



Artificial neural network for solving flow shop optimization problem with sequence independent setup time

Hajar Sadki* and Karam Allali

Abstract. In this paper, we will study the permutation flow shop scheduling problem (PFSSP) with sequence independent setup time (SIST). This constraint is the most common encountered in industrial production. 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. The purpose of this paper is to develop an artificial intelligence system and to train a neural network model for solving the flow shop scheduling problem. The objective function is to minimize the total completion time, which is known as makespan. The latter is an important task in manufacturing systems. The paper begins by suggesting an exact and four approximate methods: a mixed integer linear programming (MILP), an artificial neural network (ANN), and three efficient heuristics. The first heuristic is based on Johnson's rule algorithm (ABJR), the second on the Nawaz-Enscore and Ham algorithm (NEH), and the last on the greedy randomized adaptive search procedure algorithm (GRASP). We aim to verify the effectiveness of our resolution algorithms by considering randomly generated instances with n jobs and m machines in the flow shop factory. Our goal is to determine the optimal sequence of n jobs to be scheduled on m machines. The paper moves to the comparison between the studied heuristics. The numerical results demonstrate that the NEH algorithm outperforms the other approximate methods for our considered problem. Indeed, the NEH heuristic performs a success rate of 82.81% and achieves a minimum relative percentage deviation value of 0.0139%. It was observed that ANN method outperforms GRASP and gives sometimes best results than ABJR. The numerical simulations align with our theoretical postulations given by *RPD* values.

Keywords: Flow shop scheduling, Sequence independent setup time, Artificial neural network, Makespan, Heuristic

*Laboratory of Mathematics, Computer Science and Applications, FST Mohammedia, University Hassan II of Casablanca, PO Box 146, Mohammedia, Morocco, Sadkisadkihajar@gmail.com

1. Introduction

The area of research on the permutation flow shop scheduling problem (PFSSP) has been the subject of many studies for the last seven decades. Production scheduling is often considered as a critical aspect in planning various operations within manufacturing systems [17, 21]. It is commonly encountered in large-scale industries and includes multiple constraints related to the nature of jobs and the characteristics of the machines in the industrial context. Some important constraints include machine adjustments and the prevention of no-waiting or job blocking within machine queues during manufacturing [37, 38]. Flow shop scheduling involves organizing n jobs to be completed on m machines in a specific order to optimize performance metrics, such as minimizing the total completion time or the maximum tardiness. Flow shop scheduling is commonly employed in mechanical manufacturing and industrial production, where jobs are continuously processed across a series of machines. In a flow shop problem, jobs are processed in a predetermined sequence on a set of machines. Each machine can handle only one job at a time, and each job can be processed by only one machine. Figure 1 illustrates a set of jobs being processed in sequential machines in a flow shop industrial factory. The permutation flow shop scheduling problem with sequence independent setup time (SIST) constraint is a crucial problem in industrial optimization. The SIST constraint is influenced by the technological characteristics of the machine and the preparations required before executing a new job. It is essential to consider the setup time needed for new jobs, as machines often require adjustments or cleaning. Although setup time is considered unproductive, incorporating it into the scheduling process is crucial for effective process control.

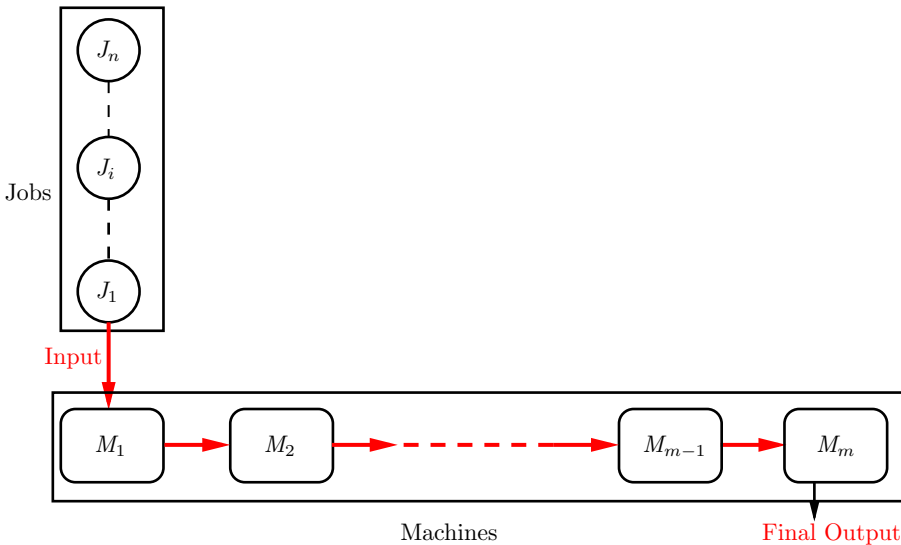


Figure 1. A flow shop scheduling problem with n jobs and m machines.

In the literature review, several researches from various disciplines have significantly contributed to the flow shop scheduling community. For instance, in [26], the authors introduced a sequence right-shift and a real-time strategy in their study, aimed at solving the multiple-order permutation flow shop scheduling problem to minimize the wastage of human resources. Also, the use of wireless body area networks for health monitoring, which has become increasingly important during the COVID-19 disease. While, the authors in [3, 5] represent an attractive area of research that has gained significant attention in recent years. This technology has the potential to revolutionize healthcare by enabling continuous and non-invasive monitoring of vital signs, thereby facilitating early detection of health issues and enhancing patient outcomes [8]. In the study conducted by [24], the work focused on robust time-discrete target-environment regulatory systems with polyhedral uncertainty and proposed a novel robust regression method for regulatory networks. Additionally, in [32] researchers explored two optimal investment problems using zero-sum and nonzero-sum stochastic game approaches. The latter leads to the derivation of regime-switching Hamilton–Jacobi–Bellman–Isaacs equations. In addition, the concept of target-environment networks in [19, 36] involved the interplay between genes and environmental factors, where both can influence each other. This reciprocal influence creates a dynamic relationship that impacts various biological processes.

On the other hand, the artificial neural network (ANN) approach is one of the methods used to address the permutation flow shop scheduling problem. This approach finds different applications in various fields that require pattern recognition, classification, medical analysis, or optimization tasks. In [16], the authors analyze the efficiency of the SFLA-ANN method compared to the SA-ANN and GA-ANN hybrid methodologies. Artificial neural network approaches are frequently employed for modeling and analyzing scheduling problems. In [2], artificial neural networks were used to model six different heuristic algorithms for solving the flow shop scheduling problem with m machines, with the goal of minimizing the makespan. Similarly, [1] proposed a technique that utilizes neural networks to solve the job scheduling problem on m machines and n jobs while considering the transport time between different jobs. Furthermore, In [22], an ANN approach was used to optimize the makespan for a problem involving two machines and n jobs using Johnson’s algorithm to generate an optimal solution. In [28], the authors aimed to develop an artificial intelligence capable of performing scheduling tasks through the process of neural network training. Also, in [27] the authors proposed two approaches in which the ANN generates priorities of a scheduling action, which is subsequently optimized by a heuristic or genetic algorithm (GA). In our research, we aim to utilize a neural network-based method to generate sequences for scheduling jobs on multiple machines in a flow shop. Our objective is to develop a simple, intuitive, and efficient neural network approach for real-time sequencing decisions in a flow shop scheduling. Also, our purpose is to compare the results obtained from this approach with different constructive sequencing heuristics. The pioneering work in this field was conducted by Johnson in 1954 [18], where an optimal solution was provided for the case of only two machines. However, solving the same problem with more than two machines is considered to be more challenging due to its classification as *NP-hard* [11]. In real-life applications, the permutation

flow shop scheduling problem becomes more complex due to a larger set of jobs and machines involved. As a result of its complexity, solving the PFSSP can be extremely challenging. Many researchers have faced significant difficulties when attempting to solve the problem using various traditional methods. Similarly, numerous researchers have recently written about new evolutionary algorithms, with a particular focus on scheduling and production problems. These writings serve as valuable guidance for future research. One important suggestion is to focus on robust optimization approach [12, 13, 15]. Additionally, we can further explore the effectiveness of linear approximation and more advanced metaheuristic algorithms in [4, 6, 14]. These approaches can provide optimal solutions to large-scale problems in a reasonable amount of time.

In our contribution, we study the analysis of the PFSSP-SIST scheduling problem with the objective of minimizing the makespan. We propose multiple approaches, including mixed integer linear programming (MILP), artificial neural networks (ANN), and three heuristics: the Nawaz-Enscore-Hamm (NEH) algorithm, a heuristic based on the Johnson rule (ABJR) algorithm, and the greedy randomized adaptive search procedure (GRASP) algorithm. The goal of this aforementioned methods and strategies for flow shop scheduling is to find the optimal sequence of jobs to achieve the problem's objective. Performance evaluation is often done using Gantt charts to assess the results visually. However, managing Gantt charts can become challenging when dealing with a large number of tasks and machines, as it involves presenting the start and end times of jobs on different machines and the completion time of all jobs. Exact mathematical methods, which are the first choice for solving PFSSP problems, have attracted significant attention from researchers due to their practical utility. Recent studies proposed the MILP models for the PFSSP-SIST problems [9, 30]. More precisely, in [31], a MILP model and constructive heuristics were used to solve the PFSSP-SIST, with the objective of minimizing the total flow time $F_{\max}$, $\sum C_{\max}$, $\sum T_{\max}$. The second class of methods used to solve this type of PFSSP is approximate methods. In [34] an algorithm has been proposed to improve the NEH to achieve a minimum makespan. While [7] aimed to enhance the makespan by developing the NEH algorithm. Additionally, we will employ the Greedy Random Adaptive Search (GRASP) [25] method to solve the PFSSP with a specific number of jobs and machines. In [23], a comparative study was conducted to evaluate the performance of GRASP in comparison to the NEH heuristic by solving reference problems.

