Journal of Theoretical and Applied Information Technology
28th February 2021. Vol.99. No 4
© 2021 Little Lion Scientific

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

PERMUTATION FLOW SHOP SCHEDULING PROBLEM WITH MAKESPAN CRITERION: LITERATURE REVIEW

ABDEL NASSER H. ZAIED[1], MAHMOUD M. ISMAIL[2], SHIMAA S. MOHAMED[3]

[1]Professor of Information Systems, Misr International University, Egypt

[2]Assistant Professor, Decision Support System, Zagazig University, Egypt

[3]Teaching Assistant, Decision Support System, Zagazig University, Egypt

E-mail: [1]nasserhr@zu.edu.eg, [2]mahsabe@yahoo.com, [3]shimaa_said1100@yahoo.com

ABSTRACT

With the evolution of the industry, the production process becomes a large-scale problem that made scheduling jobs process as a combinatorial optimization problem. The most important and most common scheduling problem was the flow-shop scheduling problem and was considered an NP-hard problem. There are different variants of flow-shop scheduling problems such as permutation, non-permutation, no-idle, no-wait, and hybrid flow shop scheduling problems. Permutation Flow-shop Scheduling problem (PFSP) is the most important variant of flow-shop problem, where its objective was to find the optimal or near to optimal sequence of n jobs on m machines with some criterion under some conditions. In this paper, it will be reviewed the PFSP with makespan criteria (completion time) and its mathematical model. Also, most of methods that used for solving PFSP will be reviewed and discussed in this paper, where each algorithm has different methodology for solving PFSP. There are three categories used for solving PFSP as traditional algorithms, heuristic algorithms, and meta-heuristic algorithms. This study concentrated on meta-heuristic algorithms in solving PFSP, due to its efficiency in solving many applications. From this study, it has been proved that meta-heuristic algorithms are playing a vital role in solving permutation flow-shop scheduling problem and it has attracted the attention of researchers in the past few years.

.

Keywords: *Flow-shop scheduling, Permutation Flow-shop Problem, Makespan, Meta-heuristic Algorithms*

1. INTRODUCTION

The scheduling process was considered the most important field in operational research as it played an important role in the industry environment, whether in the production or manufacturing process, communication, and transportation systems. The scheduling process defined as allocating resources to tasks during certain periods of time and its aim is to enhance one or more goals. The scheduling process consisted of three elements; resources, tasks, and objective function. In this paper; we deal with the shop scheduling problem (workshop), where resources represent machines, tasks refer to the processing time of jobs on each machine, and the objective function is minimizing the completion time of executing all jobs. The shop scheduling problem was classified into three types; Job-Shop Scheduling Problem **(JSSP),** Flow-Shop Scheduling Problem **(FSSP),** and the Open-Shop Scheduling Problem **(OSSP).** These problems have the same data with some different constraints and they used for solving and modeling economic and industrial applications [1]. Flow shop scheduling problem is the most important scheduling problems that known as flow continuous of some jobs on multiple machines with some criteria under some conditions. In this paper, the flow shop scheduling problem and the differences between its variants will be introduced in a general and it will be concentrated on permutation flow shop scheduling problem with makespan criteria. Also, in this paper; it will be reviewed mathematical model of PFSP and most of methods that used for solving PFSP with makespan criterion.

Rest of this paper is organized as following: Section 2 reviews the concept of Flow-shop Scheduling problem ant its variants; Section 3 review Permutation Flow-shop scheduling problem and its mathematical model; Section 4 summarize most of the algorithms that used for solving PFSP; Section 5 review simple computational analysis that compare between best algorithms used for PFSP; Section 6 represent conclusion of this study in some points.

2. FLOW-SHOP SCHEDULING PROBLEM

Flow shop problem was known as a continuous flow of a number of jobs in a sequence without interruption over multiple machines and its objective to obtain the best order of those jobs to improve some mandatory criterion as shown in figure 1. The flow-shop scheduling problem was an important problem in the scheduling field. Due to the complexity of economic and industrial applications and the difficulty to introduce an effective schedule at a desirable time, the flow problem had been focused on by many studies with varied classical assumptions and different objective functions so, the flow-shop scheduling problem becomes NP-hard combinatorial optimization problem [2], [3].

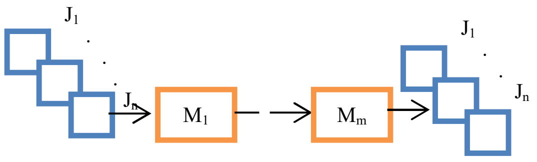


Figure 1: Flow-shop problem with n jobs and m machines.

Johnson in 1954 was the first researcher to provide the methodology for obtaining the best allocating of **n** jobs on 2-machines then worked on solving 3-machines problem under some constraints, was worked on reducing completion time of jobs. Later, many researchers worked on studying and obtaining the best solution for different types of flow shop under different criteria [2], [3], which will be shown in the next sections.

2.1 General Description of FSSP

Generally, the flow-shop scheduling problem was determined by a group of m machines (M_1, M_2, M_3…. M_m) and **n** unrelated jobs (J_1, J_2.... J_n): taking into account that the number of machines and jobs are finite. Each job consists of a group of specific processes an equal number of the machine, each process has a time $\mathbf{P_{ij}}$ ($\mathbf{P_{ij}}$ means the processing time of job **j** on the machine i). Processes of each job must be executed in the same order of given machines, i.e. for all jobs, the processes operated on machine1 then machine 2 until reach the last process on machine m. The objective was to introduce an effective schedule (sequence of jobs on the given machines) that operates all processes in the best minimal completion time that known as makespan criteria under some constraints. The feasible solution (schedule) is represented by $\boldsymbol{\pi = (\pi_1, \pi_2….. \pi_n)}$ [4].

The flow shop scheduling problem in this paper is a deterministic problem as the processing time for all processes fixed and known during the scheduling process. Makespan was the most common goal that measures the performance of feasible schedules. In addition to minimizing makespan, there are many objectives in the flow shop scheduling problem such as minimizing average-flow-time (job completion time), minimizing tardiness, and minimizing the number of tardy jobs. The notation of classical flow shop scheduling problem with makespan (C_{max}) criterion was firstly introduced by Conway et al. [4] that was $n/m/F/C_{max}$ or $F//C_{max}$ were equivalents to F/prmu/Cmax in the permutation case.
There are some constraints that must be taken into account at solving the flow shop scheduling problems, which are [5]:

- ✓ Each process starts at time zero,
- ✓ The start time of processing jobs on the first machine is zero,
- ✓ All processes must be performed,
- ✓ One process can be operated in a machine at a time,
- ✓ Each process is executed in determining time in a specific machine,
- ✓ Each machine can only operate one process at a time,
- ✓ Any process shouldn't be interrupted while it is being executed,
- ✓ If a machine busy in operation, all other processes are buffered in a queue, and
- ✓ All processes of any job must be executed in the same order of machines.

2.2 Variants of Flow Shop Problem

Researchers concluded that there are many variants of flow shop scheduling problems such as permutation, no-idle, no-wait, non-permutation, and hybrid flow shop scheduling problems. These variants differed in some constraints, which led to finding new feasible schedules and thus affected the length of the schedules this means that the optimal schedule of one of the variants may not be for another variant [3], [6].

1) Permutation flow shop problem PFSP **(F | prmu | $\mathbf{C_{max}}$)** – the most popular, the sequence of jobs on all machines must be the same. This variant will be the concentrated of study in this paper.

2) Permutation no-idle flow shop problem **(F | prmu, no – idle | C_{max})** – every machine is required to operate without any breakdown.
3) No-wait flow shop problem **(F | nwt | C_{max})** – the execution of jobs on the next machine should be started instantly after the jobs on the previous machine are completed. In this case; when the job starts its execution in the production line it continues its work without any delay on all machines in the production line. This may not be considered that there is no buffer in which the jobs wait but the jobs don't want to wait.

The first three cases have the same constrain of execution of all jobs in the same order on a given machine. Classic permutation flow shop scheduling executes jobs without any restrictions on jobs, machines, or buffer. Permutation no-idle considers condition on the machine meaning that the machine when starting not to stop. No-Wait considers condition on the job meaning that jobs when starting in a machine not interrupted on the other machines. The difference between the three variants is shown in figure 2. As the permutation flow shop problem was the popular type of flow shop problems so it intensively studied. In the next section, PFSP will be introduced with its mathematical model to calculate the completion time of the production process.

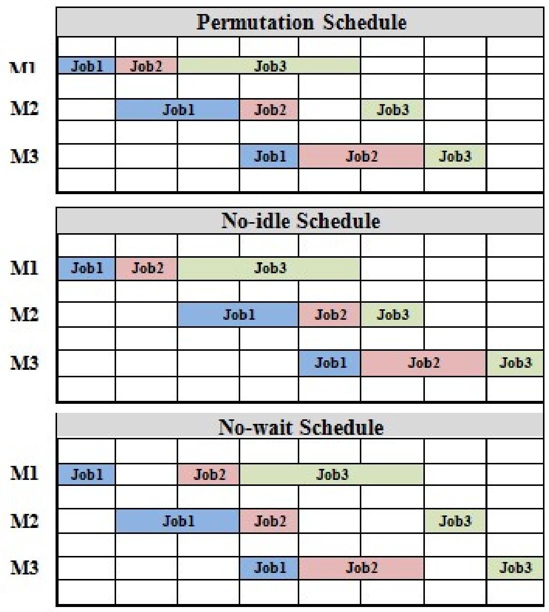


Figure 2: Schedules of the three different variants of the flow-shop problem [5]

4) Non-Permutation flow shop problem **(NPFS)** - the sequence of jobs can be changed from a machine to another.

