Received: September 8, 2021. Revised: November 26, 2021.

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

A Crossbreed Discrete Artificial Bee Colony for Permutation Flow Shop Scheduling Problem to Minimize Total Earliness and Tardiness

Annisa Kesy Garside[1]* **Ikhlasul Amallynda[1]**

[1]*University of Muhammadiyah Malang, Indonesia*
* Corresponding author's Email: annisa@umm.ac.id

Abstract: This paper considers the permutation flow shop scheduling problem to minimize total earliness and tardiness. We initiated a metaheuristic algorithm, namely a crossbreed discrete artificial bee colony. We modifications to the essential artificial bee colony and propose two versions of the crossbreed discrete artificial bee colony. Taguchi experimental design is used to test the performance of this. Several computational experiments using Vallada's benchmark instances have been carried out to prove the performance of the proposed algorithms. The statistical test results show that the proposed algorithms have the largest negative mean of BRE's value, each of -0.1959 for CDABC_ver1 and -0.1954 for CDABC_ver2. It means that the proposed algorithms perform significantly better than other algorithms. The results of the Kruskal Wallis H test show that the proposed algorithm has better performance than some dispatching rules and heuristic algorithms. Furthermore, the proposed algorithms can deliver better results in less time than mathematical model solution.

Keywords: Metaheuristic, Flow shop, Permutation, Scheduling problem, Earliness, Tardiness.

1. Introduction

Scheduling allocates resources over time to complete a series of tasks to optimize one or more specified objective functions. One of the most widely studied scheduling problems is the flow shop sheduling problem (FSP). There are different variants of FSP such as permutation, non-permutation, no-idle, nowait, and hybrid flow shop scheduling problems. The latest studies discussing review papers on FSP can be found in [1-4]. Permutation flow shop scheduling problem (PFSP) becomes the most common issue in obtaining a sequence given n jobs with the same order at m machines to find the objective function [5]. [2] discussed mathematical models and methods used by previous researchers to solve PFSP with the makespan criterion. [6] also reviewed and classified heuristics for PFSP with makespan. Meanwhile, [7] reviewed and evaluated the PFSP to minimize flowtime. From the results of the literature study, most of the PFSP still focuses on objectives based on the completion time (makespan and flowtime).

The manufacturing industry has gradually shifted from mass production to mass customization. Their production planning goal is to satisfy each customer's individual delivery dates, regardless of the amount of their order or delivery destinations. Customers' delivery dates are met, which means there are fewer early arrivals and late deliveries. As an optimality criterion, Just-In-Time (JIT) manufacturing considers both earliness and tardiness (ET) penalties. In a JIT scheduling system, jobs that are done early must be held in inventory until their due dates, whereas jobs that are performed after their due dates may decrease customer satisfaction [8]. Given the widespread adoption of just-in-time systems, there has been an increasing interest in scheduling problems in which both earliness and tardiness are penalized [9].

In the literature, some exact and approximation algorithms for the PFSP to minimize total ET have been proposed. In a multi-machine scheduling problem, [10] developed a mixed integer programming formulation. [11] proposed a heuristic method. Four new efficient heuristics are proposed by [9]. In order to reduce and accelerate the search space of the heuristics, these heuristics integrate several properties and a speed up procedure. [12] developed a variable neighbourhood search with mixed integer programming. [13] proposed a genetic algorithm. To solve PFSP, [14] presented two discrete particle swarm optimization (DPSO) algorithms. [8] developed a mix of Variable Neighbourhood Search (VNS) and Tabu Search (TS) which are stochastic search strategies capable of resolving the Multi-Objective ET Scheduling Problem (MOETSP).

ABC is a new swarm intelligence algorithm proposed by Karaboga in 2005 and inspired by honey bee behavior. Since its introduction, ABC has been used to address a variety of issues. ABC has a number of advantages: 1) Ease of hybridization with other optimization methods, 2) Use of fewer control parameters than many other search techniques, 3) Simplicity, flexibility, and robustness [15]. A survey paper on the ABC algorithm and its application was conducted by [16, 17]. However, only a few researchers have applied the ABC algorithm to PFSP. [18] proposed a DABC for the PFSP with total flow time criterion. [19] proposed a DABC with composite mutation strategies and fast local search for solving PFSP with the objective of minimizing total flow time and maximum lateness of jobs. [20] employed a hybrid DABC algorithm in solving the PFSP with the objective to minimize the makespan.

In the manufacturing industry, dispatching rules are used to handle a wide variety of scheduling problems [21]. The priority sequencing rules include shortest processing time (SPT), largest processing time (LPT), earliest due date (EDD). [22] stated that SPT and EDD are among the most frequenly used as benchmarks in the literature which focused on due date-based objectives (lateness, tardiness, etc.). Futhermore several heuristics had been developed for solving the PFSP e.g., Palmer [23] , Gupta [24], Nawaz-Enscore-Ham (NEH) [25]. The NEH heuristic is now one of the most effective constructive heuristics [22]. When compared with classical and some of the recent methods, ABC algorithm has produced promising results [26]. To the best of our knowledge, there is no one paper on ABC algorithm for minimizing the total earliness and tradiness in the PFSP. So, we proposed a Crossbreed Discrete Artificial Bee Colony (CDABC) algorithm. The difference between the CDABC algorithm and other DABC algorithms lies in the local search technique. In this case, we present several new local search techniques based on the crossover between feasible solutions. This effort was made to minimize the tendency of "getting stuck" in the local optima. Thus, we can develop more cost-effective (i.e., less time-consuming) global solutions to common real-world problems.

In this paper, a mathematical method is used to find an analytical solution. The consistency, stability, and convergence test of the proposed algorithm is carried out by comparing the numerical solution with the analytical solution. If the absolute error between the two is close to zero, the numerical solution converges to the analytical solution. This will be much more efficient than doing a comparison of the numerical solution with other numerical solutions. In addition, there are not many studies that examine the development of swarm intelligence-based metaheuristic methods to solve the ET-PSFP problem. Therefore, we decide to compare the proposed CDABC's solutions with the optimal solution of mathematical model, dispatching rules (SPT, EDD, and LPT), and some heuristic algorithms (Palmer, Gupta, and NEH). To obtain a tremendous experimental analysis, we performed the Kruskal Wallis H test to compare the quality of the solutions solved by each algorithm.