This paper's remaining sections are organized as follows: Section 2 presents the permutation flow shop scheduling problem with sequence independent setup time. Section 3 illustrates a mixed integer linear programming model (MILP) and the suggested heuristics, we also describe the artificial neural network's architecture and how it was applied to resolve the permutation flow shop scheduling problem under sequence independent setup time. Section 4 shows the analysis of the computational experiments through a comparative study. Finally, Section 5 is a conclusion with a few comments and a prediction for future study.

2. The description of the flow shop scheduling problem

The permutation flow shop scheduling problem with sequence independent setup time (PFSSP-SIST) is a issue faced by business managers in the manufacturing industry. The issue involves a set of jobs $J = \{J_1, J_2, \dots, J_n\}$, that needs to be executed on a set of machines $M = \{M_1, M_2, \dots, M_m\}$. These machines are essential for the manufacturing process of the jobs. The jobs are processed on the machines in a specific order $\sigma = (\sigma_1, \dots, \sigma_n)$, adhering to the assumptions of the traditional PFSSP problem. In this problem, each machine can handle only one job at a time, no interruptions are allowed, and each job has multiple predecessors and successors. Additionally, all the machines are available at the beginning of the scheduling process. Throughout the study, we utilized various notations to solve the PFSSP-SIST scheduling problem. These notations are listed in Table 1.

Table 1. Usage notation.

Notation	Description
n	The size of the input data of the jobs to be scheduled.
m	The size of the input data of the machines.
$p_{i,j} \geq 0$	The processing time of the job J_j in the machines M_i , where $(i=1, \dots, m)$ and $(j=1, \dots, n)$.
S_i	The setup time is independent of the sequence and a preparation time for each machine M_i .
σ	All of the jobs are in sequence where $\sigma = \{\sigma_1, \dots, \sigma_n\}$.
CT	The maximum completion time: makespan.
CT_{i,σ_j}	The completion time of the job σ_j in the i^{th} machine.
Π	The probability of selecting a job.
δ	The parameter and its value range from zero to one.
RCL	Restricted candidate list.
q	The input nodes.
X_i	The input neurons.
L_{in}	The values of the hidden neurons inputs.
L_k	The hidden neurons output values.
Z_{in}	The input values for the hidden neurons output.
Z_k	The final output value.
W	The weights of the connections links.
G	The function activation.

Because the goal is to reduce the completion time of all jobs the following equations are used to compute $C_{\max}$:

$$CT_{1,\sigma_1} = p_{1,\sigma_1} + S_1, \quad (1)$$

$$CT_{1,\sigma_j} = CT_{1,\sigma_{j-1}} + p_{1,\sigma_j} + S_1, \quad (j = 2, \dots, n), \quad (2)$$

$$CT_{i,\sigma_1} = \max \{S_i, CT_{i-1,\sigma_1}\} + p_{i,\sigma_1}, \quad (i = 2, \dots, m), \quad (3)$$

$$CT_{i,\sigma_j} = \max \{CT_{i,\sigma_{j-1}} + S_i, CT_{i-1,\sigma_j}\} + p_{i,\sigma_j}, \quad (i = 2, \dots, m), (j = 2, \dots, n), \quad (4)$$

$$C_{\max} = \max_{1 \leq j \leq n} \{CT_{m,\sigma_j}\}. \quad (5)$$

Eqs. (1) and (2) calculate the completion time of the job (σ_1, σ_j) on the first machine. Eqs. (3) and (4) facilitate the determination of the end date of the first job and the end dates of all other jobs on all other machines in a sequential manner. Eq (5) is used to determine the makespan, $C_{\max}$. The goal of this approach is to identify an optimal sequence, σ^* , from the set permutation Π , such as :

$$C_{\max}(\sigma^*) \leq C_{\max}(\sigma), \quad \sigma \in \Pi. \quad (6)$$

Job	:	J_1	J_2	J_3	J_4	S_i
M_1	:	4	5	3	6	1
M_2	:	3	4	6	5	2
M_3	:	3	6	4	3	1

Table 2. The processing time of four jobs on three machines.

2.1. Illustration with numbers

To highlight the proposed model (PFSSP-SIST), we present an illustrative example that considers an instance consisting of four jobs and three machines, forming our workshop. For this problem, the goal is to find a good permutation sequence $\sigma = \{\sigma_1, \sigma_2, \sigma_3, \sigma_4\}$ among the $4!$ achievable sequences. The objective is to minimize the job production time and determine each job's start and end times on each machine. We use Table 2 to illustrate the processing time for each of the four jobs $\sigma = \{J_1, J_2, J_3, J_4\}$ on each of the three machines. We need to schedule the sequence $\sigma = (J_3, J_1, J_2, J_4)$. Using the Eqs. (1) and (4), the following expressions provide a detailed analysis of the departure dates for each job performed by each machine in our sequence: $CT_{1,\sigma_1} = p_{1,\sigma_1} + S_1 = 3 + 1 = 4$, $CT_{1,\sigma_2} = CT_{1,\sigma_1} + p_{1,\sigma_2} + S_1 = 4 + 4 + 1 = 9$, $CT_{1,\sigma_3} = CT_{1,\sigma_2} + p_{1,\sigma_3} + S_1 = 9 + 5 + 1 = 15$, $CT_{1,\sigma_4} = CT_{1,\sigma_3} + p_{1,\sigma_4} + S_1 = 15 + 6 + 1 = 22$, $CT_{2,\sigma_1} = \max\{S_2, CT_{1,\sigma_1}\} + p_{2,\sigma_1} = \max\{2, 4\} + 6 = 10$, $CT_{3,\sigma_1} = \max\{S_3, CT_{2,\sigma_1}\} + p_{3,\sigma_1} = \max\{1, 10\} + 4 = 14$, $CT_{2,\sigma_2} = \max\{CT_{2,\sigma_1} + S_2, CT_{1,\sigma_2}\} + p_{2,\sigma_2} = \max\{10 + 2, 9\} + 3 = 15$, $CT_{2,\sigma_3} = \max\{CT_{2,\sigma_2} + S_2, CT_{1,\sigma_3}\} + p_{2,\sigma_3} = \max\{15 + 2, 15\} + 4 = 21$, $CT_{2,\sigma_4} = \max\{CT_{2,\sigma_3} + S_2, CT_{1,\sigma_4}\} + p_{2,\sigma_4} = \max\{21 + 2, 22\} + 5 = 28$, $CT_{3,\sigma_2} = \max\{CT_{3,\sigma_1} + S_1, CT_{2,\sigma_3}\} + p_{3,\sigma_2} = \max\{14 + 1, 15\} + 3 = 18$, $CT_{3,\sigma_3} = \max\{CT_{3,\sigma_2} + S_1, CT_{2,\sigma_3}\} + p_{3,\sigma_3} = \max\{18 + 1, 21\} + 6 = 27$, $CT_{3,\sigma_4} = \max\{CT_{3,\sigma_3} + S_1, CT_{2,\sigma_4}\} + p_{3,\sigma_4} = \max\{17 + 1, 28\} + 3 = 31$. Knowing that the four jobs were completed on the last machine, we can determine when they ended : $CT_{3,\sigma_1} = 14$, $CT_{3,\sigma_2} = 18$, $CT_{3,\sigma_3} = 27$, $CT_{3,\sigma_4} = 31$. Additionally, the makespan value $C_{\max} = 31$ time units. In the examined example, the three machines were scheduled using a Gantt chart with sequence independent setup time as the constraint, and the results are shown in Fig. 2.