This type of variant is similar to standard PFSP if there is an intermediate buffer (where the job wait on or between the machines until the machine in the sequence become idle) with unlimited capacity or with limited capacity but, if there is no intermediate buffer the NPFS cannot be applied for a process scheduling and optimal of PFSP is consider the optimal schedule for a given problem. Also, NPFS permitted the preemption concept in the sense that if the process is important to be executed, although it is not its order, it gives tolerance to change the order and its implementation, while making sure that the previous processes that depend on it have been accomplished. If the capacity of the buffer is unlimited, researchers consider that buffer capacity equals **n-1** (**n** equal number of jobs) to be able to store all jobs except one that is being processed [7].

5) Blocked flow shop problem – the jobs or machines can be blocked in the case that there is zero-buffer, no intermediate buffer, which led that the job cannot leave the machine until the next-machine in the sequence is free.
6) Hybrid Flow Shop problem **(F | HFS | C_{max})** - the scheduling of flow shops with multiple parallel machines per stage is completely different from the previous variants [8].

3. PERMUTATION FLOW-SHOP SCHEDULING PROBLEMS

Permutation flow-shop scheduling problems **(PFSP)** with makespan criteria aimed at finding the best order of the jobs to be executed on the specified machines, to reduce the total completion time of all jobs. Pinedo [9] studied the term PFSP were assumed that all processes of jobs must take the same sequence on all machines. In a real-problem to find the best schedule for the number of jobs **n,** this means that there are **n!** feasible schedules; for example, if the number of jobs equals **5** this means that there are **120** feasible schedules so, PFSP is known as a combinatorial optimization problem. Due to its importance in manufacturing, production and mathematical branched and PFSP has become one of the most important problems in the scheduling field, nowadays this made it an area of interest for researchers [10].

Journal of Theoretical and Applied Information Technology
28th February 2021. Vol.99. No 4
© 2021 Little Lion Scientific

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

3.1 Mathematical Model of PFSP

The **F | prmu | C_max** problem will be introduced in the form of mathematical model in three terms; decision-variables, objective-function and constraints [11], [12].

Decision-Variables:

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

k, j= {1,…………., n}

$C_{i,k}$ = completion time of job at position k on machine i, i = {1,…………, m}, k = {1,…………., n},

Objective-Function:

Minimize: $C_{\max} = C_{m,n}$ (1)

Constraints:

$$\sum_{j=0}^{n} X_{k,j} = 1 \qquad k= \{1,\ldots\ldots\ldots, n\} \qquad (2)$$

$$\sum_{k=0}^{n} X_{k,j} = 1 \qquad j= \{1,\ldots\ldots\ldots,n\} \qquad (3)$$

$$C_{1,1} \geq \sum_{j=1}^{n} X_{1,j} \times p_{i,\pi_1} \qquad (4)$$

j= {1,………,n}

$$C_{1,k} \geq C_{1,k-1} + \sum_{j=1}^{n} X_{k,j} \times p_{1,\pi_j} \qquad (5)$$

k= {2,…….,n}

$$C_{i,k} \geq C_{i-1,k} + \sum_{j=1}^{n} X_{k,j} \times p_{i,\pi_j} \qquad (6)$$

i= {2,…….,m}, k= {1,……,n}

$$C_{i,k} \geq C_{i,k-1} + \sum_{j=1}^{n} X_{k,j} \times p_{i,\pi_j} \qquad (7)$$

i= {1,…….,m}, k= {1,……,n}

$$C_{i,k} \geq 0 \qquad i= \{1,\ldots.,m\},\ k= \{1,\ldots,n\} \qquad (8)$$

$$X_{k,j} \in \{0,1\} \qquad j,\ k= \{1,\ldots\ldots n\} \qquad (9)$$

From previous equations, we can know that Eq. (1) represents minimizing the objective function **$C_{\max}$** that refers to reducing the completion time required to complete a job at the last place in the series and on the last machine. Constraints: firstly; Eq. (2) and (3) warranty that each position in the series must be occupied by precisely one job and each job must have only one position. With constraints, Eq. (4) defines the completion-time of the first job in the sequence on the first machine also Eq. (5) defines the time of accomplishment on the first-machine for the last job **k** in the order where **k ≥ 2**.

According to the completion time of the job at the **k** position on the machine **i** constraint Eq. (6) guarantees that this completion time is larger than or equal to the completion time in the previous machine **i – 1** in addition to its processing-time on the machine **i.** Also according to completion time in the machine **i** for the last job that takes position k constraint Eq. (7) guarantees that this the completion time is larger than or equal to completion time of the job in position **k-1** in addition to the processing time of the job in position **k**. Finally, constraints Eq. (8) and Eq. (9) determine the nature of the decision variables mean that to guarantee that the completion time and binary variables **$X_{k,j}$** are integer.

Method of calculating the completion time of each job on each machine it will be shown in the next equations.

$$C_{1,\pi_1} = p_{1,\pi_1} \qquad (10)$$

$$C_{1,\pi_j} = C_{1,\pi_{j-1}} + p_{1,\pi_j} \qquad j=\{2,\ldots.,n\} \qquad (11)$$

$$C_{i,\pi_1} = C_{i-1,\pi_1} + p_{i,\pi_1} \qquad i=\{2,\ldots.,m\} \qquad (12)$$

$$C_{i,\pi_j} = \max\{C_{i,\pi_{j-1}}, C_{i-1,\pi_j}\} + p_{i,\pi_j} \qquad (13)$$

i= {2,…..,m}, j= {2,…..,n}

$$C_{\max} = \max\{C_{m,\pi_j}\} \qquad (14)$$

Eq. (10) calculates the end date of the first job on the first machine. Eq. (11) calculates the end date of the other jobs on the first machine. Eq. (12) calculates the end date of the first job on all the other machines. Eq. (13) calculates the end date of all jobs on all machines. Eq. (14) calculates the production time **$C_{\max}$** that represents the makespan of a given schedule **π**. The objective of scheduling in this model is to find the best permutation π^* from all permutation ∏ that has the smallest completion time.

$$C_{\max}(\pi^*) \leq C_{\max}(\pi) \qquad \forall\ \pi \in \prod \qquad (15)$$

For example; if we have a permutation flow shop scheduling problem with 4 jobs and 3 machines and processing time of jobs on machines is given in table 1.

Table 1: Processing times of 4 jobs on 3 machines

	Job1	Job2	Job3	Job4
Machine 1	4	2	5	2
Machine 2	5	3	6	4
Machine 3	3	3	4	3

Finding optimal solution of this simple problem (4×3), this means that we have 24 available solutions with different sequence of jobs and it must be calculate all solution to determine the best that has smallest completion time. Figure 3 shows one random solution from 24 available solutions, where consider that permutation of jobs take sequence (1-4-2-3), this sequence has completion time equal 23 unit of time. So researchers proposed and used different methods to solve PFSP to reduce effort and time in finding the best solution, most of these methods will be introduced in next section.

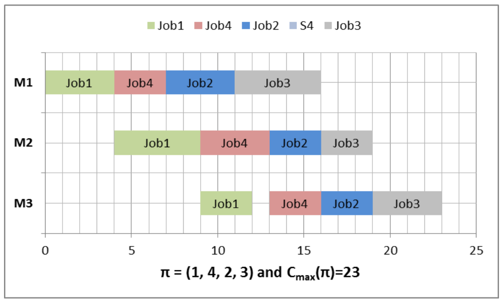


Figure 3: Gant chart of random solution of PFSP

3.2 Literature Review of PFSP

The flow shop scheduling problem was NP-hard problem as mentioned before. Researchers were solving this problem with the exact method where the scale of the problem was small. With increasing in the scale of the production system where the numbers of jobs and facilities that must be allocated on the machines become too many it was very difficult to find the best schedule in the desired time and to make re-sequence it may be significantly more costly. Also as mentioned before to make the allocation of **n** jobs on m machine this means that there are **n!** from possible schedules. So, researchers worked on finding a methodology to find the best schedule in enough time. Most researchers worked on solving the permutation flow shop scheduling problems with the makespan objective function. There are three methods; exact, heuristic, and meta-heuristic is known as the nature-inspired method. Most of these methods will be reviewed in the next sections.

3.2.1 Exact and heuristic methods

There are some exact methods proposed to obtain the optimal schedule such as dynamic programming where Held and Karp [13] used for solving the traveling salesman problem. There are other methods such as branch-and-bound, elimination rules; row generation algorithms, integer programming, and complete enumeration but these methods solved the flow shop problem on small scale.

Johnson [2] in 1954 was the first researcher that provides a heuristic methodology for solving the flow-shop scheduling problem; assumed that the scheduling problem has 2-machines then worked on a 3-machines problem with the restricted case. Johnson's rules for solving permutation flow shop scheduling problem with 2-machines were represented in four steps as following [2]:

Step 1: Find the minimum completion time of n-jobs on two machines (input the matrix of processing time) e.g. there are 3 jobs.

***Step 2*:** Select the minimum process time in the input and determine the job and machine that corresponding to this value e.g. (J1, M2)

***Step 3*:** Scheduling process

(a) If step 2 determined the machine M1 this means that the job that determined in step 2 will occupy the first available position in the schedule.

(b) If step 2 determined the machine M2 this means that the job that determined in step 2 will occupy the last available position in the schedule .

Position1	**Position2**	**J1**

***Step 4*:** Remove the job that takes a position in step 3 and back to step 1.

This process is repeated until all jobs take a position on the schedule.