2. Problem description

Flow shop is scheduling several jobs where all of these jobs must go through the same sequence of operations/production processes. Flow shop which deals with sorting n jobs on machines, where each job must be processed exactly one time on each machine in the same order, with certain processing time. The permutation shop flow scheduling (PFSP) problem is a special case in FSP, where n jobs are in the same order on each machine. Therefore, there are of n jobs completed on m machines in the same order. Each job consists of a set of operations. The processing time of job j on machine i is denoted by $p_{j,i}$. Each job can be processed on only one machine, and every machine can process only one job at the same time. Even the operation cannot be preemptable. All of the jobs to be processed on machines in the same sequence. A permutation of jobs is denoted by $\pi = \{\pi_1, \pi_2, \dots, \pi_n\}$, where n jobs are going to be sequenced through m machine. Let $C(\pi_j, m)$ denote the completion time of job π_j on machine m. The completion time of the PFSP in line with the processing sequence π is shown as follow:

$$C(\pi, 1) = p_{\pi_1,1}$$

$$C(\pi_j, 1) = C(\pi_{j-1}, 1) + p_{\pi,1}, \quad j = 2, \dots, n$$
$$C(\pi_1, i) = C(\pi_1, i-1) + p_{\pi,i}, \quad i = 2, \dots, m \tag{1}$$
$$C(\pi_j, i) = \max\left(C(\pi_{j-1}, i), C(\pi_j, i-1)\right) + p_{\pi,i}, \quad j = 2, \dots, n;\ i = 2, \dots, m$$

Meanwhile, makespan is the time needed to complete all jobs at the shop. It can be defined by:

$$C_{max}(\pi^*) \leq C(\pi_n, m), \quad \forall \pi \in \Pi \tag{2}$$

As for the PFSP with the due date constraint, let $L(\pi_j)$ the lateness of jobs π_j and be defined as:

$$L(\pi_j) = C(\pi_j, m) - d(\pi_j) \tag{3}$$

Because earliness is positive lateness and tardiness is negative lateness, both can be defined as:

$$\begin{gathered} E(\pi_j) = \max(L(\pi_j), 0) \\ T(\pi_j) = \min(L(\pi_j), 0) \end{gathered} \tag{4}$$

Where $d(\pi_j)$ is due date of jobs π_j. Then, the sum of total earliness and total tardiness can be described as:

$$f(\pi) = \sum_{j=1}^{n} E(\pi_j) + \sum_{j=1}^{n} T(\pi_j) \tag{5}$$

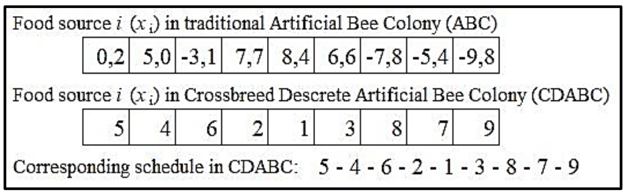
Furthermore, π^* as the optimal solution should satisfy by the following criterion:

$$f(\pi^*) \leq f(\pi), \quad \forall \pi \in \Pi \tag{6}$$

3. The Proposed algorithm

Because the basic ABC is a continuous algorithm, the standard continuous encoding scheme cannot solve PFSP directly. One of the most significant issues to solve the PFS by ABC is determining a close relationship between the job sequences and the individual vectors in ABC. Although combinatorial cases can be resolved by brute force, they will require less time (less efficiency). If the case is indeed complicated, so it cannot be accepted practice. In this case, the proposed Crossbreed Discrete Artificial Bee Colony (CDABC) algorithm can be connected to beat combinatorial problems more practically and regularly. It is very well to use because there is a structured search and evaluation mechanism. In the proposed CDABC, the food source i for scheduling problem and independent job in the PFSP is agreed with a job order $\pi = [[1], \ldots, [n]]$.

Additionally, $\varphi_\pi: \{1, \ldots, n\} \rightarrow \{1, \ldots, n\}$ is the mapping between the places during a π and the sequence job indexes. If jobs j within the pth dimension of the π, we get $\varphi_\pi(p) = j$. For example, if job 4 in the dimension of 2nd of the π= [3,4,1,5,2], therefore $\varphi_\pi(2) = 4$. The difference of food source in traditional ABC for continuous problems, CDABC, and corresponding schedule in CDABC is clearly shown in Fig. 1. Random generation is used as an initialization mechanism. In other words, there are NP solutions (food sources) within the population, and every one solution is constructed randomly at time zero.


Food source i (x_i) in traditional Artificial Bee Colony (ABC)

0,2	5,0	-3,1	7,7	8,4	6,6	-7,8	-5,4	-9,8

Food source i (x_i) in Crossbreed Descrete Artificial Bee Colony (CDABC)

5	4	6	2	1	3	8	7	9

Corresponding schedule in CDABC: 5 - 4 - 6 - 2 - 1 - 3 - 8 - 7 - 9

Figure. 1 An example of a CDABC solution

3.1 CDABC Ver1

The first proposed algorithm is modifying the traditional ABC in the employed, onlooker, and scout bee phases. Modifications are made by changing new solution generation techniques with neighborhood operators, consisting of swap operator, swap sequence, insert operator, and insert sequence. Fig. 3 shows the pseudo code of CDABC_ver1.

3.1.1. Detail CDABC_ver1

3.1.1.1. Employed bee phase

a. Swap operator (SO) or random swap

For example, the random number is SO (2,4), then the new solution generated from the initial solution with SO is as follows in Fig. 2:

$$\begin{gathered} x_i = x_i + SO \\ x_i = (2-6-8-9-1) + SO(2,4) \\ x_i = (2-9-8-6-1) \end{gathered}$$

b. Swap sequence or random swap sequences

$$\begin{gathered} x_i = x_i + SS \\ x_i = x_i + (SO_1, SO_2, SO_3, \ldots, SO_n) \end{gathered}$$

n = Number of swap sequences, then calculate the probability value.

$$Prob_i = \frac{fitness\ (x_i)}{\sum_{k=1}^{s} fitness\ (x_k)}$$

$Prob_i$ = The odds of choosing i-th employed bee.
S = The number of employed bees.

2	6	8	9	1
2	9	8	6	1

Figure. 2 An example of a swap operator

Table 1. Pseudo code of CDABC_ver1

```
1. Initialize all required parameters
2. Generate SN food source as an initial population $X_{ij}$
3. Evaluate the initial population, set as f($X_i$)
4. Find the initial best solution, and memorize it
5. Set $iter$ = 1
6. While $iter < iter_{max}$ do
7.   For $i$ = 1 : $SN$
8.     Set $trial_i$ = 0
       // Employed Bee Phase
9.     Produce a new neighbor solution $EB_i$ by the
       neighborhood operator (i.e. swap operator or
       swap sequence).
10.    Evaluate its fitness value, set as f($EB_i$)
11.    If f($EB_i$) < f($X_i$)
12.      then replace the old solution with new
         solution, $X_i = EB_i$
13.        set $trial_i$ = 0
       Else
14.      set $trial_i = trial_i + 1$
       End If
       // Probability Calculation Phase
15.    Calculate the probability values $P_i$ using
       equation below:
       $p_i = \frac{fit_i}{\sum_{j=1}^{SN} fit_i}$, where $fit_i = \begin{cases} \frac{1}{1+f_i} & f_i \geq 0 \\ 1 + abs(f_i) & f_i < 0 \end{cases}$
                                        // Onlooker Bee Phase
16.    If $P_i$ > a random value in the range of [0,1]
17.      Produce a new neighbor solution $OB_i$ by the
         neighborhood operator (i.e. insert operator
         or insert sequence).
18.        Evaluate its fitness value, set as f($OB_i$)
19.      If f($OB_i$) < f($X_i$)
20.        then replace the old solution with new
           solution, $X_i = OB_i$
21.        set $trial_i$ = 0
         Else
22.        set $trial_i = trial_i + 1$
         End If
       End If
       // Scout Bee Phase
23.    If $trial_i \geq trial_{max}$
24.      Generate new solution randomly, then set as
         $SB_i$
25.      Replace the older solution $X_i = SB_i$
       End If
     End For
26. Set $iter = iter + 1$
  End While
27. Memorize the best solution so far
```

