^{n} \eta_{q,j} \times \mathrm{p}_{1,\beta_j} \#(13)$$

$$q = 2, \ldots, n$$

$$\mathrm{Ct}_{i,q} \geq \mathrm{Ct}_{(i-1),q} + \sum_{j=1}^{n} \eta_{j,q} \times \mathrm{p}_{i,\beta_j} \#(14)$$

$$i = 2, \ldots, m;\ q = 1 \ldots, n$$

$$\mathrm{Ct}_{i,q} \geq \mathrm{St}_i + \mathrm{Ct}_{i,(q-1)} + \sum_{j=1}^{n} \eta_{j,q} \times \mathrm{p}_{i,\beta_j} \#(15)$$

$$i = 1, \ldots, m, q = 1, \ldots, n$$

$$\begin{cases} \mathrm{C}_{max} \geq \mathrm{Ct}_{m,q}, \mathrm{q} = 1, \ldots, \mathrm{n} \\ \mathrm{TFT} \geq \sum_{q=1}^{n} \mathrm{Ct}_{m,q} \end{cases} \#(16)$$

$$\mathrm{Ct}_{i,q} \geq 0, \quad i = 1, \ldots, m;\ q = 1, \ldots, n \#(17)$$

$$\eta_{j,q} \in \{0,1\}, \quad j, q = 1, \ldots, n \#(18)$$

Equation (9) shows the objective function, expressed in terms of two criteria, the makespan and total flow time, each with a weight coefficient. Constraint set (10) ensures that there is a one-to-one correspondence between positions and jobs, with each position occupied by only one job. Constraint set (11) requires each job to be assigned to a specific position. Constraint set (12) determines the estimated completion time of the first job on the first machine. Constraint set (13) a work cannot be processed on the first machine while its predecessor is processed in the same machine. Constraint set (14) regulates that a job can only be processed on a machine after it has been processed on the one before it. According to constraint set (15), each job can only begin processing once the preceding job on the same machine has finished processing. Constraint (16) is used to determine the makspean and total flow time. Equations (17) and (18) using to calculate the makspean and total flow time.

## 3.2 The proposed heuristics

Heuristics are typically quick and effective methods for solving scheduling optimization problems and delivering successful performance. In this paper, we propose four constructive approaches for solving the bi-criteria minimization problem of the Fm|prmu,SIST|(1−δ)×TFT+δ×$C_{max}$ scheduling problem

## 3.3 The Extended Johnson's Rule algorithm

The Johnson algorithm, as originally defined by Johnson (1954), is specifically designed for two-machine flow shop scheduling problems. However, it can be extended to handle flow shop scheduling problems with more than two machines through a modified approach known as Extended Johnson's Rule. In Algorithm 1 a presentation of the general outline of the Extended Johnson's Rule (EJR) for more than two machines in a PFSSP-SIST.

However, to incorporate Johnson's Rule in a heuristic for flow shop scheduling problems with more than two machines, we can a modified approach based on the following steps:

1- Identify the machine with the minimum processing time for each job (denoted as min 1) and the machine with the maximum processing time (denoted as max 1).

2- If min 1 is from the first machine, schedule the job as the first job in the sequence. If min 2 is from the last machine, schedule the job as the last job in the sequence.

3- Compare the processing times of min 1 and min 2 for the remaining machines. Schedule the job as the first job in the sequence if its processing time on min 1 is less than its processing time on min 2.

Otherwise, schedule the job as the last job in the sequence

4- Repeat steps 2 − 3 until all jobs are scheduled.

5- Once all jobs are scheduled as the first or last job in the sequence, apply Johnson's Rule to schedule the remaining jobs on the remaining machines using a two-machine flow shop approach.