There are some popular constructive heuristic methods after Johnson; Palmer [14] introduced slope index for each job to find the best sequence with 2-machines where the sequence was built based on the descending slope indices of jobs. Campbell et al. [15] introduced a new algorithm for solving PFSP with makespan called it CDS algorithm were in a heuristic way used Johnson's rule then find many schedules and choose the best one from them considered as an extension to Johnson's rules (considered extension of Johnson's rules and depended on trial and error method) also, Dannenbring [16] used Johnson's rule to obtain the

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

sequence. Gupta [17] introduced a heuristic model for m > 2 and proofed that its results were better than the palmer. Nawaz et al. [18] introduced new heuristic rules, called NEH (Nawaz–Enscore–Ham) algorithm, based on the assumption where gave higher propriety to the job with high total processing time on all machine. Most researchers used the NEH heuristic solution as an initial solution to the problem as it has been considered more effective. NEH heuristic contained from five steps showed as following:

***Step 1*:** calculate the total processing time of each job on all machines where $\mathbf{t_{ij}}$ represents the processing time of job **i** on machine **j**.

$$T_i = \sum_{j=1}^{m} t_{ij} \qquad (16)$$

***Step 2*:** Sort $\mathbf{T_i}$ in descending order, assume that result: (1-3-2-4).

***Step* 3:** Select the first two in the list of step 2 (jobs: 1, 3) and calculate the completion time of the two possible partial sequences (sequence: 1-3, 3-1). Then assume i = 3.

***Step* 4:** Select the job with **i** position in step 2 (job: 2) and calculate the makespan of all possible partial sequence with respect the sequence that found from step3 (sequence: 1-3-2, 1-2-3, 2-1-3). A Number of partial sequences equal **i**.

***Step* 5:** If n = **i**, STOP, otherwise set **i** = **i** + 1 and go to Step 4.

For knowing all heuristic methods that used for obtaining the best sequence with minimum completion time, Framinan et al. [19] reviewed most of the heuristic methods proposed for resolving permutation flow shop scheduling problems.

3.2.2 Meta-heuristic methods

As mention in the before section; obtaining on the optimal schedule need more time, also in real-problems the number of jobs and machines was more. So for resolving permutation flow shop scheduling problems with makespan criterion, most researchers used meta-heuristic algorithms. Meta-heuristic algorithms have the advantages of avoiding local optimality and starting with an initial solution better than heuristic methods. This is considered implementing the different meta-heuristic algorithms in the field of operational research to improve results.

3.2.2.1 Simulated annealing algorithm

Simulated Annealing (SA) was the first physical meta-heuristic algorithm used for solving permutation flow shop scheduling problems that were designed for improving the results of heuristics methods. Osman and Potts [20] were the first to implement simulated annealing for solving PFSP, a proposed algorithm called SAOP where firstly; determine the number of temperatures (considered the number of temperatures (considered the number of temperatures the equal number of iterations as at each temperature perform one iteration). Temperatures ($T_1, T_2, \ldots, T_K$) where **K** was the number of iteration and **K** sequence determined and evaluated at each iteration. The first temperature prosed as:

$$T_1 = \sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij} \Big/ (5mn) \qquad (17)$$

Considered $T_K=1$ as finial iteration where $m \le 20$ and $n \le 100$.

The number of iteration was determined by Eq. (18). It was important to evaluate the new sequence by Since O(mn) computations in each iteration.

$$K=\max\{3300 \ln n + 7500 \ln m - 18250, 2000\} \qquad (18)$$

All heuristic algorithms that based on neighborhood techniques such as SA has two mechanisms; N: neighborhood which used and S: how neighborhood is searched denoted as SA (N, S). Authors start with random sequence of jobs, then evaluate four simulated annealing heuristics where N $\in$ {I, S} known as perturbation scheme which determines the neighborhood structure of moving the current solution to its neighbors were used here to generate random permutation solutions (search space) and S $\in$ {O, R}. The authors evaluated four simulated annealing heuristics which were SA (I, O), SA (I, R), SA (S, O), and SA (S, R).

- **I** refers to interchange neighborhood,
- **S** refers to shift neighborhood,
- **O** refers to ordered search,
- **R** refers to random search.

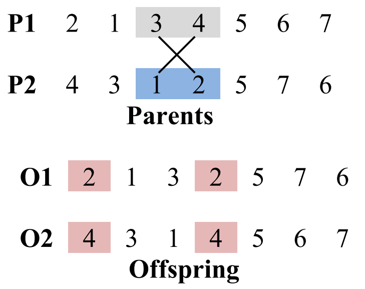
The sequence that has minimum completion time was chosen as the best solution. The time required for simulated annealing was O= (mn log (m+n)). Concluded that results of SA better than constructive heuristics.

There are other researchers who worked on improvement standard SA such as Ogbu and Smith [21] based on the previous heuristic and randomly generated solutions for improving SA also, used insertion and the pairwise exchange as the perturbation Scheme. They obtained good quality solutions in less computation time. Zegordi et al. [22] proposed a hybrid of the simulated annealing methodology with a problem-specific knowledge where create a table of 'Move Desirability for Jobs' and give index in this table then used this index as annealing scheme. This method decreased tuning of control parameters scheduling problem and results improved where obtained the optimum or near-optimum solutions at a considerably less computational time.

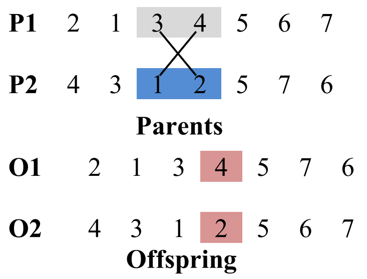
Also, Liu [23] studied the effect of neighborhood size on the traditional SA and on the simulated-annealing for flow shop problems. Results showed that the variable neighborhood sizes give excellent performance on the whole process. Tian et al. [24] used six types of perturbation schemes which are interchanging two adjacent jobs, interchanging two jobs, moving a single job, moving a subsequent of jobs, reversing a subsequent of jobs, and reversing and/or moving a subsequence of jobs. Concluded that the prosed method obtained very efficient solutions and satisfied convergence requirements. Laha and Chakraborty [25] introduce proposed simulated annealing (PSA), where has been used NEH and composite heuristics to improve solutions. Also for enhancing results and execution time of SA Bhatt [26] proposed a hybrid of the simulated annealing methodology with NEH heuristic technique.

3.2.2.2 Genetic algorithm

Genetic Algorithms (GA) simulates the biological evolution process of chromosomes through selection, crossover, and mutation operators. Reeves [27] was the first used GA for solving permutation flow shop scheduling problems where changed the standard methodology of GA to adapt the problem. General GA represented its input as chromosomes and type of data in chromosomes was numerical (0 and 1) and size of chromosomes was based on the scale of the problem in PFSP problem represent number of jobs. Being of input data 0 and 1 this made problem in PFSP problem in crossover process where PFSP order job indices. E.g. if there are two parents P1 and P2 as shown below, to make the crossover process this mean that two offspring; O1 created by using pre-X section of P1 and post-X section of P2 also O2 created similar the way of O1 with difference of parents.



As shown in the previous example in the new sequences there are repeated in some elements and some elements have not appeared this means that there are jobs not executed so Reeves to avoid this proposed crossover technique C1 where take the pre-X section from the first parent and complete elements of the new chromosome by taking in order each element from the second parent as shown below this process called conjectured.



Standard GA steps; firstly generate the initial population in these proposed algorithms generate one solution from population-based on NEH heuristic algorithm and M-1 random solutions where M means the number of chromosomes. Secondly; the Selection mechanism where select two-parent for mating. Considered the first parent selected that has minimum completion time (Fitness). To determine the best for fitness calculated for all chromosomes fitness ratio = Vmax-Vmaen (Vmax largest makespan value found so far, Vmaen population mean) but face problem of selecting the best where maybe exist more chromosomes have values close to each other, so ranked chromosomes in descending order then selected the first parent based on the fitness-rank distribution (selected randomly) and removed the chromosome that has smallest median value in the rank at the same time new offspring enter the population and selected the second parent based on the uniform distribution (also selected randomly). Thirdly; implement the crossover technique C1 where Pc=1.0. This technique improves the exploitation phase but to make the balance between exploitation and exploration implemented mutation process. To make the mutation Reeves proposed mutation rate PM and start with a high ratio (PM=0.8) and reduced by increasing diversity between populations, used a different method to measure the diversity between solutions, also mutation rate becomes the initial high rate if Vmin/Vmean > D (stat D=0.95). Two types of mutation techniques are used; exchange mutation (two elements are chosen randomly) and shift one element (the element that can be shifted and the way of shift where right or left determined randomly) and concluded that the shift mechanism better than the exchange mechanism also, improved results compared with results of SA and other heuristic algorithms [27].