3.1.1.2. Onlooker bee phase

a. Insert operator

The random number generated is IO (1,5). Then the new solutions resulting from the initial tour with IO are as follows in Fig. 3:

2	6	8	9	1
1	2	6	8	9

Figure. 3 An example for insert operator

$$x_i = x_i + IO$$
$$x_i = (2-6-8-9-1) + IO(1,5)$$
$$x_i = (1-2-6-8-9)$$

b. Insert sequence

$$x_i = x_i + IO$$
$$x_i = x_i + (IO_1, IO_2, IO_3, \dots, IO_n)$$

n = The number of insert sequence

3.1.1.3. Scout bee phase

After going through two repair solution phases, the employed bee level and the onlooker bee phase, each employed bee's quality calculation will be calculated. The number of scout bees is dynamic, depending on the number of employed bees exceeding the limit. If the bees' limit is to carry out the repair solution exceeds the specified maximum limit, the bee solution will be removed and replaced with a new solution using random techniques, updating the resulting distance, and resetting the limit back to 0.

3.2 CDABC Ver2

In the CDABC-ver2, different search operators are used to producing new solutions that have other exploration and exploitation characteristics differently. The search operator with the insert motion can be used to represent a permutation-based solution. Table. 2 shows the pseudo code of CDABC_ver2.

3.2.1. Detail CDABC_ver2

3.2.1.1. PBX (Position-based operator)

The position-based operator starts by selecting a random set of job positions from parent 1. Furthermore, this operator forces the selected jobs' position on the related jobs of the other parent. The Bernoulli test with 0,5 probability for each position is used in position selection. First, the selected job positions are copied to the corresponding offspring positions. The unselected positions are copied from the second parent to offspring, starting from the left according to the second parent order. For example, consider the parent sequence (7-6-3-2-9-1-8-5-4) and (3-2-4-6-1-7-9-5-8); and suppose that the first, third, eighth, and ninth positions are selected. So, we get

Parent 1	7	6	3	2	9	1	8	5	4
Offspring	7	2	3	6	1	9	8	5	4
Parent 2	3	2	4	6	1	7	9	5	8

Figure. 4 PBX operator scheme

the following offspring: (7-2-3-6-1-9-8-5-4) (see Fig. 4).

Table 2. Pseudo code of CDABC_ver2

```
1. Initialize all required parameters
2. Generate SN food source as an initial population $X_{ij}$
3. Evaluate the initial population, set as f($X_i$)
4. Set $iter$ = 1
5. While $iter$ =< $Max.iter$ do
    // Employed Bee Phase
6.  For each food source ($X_i$), where $i$ = 1 : $SN$
7.      Set $trial_i$ = 0
8.      Select the search operator S from (PBX, 1PX,
        OABX, and PABX) randomly
9.      Select a food source $Rand$i different from $X_i$,
        $G_{Best}$ and $L_{Best}$
10.         If S = 'OABX' or S = 'PABX'
12.         Set $R_i$ = a // a is random number between 0
            and 1
13.         If $R_i$ < ½
14.             Set (Parent1, Parent2, Parent3) as ($X_i$,
                $Rand_i$, $L_{Best}$)
            Else
15.             Set (Parent1, Parent2, Parent3) as ($X_i$,
                $Rand_i$, $G_{Best}$)
            End If
        Else
16.             Set (Parent1, Parent2) as ($X_i$, $Rand_i$)
        End If
17.     Produce a new neighbor solution $EB_i$ by S. Use
        the selected parent set with S.
18.     Evaluate its fitness value, set as f($EB_i$)
19.     If f($EB_i$) < f($X_i$)
20.         Replace the old solution with new solution,
            $X_i = EB_i$
21.         Set $trial_i$ = 0
        Else
22.         set $trial_i$ = $trial_i$ + 1
        End If
        // Probability Calculation Phase
23.     Calculate the probability values $P_i$ using
        equation below:
        $p_i = \frac{fit_i}{\sum_{j=1}^{SN} fit_i}$, where $fit_i = \begin{cases} \frac{1}{1+f_i} & f_i \ge 0 \\ 1 + abs(f_i) & f_i < 0 \end{cases}$
        // Onlooker Bee Phase
24.     If $P_i$ > a random value in the range of [0,1]
25.         Select the search operator S from (PBX,
            1PX, OABX, and PABX) randomly
26.         Select a food source $Rand$i different from
            $X_i$, $G_{Best}$ and $L_{Best}$
27.             If S = 'OABX' or S = 'PABX'
28.             Set $R_i$ = a // a is random number between
                0 and 1
29.             If $R_i$ < ½
30.                 Set (Parent1, Parent2, Parent3) as
                    ($X_i$, $Rand_i$, $L_{Best}$)
                Else
31.                 Set (Parent1, Parent2, Parent3) as
                    ($X_i$, $Rand_i$, $G_{Best}$)
                End If
            Else
32.                 Set (Parent1, Parent2) as ($X_i$, $Rand_i$)
            End If
33.         Produce a new neighbor solution $EB_i$ by S.
            Use the selected parent set with S.
34.         Evaluate its fitness value, set as f($EB_i$)
35.         If f($EB_i$) < f($X_i$)
36.             Replace the old solution with new
                solution, $X_i = EB_i$
37.             Set $trial_i$ = 0
            Else
38.             Set $trial_i$ = $trial_i$ + 1
            End If
        End If
        // Scout Bee Phase
39.     If $trial_i \ge Max.trial$
40.         Generate new solution randomly, then set as
            $SB_i$
41.         Replace the older solution $X_i = SB_i$
        End If
    End For
42. Set $iter$ = $iter$ + 1
   End While
43. Memorize the best solution so far
```

Parent 1	7	6	3	**2**	9	1	8	5	4
Offspring	7	6	3	2	4	1	9	5	8
Parent 2	3	2	4	6	1	7	9	5	8