2.2. Managerial implications

In this paper, we study the permutation flow shop scheduling problem under the constraint of sequence independent setup time. 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, its introduction into the scheduling is of great use for the control of the process. The setup time may be dependent on the nature of the jobs; hence any job may require a

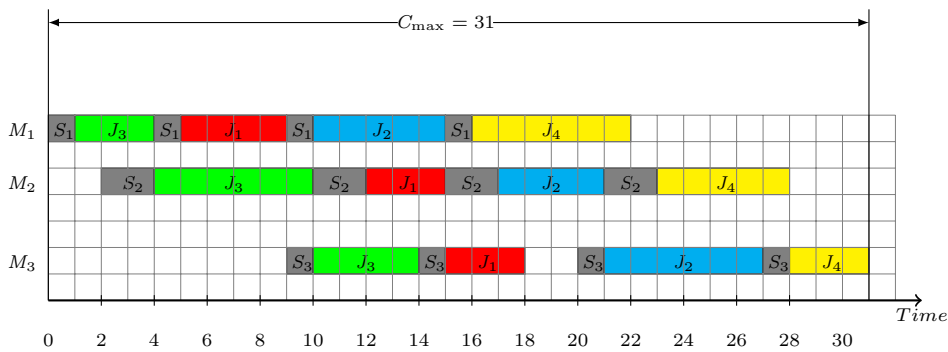


Figure 2. Gantt chart for $\sigma = (J_3, J_1, J_2, J_4)$ jobs in three machine.

certain setup time before it can be processed in the machine. This type of problem is exciting 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 setup time can be interpreted in the same industrial cases as cleaning, parts assembly-disassembly, or maintenance of a certain machine. The setup time may be also independent of jobs nature. An example of this situation is when the setup time depends of the nature of the machine instead of jobs. The interest of PFSSP-SIST scheduling study is when the manager fix automatically the setup time of each working machine in order to process the job arrivals. The interpretation of this setup time is when the machines need to be cleaned, parts assembled-disassembled and/or maintained. In fact this setup time may be the subject of several cases such as assembly and dis-assembly of parts or tools in the machine, cleaning and evacuation of production means, etc. These activities can be carried out by robots or manipulators independently 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. This case is inspired by the production of plastic material. The manufacture of plastic in 5 steps: naphtha, the basic ingredient of plastic, cracking, polymerization, shaping and recycling of plastic waste. these steps presented in Fig. 3

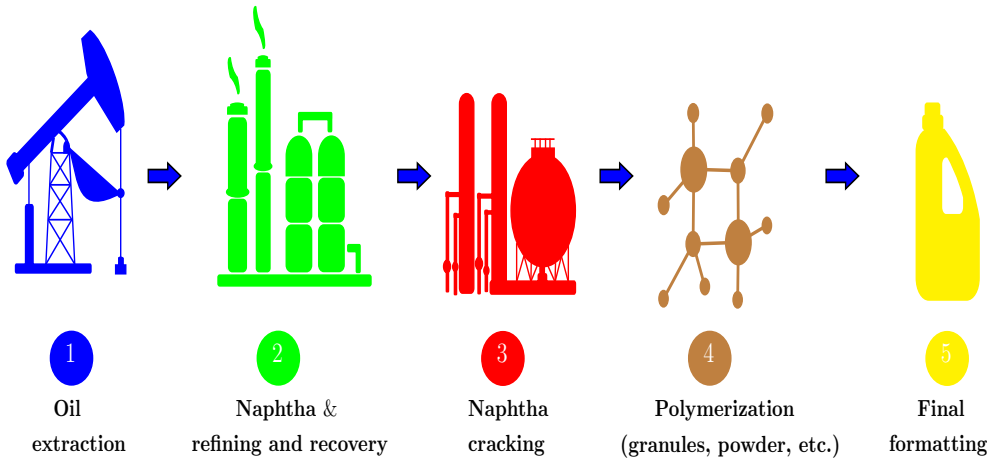


Figure 3. The manufacture of plastic in 5 steps.

3. Resolution approaches

3.1. The mixed integer linear programming model approach

In recent years, one of the important areas of focus in operations research has been the use of linear programming to model scheduling problems with constraints. These models provide a precise approach by utilizing optimization solvers. Among various mathematical techniques, mixed integer linear programming (MILP) has proven to be highly effective. Numerous MILP models have been proposed for different case studies of the permutation flow shop scheduling problem [35]. The MILP model for the PFSSP, aiming to minimize the makespan, consists of the following variables:

$$y_{k,j} = \begin{cases} 1, & \text{if the job } j \text{ is placed in the order } k^{th} \text{ location in the sequence} \\ 0, & \text{otherwise} \end{cases}$$

$$(j, k = 1, \dots, n).$$

The purpose is to :

$$\text{minimize } C_{\max} = CT_{m,n} \quad (7)$$

Subject to :

$$CT_{1,1} \geq S_1 + \sum_{j=1}^n y_{1,j} \times p_{1,\sigma_1}, \quad (8)$$

$$CT_{1,k} \geq S_1 + CT_{1,k-1} + \sum_{j=1}^n y_{k,j} \times p_{1,\sigma_j}, \quad (k = 2, \dots, n), \quad (9)$$

$$CT_{i,k} \geq CT_{i-1,k} + \sum_{j=1}^n y_{j,k} \times p_{i,\sigma_j}, \quad (i = 2, \dots, m), \quad (k = 1, \dots, n), \quad (10)$$

$$CT_{i,k} \geq S_i + CT_{i,k-1} + \sum_{j=1}^n y_{j,k} \times p_{i,\sigma_j}, \quad (i = 1, \dots, m), \quad (k = 1, \dots, n), \quad (11)$$

$$\sum_{j=1}^n y_{k,j} = 1, \quad (k = 1, \dots, n), \quad (12)$$

$$\sum_{k=1}^n y_{k,j} = 1, \quad (j = 1, \dots, n), \quad (13)$$

$$C_{\max} > CT_{m,k}, \quad (k = 1, \dots, n), \quad (14)$$

$$CT_{i,k} \geq 0, \quad (i = 1, \dots, m), \quad (k = 1, \dots, n), \quad (15)$$

$$y_{j,k} \in \{0, 1\}, \quad (j, k = 1, \dots, n). \quad (16)$$

The main objective of the function is to reduce the makespan (7). The initial job's completion time on the first machine is estimated by constraint set (8). According to constraint set (9), no job on the first machine can be processed before its predecessor on the same machine. The similarity constraint set (10) ensures that each machine can only process a job after the previous machine has completed processing it. The constraint set (11) states that each job must wait until the processing of its predecessor job on the same machine is finished before it may start. Constraints set (12), (13) each job must be assigned to a position, which also limits the number of jobs that can occupy each position to one. Constraint set (14) is used to determine the completion time $C_{\max}$. Formulations (15) and (16) specify the domain of decision variables.

3.2. The artificial neural network model approach

3.2.1. Description of the artificial neural network

Over the years, numerous complex tools have been developed to assist humanity, making human existence simpler and more comfortable. Among these tools, digital computers and smart machines have played a significant role. Researchers have been inspired by the neural network of the human brain [33, 10] and have created artificial neural networks (ANNs). Neural networks are simplified models of the biological nerve structure and are designed to mimic the calculations performed by the human brain. An ANN is composed of interconnected processing units called neurons, which can comprehend, learn, and utilize knowledge.

The fundamental model of an ANN consists of three layers: the input layer, the hidden layer, and the output layer. Similar to the human brain, a neuron in an ANN receives input and transmits it to another neuron. These neurons are connected by links, and each link has a weight that regulates the signal between the

nodes/neurons. Furthermore, the structure of the ANN is influenced by the information passing through the network, as neural networks are capable of learning based on input and output. Figure 4 illustrates a simple model that accurately represents the behavior of a neuron.

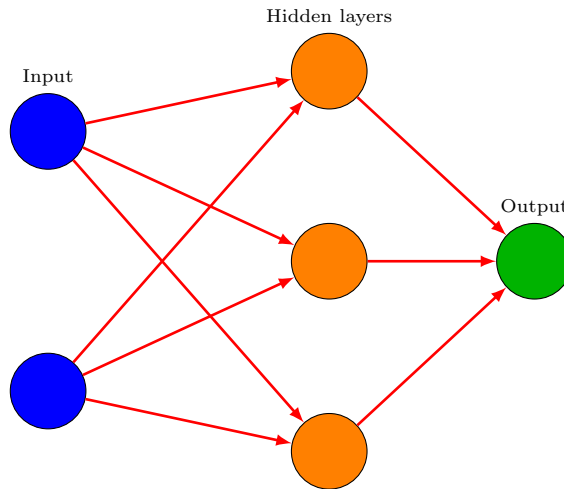


Figure 4. An ANN's basic structure

3.2.2. The ANN model of the flow shop scheduling problem

Based on the description of the flow shop scheduling problem provided above, the makespan of a specific flow shop scheduling problem is influenced by various factors, including the number of jobs $J = \{J_1, J_2, \dots, J_n\}$, the number of machines $M = \{M_1, M_2, \dots, M_m\}$, the processing times at each machine $p_{i,j}$, and the setup time S_i . Integrating all these variables into a flow shop ANN model would require creating and training a separate ANN for each problem instance. The objective is to determine a job processing sequence on the machines that minimizes the total completion time or makespan ($C_{\max}$) of the schedule. Instead of using the average and standard deviation of processing times at each machine, we suggest utilizing the actual processing time of a single job at each machine as the specific inputs for our ANN model of the flow shop scheduling problem. The ANN model's structure would depend on the number of machines employed in a given business. However, this approach is acceptable because, in reality, the number of machines is unlikely to change significantly in the short- to medium-term. In [20], the authors employed this method to simulate and address the flow shop scheduling problem with ANNs. Based on their research, the following list of steps were developed to train the neural network and create an ANN model for a flow shop scheduling problem.