There are other researchers who worked on improvement standard GA such as Chen et al. [28] were used several constructive heuristics for generating the initial population and trial many combinations from Goldberg's partially mapped crossover (PMX) operator and combination from mutation mechanisms. Concluded reducing execution where results become stable at twenty generations and result improved. Murata et al. [29] used the permutation code for choosing the parents and tested three different selection probabilities. They also tested 10 crossover operators and examined two-hybrid genetic algorithms: genetic local search and genetic simulated annealing. Concluded that the one-pint and three variants of the two-point crossover also the shift change mutation are effective for PFSP and achieved high performance with genetic local search and genetic simulated annealing. Reeves and Yamada [30] improved GA for solving PFSP by introducing a new crossover operator denoted as MSXF (multi-step crossover fusion) and introducing path relinking. They also introduced a multi-step mutation fusion (MSMF) operator. Results showed the effectiveness of the proposed algorithm. Lee and Shaw [31] proposed a new algorithm where made a hybridization neural network with a GA algorithm. Results showed increasing the quality of solutions and reducing computational time. Wang and Zheng [32] proposed an effective hybrid heuristic for flow shop scheduling were used the NEH heuristic to generate the initial population then used multi-crossover operators. Finally; they replaced classical mutation with a metropolis sample of simulated annealing. Results showed the effectiveness of the hybrid heuristic where avoided premature convergence, enhanced the exploring the neighbor searchability.

Etiler et al. [33] proposed a new GA-based heuristic where used uses the LOX crossover operator (linear Order crossover) with PC =1.0 and the shift mutation with a probability of PM =0:05. Concluded that proposed algorithm was easily implementable and performs quite effectively. Iyer and Saxena [34] Improved standard GA by using structural information from the problem to enhance the results of standard GA. Chang et al. [35] proposed a combination from a GA with a mutation-based local search, also concluded that the results of the hybrid algorithm were better than the NEH and the simple GA. Ruiz et al. [36] proposed a robust genetic algorithm and a fast hybrid implementation were used a new generational scheme which considered an efficient population initialization. Then they used new genetic operators where used four new crossover operators, used two best of them, SBOX (similar block order crossover) and SB2OX (similar block two-point order crossover). They also made hybridization GA with local search. Results were better than results from many other well know algorithms.

Rajkumar and Shahabudeen [37] proposed an improved Genetic Algorithm (IGA) for PFSP where a merge between multi-crossover operators, multi-mutation operators, and proposed hyper-mutation. Results showed that IGA gives a better solution. Zobolas et al. [38] proposed a hybrid meta-heuristic algorithm called (NEGAVNs) where it has been combined GA with Variable Neighborhood Search (VNS) also, combined GA with four heuristics: NEH, CDS heuristic, Palmer's heuristic, and Gupta's heuristic. Chang et al. [39] proposed a new algorithm where introduce a new heuristic procedure called re-combining the blocks mining to traditional GA, where it had been created new chromosomes called artificial chromosomes (ACs). This modification improved the quality of newly generated chromosomes so improve results. Also, authors in [40] proposed a combination between GA and SA for solving PFSP with makespan criterion.

3.2.2.3 Tabu search algorithm

For obtaining the suitable solution in the fastest time by using the previous methods the solution quality becomes less (less optimal). To increase the quality of the solution and reduce computational time Widmer and Hertz [41] used the tabu search algorithm to find the best solution that was near to the optimal solution. The proposed new heuristic technique called SPIRIT refers to Sequencing

Problem Involving a Resolution by Integrated Taboo search techniques.

Firstly; they generated initial feasible solution s by using the insertion method where s refers to any sequence of jobs. Insertion method depended on the way of solving the traveling salesman problem and determined in the following steps:

Step 1: make all possible pairs from all jobs that must be sequenced where each tow pair consisted of two jobs **a** and **b** then measures the distance between two jobs for each pair. Eq. (19) was a satisfactory method for measuring the distance between two jobs a and b where $\mathbf{t_{i,j}}$ was the processing time of job **i** on machine **j**.

$$d_{a,b}=t_{a,1}+\sum_{j=2}^{m}(m-j)\times\left|t_{a,j}-t_{b,j-1}\right|+t_{b,m} \qquad (19)$$

***Step* 2:** Select two jobs **a** and **b** that have a minimum distance from all, $d_{a,b} = \min_{i,j}*d_{i,j}$ this mean that we have sequence a-b.

***Step 3*:** Select an un-sequenced job randomly and add it to the sequence a-b.

Step 4: Repeat step 3 until all jobs sequenced.

Secondly; taboo search techniques were used for improving this solution, where determined in the next steps:

***Step 1*:** generate neighborhood N(s) which represent all sequences that can be obtained from reorder initial feasible solution s.

***Step* 2:** the best neighbor is the sequence s* that has minimum makespan from all sequences N(s) and doesn't lead to taboo moves. There is taboo list T where authors considered taboo size │T│=7 this means that there are seven pairs; each pair consisted of (Job, position) meaning that the job in this pair prevented from taking this position in the sequence through a certain number of iteration. Each iteration the taboo list changed where the two oldest pairs will be removed from the list.

***Step* 3:** repeat step 2 until nbmax (maximum-number-of-iterations), authors assume nbmax = 4 to avoid falling into local minimum, and obtained the best solution with minimum makespan [41].

There are other researchers who worked on improvement standard TS such as Taillard [42] where they worked on two criteria improve the quality of solution and reduce the time of obtaining the best sequence (computation time). Firstly; they generated initial solutions based on the insertion method that used in NEH heuristics then used the tabu search technique. They concluded that this method can execute the evaluation of all makespans in time **O(n²m)**. They found that taboo search needs great calculation times; to reduce them they proposed two methods of parallelization that can be executed simultaneously applied where the time reduced to **O(n)**. Reeves [43] introduced a report about the basic idea of TS techniques and its computational experiments in the machine sequencing field. From results; it has been shown that it is important to work with neighborhoods carefully and to improve any technique must work to make a balance between exploitation and exploration phase. Also, it has been concluded that the tabu search technique gave results more effective than the simulated annealing technique. Mocellin [44] proposed a new algorithm called FSHOPH (flow shop heuristic). It has been depended on SPIRIT that proposed by Widmer and Hertz with the difference in the way of measure the distance between any two jobs, in the proposed method depended on the property of permutation flow shop scheduling problem where calculated the difference between job u and v with position **j** and **j+1** (j=0,1,….., n-1) for any sequence ***S*** as follows:

$$UBX_{uv}^{k+1}=\max\{0,UBX_{uv}^{k}+(p_{kv}-p_{k+1,u})\} \qquad (20)$$

Where k=1,2,……,m-1 and $UBX_{uv}^{1} = 0$ also, UBX_{uv}^{k}: upper bound on idle time of machine k between the end of job u and the start of job v and $\mathbf{p_{kv}}$: processing time of job v on machine k. Upper bound on makespan M(S) of any n-job sequence S measured by using Eq. (21). Mocellin worked on finding the path that minimizes the upper bound UBM(s).

$$UBM(s)=\sum_{j=1}^{n}p_{mj}+\sum_{j=0}^{n-1}UBX_{[j][j-1]}^{m} \qquad (21)$$

Nowicki and Smutnicki [45] proposed a new algorithm named TSAB (Tabu Search Algorithm with Back Jump Tracking). It has been used certain block properties, where in addition to taboo list store elite solutions that found during the search to be visited and the moves that are not effective to be ignored. This algorithm found a better solution with less time. Ben-Daya and Al-Fawzan [46]; for implementing the tabu search technique efficiently, they proposed some extra features. They proposed simple techniques for generating neighborhoods of a given sequence and proposed a combined scheme that used for the first time where it has been merged intensification schema with diversification schema. Moccellin and dos Santos [47] proposed a hybrid tabu search-simulated annealing heuristic where made hybridization between simple tabu search and simple simulated annealing. Solimanpur et al. [48] proposed a new algorithm called EXTS where improved TS by a combined neural network with TS. They generate an initial sequence based on the NEH heuristic then used a neuro-dynamical structure to enhance the initial permutation iteratively and reduces the tabu effect exponentially.

Also, Pararach [49] proposed a procedure based on tabu search approach where it has been shown that it leads to new better upper-bound values for the datasets of FSSPs. Bargaoui and Driss [50] proposed a Multi-Agent model based on a tabu-search method for enhancing results of PFSP, where this model depended on two classes of agents: Supervisor agent and Scheduler agents. Results have been shown that the proposed model obtained high-quality solutions. Dewi et al. [51] implement the tabu search technique in manufacturing companies to produce dining chair products. Authors convert this problem into a flow shop scheduling problem with 6-machines and the job determined by the needs of the customer. There are many jobs in addition to dinning chair that has been worked on the same machine so it has been must take into consideration idle and wait time of job on the machine. According to these constraints; they proposed a new method based on tabu search for introducing the best order of jobs with minimum completion time.