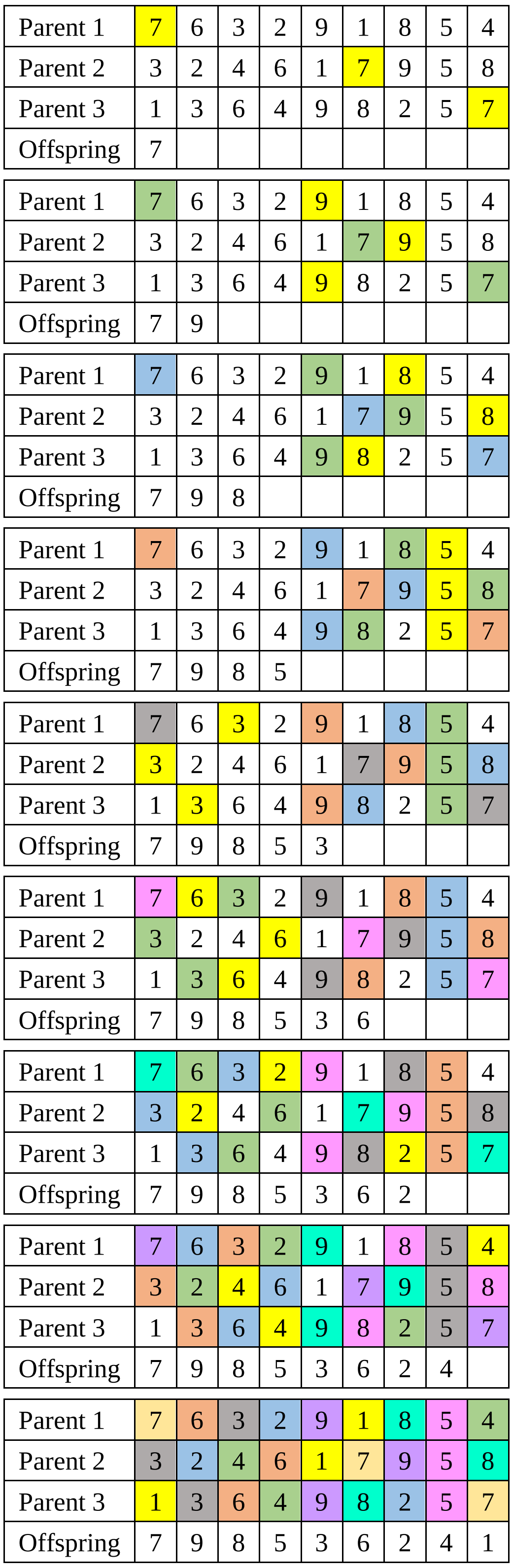
Figure. 5 1PX operator scheme

3.2.1.2. IPX (one-point operator)

One arbitrary point is selected from Parent 1 and splits Parent 1 into two job sequences. Jobs in the left-hand sequence are copied from Parent 1 to offspring directly. The order of the jobs on the right is arranged according to the job position in parent 2. Fig. 5 depicts an example of a 1PX operator, where the points between the fourth and fifth positions are randomly selected for the dividing point.

3.2.1.3. OABX (occurrence adjacency-based operator)

The first job on Parent 1 and copies to offspring. Then, at each step, the first successor job of previously copied jobs, which is not offspring, is selected from the respective parent. Fig. 6 illustrates an example of the OABX operator. The yellow cells show each parent's selected occupation, green cells indicate the job copied in the previous step, and other color cells show the offspring's value set.



Parent 1	7	6	3	2	9	1	8	5	4
Parent 2	3	2	4	6	1	7	9	5	8
Parent 3	1	3	6	4	9	8	2	5	7
Offspring	7								

Parent 1	7	6	3	2	9	1	8	5	4
Parent 2	3	2	4	6	1	7	9	5	8
Parent 3	1	3	6	4	9	8	2	5	7
Offspring	7	9							

Parent 1	7	6	3	2	9	1	8	5	4
Parent 2	3	2	4	6	1	7	9	5	8
Parent 3	1	3	6	4	9	8	2	5	7
Offspring	7	9	8						

Parent 1	7	6	3	2	9	1	8	5	4
Parent 2	3	2	4	6	1	7	9	5	8
Parent 3	1	3	6	4	9	8	2	5	7
Offspring	7	9	8	5					

Parent 1	7	6	3	2	9	1	8	5	4
Parent 2	3	2	4	6	1	7	9	5	8
Parent 3	1	3	6	4	9	8	2	5	7
Offspring	7	9	8	5	3				

Parent 1	7	6	3	2	9	1	8	5	4
Parent 2	3	2	4	6	1	7	9	5	8
Parent 3	1	3	6	4	9	8	2	5	7
Offspring	7	9	8	5	3	6			

Parent 1	7	6	3	2	9	1	8	5	4
Parent 2	3	2	4	6	1	7	9	5	8
Parent 3	1	3	6	4	9	8	2	5	7
Offspring	7	9	8	5	3	6	2		

Parent 1	7	6	3	2	9	1	8	5	4
Parent 2	3	2	4	6	1	7	9	5	8
Parent 3	1	3	6	4	9	8	2	5	7
Offspring	7	9	8	5	3	6	2	4	

Parent 1	7	6	3	2	9	1	8	5	4
Parent 2	3	2	4	6	1	7	9	5	8
Parent 3	1	3	6	4	9	8	2	5	7
Offspring	7	9	8	5	3	6	2	4	1

Figure. 6 OABX operator scheme

3.2.1.4. PABX (probability adjacency-based operator)

Generally, the PABX's mechanism is similar to the OABX. However, this operator uses a probabilistic selection to determine the parent where the job will be copied. Three probability selections, i.e., p_1; p_2; and p_3, are predefined for each parent such that $p_1 + p_2 + p_3 = 1$. At each selection step, the parent's determination is made based on this probability, and the offspring will inherit a job from the first successor of the selected parent.

4. Result and discussion

4.1 Parameter tuning

The quality and effectiveness of algorithm are affected by parameter configuration. Therefore, to obtain optimal performance, these parameters must be adjusted. Taguchi's experimental design method was used to analyze the impacts of the whole set of parameters using a small number of experiments based on an extraordinary orthogonal array design. There are three control parameters considered for factor analysis to optimize CDABC performance: (1) population size, (2) number of maximum (limit) increase trials (3) selection probability of the PABX operator. Each parameter is categorized into five levels and summarized in Table 3. An orthogonal array $L_{25}(5^5)$ with two dummy factors was chosen as the experimental design.

The signal-to-noise (S/N) ratio is used to analyze variations of CDABC performance in response to each set of parameters. The signal describes the desired impact of the yield, while the noise represents the undesired impact. Because the S / N ratio comes from the quadratic loss function, three examples are widely used which are (1) the nominal is best, (2) the smaller is better and (3) the bigger is better. In this study, the smaller the better it is used, because the objective function is a minimization. Then, the S / N ratio is calculated by:

Table 3. Levels of parameter

No	**Factor**	**Level Factor**				
		1	**2**	**3**	**4**	**5**
1	Population Size (SN)	50	100	150	200	250
2	Max. Trial Number (Max trial)	25	50	75	100	125
3	Selection Probabilities (p_1, p_2, p_3)	0.2; 0.6; 0.2	0.6; 0.2; 0.2	0.3; 0.35; 0.3	0.2; 0.2; 0.6	0.45; 0.1; 0.45

Table 4. Response table of factors (SN, Max.Trial, SP)

Received: September 8, 2021. Revised: November 26, 2021.

SN Ratios Response Table			
Smaller is better			
Level	**SN**	**Max.Trial**	**SP**
1	-11,318	-9,728	-6,631
2	-9,823	-8,513	-11,708
3	-8,163	-8,933	-9,823
4	-7,147	-9,599	-11,584
5	-11,745	-11,422	-8,449
Delta	4,597	2,909	5,077
Rank	2	3	1

$$\frac{S}{N} = -10\log\left(\frac{1}{n}\sum_{i=1}^{n} y_i^2\right) \quad (7)$$

where n is the number of replication and y is the output of the ith replication.

In this study, five instances are selected based on each job size (50, 150, 250, 350) from benchmark sets, and five replications are done for each problem. The CPU time limit is selected as the secondary stopping criteria and set to 60 minutes. The best relative error (BRE) is taken as the output. Table 4 shows that the SN-5, Max.Trial-5 and SP-2 give the smallest S/NR value, it means that these factors give the smallest BRE. Then, the row with rank values denotes the descending rank of factors due to descending Δ values. The results show that SP-1 is the factor that most influences the algorithm performance (BRE's value), followed by Max Trial-2 and SN-4.