**Algorithm 1** Extended Johnson's Rule

**Input:** $\beta_0=\{\beta_1, \beta_2, \ldots, \beta_n\}$, $\beta \leftarrow \beta_0$

Step 1:

$p'_{1,\beta_j} \leftarrow p_{1,\beta_j} + St_1$

$p'_{2,\beta_j} \leftarrow \sum_{k=2}^{m} \left(p_{k,\beta_j} + St_k\right), \mathrm{j} = 1, \ldots, \mathrm{n}$

$\beta^* \leftarrow \beta'$

Step 2:

**For** i=2 to m-1 **do**

$p'_{1,\beta_j} \leftarrow \sum_{k=1}^{i} \left(p_{k,\beta_j} + St_k\right), \mathrm{j} = 1, \ldots, \mathrm{n}$

$p'_{2,\beta_j} \leftarrow \sum_{k=i+1}^{m} \left(p_{k,\beta_j} + St_k\right), \mathrm{j} = 1, \ldots, \mathrm{n}$

$\beta'' \leftarrow$ Johnson rule($\beta'$)

**if** $F(\beta'') < F(\beta')$ **then**

$\beta \leftarrow \beta''$

**if** $F(\beta) < F(\beta^*)$ **then**

$\beta \leftarrow \beta$

**end if**

**end if**

**end for**

$F(\beta) \leftarrow F(\beta^*)$

**Output:** $\beta^*$

### 3.4 The NEH algorithm

The NEH (Nawaz, Enscore, and Ham) heuristic Nawaz et al. (1983) is a constructive algorithm used to find an optimal sequence for the permutation flow shop scheduling problem.

Here is a general outline of the NEH heuristic:

1- Start with an empty sequence.

2- Calculate the total processing time for each job by summing up its processing times and setup time on all machines.

3- Sort the jobs in non-increasing order based on their total processing times.

4- Initialize the sequence by inserting the job with the highest total processing time at the beginning.

5- For each remaining job, insert it at each possible position in the current sequence and calculate the new value of F.

6- Select the position that results in the smallest makespan and insert the job there.

7- Repeat Step 6 for all remaining jobs, iteratively building the sequence.

8- The final sequence obtained after inserting all jobs is an optimal solution.

Algorithm 2, describes the processes for scheduling n jobs on m machines for a set of units in PFSSPSIST with a bi-criteria minimization scheduling problem.

**Algorithm 2** The algorithm of NEH

Input: A processing time p1,βj, . . . , pm,βj have the setup time Sti in m machines.

Calculate sum the total processing time and setup time of each job by:

$$T(\beta_j) = \sum_{i=1}^{m} \left(p_{i,\beta_j} + St_k\right)$$

Construct the sequence β = (β1, . . ., βn) by sorting the jobs in descending order of T(βj ).

Schedule the first two jobs β1, β2 and choose the best sequence β of the first two jobs.

for j = 3 to n do

Insert βj in all positions of the current sequence built select the best sequence

with the minimum F in the last machine and update βNEH of the optimal sequence.

end for

Output: βNEH

### 3.5 The greedy randomized adaptive search procedure algorithm

The Greedy Randomized Adaptive Search Procedure (GRASP) algorithm is an approach used to solve optimization problems, including the flow shop scheduling problem Prabhaharan et al. (2006). Here is a general outline of the GRASP algorithm for the flow shop scheduling problem:

1- Initialize an empty solution.

2- Create a list, called RCL (Restricted Candidate List), consisting of candidate jobs for selection.

3- Define the RCL of candidate jobs based on an interval established using the objective function values of neighboring solutions.

4- Determine the interval for selecting candidate jobs using the parameter α, where α ∈ [0.5, 0.9].

5- Set the condition for selecting jobs in the RCL using the following equation: F(βj) ∈ [F1, F1 +α× (F2 − F1)].

The GRASP's constructive steps are illustrated in Algorithm 3.

**Algorithm 3** GRASP algorithm

Input: β = {β1, β2, . . . , βn} % the best current sequence

βgr ← ∅ % the GRASP sequence

Ω ← β, and evaluate the makespan βj ∈ Ω

while Ω≠ ∅ do

F1 ← min {F(βj ), βj ∈ Ω}, F2←max{F(βj), βj ∈ Ω}

Create the RCL←{βj∈Ω,F(βj)∈[F1, F1+α×(F2−F1)]}

Choose the sequence βu from the RCL that will give the created

solution's optimum place that minimizes value of F.