3.2.2.4 Ant-colony optimization algorithm

Stützle [52] proposed new algorithm Max–Min Ant System (MMAS). It has been indicated that the flow shop scheduling problem (FSSP) differ from travelling salesman problem (TSP) where in TSP sequence π1=(1,2,…..,n) is the same as π2=(n,1,2,…,n-1) where the two sequence obtain the same objective but FSSP each sequence has different makesapn and π1 and π2 are different solutions where the position of job is absolute. The pheromone trail of real ants in ACO is mimicked by some real numbers $\mathbf{T_{ij}}$ (solution attribute, trial that represents the wish of setting job i at the jth position in the permutation) to give position of job absolute value). To generate the sequence, it has been introduced dummy job j_0 that the ants were set on in the beginning (where each ant begins with empty sequence). The ants generate the permutations by using probability distribution as shown in next equation.

$$p_{ij} = \begin{cases} \dfrac{T_{ij}}{\sum_{i\ not\ schedukes} Tij}, & if\ job\ i\ is\ not\ yet\ scheduled \\ 0, & otherwise \end{cases} \tag{22}$$

Stützle proposed MMAS where allow to only one ant modify the trails. Eq. (23) determines the updating of trial.

$$T_{ij}^{new} = p * T_{ij}^{old} + \Delta T_{ij} \tag{23}$$

Where:

$$\Delta T_{ij} = \begin{cases} 1/C_{best}, & if\ job\ j\ is\ placed\ on\ position\ i \\ 0, & otherwise \end{cases} \tag{24}$$

C_{best} is the makespan of the ant that updates and the interval for the trails is determined by $[T_{min},T_{max}]$. To improve the feasible solution it has been used local search strategy where use some neighborhood structures such as swap, interchange and insertion. Then improved results of local search by using insertion neighborhood strategy; when there are pair of position (i,j), to generate new sequence π* the job π(i) in position i is changed to take position j in new permutation [52]. Then it has been concluded that using the NEH as generating of initial solutions to improve results.

i<j $\pi^{new}=(\pi(1),.., \pi(i-1), \pi(i+1),.., \pi(j), \pi(j+1),.., \pi(n))$
i>j $\pi^{new}=(\pi(1),.., \pi(j-1), \pi(i), \pi(j),.., \pi(i-1), \pi(i+1),.., \pi(n))$

There are some researchers that used ACO in solving permutation flow shop scheduling problem such as Ying and Liao [53] proposed Ant Colony System named ACS, where represented $n/m/P/C_{max}$ problem with a disjunctive graph, where this graph represented by nodes (nm+2) that equal number of processes for all jobs in addition to two dummy nodes (nest and food source node) and represent the relationship between processes and machines. When generating a feasible solution, it has been used the transition rule of ACO to able an ant to choose the next node that must move to it. Generate tabu list that save chosen nodes in each trial, differ from tabu list in TS where here tabu list in final trial rtepresent the sequence. Then to reach to a near optimum solution, Palmer's method [14] used as it was a quick method for this process. It has been shown that the ACS approach is an effective meta-heuristic for solving PFSP. Blum and Sampels [54] improved ACO by proposing two mechanisms. Firstly, proposed a new neighborhood structure. Then, for constructing solutions improve an ACO approach, where used strong non-delay guidance then to improve these solutions proposed black-box local search procedures. Gajpal and Rajendran [55] proposed new ant-colony algorithm (NACO) to minimize the completion-time variance of jobs. Where based on Stützle with using random-insertion local search (RJILS) procedures to enhance initial sequence that generated by NEH heuristic. Ahmadizar et al. [56] proposed Ant Colony Algorithm (ACA) where introduced two-priority transition rules are introduced as heuristic-information based on Johnson's Rule and total processing times. Chen et al. [57] improved ACO where used three schemas of transition rule; In order, Random number and Pheromone-related schema. Then test the three transition rule with and without interchange-movie of local search approach. It has been concluded that Random-order with local search is more effective has lower complexity. Authors in [58] improved ACO where combined Nawaz-Enscore-Ham heuristic with ant colony optimization. Tasmin [59] firstly; solved TSP using ACO technique which generates a permutation or tour. Then used solution of TSP as an initial solution for PFSP also, used 2-opt exchanges procedure to improve previous solution.

3.2.2.5 Particle swarm optimization algorithm

Because of increasing the evolution in natural inspired algorithms, researchers used Particle Swarm Optimization algorithm (PSO) for increasing optimality of results the PFSP. These algorithms summarized in table 2.

Table 2: Summary of PSO algorithms for PFSP

Reference	Notations
Tasgetiren et al. [60] PSO$_{VNS}$	- Adopted the PSO for solving PFSP - Then used Variable Neighborhood Search (VNS) in local search
Pan et al. [61] DPSO	- Proposed discrete particle swarm optimization (DPSO) algorithm - Proposed new crossover operator (PTL crossover), - Combined new algorithm with a simple local search algorithm.
Liu et al. [62] PSO$_{MA}$	- Adapted the PSO to the PFSP - Used NEH as initial solution - Proposed new local search procedure (NEH_1) - Combined this algorithm with SA
Tasgetiren et al. [63] PSO$_{VNS}$	- Used a heuristic rule named the Smallest Position Value (SPV) - Combined variable neighborhood search (VNS) local search technique to PSO - Implemented PSO$_{VNS}$ for solving multi-objective PFSP.
Lian et al. [64] NPSO	- Used mutation and crossover operators
Chen and Chen [65]	- Used previous algorithm (DPSO) - Used the smallest position value (SPV) - R-defined the particle and the velocity.
Tang and Ye [66] HPSO	- Incorporated knowledge evolution algorithm (KEA) with particle swarm optimization (PSO) algorithm
Zhang and Wu [67] SA-PSO$_{VNS}$	- Used path relinking to improve exploration - Combined simulated annealing and stochastic VNS with PSO - Used a population update method to enhance the search diversification of PSO
Chen et al. [68]	- Used DPSO (Discrete PSO) - Used new filtered local search RDPSO
Ramanan et al. [69]	- Proposed new algorithm called PSO-NEH-VNS, - Used NEH heuristic method and used a variable neighborhood search (VNS).
Eddaly el al. [70]	- Proposed hybrid combinatorial PSO algorithm - Used different priority rules in the initialization stage - Used probabilistic perturbation to repeat the local search procedure. HCPSO

3.2.2.6 Cuckoo search algorithm

Li and Yin [71] proposed a cuckoo search (CS)-based memetic algorithm named HCS where used Largest-Ranked-Value (LVR) rule to deal with discrete positions as shown in table 3 and used NEH with random initialization to create 10 % solutions from initial populations.

Table 3: Largest-Ranked-Value Rule for PFSP

Jobs	1	2	3	4	5	6
Position	0.98	0.96	1.20	0.30	0.72	0.28
Job (π)	**2**	**3**	**1**	**5**	**4**	**6**

Zheng et al. [72] introduced modified cuckoo search (MCS) algorithm where CS algorithm generate continuous values so, it has been used Largest Position Value (LPV) to solve this problem. Wang et al. [73] proposed new cuckoo search algorithm (NCS) where used four strategies; smallest position value (SPV) rule, NEH heuristic, improved generalized opposition-based learning (GOBL) and local search strategy. Zhang et al. [74] proposed ACOCS algorithm where combined ant colony to cuckoo search to improve quality and time of solutions. Zhang et al. [75] proposed a self-adaptive-step cuckoo search algorithm. It has been introduced two parameters; iteration-number ratio parameter and adaptability-ratio parameter and used a dynamic-balance factor parameter two make balancing between two previous parameters. Then, it has been used Skewness-value calculation method and self-adaptive step strategy to improve results.

3.2.2.7 Other algorithms

Lately; some researches introduced new meta-heuristic to improve results and desire to reach to optimality. Most of meta-heuristic algorithms are summarized in table 4:

Table 4: Summary of Other Meta-heuristic algorithms for PFSP

Reference	Notations
Ruiz and Stützle [76]	- Used iterated greedy algorithm (IG) - Worked on destruction and construction phase. - Used NEH heuristic to construct solutions - Used local search procedure.
Pan et al. [77]	- Proposed a discrete differential evolution (DDE) algorithm for solving PFSP - Used local search procedure to improve results.
Saravanan et al. [78]	- Adapted the Scatter Search for solving PFSP - Used NEH with SS.
Qian et al. [79]	- Used Differential Evolution (DE) - Used Largest-Order-Value - Used simple local search to improve exploitation.
Zheng and Yamashiro [80]	- Combined differential evolution strategy and variable neighborhood search with quantum evolutionary algorithm (QEA).
Sayadi et al. [81]	- Used a local search procedure to improve solutions.
Liu et al. [82]	- Combined estimation of distribution algorithm (EDA) with PSO - Used the minimization-of-waiting-time local search (MWL) algorithm
Wang et al. [83]	- Proposed hybrid modified global-best harmony search algorithm - Used NEH heuristic to generate initial harmony - Used Largest Position Value to adapt PFSP.
Chen et al. [84]	- Combined Distribution Algorithms (EDAs) with GA.
Kadarkarainadar and Geetha [85]	- Used a discrete-African-wild-dog algorithm for solving PFSP.
Li and Yin [86]	- Proposed opposition-based differential evolution algorithm - Used Largest Rank Value rule - Used opposition based learning (OBL)
Wang et al. [87]	- Proposed new hybrid differential algorithm - Used Largest Rank Value rule - Used DE/rand/1/bin strategy for global search - Used insert based local-search method.
Ceberio et al. [88]	- Proposed a novel general estimation of distribution algorithm (EDA) to solve PFSP - Used Generalized Mallows (GM) model for permutation. - Combined a variable neighborhood search with new EDA.
Fong et al. [89]	- Proposed Firefly Algorithm (FA) - Used Ranked Order Value (ROV) to adapt PFSP.
Hariharan and Nimal [90]	- Proposed genetic scatter search algorithm - Combined an effective constructive heuristics with the initial solutions.
Luo et al. [91]	- Proposed discrete bat algorithm DBA - Used NEH to generate some initial solutions and the reset solutions generated randomly generated - Used an intensive virtual population neighborhood search.
Su and Li [92]	- Used Largest Ranked Value rule - Implemented a Levy flight function to improve global search domain - Used a local search algorithm to improve local search process.
Xie et al. [93]	- Proposed Teaching–Learning-Based Optimization algorithm - Used a largest order value (LOV) rule - Used a variable neighborhood search (VNS) - Used a simulated annealing (SA) as the local search method of VNS.
Bouzidi and Riffi [94]	- Adapted traditional cat swarm optimization algorithm (CSO) for solving PFSP.
Lin et al. [95]	- Combined traditional backtracking search algorithm (BSA) algorithm with crossover, mutation operations, simulated annealing and random insertion local search.
Tosun and Marichelvam [96]	- Combined local search mechanisms with standard bat algorithm.