Table 5. ANOVA for parameter tunning

Analysis of Variance						
		Seq	**Adj**	**Adj**	**F-**	**P-**
Source	**DF**	**SS**	**SS**	**MS**	**Value**	**Value**
SN	4	7,850	7,850	1,9624	11,06	0,001
Max.Trial	4	3,517	3,517	0,8793	4,96	0,014
SP	4	10,215	10,215	2,5537	14,40	0,000
Error	12	2,129	2,129	0,1774		
Total	24	23,711				

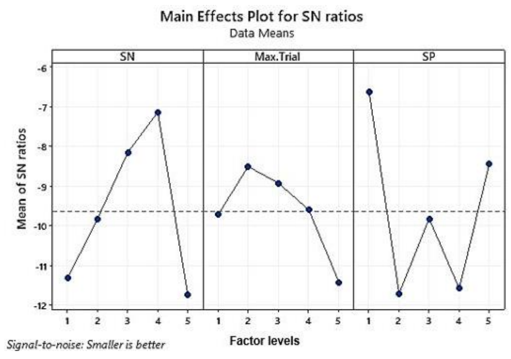


Figure. 7 Factor level patterns (SN, Max.Trial, SP)

ANOVA analysis was also performed to measure the level of statistical significance of the effect of the factor on the outcome. Then the results are shown in Table 5. It can be reasoned that all factors have a measurably significant impact on the results with a 95% confidence level. Since all p-values are less than 0.05 and the residue is extensively small. In addition, the factor level pattern for each level is shown in Fig. 7. Based on the maximum response rate obtained for each factor, the best configuration for the three control parameters is as follows: $SN = 250$, $Max.Trial = 125$ and $p_1, p_2, p_3 = (0.60, 0.20, 0.20)$. The parameter setting results show that bigger population and higher trial limit give better results. In addition, improved execution can be reached by using a higher selection probability for the third parent in the exploitation behaviour of proposed CDABC_ver2.

4.2 Comparison algorithm and evaluation

To analyze the proposed algorithm efficiency, the benchmark instances are solved. In this paper, we have selected the benchmark instances proposed by [27]. There are two considerations: 1) the instance set is explained in detail, and we could download it via http://soa.iti.es/rruiz, 2) the number of jobs up to 350 and the number of machines up to 50; therefore, we get instances to vary widely. Regarding the number of jobs and machines, the following values are selected: $n = \{50,150,250,350\}$, and the number of machines $m = \{10,30,50\}$. The due dates are generated with a uniform distribution according to the tardiness factor (T) and the due date range (R). They proposed the following values: $T = \{0.2, 0.4, 0.6\}$ and $R = \{0.2, 0.6, 1\}$. Vallada used a full factorial experiment with five replications in each combination, so there are 540 test problems related to the n, m, T, R configurations described previously.

This research has used taguchi experimental design to test the performance of the proposed algorithms. As we know, the number of experiments to be conducted under full factorial design is very high as many factors are to be examined. When the number of factors increases, it becomes laborious and complex. To overcome this, Taguchi suggested experimental design using an orthonal array. We can perform the more factors with lesser number of experiments, so that costs and time can be saved [28].

Based on the Vallada configuration described in the benchmark instances. We have chosen the following configuration for the experimental Taguchi design: $T = (0.2, 0.4, 0.6)$, $R = (0.2, 0.6, 1.0)$, $n = (50, 150, 250)$, and $m = (10, 30, 50)$. We have 4 factors and each of them has 3 levels, then an $L_9(3^2)$

Table 6. BRE and ARE value of the proposed algorithms

Experiment Number	**Instance**	**Test Problem T | R | n | m | replication**	**C***	**CDABC_ver1**		**CDABC_ver2**	
				BRE	**ARE**	**BRE**	**ARE**
1	I_0,2_0,2_50_10	0,2_0,2_50_10_1	14820	0,974	1,004	1,069	1,106
		0,2_0,2_50_10_2	14175	1,004	1,054	1,004	1,048
		0,2_0,2_50_10_3	15146	0,830	0,961	0,971	0,992
		0,2_0,2_50_10_4	14844	0,986	1,046	0,937	1,062
		0,2_0,2_50_10_5	18050	0,514	0,619	0,557	0,599
2	I_0,2_0,6_150_30	0,2_0,6_150_30_1	62638	1,598	1,612	1,498	1,506
		0,2_0,6_150_30_2	66167	1,202	1,283	1,217	1,220
		0,2_0,6_150_30_3	85190	1,216	1,234	1,176	1,205
		0,2_0,6_150_30_4	79305	0,974	0,997	0,919	0,946
		0,2_0,6_150_30_5	328471	-0,483	-0,475	-0,471	-0,468
3	I_0,2_1_250_50	0,2_1_250_50_1	Unknown	-1,000	-1,000	-1,000	-1,000
		0,2_1_250_50_2	Unknown	-1,000	-1,000	-1,000	-1,000
		0,2_1_250_50_3	Unknown	-1,000	-1,000	-1,000	-1,000
		0,2_1_250_50_4	Unknown	-1,000	-1,000	-1,000	-1,000
		0,2_1_250_50_5	Unknown	-1,000	-1,000	-1,000	-1,000
4	I_0,4_0,2_150_50	0,4_0,2_150_50_1	Unknown	-1,000	-1,000	-1,000	-1,000
		0,4_0,2_150_50_2	Unknown	-1,000	-1,000	-1,000	-1,000
		0,4_0,2_150_50_3	Unknown	-1,000	-1,000	-1,000	-1,000
		0,4_0,2_150_50_4	Unknown	-1,000	-1,000	-1,000	-1,000
		0,4_0,2_150_50_5	Unknown	-1,000	-1,000	-1,000	-1,000
5	I_0,4_0,6_250_10	0,4_0,6_250_10_1	930579	-0,589	-0,588	-0,586	-0,585
		0,4_0,6_250_10_2	822150	-0,511	-0,507	-0,507	-0,497
		0,4_0,6_250_10_3	945230	-0,596	-0,596	-0,594	-0,594
		0,4_0,6_250_10_4	239158	0,663	0,667	0,698	0,764
		0,4_0,6_250_10_5	269043	0,439	0,455	0,432	0,437
6	I_0,4_1_50_30	0,4_1_50_30_1	96254	-0,201	-0,172	-0,174	-0,163
		0,4_1_50_30_2	99089	-0,264	-0,241	-0,241	-0,227
		0,4_1_50_30_3	67972	-0,073	-0,069	-0,063	-0,055
		0,4_1_50_30_4	70799	0,012	0,022	-0,023	0,018
		0,4_1_50_30_5	62145	-0,052	0,005	-0,007	0,044
7	I_0,6_0,2_250_30	0,6_0,2_250_30_1	Unknown	-0,999	-0,999	-0,999	-0,999
		0,6_0,2_250_30_2	Unknown	-0,999	-0,999	-0,999	-0,999
		0,6_0,2_250_30_3	Unknown	-0,999	-0,999	-0,999	-0,999
		0,6_0,2_250_30_4	Unknown	-0,999	-0,999	-0,999	-0,999
		0,6_0,2_250_30_5	Unknown	-0,999	-0,999	-0,999	-0,999
8	I_0,6_0,6_50_50	0,6_0,6_50_50_1	166159	-0,024	-0,017	-0,069	-0,062
		0,6_0,6_50_50_2	163556	-0,064	-0,059	-0,075	-0,063
		0,6_0,6_50_50_3	158970	-0,034	-0,015	-0,058	-0,029
		0,6_0,6_50_50_4	155808	-0,070	-0,055	-0,063	-0,042
		0,6_0,6_50_50_5	165306	-0,052	-0,040	-0,068	-0,064
9	I_0,6_1_150_10	0,6_1_150_10_1	188782	0,157	0,181	0,162	0,166
		0,6_1_150_10_2	158740	0,234	0,263	0,146	0,223
		0,6_1_150_10_3	458305	-0,497	-0,494	-0,501	-0,497
		0,6_1_150_10_4	411084	-0,584	-0,570	-0,573	-0,569
		0,6_1_150_10_5	450930	-0,533	-0,526	-0,518	-0,513
			Average	**-0,196**	**-0,178**	**-0,195**	**-0,180**