βgr ← βgr ∪ βu and update from Ω.

end while

Output: βgr

### 3.6 The path relinking algorithm

Path relinking Vallada and Ruiz (2010) is an approach that can be used to solve optimization problems, including the flow shop scheduling problem. The path relinking heuristic combines solutions from different parts of the search space to generate improved solutions. Here is a general outline of the path relinking heuristic for the flow shop scheduling problem:

1- Select two individuals, these individuals represent solutions in the population.

2- Perform a series of movements (e.g., swap moves or insertion moves). Each movement results in an intermediate individual.

3- Carry out a movement, producing an intermediate individual.

4- Repeat step 3, performing additional movements to generate more intermediate individuals.

5- With increasing movements, the intermediate individuals become progressively more similar

6- Evaluate all the obtained intermediate individuals, typically using an objective function Algorithm 4 is a pseudo code of the PR heuristic.

**Algorithm 4** The algorithm of path relinking

Input: β = {β1, β2, . . . , βn} % the best current sequence
B ← ∅, βpr ← β
for j = 1 to n − 1 do
if β(j) ≠ βbest(j) then
Locate the position k of the job βbest in the sequence β and insert the job β(k) at position β(j) to obtain the new sequence β
B ← B ∪ β
End if
Choose the sequence βpr that results in the smallest value of F among all sequences in B
end for
Output: βpr

## 4 EXPERIMENTAL RESULTS

In this section, we will present the computational findings used to evaluate the performance of our methods in this part. We highlight a set of heuristics in our approach for solving the bi-objective SIST/PFSSP scheduling problem. The research is conducted by comparing the small size. In small instances, EJR, NEH, and GRASP algorithms and the exact procedure specified by the MILP approach are compared. For the problem testing, we have selected the simulation parameters. The number of jobs n, the number of machines m, the processing times pi,j ∈ [1, 45], and the setup time Sti are also created at random in intervals according to a uniform law [1, 5] and δ ∈ [0, 1] are all specified. For small and medium-sized instances, we changed the number of jobs n ∈ {5, 10, 15} and the numbers of machines m ∈ {2, 3, 5}. The measured parameter is based on calculating the relative deviation in percentage (RPD) and is concerned with the investigation of resolution algorithm convergence for medium instances. Eq. (19) provides this parameter of the RPD:

$$RDP = \left(\frac{F^{algo} - F^{best}}{F^{best}}\right) \times 100\% \#(19)$$

Knowing that Fbest is the optimum objective function value among all algorithms and that FAlgo is the function value for a specific algorithm in this instance. One should perform a set of tests and keep an average of 10 instances for each test problem, to create a good calibration to measure the required performances. In this type of research, a statistical analysis is an important tool for verifying the best method for solving the problem out of all the ones that have been proposed.

In this study, we are interested in comparing the results of our heuristics and the MILP for small instances. We examined forty instances, each with ten cases that varied based on the value of β, resulting in a total of 440 problems. The instances for a specific size, with n machines and m being the number of machines, are defined as combinations of m × n. Table 2 shows the objective function values for different instances. The weighting factor β is used to calculate the results of the objective function criteria. Based on our initial analysis and by comparing with the exact answer provided by the MILP, we can conclude that the NEH algorithm produces better results. Furthermore, we observe that the MILP provides the best results within seconds, closely matching the NEH heuristic answer. It is also important to note that when m = 2 and the parameter β = 1 when the problem can note it by Fm|prmu,SIST|$C_{max}$, the heuristics findings are identical to those provided by the MILP. The findings recorded in Table 3 further demonstrate the superiority of the NEH algorithm. We present in Fig. 2 (a) the ANOVA diagrams show the statistical study findings for the m = 5 instance. We take into account 10 replications for each instance by changing the coefficient so that it is either δ = 0 and equivalent to Fm|prmu, SIST| $\sum_{j=1}^{n} \mathrm{Ct}_{m,j}$ *the* analysis is then done on the median RPD value, the NEH heuristic give the best RPD. Similarly, in Fig. 2 (b) the mean plot is given for δ = 0.6 for the m = 3. The NEH heuristic is better RPD compared to other heuristics algorithms. From the last analysis, we have confirmed that NEH algorithm is an effective method compared with EJR, GRASP, and PR heuristics for solving the PFSSP-SIST bi-objective scheduling problem.