Reference	Notations
Govindan et al. [97]	- Combined Decision Tree (DT) with Scatter Search (SS) algorithms - Entropy function is used in DT to convert PFSP into a tree structured format / set of rules - Diversification in SS improved search space.
Marichelvam et al. [98]	- Proposed Hybrid Monkey Search Algorithm (HMSA) - Used the shortest processing time (SPT) and longest processing time (LPT) dispatching rules - Combined NEH heuristics to this modification.
Abdel-Basset et al. [99]	- Proposed Hybrid Whale algorithm (HWA), - Used Largest Ranked value (LRV) to adapt continuous values of Whale Optimization Algorithm, - Used NEH heuristic rule for 10% from initial solutions, - Used swap mutation operation, - Used insert-reversed block operation, - Used local search strategy.
Nurdiansyah et al [100]	- To increase variety in population, it has been used adaptive parameter - Modified cross over operator - Used local search technique to enhance results of DE.
Huang et al. [101]	- Used Crow search algorithm (CSA) - Used NEH heuristic to generate initial population - used Simulated annealing (SA) and Variable neighborhood search (VNS) as local search method.
Arık [102]	- Used artificial bee colony (ABC) algorithm combined with: - Local search (LS)operators - Destruction/Construction (DC) operators of the variants of iterated greedy (IG) algorithm.

4. COMPUTATIONAL ANALYSIS

Recently; in 2016 Mirjalili and Lewis proposed new meta-heuristic algorithm called Whale Optimization Algorithm, and proved efficiency and speed of the proposed algorithm in solving optimization problems. Abdel-Basset et al [99] proposed Hybrid Whale Algorithm (HWA) and compared proposed algorithm with ten algorithms used for PFSP: Differential evolution algorithm with the iterated improving scheme-based local search and the greedy based local search (L-HDE), Opposition based differential evolution algorithm (ODDE), Discrete bat algorithm (DBA), Hybrid backtracking search algorithm (HBSA), Particle swarm optimization based on variable neighborhood search (PSOVNS), Particle swarm optimization based memetic algorithm (PSOMA), Hybrid teaching–learning-based optimization algorithm (HTLBO), Quantum differential evolutionary algorithm (QDEA), Hybrid genetic algorithm (HGA), Hybrid bat algorithm (HBA) and Hybrid cuckoo search (HCS) also, we used Cat Swarm Algorithm (CSO) [94].

The comparison based on calculating Best-Related-Error for each data set for each algorithm as shown in Eq. (25).

$$BRE = (C^{*} - C_{best}) / C_{best} \qquad (25)$$

Figure 4 shows comparison between 11 best algorithms used for solving PFSP with makespan criterion based on average BRE values. The datasets that used for comparison are Carlier and Reeves datasets.

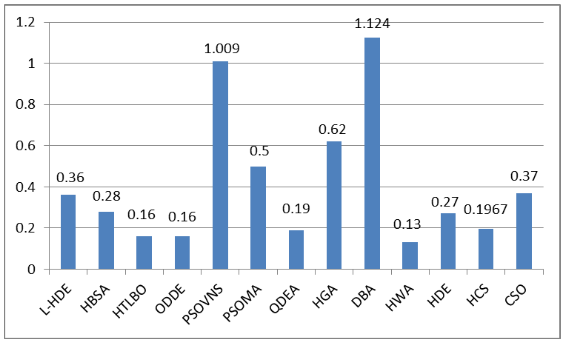


Figure 4: Comparison best algorithms based on average BRE

From figure 4; it has been illustrated that HWA is the best algorithm in solving PFSP with makespan criteria. HTLBO and ODDE have the same results but, PSO-VNS and DBA give small results. But it must be taken in consideration that each meta-heuristic algorithm has some parameters that lead to improve results of algorithm, so it is important to study the algorithm before use it. All these studies are the Strivings of researchers for obtaining on the best solutions for PFSP with makespan.

5. CONCLUSION

Due to the evolution of the industry and the increasing complexity of the production process, the scheduling jobs process in a sequence became a combinatorial optimization problem and NP-hard problem. The most important and most common flow-shop scheduling problems was permutation flow-shop scheduling problem (PFSP) where its objective to obtain the best order of n jobs on m machines that have minimum completion-time that referrer to makespan criterion, this process constraining that all processes of jobs take the same order on machines. This problem represented as **F/prmu/C$_{max}$** where **F** referred to the Flow-scheduling problem, **Prmu** referred to the Permutation variant of the flow-shop problem, and **C$_{max}$** referred to the completion-time of execution of all jobs.

In this paper, we reviewed 102 papers introduced the PFSP with makespan criteria and 81% from all researches introduced in this study used meta-heuristic algorithms for solving this problem. From this reviewing it has been concluded that these algorithms give results better than exact and heuristic rules methods. It has been used meta-heuristic algorithm for PFSP firstly in 1989 when Osman and Potts used simulated annealing algorithm and proved efficiency of SA for PFSP. Although this field of study is very old, however, it is still the center of attention for researchers, some studies worked in 2020 for PFSP, due to its importance in the field of industry and production systems. Also, NEH heuristic rules play vital role in generating initial population in modified algorithms. There are still some meta-heuristic algorithms not used yet in use to solve PFSP such as Grey-Wolf Optimization Algorithm, Ant-Lion Optimization Algorithm, Dragonfly Algorithm and Grasshopper Optimization algorithm. Each algorithm has advantage, so researchers used algorithms and work on modifying it to enhance results. In future, we will work in modified whale optimization algorithm for PFSP.

REFERENCES

[1] A. Bouzidi, and M.E. Riffi, "CSO to Solve the Shop Scheduling Problem: Survey", *Springer Nature Switzerland AG 2020, M. Ezziyyani (Ed.): Advanced Intelligent Systems for Sustainable Development (AI2SD 2019)*, vol. 1106, 2020, pp. 34–44.

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

[3] S.R. Hejazi, and S. Saghafian, "Flowshop-scheduling problems with makespan criterion: a review", *International Journal of Production Research*, Vol. 43, No. 14, 2005, pp.2895–2929.

[4] R.W. Conway, W.L. Maxwell and L.W. Miller, "Theory of Scheduling", *Addison-Wesley: Reading, Mass*, 1967.

[5] A. Bouzidi, and M.E. Riffi, "Cat Swarm Optimization to Solve Flow Shop Scheduling Problem", Journal of Theoretical and Applied Information Technology, Vol. 72, No. 2, 2015, pp. 239-243.

[6] M. Makuchowski, "Permutation, no-wait, no-idle flow shop problems", *Archives of Control Sciences*, Vol. 25, No. 2, 2015, pp. 189-199.

[7] D.A. Rossit, F. Tohm, and M. Frutos, "The Non-Permutation Flow-Shop scheduling problem: a literature review", *Omega*, Vol. 77, 2018, pp. 143-153.

[8] R. Ruiz, and J.A.V. Rodríguez, "The hybrid flow shop scheduling problem", *European Journal of Operational Research*, Vo. 205, No. 1, 2010, pp. 1-18.

[9] M.L. Pinedo, "Scheduling: Theory, Algorithms and Systems", *Prentice Hall, New Jersey, second edition,* 2002.

[10] R.L. Graham, E.L. Lawler, J.K Lenstra, and A.H.G.K. Rinnooy, "Optimization and approximation in deterministic sequencing and scheduling: a survey", *Annals of Discrete Mathematics*, Vol. 5, 1979, pp. 287–326.

[11] V.G. Tonge, and P. Kulkarni, "Solving Permutation Flowshop Scheduling Problem Using Improved Differential Evolutionary Algorithm", *International Journal of Engineering Research & Technology (IJERT)*, Vol. 2, No. 10, 2013, pp. 456-261.

[12] J. B., S. Aqil, and K. Allali, "Solving Permutation Flow Shop Scheduling Problem with Sequence-Independent Setup Time", *Journal of Applied Mathematics*, 2020.

[13] M. Held, and R.M. Karp, "A Dynamic Programming Approach to Sequencing Problems", *Journal of the Society for Industrial and Applied Mathematics*, Vol. 10, No. 1, 1962, pp. 196-210.

[14] D.S. Palmer, "Sequencing Jobs Through a Multi-Stage Process in the Minimum Total Time—A Quick Method of Obtaining a Near

Optimum", *Journal of Operational Research Society*, Vol. 16, No.1, 1965, pp. 101–107.

[15] H.G. Campbell, R.A. Dudek, and M.L Smith, "A Heuristic Algorithm for the n Job, m Machine Sequencing Problem", *Management Science*, Vol. 16, No. 10, 1970, pp. 630-737.

[16] D.G. Dannenbring, "An Evaluation of Flow Shop Sequencing Heuristics", *Management Science*, Vol. 23, No. 11, 1977, pp. 1174-1182.

[17] J.N.D. Gupta, A Functional Heuristic Algorithm for the Flowshop Scheduling Problem, *Journal of Operational Research Society*, Vol. 22, No.1, 1971, pp.39–47.

[18] M. Nawaz, E.E. Enscore, and I. Ham, "A heuristic algorithm for the m-machine, n-job flowshop sequencing problem", *Omega*, Vol. 11, No. 1, 1983, pp. 91–95.

[19] J.M. Framinan, J.N.D. Gupta, and R. Leisten, "A review and classification of heuristics for permutation flow-shop scheduling with makespan objective", *Journal of Operational Research Society*, Vol. 55, No. 12, 2004, pp. 1243–1255.