Received: September 8, 2021. Revised: November 26, 2021.

orthogonal array (OA) is most suitable for experimental design. This OA assumes that there is no interaction between any two factors. There are totally nine experiments to be conducted and five replications in each experiment. In total, we only need 45 of the 540 test problems in Vallada's benchmark instances.

Each test problem is solved using an exact model, dispatching rules and heuristic algorithms, and proposed algorithms. All algorithms are coded in MATLAB R2014a and run on a computer with Intel Core i3-6006U CPU at 2.0 GHz, 8Gb RAM, and 1TB HDD in Windows 7 system. Meanwhile, the solution of the PFSP mathematical model using an exact algorithm will be obtained by using LINGO solver. PFSP problem is an NP-hard optimization problem. LINGO software is not able to solve large-scale problems in an acceptable time. To get a fair performance comparison between exact and heuristic algorithms, the computation time to solve each instance is equal to 60 minutes. The algorithm parameters are determined based on the results of the previous tuning parameters, i.e. $SN = 250$, $Max.Trial = 125$ and $p_1, p_2, p_3 = (0.60, 0.20, 0.20)$.

The best objective of the mathematical model solution obtained from the LINGO software (C*) are shown in Table 6. We compared the relative error of the two proposed algorithms in Table 6. BRE represents the mean of the relative error to C*. Then, ARE represents the mean of the average relative error to C*. BRE and ARE are defined:

$$BRE = \frac{Heu_{best} - Best_{known}}{Best_{known}} \tag{8}$$

$$RE = \sum_{i=1}^{n} \left[\frac{Heu_i - Best_{know}}{Best_{know}} \times \frac{1}{n}\right] \tag{9}$$

From Table 6, we can see running for 60 minutes with LINGO software on experiments 3, 4, and 7 (the combination of the number of jobs and machines are 250 and 50, 150 and 50, 250 and 30), then the solution is still unknown. The mean BRE and ARE values for the whole experiment between CDABC_ver1 and CDABC_ver2 are only different by 0.001 and 0.002, respectively. Therefore, we can conclude that the performance of the two algorithms is almost the same.

The computing result in Table 7 and Fig. 8 show that the number of machines *m* (Delta 52.278, Rank = 1) has the largest effect on the S/N ratio, followed by number of jobs *n* (Delta 48.735, Rank = 2), then followed by due date range *R*, and tardiness factor *T*. Besides, the Table 7 indicates that the number of machine *m* has the largest effect on the signal-to-

Table 7. Response table of factors (T, R, n, m)

a) **SN Ratios Response Table (CDABC_ver1)**

Smaller is better

Level	T	R	n	m
1	32,549	49,013	6,962	6,882
2	31,391	6,271	29,627	26,235
3	28,612	37,268	55,963	59,435
Delta	3,937	42,743	49,001	52,553
Rank	4	3	2	1

b) **SN Ratios Response Table (CDABC_ver2)**

Smaller is better

Level	T	R	n	m
1	31,914	48,519	6,564	6,518
2	30,909	6,015	29,349	25,898
3	28,390	36,679	55,299	58,796
Delta	3,525	42,504	48,735	52,278
Rank	4	3	2	1

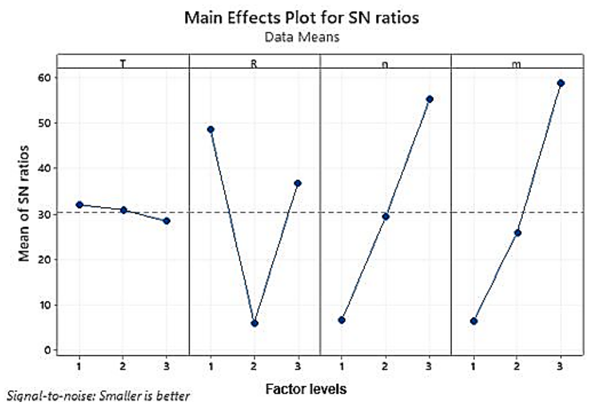


Figure. 8 Factor level trends (T, R, n, m)

Table 8. ANOVA for normalized BRE in different problem size

a) **Analysis of Variance (CDABC_ver1)**

Source	DF	Seq SS	Adj SS	Adj MS	F-Value	P-Value
T	2	0,6761	0,6761	0,33807	16,44	0,000
R	2	0,6529	0,6529	0,32643	15,88	0,000
N	2	1,0001	1,0001	0,50003	24,32	0,000
M	2	0,8524	0,8524	0,42620	20,73	0,000
Error	36	0,7403	0,7403	0,02056		
Total	44	3,9217				

b) **Analysis of Variance (CDABC_ver2)**

Source	DF	Seq SS	Adj SS	Adj MS	F-Value	P-Value
T	2	0,7449	0,7449	0,37244	17,64	0,000
R	2	0,6621	0,6621	0,33104	15,68	0,000
N	2	1,1000	1,1000	0,55002	26,05	0,000
M	2	0,9568	0,9568	0,47839	22,66	0,000
Error	36	0,7601	0,7601	0,02112		
Total	44	4,2239				