Step 1: Establish the neural network's architecture.

This article discusses a neural network with three layers: an input layer, a hidden layer, and an output layer. The proposed neural network architecture includes two hidden layers. This design requires distinct and individually trained networks for flow shop with different machine counts, where the input layer size is determined by the number of machines. To ensure compatibility with the neural network, the input patterns must have values ranging from zero to one, necessitating data normalization. In this study, the considered flow shop has minimum processing times of 1 minute and maximum processing times of 49 minutes. The divisor of 100 in Eq. (17) is used to normalize the processing times within the range of (1, 49). Each job's processing time and scheduled jobs are represented as a vector with several elements.

The input layer of the proposed neural network includes three times the number of machines for each job. This vector's size ($3 \times m$) accounts for three times as many machines in the proposed neural network's input layer, and they are distributed as follows:

- The processing times of job on each of the m machines are contained in the first m nodes.
- The average processing time requirements for each of the m machines are contained in the middle m nodes.
- The standard deviation of the processing times for each of the m machines are contained in the last m nodes.

The log sigmoidal transfer function utilized in these training techniques has a range of values for the single output node between 0.1 and 0.9.

Step 2: Describe the value of each ANN input node.

The following equations are used to calculate the values that the input layer nodes assume when a job j is delivered to the network :

$$\text{Node}_q = \begin{cases} \frac{p_{i,q} + S_q}{100}, & (q = 1, \dots, m), \\ \frac{\frac{1}{n} \sum_{i=1}^n p_{i,q-m} + S_{q-m}}{100}, & (q = m+1, \dots, 2m), \\ \sqrt{\frac{Y_{(q-2m)} - n\bar{p}_{(q-2m)}^2}{(n-1) \times 10^4}}, & (q = 2m+1, \dots, 3m), \end{cases} \quad (17)$$

where :

$$\bar{p}_{(k)} = \frac{1}{n} \sum_{i=1}^n p_{i,k} + S_k, \text{ where } (k = q - m), \quad (18)$$

and :

$$Y_{(r)} = \sum_{i=1}^n (p_{i,k} + S_k)^2, \text{ where } (r = q - 2m) \quad (19)$$

Eq. (17), utilizes a trained network to construct a sequence by creating input patterns that represent the n jobs. These patterns are then fed into the trained network's input layer one at a time. Each input pattern generates an output value. After processing all the input patterns and sorting them in ascending order, the resulting values and the corresponding jobs are noted. The jobs connected to each output are arranged in the order recommended by the neural network. This approach can also be applied to more complex problems with a larger set of jobs and machines. The method for training large networks using a multi-layer neural network architecture is straightforward to understand and implement. The working process architecture of the proposed neural model for the flow shop scheduling problem is depicted in Figure 5.

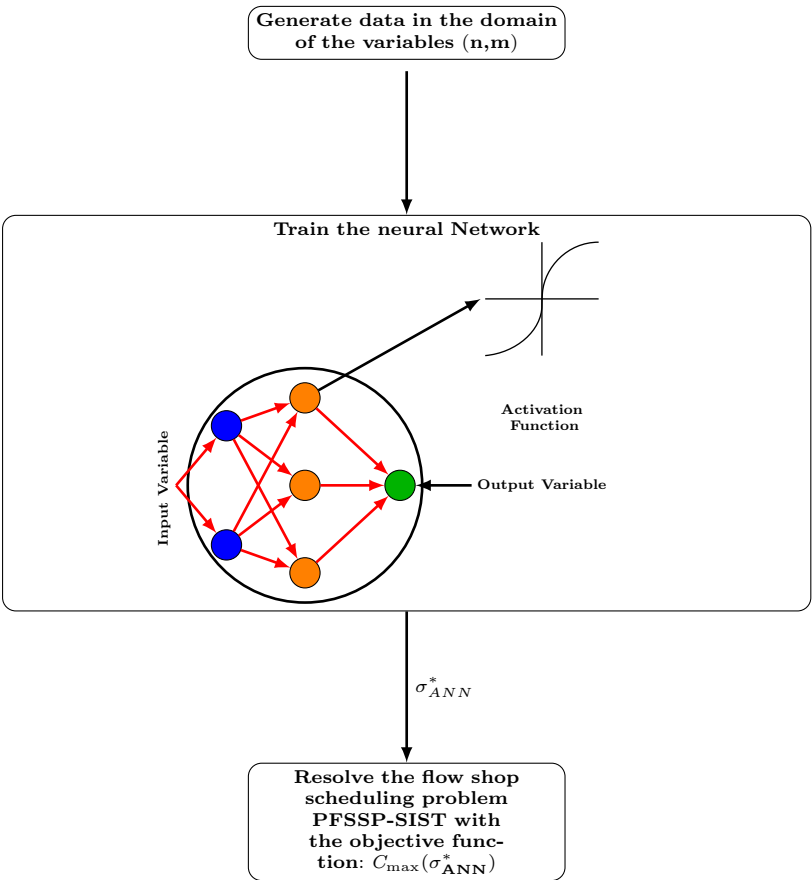


Figure 5. The trained neural network

To illustrate how a neural network may be used to build a sequence out of a set of jobs, we will examine the example of a problem with 5 jobs and 10 machines from the datasets. Table 3 shows the processing times on each machine. The input nodes

Job	:	J_1	J_2	J_3	J_4	J_5	S_i
Machine 1	:	6	48	1	38	41	3
Machine 2	:	43	5	20	13	40	4
Machine 3	:	22	45	9	13	8	9
Machine 4	:	7	43	29	27	8	1
Machine 5	:	42	31	18	26	20	1
Machine 6	:	4	12	7	10	12	2
Machine 7	:	21	3	45	47	25	7
Machine 8	:	24	17	45	19	6	8
Machine 9	:	39	20	12	20	5	7
Machine 10	:	7	47	47	29	3	5

Table 3. The processing time of five jobs on ten machines.

Sorted output	:	Sequence
0.50556	:	5
0.50567	:	1
0.50571	:	3
0.50572	:	4
0.50580	:	2

Table 4. The processing time of five jobs on ten machines.

for each of the five jobs are first expressed as a thirty-element vector (vector with thirty rows) using Eqs. (17) – (19). Table 5 displays the results of this step. These vectors are then presented one at a time at the input layer of the neural network that has been trained for the 10-machine flow shop. The network output corresponding to each instance is given in the last row of Table 5. The five jobs are arranged in increasing order of their neural outputs to derive the neural sequence, which is shown in Table 4. The enhanced ANN algorithm adopted to solve the PFSSP-SIST problem is described by the algorithm 1. Figure 6 illustrates the neural network model architecture, including weight values and connecting links, starting from initial jobs J_1 .

Input elements	<i>Jobs</i>				
	J_1	J_2	J_3	J_4	J_5
1	0.0900	0.5100	0.0400	0.4100	0.4400
2	0.4700	0.0900	0.2400	0.1700	0.4400
3	0.3100	0.5400	0.1800	0.2200	0.1700
4	0.0800	0.4400	0.3000	0.2800	0.0900
5	0.4300	0.3200	0.1900	0.2700	0.2100
6	0.0600	0.1400	0.0900	0.1200	0.1400
7	0.2800	0.1000	0.5200	0.5400	0.3200
8	0.3200	0.2500	0.5300	0.2700	0.1400
9	0.4600	0.2700	0.1900	0.2700	0.1200
10	0.1200	0.5200	0.5200	0.3400	0.0800
11	0.2980	0.2980	0.2980	0.2980	0.2980
12	0.2820	0.2820	0.2820	0.2820	0.2820
13	0.2840	0.2840	0.2840	0.2840	0.2840
14	0.2380	0.2380	0.2380	0.2380	0.2380
15	0.2840	0.2840	0.2840	0.2840	0.2840
16	0.1100	0.1100	0.1100	0.1100	0.1100
17	0.3520	0.3520	0.3520	0.3520	0.3520
18	0.3020	0.3020	0.3020	0.3020	0.3020
19	0.2620	0.2620	0.2620	0.2620	0.2620
20	0.3160	0.3160	0.3160	0.3160	0.3160
21	0.2164	0.2164	0.2164	0.2164	0.2164
22	0.1669	0.1669	0.1669	0.1669	0.1669
23	0.1533	0.1533	0.1533	0.1533	0.1533
24	0.1527	0.1527	0.1527	0.1527	0.1527
25	0.0963	0.0963	0.0963	0.0963	0.0963
26	0.0346	0.0346	0.0346	0.0346	0.0346
27	0.1825	0.1825	0.1825	0.1825	0.1825
28	0.1434	0.1434	0.1434	0.1434	0.1434
29	0.1271	0.1271	0.1271	0.1271	0.1271
30	0.2109	0.2109	0.2109	0.2109	0.2109
Output	0.50567	0.50580	0.50571	0.50572	0.50556