**Table 2: Objective function value of small instances**

| n×m | RPD | β | | | | | |
|---|---|---|---|---|---|---|---|
| | | 0 | 0.2 | 0.4 | 0.6 | 0.8 | 1 |
| 5×2 | MILP | 482 | 416 | 350 | 284 | 218 | 153 |
| | NEH | 482 | 416 | 350 | 284 | 218 | 153 |
| | GRASP | 530 | 462 | 360 | 330 | 266 | 165 |
| | EJR | 560 | 457 | 364 | 318 | 225 | 161 |
| | PR | 558 | 451 | 364 | 293 | 221 | 153 |
| 10×2 | MILP | 1358 | 1146 | 932 | 715 | 498 | 281 |
| | NEH | 1414 | 1190 | 984 | 749 | 521 | 281 |
| | GRASP | 1597 | 1195 | 982 | 904 | 612 | 299 |
| | EJR | 1573 | 1282 | 1130 | 825 | 577 | 300 |
| | PR | 1579 | 1292 | 1041 | 825 | 577 | 299 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 5×3 | MILP | 664 | 574 | 484 | 394 | 299 | 205 |
| | NEH | 665 | 574 | 484 | 394 | 299 | 205 |
| | GRASP | 721 | 660 | 513 | 477 | 340 | 224 |
| | EJR | 778 | 626 | 515 | 456 | 354 | 214 |
| | PR | 778 | 577 | 515 | 409 | 326 | 214 |
| 10×3 | MILP | 1998 | 1674 | 1351 | 1027 | 704 | 381 |
| | NEH | 2050 | 1716 | 1382 | 1048 | 714 | 381 |
| | GRASP | 2260 | 1891 | 1549 | 1073 | 739 | 419 |
| | EJR | 2386 | 1855 | 1572 | 1174 | 828 | 409 |
| | PR | 2386 | 1855 | 1413 | 1117 | 828 | 409 |
| 5×5 | MILP | 916 | 780 | 645 | 509 | 374 | 237 |
| | NEH | 916 | 780 | 645 | 509 | 374 | 239 |
| | GRASP | 962 | 863 | 735 | 591 | 437 | 283 |
| | EJR | 979 | 822 | 686 | 543 | 410 | 263 |
| | PR | 930 | 813 | 686 | 534 | 401 | 259 |
| 10×5 | MILP | 2280 | 1916 | 1542 | 1155 | 767 | 360 |
| | NEH | 2321 | 1928 | 1542 | 1155 | 767 | 360 |
| | GRASP | 2544 | 2188 | 1708 | 1356 | 875 | 379 |
| | EJR | 2668 | 2222 | 1826 | 1412 | 879 | 371 |
| | PR | 2668 | 2159 | 1708 | 1400 | 863 | 380 |
| 15×3 | MILP | 3570 | 2952 | 2334 | 1716 | 1098 | 480 |
| | NEH | 3798 | 3136 | 2474 | 1812 | 1171 | 480 |
| | GRASP | 4155 | 3619 | 2650 | 2084 | 1192 | 483 |
| | EJR | 4158 | 3620 | 2845 | 2116 | 1268 | 495 |
| | PR | 4156 | 3620 | 2776 | 2040 | 1268 | 495 |
| 15×5 | MILP | 4118 | 3426 | 2701 | 1967 | 1243 | 455 |
| | NEH | 4282 | 3531 | 2780 | 2030 | 1290 | 470 |
| | GRASP | 4829 | 4141 | 3049 | 2183 | 1423 | 500 |
| | EJR | 4877 | 4078 | 3149 | 2152 | 1429 | 494 |
| | PR | 4609 | 3819 | 3146 | 2152 | 1429 | 481 |