Table 9. Result of comparison between proposed CDABC, dispatching rules, and heuristic algorithms

s	instances	BRE	SPT [21]	EDD [21]	LPT [21]	Palmer [23]	Gupta [24]	NEH [25]	DABC_ver1	DABC_ver2
1	I_0,2_0,2_50_10	Mean (%)	1,974	1,179	1,575	1,423	1,633	1,024	0,862	0,908
		Max (%)	2,266	1,337	1,935	1,674	1,784	1,182	1,004	1,069
		Best (%)	1,502	0,958	1,070	0,887	1,217	0,578	0,514	0,557
2	I_0,4_1_50_30	Mean (%)	4,201	0,896	3,869	3,561	3,793	∞	0,901	0,868
		Max (%)	6,420	1,511	5,347	5,607	5,452	∞	1,598	1,498
		Best (%)	0,335	-0,469	0,264	0,229	0,318	∞	-0,483	-0,471
3	I_0,6_0,6_50_50	Mean (%)	-0,999	-1,000	-0,999	-0,999	-0,999	∞	-1,000	-1,000
		Max (%)	-0,999	-1,000	-0,999	-0,999	-0,999	∞	-1,000	-1,000
		Best (%)	-0,999	-1,000	-0,999	-0,999	-0,999	∞	-1,000	-1,000
4	I_0,6_1_150_10	Mean (%)	-0,999	-1,000	-0,999	-1,000	-0,999	∞	-1,000	-1,000
		Max (%)	-0,999	-1,000	-0,999	-0,999	-0,999	∞	-1,000	-1,000
		Best (%)	-0,999	-1,000	-0,999	-1,000	-0,999	∞	-1,000	-1,000
5	I_0,2_0,6_150_30	Mean (%)	1,312	-0,106	1,237	1,084	1,253	∞	-0,119	-0,111
		Max (%)	3,070	0,698	3,401	2,969	3,265	∞	0,663	0,698
		Best (%)	0,125	-0,594	-0,018	-0,030	0,028	∞	-0,596	-0,594
6	I_0,4_0,2_150_50	Mean (%)	0,158	-0,008	0,265	0,103	0,225	-0,258	-0,116	-0,101
		Max (%)	0,318	0,109	0,484	0,283	0,510	-0,162	0,012	-0,007
		Best (%)	-0,053	-0,107	-0,015	-0,106	-0,061	-0,381	-0,264	-0,241
7	I_0,4_0,6_250_10	Mean (%)	-0,999	-0,999	-0,999	-0,999	-0,999	∞	-0,999	-0,999
		Max (%)	-0,999	-0,999	-0,999	-0,999	-0,999	∞	-0,999	-0,999
		Best (%)	-0,999	-0,999	-0,999	-0,999	-0,999	∞	-0,999	-0,999
8	I_0,6_0,2_250_30	Mean (%)	-0,045	0,042	0,030	-0,032	0,030	-0,145	-0,049	-0,067
		Max (%)	-0,019	0,091	0,049	-0,013	0,049	-0,116	-0,024	-0,058
		Best (%)	-0,083	0,018	0,004	-0,049	0,004	-0,171	-0,070	-0,075
9	I_0,2_1_250_50	Mean (%)	0,580	-0,215	0,759	0,560	0,697	∞	-0,245	-0,257
		Max (%)	1,744	0,277	1,844	1,820	1,752	∞	0,234	0,162
		Best (%)	-0,054	-0,557	0,036	-0,126	0,031	∞	-0,584	-0,573

noise ratio. On average, experimental runs with number of machines in level 3 had much higher signal-to-noise ratios than the other at different levels. The tardiness factor had a small effect or no effect on the signal-to-noise ratio.

Based on ANOVA output in Table 8, it can be seen that all the selected factors significantly affect the BRE value. This can be seen by comparing the F-ratio using alpha 5%. From the F table, $F_{0,05;2;9} = 4{,}256$ is much smaller than the calculated F-ratio of each factor. In addition, Sig. value at P-value = 0.000 across all factors. Thus, at the real level = 0.05, we reject Ho. It means that the problem complexity has a significant effect on the quality of the resulting solution (based on the normalized BRE value).

Our proposed algorithms are compared with some dispatching rules such as SPT, EDD, and LPT. In addition, we compared the solution with several heuristic algorithms such as Palmer, Gupta and NEH. Any best feasible solutions are generated by each algorithm. It will be compared with the optimal solution which is obtained from the mathematical model. So that the mean, max, and best value of BRE are obtained (see Table 9).

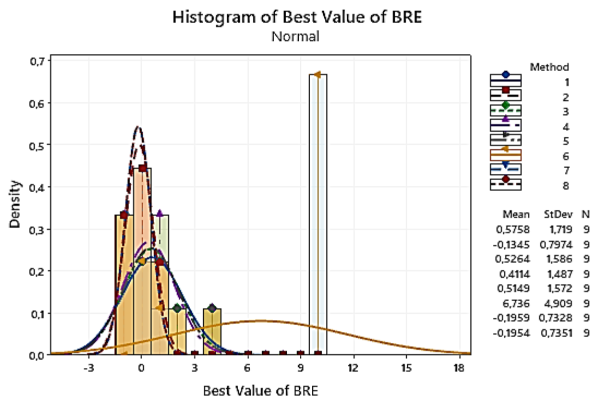
Furthermore, we performed the Kruskal Wallis H test to compare the quality of the solutions solved by each algorithm. The Kruskal Wallis test is a ranking-based nonparametric test whose objective is to determine whether there is a significant difference between two or more independent variables on the dependent variable on the numerical data scale (interval/ratio) or ordinal scale. The Kruskall Wallis H value is indicated by the H-value of 15.68 on DF 7 with a P-value of 0.028, which is smaller than the critical limit of 0.05 so that the hypothesis decision is to accept H1 or which means that each method makesa significant difference to the quality of the resulting solution (best value of BRE).

The Kruskall Wallis value for TIES is also not much different, 15.69 on DF 7 with a P-value of 0.028 (see Table 10), which is still smaller than the critical limit of 0.05. It means that the two algorithms

Received: September 8, 2021. Revised: November 26, 2021.

Table 10. Kruskall Wallis H test for comparing result

Wallis H Test			
Null hypothesis	H_0: All medians are equal		
Alternative hypothesis	H_1: At least one median is different		
Method	**DF**	**H-Value**	**P-Value**
Not adjusted forties	7	15,68	0,028
Adjusted forties	7	15,69	0,028



Mean	StDev	N
0,5758	1,719	9
-0,1345	0,7974	9
0,5264	1,586	9
0,4114	1,487	9
0,5149	1,572	9
6,736	4,909	9
-0,1959	0,7328	9
-0,1954	0,7351	9

Figure. 9 Histogram of BRE's best value

can provide the best solution from other algorithms. Furthermore, it is known that the two proposed algorithms have the largest negative mean of BRE's best value, each of -0.1959 for CDABC_ver1 and -0.1954 for CDABC_ver2 (see Fig. 9). It means that the two algorithms can provide the best solution from other algorithms.

5. Conclusion

In this paper, we consider the permutation flow shop scheduling problem by minimizing total earliness and tardiness. We propose two versions of the Crossbreed Discrete Artificial Bee Colony (CDABC). Several computational experiments using benchmark instances have been carried out to prove the performance of the proposed algorithms. This research has used taguchi experimental design to test the performance of the proposed algorithms. We can perform with lesser number of experiments, so that cost and time can be saved. The statistical test results show that the proposed algorithms have the largest negative mean of BRE's value, each of -0.1959 for CDABC_ver1 and -0.1954 for CDABC_ver2. It means that the proposed algorithms perform significantly better than other algorithms. The results of the Kruskal Wallis H test show that the proposed algorithm has better performance than some dispatching rules and heuristic algorithms. Furthermore, the proposed algorithms can deliver better results in less time than mathematical model solution. Therefore, future research can be directed to comparisons with several metaheuristic algorithms developed by other researchers. The proposed algorithms are modified to solve the permutation flow shop scheduling problem with other objective functions such as makespan, mean flow time, and maximum tardiness.