Table 5. Input/Output values of neurons for five jobs and ten machine

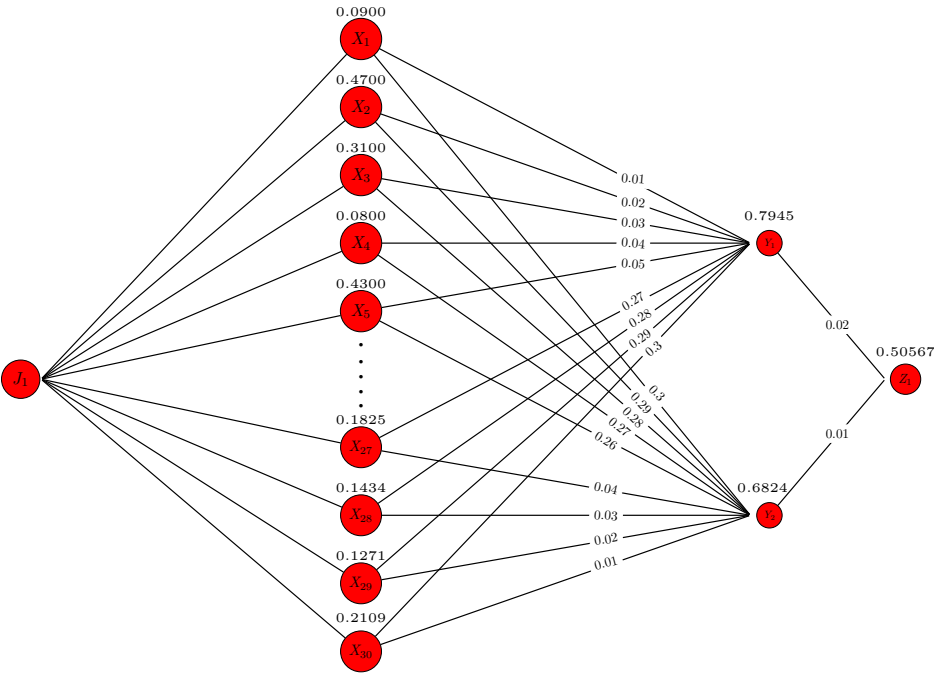


Figure 6. Input/Output values of neurons for J_1 and ten machine.

Algorithm 1 The artificial neural network algorithm.

Input: Generate the P using the processing time $p_{i,j}$ and S_i

Step 1: Apply the same procedure as in Eqs (17) – (19) to determine the input values for a job j when it is presented to the network

Step 2: The weight values on the connections between the hidden neuron and the others should be calculated as follows:

- * For the first hidden neurons
 - Order the input values in ascending order
 - Rank their priorities in order from 1 to $3 \times m$, then divide each one by 100
- ** For the second hidden neurons
 - Order the input values in decreasing order
 - Rank their priorities in order from 1 to $3 \times m$, then divide each one by 100

Step 3: Apply the activation function to the net input, L_{ink} for the first and second hidden neurons use:

$$L_{ink} = \sum_{n=1}^n X_n W_{n,k}, \quad k \in (1, 2)$$

Step 4: Using the following function, determine the output values L_k for the first and second hidden neurons:

$$L_k = G(L_{in}), \quad k \in (1, 2)$$

Where: $G(x) = 1/(1 + \exp(-x))$

Step 5: The values used as inputs for the output neurons come from L_k , where $k \in (1, 2)$

Step 6: The weights for the connections between hidden neurons and output neurons should be determined as follows:

- Sort the hidden neurons Y_k values in ascending order
- Divide each of their priorities by 100 after ranking them from 1 to 2

Step 7: Pass the net input values Z_{in} , through the activation function for the output neuron:

$$Z_{in} = \sum_{n=1}^n X_n W_n$$

Step 8: The final output values Z_k can be calculated using the following function:

$$Z_k = G(Z_{in}), \quad k \in (1 \dots n)$$

Where: $G(x) = 1/(1 + \exp(-x))$

Step 9: Arranged the output values of Z_k in increasing order to obtain the sequence of jobs

Output: $\sigma^*, C_{\max}(\sigma^*)$

3.3. The NEH algorithm

The NEH algorithm has emerged as a valuable heuristic for addressing the Permutation Flow Shop Scheduling Problem across both small and large problem instances. In their work [29], Ruiz and Maroto recommended the NEH algorithm as one of the best-performing methods for solving PFSSP problems, surpassing even the most complex and modern heuristics. The strength of the NEH method lies in its prioritization of jobs during the insertion phase, where jobs are selected for scheduling. Algorithm

2 outlines the process for solving the PFSSP-SIST scheduling problem with n jobs on m machines.

Algorithm 2 The algorithm of NEH.

Input: For a processing time $p_{1,\sigma_j}, \dots, p_{m,\sigma_j}$ of a the setup time is S_i in m machines.
 Total the processing time and setup time for each job by:
 $T(\sigma_j) = \sum_{i=1}^m (p_{i,\sigma_j} + S_i)$
 Construct the sequence $\sigma = (\sigma_1, \dots, \sigma_n)$ by sorting the jobs in descending order of $T(\sigma_j)$
 Determine the optimum sequence for the first two jobs by taking σ_1 and σ_2
for $j = 3$ to n **do**
 Add σ_j to every point of the current sequence being created, choose the sequence with the minimum $C_{\max}$ in the previous machine, and update the σ_{NEH} of the best sequence
end for
Output: σ_{NEH}

3.4. Greedy Randomized Adaptive Search Procedure algorithm

The GRASP algorithm is well-known and effective heuristic for resolving PFSSP scheduling issues [25]. In this approach, we start by setting a machine's work completion time initialization value to zero. We note that the basic technique used in this procedure is selecting a job from the created sequence and sorting the candidate jobs into an *RCL* list. This set of candidates is established using an interval based on the values of the neighboring solutions objective functions C_{σ_j} . C_{σ_j} is represented the objective function's value, when a job number σ_j is inserted into the construction sequence. The two terms $C_{\min}$ and $C_{\max}$ denote, the minimum and maximum values of the objective function during the production of the sequences. A randomly selected element from the *RCL* list is taken at each iteration of the construction phase, tested in a variety of positions, and the position with the minimum makespan is retained. The GRASP's constructive steps are illustrated in Algorithm 3.

Algorithm 3 GRASP algorithm.

Input: $\sigma = \{\sigma_1, \sigma_2, \dots, \sigma_n\}$ % the best current sequence
 $\sigma^{gr} \leftarrow \emptyset$ % the GRASP sequence
 $\Omega \leftarrow \sigma$, and evaluate the makespan $\sigma_j \in \Omega$
while $\Omega \neq \emptyset$ **do**
 $C_{\min} \leftarrow \min \{C_{\sigma_j}, \sigma_j \in \Omega\}$, $C_{\max} \leftarrow \max \{C_{\sigma_j}, \sigma_j \in \Omega\}$
Create the $RCL \leftarrow \{\sigma_j \in \Omega, C_{\sigma_j} \in [C_{\min}, C_{\min} + \delta \times (C_{\max} - C_{\min})]\}$
Choose the sequence σ_u from the RCL that will give the created solution's optimum place that minimizes value of $C_{\max}$
 $\sigma^{gr} \leftarrow \sigma^{gr} \cup \sigma_u$ and update from Ω
end while
Output: σ^{gr}

3.5. The ABJR algorithm

The permutation flow shop scheduling problems were studied in the early 1950s. Johnson, was the first to tackle the $F_2 || C_{\max}$ flow shop problem, which is a fundamental problem in flow shop scheduling [18]. The objective of our approach is to minimize the manufacturing time for each problem in PFSSP-SIST. We investigate the use of heuristics with the Johnson rule (ABJR) to solve the $F_m | \text{prmu}, \text{SIST} | C_{\max}$. The characteristics of our heuristic solution are presented in Algorithm 4. This approach enables us to solve PFSSP-SIST scheduling problems involving more than two machines using the Johnson algorithm.

Algorithm 4 algorithm with Johnson rule.

Input: $\sigma_0 = \{\sigma_1, \sigma_2, \dots, \sigma_n\}$ % The initial sequence

 $\sigma \leftarrow \sigma_0$

 % The construction of σ' %

 $p'_{1,\sigma_j} \leftarrow p_{1,\sigma_j} + S_1,$
 $p'_{2,\sigma_j} \leftarrow \sum_{l=2}^m (p_{l,\sigma_j} + S_l), j = 1, \dots, n$
 $\sigma^* \leftarrow \sigma'$
for $i = 2$ **to** $m - 1$ **do**

% Reconstruction of new two machines %

 $p'_{1,\sigma_j} \leftarrow \sum_{l=1}^i (p_{l,\sigma_j} + S_l), j = 1, \dots, n$
 $p'_{2,\sigma_j} \leftarrow \sum_{l=i+1}^m (p_{l,\sigma_j} + S_l), j = 1, \dots, n$
 $\sigma'' \leftarrow \text{Johnson rule}(\sigma')$
if $(C_{\max}(\sigma'') < C_{\max}(\sigma'))$ **then**
 $\sigma \leftarrow \sigma''$
if $(C_{\max}(\sigma) < C_{\max}(\sigma^*))$ **then**
 $\sigma^* \leftarrow \sigma$
end if
end if
end for
 $\text{makespan} \leftarrow C_{\max}(\sigma^*)$
Output: σ^*

4. Computational results and comparisons

This paper focuses on the study of typical flow shop problem that is PFSSP under SIST constraint. To evaluate the developed algorithms, several instances of different sizes are created, including small $n \in [5, 15]$ and $m \in [3, 5]$, medium/ large $n \in [30, 700]$ and $m \in [5, 30]$. Processing times are uniformly distributed in the interval $p_{i,\sigma_j} \in [1, 50]$ and $S_i \in [1, 5]$ for all instance sizes. We present the results of our experimental study in order to evaluate the performance of the proposed heuristics and artificial neural networks method. To improve the evaluation of our algorithms, we adjust the parameters of each algorithm according to the data of the tested instances. The first evaluation concerns the difference in makespan between the different methods, for this we calculate the relative percentage deviation (*RPD*) using the following expression:

$$RPD(C) = \left(\frac{C^{Algo} - C^{\text{best}}}{C^{\text{best}}} \right) \times 100\% \quad (20)$$

The makespan with an average of ten tests is considered for each addressed problem, with C^{Algo} representing the average makespan value for each algorithm, and C^{best} representing the best makespan value among the 10 tests.