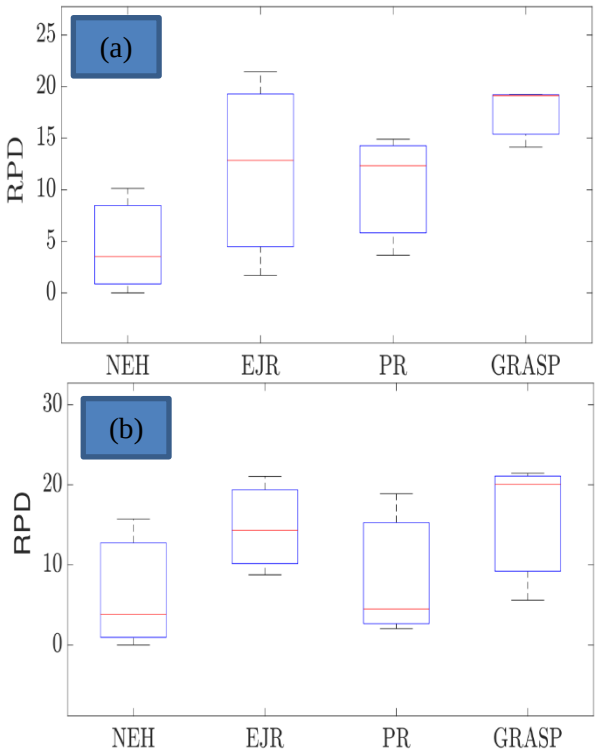


**Figure 2 : Box plot of RPD for different heuristics.**

**Table 3 : RPD value of small instances**

| n×m | RPD | β (×10-2) | | | | | |
|---|---|---|---|---|---|---|---|
| | | 0 | 0.2 | 0.4 | 0.6 | 0.8 | 1 |
| 5×2 | NEH | 0 | 0 | 0 | 0 | 0 | 0 |
| | GRASP | 995 | 1105 | 285 | 1619 | 2201 | 784 |
| | EJR | 1618 | 985 | 400 | 1197 | 321 | 522 |
| | PR | 1576 | 841 | 400 | 316 | 137 | 0 |
| 10×2 | NEH | 412 | 383 | 557 | 475 | 461 | 0 |
| | GRASP | 1759 | 427 | 536 | 1258 | 2289 | 640 |
| | EJR | 1583 | 1186 | 2124 | 1538 | 1568 | 676 |
| | PR | 1627 | 1273 | 1169 | 1538 | 1568 | 640 |
| 5×3 | NEH | 15 | 0 | 0 | 0 | 0 | 0 |
| | GRASP | 858 | 1498 | 599 | 2106 | 1371 | 926 |
| | EJR | 1716 | 905 | 640 | 1573 | 1839 | 439 |
| | PR | 1716 | 52 | 640 | 380 | 903 | 439 |
| 10×3 | NEH | 260 | 250 | 229 | 204 | 142 | 0 |
| | GRASP | 1311 | 1296 | 1465 | 447 | 497 | 997 |
| | EJR | 1941 | 1081 | 1635 | 1431 | 1761 | 734 |
| | PR | 1941 | 1081 | 458 | 876 | 1761 | 734 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 5×5 | NEH | 0 | 0 | 0 | 0 | 0 | 0 |
| | GRASP | 502 | 1064 | 1395 | 1611 | 1684 | 1940 |
| | EJR | 687 | 538 | 635 | 667 | 962 | 1097 |
| | PR | 152 | 423 | 635 | 667 | 962 | 928 |
| 10×5 | NEH | 179 | 62 | 0 | 0 | 0 | 0 |
| | GRASP | 1157 | 1419 | 1076 | 1740 | 1408 | 527 |
| | EJR | 1701 | 1597 | 1841 | 5525 | 1460 | 305 |
| | PR | 1701 | 1268 | 1076 | 2121 | 1251 | 555 |
| 15×3 | NEH | 638 | 623 | 599 | 559 | 664 | 0 |
| | GRASP | 1638 | 2255 | 1353 | 2144 | 856 | 62 |
| | EJR | 1647 | 2262 | 2189 | 2004 | 1548 | 312 |
| | PR | 1641 | 2262 | 1893 | 1888 | 1548 | 312 |
| 15×5 | NEH | 398 | 306 | 292 | 320 | 378 | 392 |
| | GRASP | 1726 | 2086 | 1288 | 1098 | 1448 | 989 |
| | EJR | 1843 | 1903 | 1658 | 940 | 1496 | 769 |
| | PR | 1192 | 1147 | 1647 | 940 | 1496 | 571 |