References

[1] H. Singh, J. S. Oberoi, and D. Singh, "Multi-Objective Permutation and Non-Permutation Flow Shop Scheduling Problems with No-Wait: A Systematic Literature Review", *RAIRO - Operations Research*, Vol. 55, No. 1, pp. 27-50, 2021.

[2] A. N. H. Zaied, M. M. Ismail, and S. S. Mohamed, "Permutation Flow Shop Scheduling Problem with Makespan Criterion : Literature Review", *Journal of Theoritical and Applied Information Technology*, Vol. 99, No. 4, pp. 830-848, 2021.

[3] D. A. Rossit, F. Tohmé, and M. Frutos, "The Non-Permutation Flow-Shop Scheduling Problem: A Literature Review", *Omega*, Vol. 77, pp. 143-153, 2018.

[4] Ö. Tosun, M. K. Marichelvam, and N. Tosun, "A Literature Review on Hybrid Flow Shop Scheduling", *International Journal of Advanced Operations Management*, Vol. 12, No. 2, pp. 156-194, 2020.

[5] M. M. Yenisey and B. Yagmahan, "Multi-Objective Permutation Flow Shop Scheduling Problem: Literature Review, Classification and Current Trends", *Omega*, Vol. 45, pp. 119-135, 2014.

[6] J. M. Framinan and R. Leisten, "A Heuristic for Scheduling A Permutation Flowshop with Makespan Objective Subject to Maximum Tardiness", *International Journal of Production Economics*, Vol. 99, No. 1-2, pp. 28-40, 2006.

[7] Q. K. Pan and R. Ruiz, "A Comprehensive Review and Evaluation of Permutation Flowshop Heuristics to Minimize Flowtime", *Computers and Operations Research*, Vol. 40, pp. 117-128, 2013.

[8] I. A. Gbolahan, A. O. Muminu, and S. A. Babatunde, "Hybrid Metaheuristic for Just In Time Scheduling in a Flow Shop with Distinct Time Windows", *International Journal of Methematical Analysis and Optimization: Theory and Applications*, Vol. 1, pp. 741-756, 2020.

[9] V. F. Viagas, M. Dios, and J. Framinan, "Efficient Constructive and Composite Heuristics for The Permutation Flowshop To Minimise Total Earliness And Tardiness", *Computers and Operations Research*, Vol. 75, pp. 38-48, 2016.

[10] Z. Zhu and R. B. Heady, "Minimizing The Sum of Earliness/Tardiness in Multi-Machine Scheduling: A Mixed Integer Programming Approach", *Computers and Industrial Engineering*, Vol. 38, No. 2, pp. 297-305, 2000.

[11] P. Chandra, P. Mehta, and D. Tirupati, "Permutation Flow Shop Scheduling with Earliness and Tardiness Penalties", *International Journal of Production Research*, Vol. 47, No. 20, pp. 5591-5610, 2009.

[12] R. M'Hallah, "Minimizing Total Earliness and Tardiness on A Permutation Flow Shop Using VNS and MIP", *Computers and Industrial Engineering*, Vol. 75, pp. 142-156, 2014.

[13] J. Schaller and J. M. S. Valente, "A Comparison of Metaheuristic Procedures to Schedule Jobs In A Permutation Flow Shop to Minimise Total Earliness and Tardiness", *International Journal of Production Research*, Vol. 51, No. 3, pp. 772-779, 2013.

[14] I. Amallynda, "The Discrete Particle Swarm Optimization Algorithms for Multi-Objective Permutation Flowshop Scheduling Problem", *Jurnal Teknik Industri*, Vol. 20, No. 2, pp. 105-116, 2019.

[15] M. S. Beg and A. A. Waoo, "A Comprehensive Study of Artificial Bee Colony (ABC) Algorithms and Its A Applications", *International Research Journal of Engineering and Technology*, Vol. 6, No. 5, pp. 5086-5093, 2019.

[16] D. Karaboga, B. Gorkemli, C. Ozturk, and N. Karaboga, "A Comprehensive Survey: Artificial Bee Colony (ABC) Algorithm and Applications", *Artificial Intelligence Review*, Vol. 42, No. 1, pp. 21-57, 2014.

[17]A. Nabaei, M. Hamian, M. R. Parsaei, R. Safdari, T. S. Soltani, H. Zarrabi, and A. Ghassemi, "Topologies and Performance of Intelligent Algorithms: a Comprehensive Review", *Artificial Intelligence Review*, Vol. 49, No. 1, pp. 79-103, 2018.

[18] M. F. Tasgetiren, Q. K. Pan, P. N. Suganthan, and A. H. L. Chen, "A Discrete Artificial Bee Colony Algorithm for The Total Flowtime Minimization in Permutation Flow Shops", *Information Sciences*, Vol. 181, No. 16, pp. 3459-3475, 2011.

[19] X. Li and M. Yin, "A Discrete Artificial Bee Colony Algorithm with Composite Mutation Strategies for Permutation Flow Shop Scheduling Problem", *Scientia Iranica*, Vol. 19, No. 6, pp. 1921-1935, 2012.

[20] Y. F. Liu and S. Y. Liu, "Hybrid Discrete Artificial Bee Colony Algorithm for Permutation Flowshop Scheduling Problem", *Applied Soft Computing*, Vol. 13, No. 3, pp. 1459-1463, 2013.

[21] M. Suryaprakash, R. Sridhar, R. Jayachitra, and M. G. Prabha, "Comparison of Dispatching Rules in a Flow Shop Scheduling Problem Using Simulation", In: *Proc. of Lecture Notes in Mechanical Engineering*, 2021.

[22] T. C. Chiang and L. C. Fu, "Using Dispatching Rules for Job Shop Scheduling with Due Date-Based Objectives", *International Journal of Production Research*, Vol. 45, No. 14, pp. 3245-3262, 2007.

[23] D. S. Palmer, "Sequencing Jobs Through A Multi-Stage Process in The Minimum Total Time - A Quick Method Of Obtaining A Near Optimum", *Operational Research Quarterly*, Vol. 16, pp. 101-107, 1965.

[24] J. N. D. Gupta, "A Functional Heuristic Algorithm for The Flow Shop Scheduling Problem", *Journal of the Operational Research*, Vol. 22, No. 1, pp. 39-47, 1971.

[25] M. Nawaz, E. Enscore, and I. Ham, "A Heuristic Algorithm for the m-Machine n-Job Flowshop Sequencing Problem", *Omega*, Vol. 11, No. 1, pp. 91-95, 1983.

[26] K. Hussain, M. Salleh, M. Najib, S. Cheng, Y. Shi, and R. Naseem, "Artificial Bee Colony Algorithm: A Component-Wise Analysis Using Diversity Measurement", *Journal of King Saud University - Computer and Information Sciences*, Vol. 32, No. 7, pp. 794-808, 2020.

[27] E. Vallada, R. Ruiz, and G. Minella, "Minimising Total Tardiness in the m-Machine Flowshop Problem: A Review and Evaluation of Heuristics and Metaheuristics", *Computers and Operations Research*, Vol. 35, No. 4, pp. 1350-1373, 2008.

[28] G. Taguchi, S. Chowdhury, and Y. Wu, *Taguchi's Quality Engineering Handbook.*, John Wiley and Son's Inc., 2005.