An initial study is carried out on relatively small instances, comparing the results

obtained from the MILP formulation with four constructive approaches: NEH, ANN, AWJH, and GRASP. The instances are simulated by varying the processing times $p_{i,\sigma_j} \in [1, 50]$ and $S_i \in [1, 5]$. The number of jobs is defined as $n \in \{5, \dots, 15\}$, and the number of machines is denoted as $m \in \{3, \dots, 5\}$. In Table 6, the results value of the objective function are presented for 15 instances. Using MILP as an alternative to solvers is very interesting for decision-makers in industrial production. It has the advantage of providing an exact solution, which can save a lot of time and money, especially for small batch sizes. However, it's important to note that finding an optimal solution with MILP can sometimes take a long time to compute. Furthermore, other approaches like NEH, ANN, AWJH, and GRASP can find highly efficient solutions in just a few seconds. The purpose of comparing these four approaches is to evaluate how their solutions differ from the MILP method. The main goal is to assess the performance of these approaches and determine if they can be used in future work as effective procedures.

Based on the analysis, the NEH algorithm consistently produces results that are very close to the optimal solution. In 9 out of 15 cases, there were no differences between the NEH algorithm's results and the optimal solution obtained using MILP. This means that the NEH algorithm has a success rate of 60% in reaching the optimal solution as compared to the MILP method. It is also conclude that NEH algorithm produces better results than the other heuristics. It was observed that ANN method outperforms GRASP and gives sometimes best results than ABJR.

To evaluate the efficiency of different algorithms, we conducted numerical tests on medium-large-sized instances. We calculated the Relative Percentage Deviation (*RPD*) for the average of ten instances for each algorithm. The tests were performed with varying numbers of jobs n ranging from 30 to 700 and the number of machines m ranging from 5 to 30. We conducted two separate analyses, one with $\delta = 0.5$ and another with $\delta = 0.6$ for GRASP algorithm. In Table 7, for the first analysis with $\delta = 0.5$, we compared the efficiency of four algorithms. The results showed that the NEH algorithm performed the best. The minimum *RPD* value of 0.0634% was achieved by NEH, while the largest value of 2.6692% was obtained by the GRASP algorithm. Out of the 32 simulated cases, NEH achieved the minimum value in 27 cases, resulting in a success rate of 84.37%. In the second analysis with $\delta = 0.6$, we again observed the excellent performance of the NEH algorithm based on the *RPD* values. NEH achieved the minimum *RPD* value of 0.0139%, similar to the previous case, while the largest value of 3.9691% was obtained by the ANN algorithm. In this analysis, NEH achieved the minimum *RPD* value in 26 out of 32 tested cases, resulting in a success rate of 81.25%.

To assess how well the algorithms perform in different size, I conducted an experimental study using medium and large instances. These instances were selected from the initial comparison to assess the robustness of the algorithms. The following diagram represents the second analysis of the comparison research between the several four algorithms. The evaluation will be carried out by computing the *RPD* term in accordance with the change of the coefficient δ , while adjusting the quantity of jobs

Table 6. The objective value for small instance

$n \times m$	<i>MILP</i>	<i>NEH</i>	<i>ANN</i>	<i>ABJR</i>	<i>GRASP</i>
5×3	233	233	291	233	253
8×3	246	246	294	246	285
10×3	301	301	343	316	340
12×3	409	409	490	409	495
14×3	543	543	615	576	610
15×3	582	582	619	607	623
5×4	261	261	284	274	299
8×4	408	408	489	416	501
10×4	424	424	529	458	514
12×4	421	430	526	440	520
14×4	472	479	538	497	537
15×4	442	454	555	474	530
5×5	264	264	305	271	322
10×5	411	412	473	426	475
15×5	561	567	673	607	644

and machines for each investigated problem. In Fig. 7 we present ANOVA diagrams show statistical analysis findings for five instance. For $\delta = 0.5$, in (a) the heuristic NEH provides the best *RPD* value of 0.18% followed by the heuristic ABJR with a value of 0.21% then the ANN with 0.30%, and the greatest *RPD* is recorded for GRASP at 0.31%. In the case of (c) the heuristic NEH gives the best *RPD* value of 0.01% followed by the heuristic ABJR with a value of 0.01%, and the ANN by 0.02%; while the highest value of *RPD* is 3.05% recorded for GRASP.

In Fig. 8 for $\delta = 0.6$, for the case of (b) the heuristic NEH gives the best of *RPD* 0.01% followed by ANN with a value of *RPD* 0.02% then the ABJR 0.05%, while the largest value is recorded by ILS with a value of 8.05%. We find that for the case of (c), the heuristic NEH gives the best *RPD* value, with a value of 0.10%, continued by the heuristic ABJR whose value is 0.19% and the ANN with 0.20%, and the greatest worth *RPD* is registered for GRASP at 1.10%. According to our most recent analysis, the NEH algorithm is a more efficient way to solve the PFSSP-SIST in comparison to the other algorithms. It was observed that ANN method outperforms GRASP and gives sometimes best results than ABJR.

Table 7. The RPD value for all algorithm for $\delta = 0.5$ and $\delta = 0.6$, the minimum values are in bold.

$n \times m$	$\delta = 0.5$				$\delta = 0.6$			
	NEH	ANN	GRASP	ABJR	NEH	ANN	GRASP	ABJR
30×5	0.8328	1.6130	2.4143	1.0437	3.3406	3.9691	2.7232	3.5611
50×5	0.8172	1.4086	1.7740	0.9634	1.1420	1.6600	2.1779	1.2682
100×5	1.0228	1.2939	1.0708	1.0434	0.4052	0.7259	1.1953	0.4659
120×5	1.5331	1.7920	1.9214	1.5993	0.9788	1.2593	1.6113	1.0618
150×5	1.1422	1.3961	1.2484	1.2414	1.5006	1.7153	1.4313	1.5652
30×10	1.1027	2.5730	3.4024	1.4043	0.3844	1.3502	2.0721	0.5438
50×10	0.1149	0.8956	1.5431	0.4538	1.3695	2.1344	1.1165	1.5114
100×10	0.5240	1.0774	0.9472	0.2115	1.3774	2.0497	2.3793	1.5455
120×10	1.6996	2.1621	1.5463	1.8138	1.0723	1.5411	2.2197	1.2418
150×10	0.2301	0.6661	0.9339	0.3806	0.1684	0.5142	1.1592	0.2859
200×10	0.2577	0.5627	0.2535	0.3436	1.4205	1.7176	1.7577	1.4888
50×15	0.2041	0.8846	1.9109	0.5160	0.9693	1.9214	3.6822	1.3650
100×15	0.8358	1.4634	0.6276	1.1039	0.1403	0.8702	1.6500	0.4335
120×15	1.0145	1.5688	1.9375	1.2648	1.0384	1.7181	2.0500	1.2150
150×15	1.3911	1.8490	1.8271	1.5664	1.2990	1.7765	2.3307	1.4041
180×15	0.1891	0.6458	1.0278	0.3347	1.0699	1.5911	2.1270	1.2412
200×15	0.3195	0.6556	0.9900	0.4678	0.6159	0.6159	1.0869	0.3738
100×20	0.3161	0.8761	2.3814	0.5359	0.3893	0.9084	1.6387	0.6398
180×20	1.2512	1.8001	2.2533	1.4354	0.1869	0.7425	1.4388	1.6133
200×20	0.4033	1.1127	1.0673	0.7434	0.1209	0.5953	1.2406	0.2758
300×20	0.1355	0.6205	0.8920	0.3659	0.0903	0.4641	0.8863	0.2149
400×20	1.2804	1.6035	0.2185	1.3935	0.0878	0.3841	0.6628	0.2234
500×20	0.1896	0.5231	0.3017	0.2902	0.0715	0.4108	0.5248	0.1767
600×20	0.0813	0.3648	0.4518	0.1603	0.0734	0.3339	0.4876	0.1632
700×20	0.1893	0.4488	0.5208	0.5208	0.0987	0.3235	0.5075	0.1649
100×30	0.1825	0.9773	1.4520	0.5195	0.1374	1.0235	2.5687	0.5468
150×30	0.1875	0.9309	1.6941	0.4874	0.5435	1.2235	2.3167	0.7140
200×30	0.0978	0.5684	1.8018	0.3261	0.2692	1.0379	1.2433	0.5944
400×30	0.1033	0.6168	1.0352	0.3199	0.3199	0.0861	1.0822	0.2781
500×30	1.2023	2.3645	2.6692	2.1371	0.0139	0.4683	0.7814	0.2852
600×30	0.0791	0.5003	0.5003	0.2568	0.0986	0.4470	0.3216	0.3032
700×30	0.0634	0.3930	0.5395	0.1710	0.0811	0.3425	0.5985	0.1825