## 5 CONCLUSION

This work focuses on finding the optimal solution to the flow shop scheduling problem with sequence independent setup time. To this goal, we suggested two methods: an exact approach and a set of heuristics. The makespan and the total flow time, each of which has a weighting coefficient, have been combined to form an objective function. The first technique uses mixed integers to formulate mathematics, whereas the second way is based on heuristics Johnson's Rules algorithm, Nawaz Enscore and Ham heuristic, Greedy Randomized Adaptive Search Procedure, and Path Relinking. To achieve this, we have made adjustments to the number of machines, jobs, and the weighting parameter of the problem. This ensures that one criterion is appropriately favored over the other. We conducted simulations using a series of short sequences to evaluate the effectiveness of the approximation methods. The results indicate that, among the proposed heuristics, the NEH algorithm is the most effective. In future work, we plan to tackle larger instances of the problem by employing different techniques such as metaheuristics. Additionally, we will consider other constraints commonly encountered in modern industry, such as no-wait and blocking scenarios. By incorporating these factors, we aim to further enhance our approach's applicability to real-world industrial settings.

## 6 REFERENCES

Belabid J, Aqil S, Allali K (2020) Solving permutation flow shop scheduling problem with sequenceindependent setup time. Journal of Applied Mathematics 2020:1–11.

Gmys J, Mezmaz M, Melab N, Tuyttens D (2020) A computationally efficient branch-and-bound algorithm for the permutation flow-shop scheduling problem. European Journal of Operational Research 284(3):814–833.

Johnson SM (1954) Optimal two-and three-stage production schedules with setup times included. Naval research logistics quarterly 1(1):61–68.

Kim HJ, Lee JH (2019) Three-machine flow shop scheduling with overlapping waiting time constraints. Computers & Operations Research 101:93–102.

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

Prabhaharan G, Khan BSH, Rakesh L (2006) Implementation of grasp in flow shop scheduling. The International Journal of Advanced Manufacturing Technology 30:1126–1131.

Rajendran C (1993) Heuristic algorithm for scheduling in a flowshop to minimize total flowtime. International Journal of Production Economics 29(1):65–73.

Rossi FL, Nagano MS, Sagawa JK (2017) An effective constructive heuristic for permutation flow shop scheduling problem with total flow time criterion. The International Journal of Advanced Manufacturing Technology 90:93–107.

Ruiz R, Maroto C (2005) A comprehensive review and evaluation of permutation flowshop heuristics. European journal of operational research 165(2):479–494.

Sadki H, Aqil S, Belabid J, Allali K (2023) Multi-objective optimization flow shop scheduling problem solving the makespan and total flow time with sequence independent setup time. Journal of Advanced Manufacturing Systems pp 1–22.

Saravanan M, Vijayakumar S, Srinivasan R, Singarayar SP (2015) Scheduling to minimize the sum of weighted total flow time and makespan in a permutation flow shop with setup time. In: Applied Mechanics and Materials, Trans Tech Publ, vol 766, pp 989–994.

Shao Z, Pi D, Shao W (2018) Estimation of distribution algorithm with path relinking for the blocking flow-shop scheduling problem. Engineering Optimization 50(5):894–916.

Srikar BN, Ghosh S (1986) A milp model for the n-job, m-stage flowshop with sequence dependent set-up times. International Journal of Production Research 24(6):1459–1474.

Stafford E, Tseng FT, Gupta JN (2005) Comparative evaluation of milp flowshop models. Journal of the Operational Research Society 56:88–101.

Taillard E (1990) Some efficient heuristic methods for the flow shop sequencing problem. European journal of Operational research 47(1):65–74.

Tseng FT, Stafford Jr EF (2008) New milp models for the permutation flowshop problem. Journal of the Operational Research Society 59(10):1373–1386.

Vallada E, Ruiz R (2010) Genetic algorithms with path relinking for the minimum tardiness permutation flowshop problem. Omega 38(1-2):57–67.

Zobolas G, Tarantilis CD, Ioannou G (2009) Minimizing makespan in permutation flow shop scheduling problems using a hybrid metaheuristic algorithm. Computers & Operations Research 36(4):1249–1267.