[20] I.H. Osman, and C.N. Potts, "Simulated Annealing for Permutation Flow-Shop Scheduling", *Omega*, Vol. 17, No. 6, 1989, pp. 551–557.

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

[22] S.H. Zegordi, K. Itoh, and T. Enkawa, "Minimizing makespan for flow-shop scheduling by combining simulated annealing with sequencing knowledge", *European Journal of Operational Research*, Vol. 85, No.3, 1995, pp. 515–531.

[23] J. Liu, "The impact of neighbourhood size on the process of simulated annealing: computational experiments on the flowshop-scheduling problem", *Computers & Industrial Engineering*, Vol. 37, No. 1-2, 1999, pp. 285–288.

[24] P. Tian, J. Ma, and D-M. Zhang, "Application of the simulated annealing algorithm to the combinatorial optimisation problem with permutation property: An investigation of generation mechanism", *European Journal of Operational Research*, Vol. 118, No. 1, 1999, pp. 81-94.

[25] D. Laha, and U.K. Chakraborty, "An efficient hybrid heuristic for makespan minimization in permutation flow shop scheduling", *International Journal of Advanced Manufacturing Technology*, Vol. 44, No. 5, 2009, pp. 559-569.

[26] P. Bhatt, "Permutation Flow Shop via Simulated Annealing and NEH", *UNLV Theses, Dissertations, Professional Papers, and Capstones,* 2019.

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

[28] C-L. Chen, V.S. Vempati, and N. Aljaber, "An application of genetic algorithms for flow shop problem", *European Journal of Operational Research*, Vol. 80, No. 2, 1995, pp. 389–396.

[29] T. Murata, H. Ishibuchi, and H. Tanaka, "Genetic algorithms for Flowshop Scheduling Problems", *Computer & Industrial Engineering*, Vol. 30, No. 4, 1996, pp. 1061–1071.

[30] C.R. Reeves, and T. Yamada, "Genetic Algorithms, Path Relinking and the Flowshop Sequencing Problem", *Evolutionary Computation*, Vol. 6, No. 1, 1998, pp. 230–234.

[31] I. Lee, and M.J. Shaw, "Neural-net approach to real-time flow-shop sequencing", *Computers & Industrial Engineering*, Vol. 38, No. 1, 2000, pp. 125–147.

[32] L. Wang, and D-Z. Zheng, "An effective hybrid heuristic for flow shop scheduling", *International Journal of Advanced Manufacturing Technology*, Vol. 21, No. 1, 2003, pp. 38–44.

[33] O. Etiler, B. Toklu, M. Atak, and J. Wilson, "A genetic algorithm for flow shop scheduling problem", *Journal of Operational Research Society*, Vol. 55, No. 8, 2004, pp. 830–835.

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

[35] P-C Chang, C-H. Liu and C-Y. Fan, "A depth-first mutation-based genetic algorithm for flow shop scheduling problems", *2006 International Conference on Hybrid Information Technology, Cheju Island*, 2006, pp. 25-32.

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

[37] R. Rajkumar, and P. Shahabudeen, "An improved genetic algorithm for the flowshop scheduling problem", *International Journal of Production Research*, Vol. 47, No. 1, 2009, pp. 233– 249.

[38] G.L Zobolas, C.D. Tarantilis, and G. Ioannou, "Minimizing makespan in permutation flow shop scheduling problems using a hybrid metaheuristic algorithm", *Computers & Operations Research*, Vol. 36, No. 4, 2009, pp. 1249-1267.

[39] P-C. Chang, W-H. Hunag, J-L.Wu, and T.C.E. Cheng, "A block mining and re-combination enhanced genetic algorithm for the permutation flowshop scheduling problem", *International Journal of Production Economics*, Vol. 141, No. 1, 2013, pp. 45-55.

[40] H. Wei, S. Li, H. Jiang, J. Hu, and J. Hu, "Hybrid genetic simulated annealing algorithm for improved flow shop scheduling with makespan criterion", *Applied Sciences*, Vol. 8, No. 12, 2018. pp. 1-20.

[41] M. Widmer, and A. Hertz, "A new heuristic method for the flow shop sequencing problem", *European Journal of Operational Research*, Vol. 41, No. 2, 1989, pp. 186–193.

[42] E. Taillard, "Some efficient heuristic methods for the flowshop sequencing problem", *European Journal of Operational Research*, Vol. 47, No. 1, 1990, pp. 65–74.

[43] C.R. Reeves, "Improving the efficiency of tabu search for machine scheduling problems", *Journal of Operational Research Society*, Vol. 44, No. 4, 1993, pp. 375–382.

[44] J.V. Moccellin, "A new heuristic method for the permutation flow-shop scheduling problem", *Journal of Operational Research Society*, Vol. 46, No. 7, 1995, pp. 883–886.

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

[46] M. Ben-Daya, and M. Al-Fawzan, "A tabu search approach for the flow shop scheduling problem", *European Journal of Operational Research*, Vol. 109, No. 1, 1998, pp. 88-95.

[47] J.V. Moccellin, and M.O. Santos, "An adaptive hybrid metaheuristic for permutation ftowshop scheduling", *Control and Cybernetics*, Vol. 29, No. 3, 2000, pp. 761-771, 2000.

[48] M. Solimanpur, P. Vrat, and R. Shankar, "A neuro-tabu search heuristic for the flow shop scheduling problem", *Computers & Operations Research*, Vol. 31, No. 13, 2004, pp. 2151–2164.

[49] S. Pararach, "A tabu search approach for makespan minimization in a permutation flow shop scheduling problems", *2011 International Conference on Industrial Engineering and Operations Management, Kuala Lumpur, Malaysia*, 2011, pp. 300-305.

[50] H. Bargaoui, and O.B. Driss, "Multi-agent Model Based on Tabu Search for the Permutation Flow Shop Scheduling Problem", *Distributed Computing and Artificial Intelligence, 11th International Conference, Advances in Intelligent Systems and Computing, Springer, Cham*, Vol. 290, 2014, pp. 519-527.

[51] S.S. Dewi1, A. Andriansyah, and Syahriza, "Optimization of Flow Shop Scheduling Problem using Tabu Search Algorithm: A Case Study", *IOP Conference Series: Materials Science and Engineering*, Vol. 206, No. 1, 2019.

[52] T. Stützle, "An ant approach for the flow shop problem", *6th European Congress on Intelligent Techniques & Soft Computing (EUFIT'98), Aachen, Germany*, Vol. 3, 1998, pp. 1560- 1564.

[53] K-C. Ying, and C-J. Liao, "An ant colony system for permutation flow-shop sequencing", *Computers & Operations Research*, Vol. 31, No. 5, 2004, pp. 791–801.

[54] C. Blum and M. Sampels, "An Ant Colony Optimization Algorithm for Shop Scheduling Problems", *Journal of Mathematical Modelling and Algorithms*, Vol. 3, No. 3, 2004, pp. 285-308.

[55] Y. Gajpal, and C. Rajendran, "An ant-colony optimization algorithm for minimizing the completion-time variance of jobs in flowshops", *International Journal Production Economics*, Vol. 101, No. 2, 2006, pp. 259-272.

[56] F. Ahmadizar, F. Barzinpour, and J. Arkat, "Solving Permutation Flow Shop Sequencing using Ant Colony Optimization", *2007 IEEE International Conference on Industrial Engineering and Engineering Management, Singapore,* 2007, pp. 753-757.

[57] R-M. Chen, S-T. Lo, C-L. Wu, and T-H Lin, "An Effective Ant Colony Optimization-Based Algorithm for Flow Shop Scheduling", *2008 IEEE Conference on Soft Computing in Industrial Applications (SMCia/08), Muroran, Japan,* 2008, pp. 101-106.

[58] Y-F. Liu, and S-Y. Liu, "Permutation flow shop scheduling algorithm based on ant colony optimization", *Journal of Computer Applications*, vol. 28, No. 2, 2008, pp. 302-304.

[59] C. Tasmin, "Combinatorial Ant Optimization and the Flowshop Problem", *UNLV Theses, Dissertations, Professional Papers, and Capstones*, 55 pages, 2018.

[60] M.F. Tasgetiren, M. Sevkli, Y.C. Liang, and G. Gencyilmaz, "Particle swarm optimization algorithm for permutation flowshop sequencing problem", *Conference: Ant Colony Optimization and Swarm Intelligence, 4th International Workshop, ANTS 2004, Brussels, Belgium*, 2004, pp. 382-389.

[61] Q-K. Pan, M.F. Tasgetiren, and Y-C. Liang, "A Discrete Particle Swarm Optimization Algorithm for the Permutation Flowshop Sequencing problem with Makespan Criterion", *SGAI 2006: Research and Development in Intelligent Systems XXIII, Springer, London*, 2007, pp. 19-31.

[62] B. Liu, L. Wang, and Y.H. Jin, "An Effective PSO-based Memetic Algorithm for Flow Shop Scheduling", *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics),* Vol. 37, No. 1, 2007, pp. 18-27.

[63] M. Tasgetiren, Y. Liang, M. Sevkli, and G. Gencyilmaz, "A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flow shop sequencing problem", *European Journal of Operational Research*, Vol. 177, No. 3, 2007, pp. 1930-1947.

[64] Z. Lian, X. Gu, and B. Jiao, "A novel particle swarm optimization algorithm for permutation flow-shop scheduling to minimize makespan", *Chaos, Solitons & Fractals*, Vol. 35, No. 5, 2008, pp. 851-861.