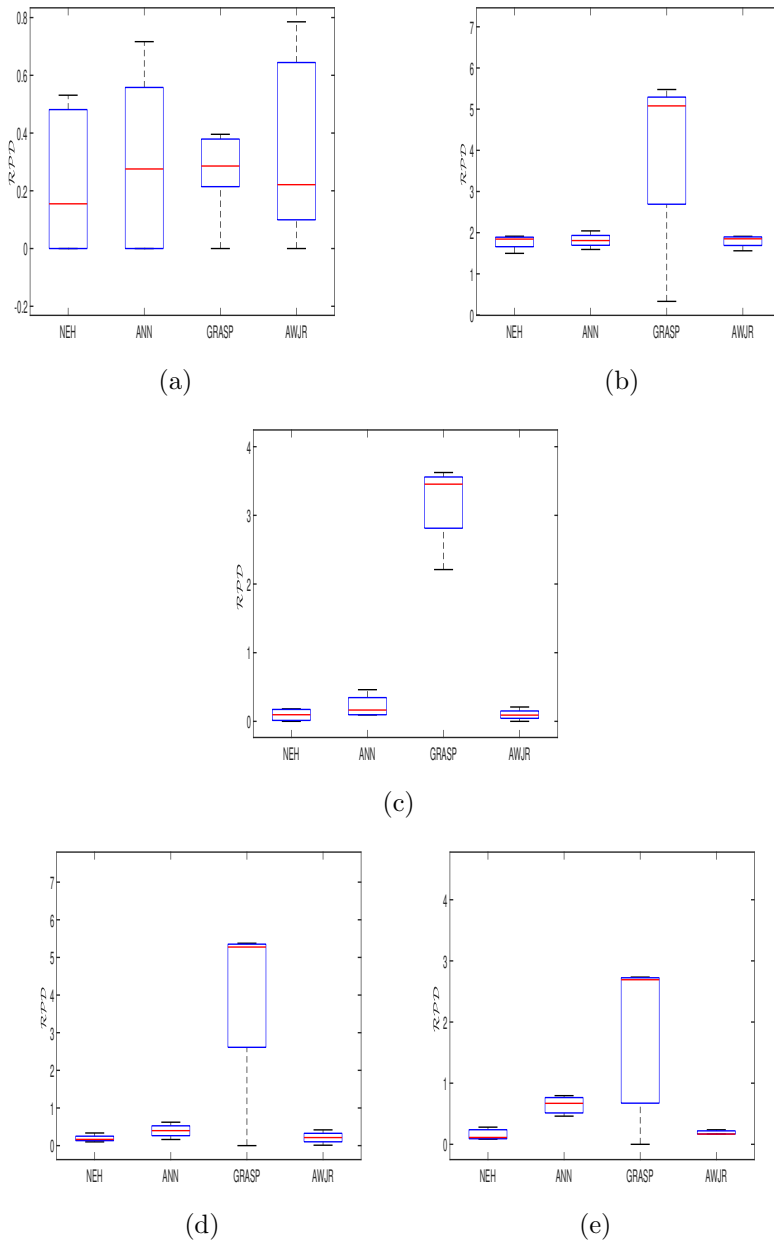


Figure 7. Box plot of RPD for different used methods with $\delta = 0.5$ at instance $n \times m$: (a) 150×10 , (b) 200×15 , (c) 300×20 , (d) 700×20 , (e) 100×30

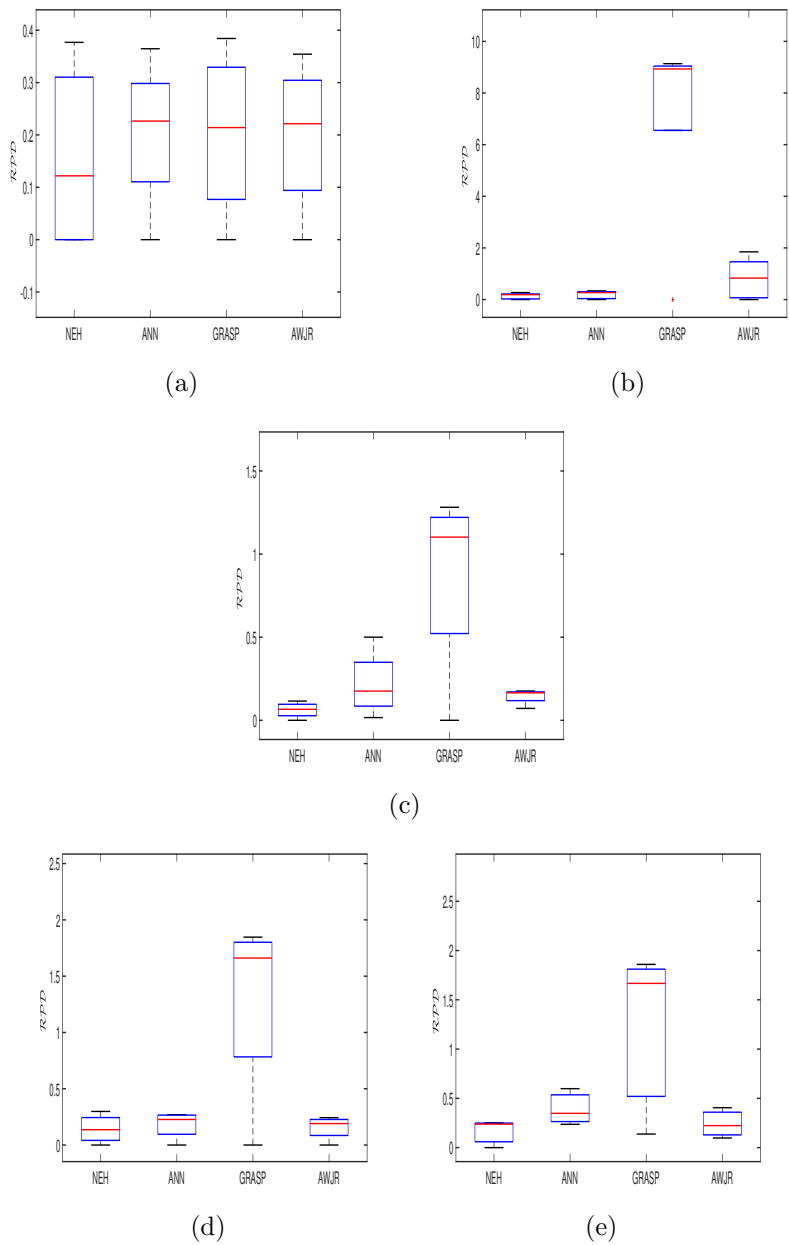


Figure 8. Box plot of RPD for different used methods with $\delta = 0.6$ at instance $n \times m$:(a) 150×10 , (b) 200×15 , (c) 300×20 , (d) 700×20 , (e) 100×30

5. Conclusion

In this paper, we address the Permutation Flow Shop Scheduling Problem under the constraint of the Sequence Independent Setup Time. In this case, the SIST constraint of each machine requires a certain setup time to process all the assigned jobs. Hence, this setup time will be independent of the sequence of jobs; but will depend only on the nature of the machine. The objective is to minimize the makespan of all jobs to be completed. The primary research objective is to create an artificial neural network capable of performing this type of scheduling through the process of neural network training. The trained network can provide a priority that shows the sequence of the job and will be very close to the optimal sequence. The performance of the ANN model has been evaluated for various problem instances. For small-sized instances of the problem, we found solutions using a mixed integer linear programming (MILP) approach. However, MILP may not be effective for resolving large-scale dimensions of the investigated problem. Therefore, we have employed multiple effective approximation methods to find appropriate solutions. Specifically, we used the NEH algorithm, the Algorithm based on Johnson Rule (ABJR), the Greedy Randomized Adaptive Search Procedure (GRASP) algorithm, and the ANN model. Numerical computations demonstrate that the NEH algorithm converges to the optimal solution. It is worth noting that the NEH algorithm consistently yields better results for all considered instance sizes of the PFSSP problem. Simulation results indicate that the performance of the NEH algorithm superior that of other approximation methods for our specific problem. Indeed, the NEH heuristic performs a success rate of 82.81% and achieves a minimum relative percentage deviation value of 0.0139%. Additionally, the ANN method outperforms GRASP in some cases and occasionally provides superior results compared to ABRJ. The developed strategy shows promising potential for tackling more difficult optimization problems.

As future directions of this work, one can further explore manufacturing problems by employing exact and approximate methods, while integrating additional constraints such as blocking and no-idle. Additionally, it can be useful to investigate other criteria for optimization, such as energy efficiency and production cost, as well as analyze the performance of machines in the scheduling process. Furthermore, developing and implementing multivariate time series utilizing machine learning and deep learning algorithms can enhance the search for optimal or near-optimal solutions to similar problems of this present work.

Funding:

This research received no external funding.

Conflicts of Interest:

The authors declare no conflict of interest.

Author Contributions:

Authors contributed equally in this work.

Data Availability:

Our manuscript has no associated data.

Acknowledgements:

We would like to thank the reviewers for their helpful remarks that have lead to the significant improvement of this paper.

References

- [1] Ahmad, Q. S., and Khan, M. H., Application of Neural Networks to Scheduling Problem including Transportation Time, *International Journal of Computer Applications*, **54**, (5), 2012.
- [2] Akyol, D. E., Application of neural networks to heuristic scheduling algorithms, *Computers & Industrial Engineering*. **46**, (4), 2004, 679–696.
- [3] Al-Barazanchi, I., Hashim, W., Alkahtani, A. A., Abdulshaheed, H. R., Gheni, H. M., Murthy, A., ... and Jaaz, Z. A., Remote monitoring of COVID-19 patients using multisensor body area network innovative system. *Computational Intelligence and Neuroscience*, 2022.
- [4] Allali, K., Aqil, S., and Belabid, J., Distributed no-wait flow shop problem with sequence dependent setup time: Optimization of makespan and maximum tardiness. *Simulation Modelling Practice and Theory*, 116, 102455.
- [5] Anh, B. T., and Hiep, P. T., Developing the max-min power control algorithm for distributed wireless body area networks, *AEU-International Journal of Electronics and Communications*, **158**, 154448, 2023.
- [6] Aqil, S., Effective Population-Based Meta-heuristics with NEH and GRASP Heuristics Minimizing Total Weighted flow Time in No-Wait Flow Shop Scheduling Problem, *Mathematics*, **11**, 102455, 2023.

- ing Problem Under Sequence-Dependent Setup Time Constraint. *Arabian Journal for Science and Engineering*, 2024, 1–24.
- [7] Baskar, A., Minimizing the makespan in permutation flow shop scheduling problems using simulation, *Indian Journal of Science and Technology*, **8**, (22), 2015, p. 1.
- [8] Deepak, K. S., and Babu, A. V., Packet size optimization for energy efficient cooperative wireless body area networks. In *2012 Annual IEEE India Conference (INDICON)*, 736–741.
- [9] Elissaouy, O., and Allali, K., Minimizing the maximum tardiness for a permutation flow shop problem under the constraint of sequence independent setup time. *RAIRO-Operations Research*, **58**, (1), 2024, 373–395.
- [10] de Garis, H., An artificial brain ATR's CAM-Brain Project aims to build/evolve an artificial brain with a million neural net modules inside a trillion cell Cellular Automata Machine, *New Generation Computing*, **12**, 1994, 215–221.
- [11] Garey MR., Johnson DS., Sethi R., The complexity of flowshop and jobshop scheduling, *Mathematics of operations research*. **1**, (2), 1976, 117–129.
- [12] Goli, A., Integration of blockchain-enabled closed-loop supply chain and robust product portfolio design. *Computers & Industrial Engineering*, 179, 109211.
- [13] Goli, A., Ala, A., and Mirjalili, S., A robust possibilistic programming framework for designing an organ transplant supply chain under uncertainty. *Annals of Operations Research*, **328**, (1), 493–530.
- [14] Goli, A., Ala, A., and Hajiaghaei-Keshteli, M., Efficient multi-objective meta-heuristic algorithms for energy-aware non-permutation flow-shop scheduling problem. *Expert Systems with Applications*, 213, 119077.
- [15] Goli, A., and Tirkolaee, E. B., Designing a portfolio-based closed-loop supply chain network for dairy products with a financial approach: Accelerated Benders decomposition algorithm. *Computers & Operations Research*, 155, 106244.
- [16] Goli, A., Tirkolaee, E. B., and Weber, G. W., An integration of neural network and shuffled frog-leaping algorithm for CNC machining monitoring, *Foundations of Computing and Decision Sciences*, **46**, (1), 2021, 27–42.
- [17] Hees, A., Schutte, C. S., and Reinhart, G., A production planning system to continuously integrate the characteristics of reconfigurable manufacturing systems. *Production Engineering*, **11**, 2017, 511–521.
- [18] Johnson SM., Optimal two-and three-stage production schedules with setup times included. *Naval research logistics quarterly*, **1**, (1), 1954, 61–68.
- [19] Kropat, E., Weber, G. W., and Akteke-Öztürk, B., Eco-finance networks under uncertainty. In *Proceedings of the international conference on engineering optimization*, 353–377.

- [20] Kumar, H., and Giri, S., Optimisation of makespan of a flow shop problem using multi layer neural network. *International Journal of Computing Science and Mathematics*, **11**, (2), 2020, 107–122.
- [21] Kumar, S., Manjrekar, V., Singh, V., and Lad, B. K., Integrated yet distributed operations planning approach: A next generation manufacturing planning system. *Journal of Manufacturing Systems*, **54**, 2020, 103–122.
- [22] Lee, I., and Shaw, M. J., A neural-net approach to real time flow-shop sequencing. *Computers & Industrial Engineering*, **38**, (1), 2000, 125–147.
- [23] Nawaz, M., Ensco Jr, E. E., and Ham, I., A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem, *Omega*, **11**, (1), 91–95.
- [24] Özmen, A., Kropat, E., and Weber, G. W., Robust optimization in spline regression models for multi-model regulatory networks under polyhedral uncertainty. *Optimization*, **66**, (12), 2135–2155.
- [25] Prabhakaran, G., Khan, B. S. H., and Rakesh, L., Implementation of grasp in flow shop scheduling, *The International Journal of Advanced Manufacturing Technology*, **30**, 2006, 1126–1131.
- [26] Rahman, Humyun Fuad and Sarker, Ruhul and Essam Daryl., Multiple-order permutation flow shop scheduling under process interruptions, *The International Journal of Advanced Manufacturing Technology*, **97**, 2018, 2781–2808.
- [27] Ramanan, T. R., Sridharan, R., Shashikant, K. S., and Haq, A. N., An artificial neural network based heuristic for flow shop scheduling problems. *Journal of Intelligent Manufacturing*, **22**, 2011 , 279–288.
- [28] Rouhani, S., Fathian, M., Jafari, M., and Akhavan, P., Solving the problem of flow shop scheduling by neural network approach. In *Networked Digital Technologies: Second International Conference*, July 7-9, 2010, pp. 172–183.
- [29] Ruiz, R., and Maroto, C., A comprehensive review and evaluation of permutation flowshop heuristics, *European journal of operational research*, **165**(2), 2005, 479–494.
- [30] Sadki, H., Aqil, S., Belabid, J., and Allali, K., Multi-Objective Optimization Flow Shop Scheduling Problem Solving the Makespan and Total Flow Time with Sequence Independent Setup Time, *Journal of Advanced Manufacturing Systems*, 2023, 1–22.
- [31] Sadki, H., Belabid, J., Aqil, S., and Allali, K., On Permutation Flow Shop Scheduling Problem with Sequence-Independent Setup Time and Total Flow Time, In *International Conference on Advanced Technologies for Humanity*, 2022, 507–518.

-
- [32] Savku, E., and Weber, G. W., Stochastic differential games for optimal investment problems in a Markov regime-switching jump-diffusion market. *Annals of Operations Research*, **312**, (2), 1171–1196.
- [33] Sharma, S., and Mehra, R., Implications of pooling strategies in convolutional neural networks: A deep insight, *Foundations of Computing and Decision Sciences*, **44**, (3), 2019, 303–330.
- [34] Singhal, E., and Hemrajani, N., An improved NEH algorithm applied to permutation flow shop scheduling, *International Journal of Engineering Sciences & Research Technology*, **2**, (5), 2013, 1164–1170.
- [35] Stafford, E. F., Tseng, F. T., and Gupta, J. N., Comparative evaluation of MILP flowshop models, *Journal of the Operational Research Society*, **56**, 2005, 88–101.
- [36] Weber, G. W., Kropat, E., Tezel, A., and Belen, S., Optimization applied on regulatory and eco-finance networks-survey and new developments.
- [37] Xu, Z., Xu, D., He, J., Wang, Q., Liu, A., and Xiao, J., Mixed integer programming formulations for two-machine flow shop scheduling with an availability constraint. *Arabian Journal for Science and Engineering*, **43**, (2), 2018, 777–788.
- [38] Yenisey, M. M., and Yagmahan, B., Multi-objective permutation flow shop scheduling problem: Literature review, classification and current trends. *Omega*, **45**, 2014, 119–135.

Received 21.12.2023, Accepted 17.07.2024