[65] T-Y. Chen, and C-L. Chen, "New Particle Swarm Optimization Algorithm for Makespan Minimization in Permutation Flowshop Sequencing", *2009 Fourth International Conference on Innovative Computing, Information and Control (ICICIC), Kaohsiung,* 2009, pp. 868-871.

[66] H-B. Tang, and C-M. Ye, "Permutation flow shop scheduling algorithm based on a hybrid particle swarm optimization", *2010 IEEE 17th International Conference on Industrial Engineering and Engineering Management, Xiamen*, 2010, pp.557-560, 2010.

[67] L. Zhang, and J. Wu, "A PSO-Based Hybrid Metaheuristic for Permutation Flowshop Scheduling Problems", *The Scientific World Journal*, Vol. 2014, Article ID 902950, 2014, 8pages.

[68] C-L. Chen, S-Y. Huang, Y-R. Tzeng, and C-L. Chen, "A revised discrete particle swarm optimization algorithm for permutation flow-shop scheduling problem", Soft Computing, Vol. 18, 2014, pp. 2271–2282.

[69] T.R. Ramanan, M. Iqbal, and K. Umarali, "A particle swarm optimization approach for permutation flow shop scheduling problem", *International Journal for Simulation and multidisciplinary Design Optimization*, Vol. 5, 6 pages, 2014.

[70] M. Eddaly, B. Jarbouia, and P. Siarryb, "Combinatorial particle swarm optimization for solving blocking flowshop scheduling problem", *Journal of Computational Design and Engineering*, Vol. 3, No. 4, pp. 295-311, 2016.

[71] X. Li, and M. Yin, M, "A hybrid cuckoo search via Lévy flights for the permutation flow shop scheduling problem", *International Journal of Production Research*, Vol. 51, No. 16, 2013, pp. 4732-4754.

[72] H.Q. Zheng, Y.Q. Zhou and C. Xie, "Modified Cuckoo Search Algorithm for Solving Permutation Flow Shop Problem", *Intelligent Computing Theories and Application, ICIC 2016, Lecture Notes in Computer Science, Springer, Cham*, Vol. 9771, 2016, pp. 714–721.

[73] H. Wang, W. Wang, H. Sun, Z. Cui, S. Rahnamayan, and S. Zeng, "A new cuckoo search algorithm with hybrid strategies for flow shop scheduling problems", *Soft Computing*, Vol. 21, 2017, pp. 4297-4307.

[74] Y. Zhang, Y. Yu, S. Zhang, Y. Luo and L. Zhang, "Ant colony optimization for Cuckoo Search algorithm for permutation flow shop scheduling problem", Systems Science & Control Engineering, Vol. 7, No. 1, 2019, pp. 20-27.

[75] L. Zhang, Y. Yu, Y. Luo, and S. Zhang, "Improved cuckoo search algorithm and its application to permutation flow shop scheduling problem", Journal of Algorithms & Computational Technology, Vol. 14, 2020, pp. 1-12.

[76] R. Ruiz, and T. Stützle, "A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem",

European Journal of Operational Research, Vol. 177, No. 3, 2007, pp. 2033–2049.

[77] Q. Pan, M. Tasgetiren, and Y. Liang, "A discrete differential evolution algorithm for the permutation flowshop scheduling problem", *Computers & Industrial Engineering*, Vol. 55, No. 4, 2008, pp. 795-816.

[78] M. Saravanan, A.N. Haq, A.R. Vivekraj, and T. Prasad, "Performance evaluation of the scatter search method for permutation flowshop sequencing problems", *International Journal of Advanced Manufacturing Technology*, Vol. 37, 2008, pp. 1200–1208.

[79] B. Qian, L. Wang, R. Hu, W.L. Wang, D.X. Huang, and X. Wang, "A hybrid differential evolution method for permutation flow-shop scheduling", *International Journal of Advanced Manufacturing Technology*, Vol. 38, 2008, pp. 757-777.

[80] T. Zheng, and M. Yamashiro, "Solving flow shop scheduling problems by quantum differential evolutionary algorithm", *International Journal of Advanced Manufacturing Technology*, Vol. 49, 2010, pp. 643-662.

[81] M. Sayadi, R. Ramezanian, and N. Ghaffari-Nasab, "A discrete firefly meta-heuristic with local search for makespan minimization in permutation flow shop scheduling problems", *International Journal of Industrial Engineering Computations*, Vol. 1, No. 1, 2010, pp. 1-10.

[82] H. Liu, G. Liang, and Q. Pan, "A hybrid particle swarm optimization with estimation of distribution algorithm for solving permutation flowshop scheduling problem", *Expert Systems with Applications*, Vol. 38, No. 4, 2011, pp. 4348–4360.

[83] L. Wang, Q-K. Pan, and A.F. Tasgetiren, "A hybrid harmony search algorithm for the blocking permutation flow shop scheduling problem", *Computers & Industrial Engineering*, Vol. 61, No. 1, 2011, pp. 76-83.

[84] S-H. Chen, P-C. Chang, T.C.E. Cheng and Q. Zhang, "A Self-guided Genetic Algorithm for permutation flowshop scheduling problems", *Computers & Operations Research*, Vol. 39, No. 7, 2012, pp. 1450–1457.

[85] M.M. Kadarkarainadar and M. Geetha, "Solving flowshop scheduling problems using a Discrete African Wild Dog algorithm", *ICTACT Journal on Soft Computing,* Vol. 3, No. 3, 2013, pp. 555-558.

[86] X. Li and M. Yin, "An opposition-based differential evolution algorithm for permutation flowvshop scheduling based on diversity measure", *Advances in Engineering Software*, Vol. 55, 2013, pp. 10-31.

[87] H-Y. Wang, Y-B. Lu, and W-L. Peng, "Permutation flow-shop scheduling using a hybrid differential evolution algorithm", *International Journal of Computing Science and Mathematics,* Vol. 4, No. 3, 2013, pp. 298-307.

[88] J. Ceberio, E. Irurozki, A. Mendiburu, and J.A. Lozano, "A Distance-Based Ranking Model Estimation of Distribution Algorithm for the Flowshop Scheduling Problem", *IEEE Transactions on Evolutionary Computation*, Vol. 18, Vo. 2, 2014, pp. 286-300.

[89] S. Fong, H.L. Lou, Y. Zhuang, S. Deb, and T. Hanne, "Solving the Permutation Flow Shop Problem with Firefly Algorithm", *2014 2nd International Symposium on Computational and Business Intelligence, New Delhi,* 2014, pp. 25-29.

[90] R. Hariharan and G. Nimal, "Solving flow shop scheduling problems using a hybrid genetic scatter search algorithm", *Middle-East Journal of Scientific Research*, Vol. 20, No. 3, 2014, pp. 328-333.

[91] Q. Luo, Y. Zhou, J. Xie, M. Ma, and L. Li, "Discrete bat algorithm for optimal problem of permutation flow shop scheduling", *The Scientific World Journal*, Vol. 2014, 2014, 15 pages.

[92] F. Su, and Y. Li, "An Improve Firefly Algorithm and its application in permutation flow shop scheduling problem", *Applied Mechanics and Materials*, Vol. 651-653, 2014, pp. 2125-2129.

[93] Z. Xie, C. Zhang, X. Shao, W. Lin, and H. Zhu, "An effective hybrid teaching–learning-based optimization algorithm for permutation flow shop scheduling problem", *Advances in Engineering Software*, Vol. 77, 2014, pp. 35-47.

[94] A. Bouzidi, and M.E. Riffi, "Cat swarm optimization to solve flow shop scheduling problem", *Journal of Theoretical and Applied Information Technology*, Vol. 72, No. 2, 2015, pp. 239-243.

[95] Q. Lin, L. Gao, X. Li, and C. Zhang, "A hybrid backtracking search algorithm for permutation flow-shop scheduling problem", *Computers & Industrial Engineering*, Vol. 85, 2015, pp. 437-446.

[96] Ö. Tosun and M.K. Marichelvam, "Hybrid bat algorithm for flow shop scheduling problems", *International Journal of Mathematics in Operational Research*, Vol. 9, No. 1, 2016, pp. 125-138.

[97] K. Govindan, R. Balasundaram, N. Baskar, and P. Asokan, "A hybrid approach for minimizing makespan in permutation flowshop scheduling", *Journal of Systems Science and systems Engineering*, Vol. 26, 2017, pp. 50–76.

[98] M.K. Marichelvam, Ö. Tosun and M. Geetha, "Hybrid monkey search algorithm for flow shop scheduling problem under makespan and total flow time", *Applied Soft Computing*, Vol. 55, 2017, pp. 82-92.

[99] M. Abdel-Basset, M. Gunasekaran, D. El-Shahat and S. Mirjalili, "A hybrid whale optimization algorithm based on local search strategy for the permutation flow shop scheduling problem", *Future Generation Computer Systems*, Vol. 85, 2018, pp. 129-145.

[100] R. Nurdiansyah, O.A.W. Rijanto, B. Santosa, and S.E. Wiratno, "An Improved Differential Evolution Algorithm for Permutation Flow Shop Scheduling Problem", *International Journal of Operations Research*, Vol. 16, No. 2, 2019, pp. 37-44.

[101] K-W. Huang, G. Girsang, Z-X. Wu, and Y-W. Chuang, "A Hybrid Crow Search Algorithm for Solving Permutation Flow Shop Scheduling Problems", *Applied Sciences*, Vol. 9, No. 7, 2019.

[102] O.A. Arık, "Artificial bee colony algorithm including some components of iterated greedy algorithm for permutation flow shop scheduling problems", *Neural Computing and Applications*, 2020.