


## Research Article

# Solutions to No-Wait Flow Shop Scheduling Problem Using the Flower Pollination Algorithm Based on the Hormone Modulation Mechanism

Chiwen Qu <sup>1</sup>, Yanming Fu,<sup>2</sup> Zhongjun Yi,<sup>3</sup> and Jun Tan<sup>1</sup>

<sup>1</sup>School of Information Engineering, Baise University, Baise 533000, China

<sup>2</sup>Computer and Electronic Information College, Guangxi University, Nanning 530004, China

<sup>3</sup>School of Politics and Public Affair Management, Baise University, Baise 533000, China

Correspondence should be addressed to Chiwen Qu; [quchiwen@163.com](mailto:quchiwen@163.com)

Received 9 March 2018; Revised 21 June 2018; Accepted 5 July 2018; Published 6 August 2018

Academic Editor: Irene Otero-Muras

Copyright © 2018 Chiwen Qu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

A flower pollination algorithm is proposed based on the hormone modulation mechanism (HMM-FPA) to solve the no-wait flow shop scheduling problem (NWFSP). This algorithm minimizes the maximum accomplished time. Random keys are encoded based on an ascending sequence of components to make the flower pollination algorithm (FPA) suitable for the no-wait flow shop scheduling problem. The hormone modulation factor is introduced to strengthen information sharing among the flowers and improve FPA cross-pollination to enhance the algorithm global search performance. A variable neighborhood search strategy based on dynamic self-adaptive variable work piece blocks is constructed to improve the local search quality. Three common benchmark instances are applied to test the proposed algorithm. The result verifies that this algorithm is effective.

## 1. Introduction

The flow shop scheduling problem is a simplified model of many manufacturing enterprise processes, belonging to a kind of important combinatorial optimization problem. The no-wait flow shop scheduling problem is a scheduling problem developed on the flow shop scheduling problem. In manufacturing, such as chemical industry manufacturing, smelting, food processing, and pharmaceutical manufacturing, the processing of work pieces cannot be interrupted from start to finish due to manufacturing craft limitations or storage space. Optimization scheduling of this kind of process can be concluded as a solution to the no-wait flow shop scheduling problem. In a more competitive market, optimal production planning and scheduling methods can reduce

enterprises' production costs and improve their competitiveness. However, when the number of work pieces is over 3, the no-wait flow shop scheduling problem has been proven to be an NP-hard problem [1]. This problem has great engineering research value and important theoretical significance. In the past few decades, many scholars have put forward several approaches to solve such problems, which can be divided into three categories: exact solution, heuristic algorithm, and meta-heuristic method. The exact solution method includes the dynamic programming method, branch and bound method [2], enumeration method, and cutting plane method. Due to the difficulty of the no-wait flow shop scheduling problem, the exact solution is only suitable for problems with relatively small scales. With improvement in the problem size, the time complexity of the

algorithm increases rapidly. Nagano and Miyata [3] presented a mechanism that describes in detail how to construct a heuristic algorithm to solve the no-wait flow shop scheduling problem. Laha and Chakraborty [4] developed a constructive heuristic algorithm based on the job insertion principle. Framinan et al. [5] tackled the no-wait flow shop problem with a constructive heuristic algorithm based on an analogy with the objective of minimizing makespan. In order to obtain good approximate solutions, a new constructive heuristic named QUARTS is proposed by Nagano et al. [6]. With the total flow time as the criterion, many heuristics were examined by Bonney and Gundry [7], King and Spachis [8], Nawaz et al. [9], Rajendran [10], Gangadharan and Rajendran [11], and Ronconi and Armentano [12]. Grabowski and Pempera [13] used different local search algorithms to solve the no-wait flow shop problem for minimizing makespan. A heuristic algorithm is able to acquire the solutions within short time, but the solutions are usually worse in quality.

The meta-heuristic algorithm based on swarm optimization algorithm can acquire the optimal solution or approximate optimal solution to the no-wait flow shop scheduling problem within feasible time and space complexity. Considering the no-wait flow shop scheduling problem with makespan minimization, Aldowaisan and Allahverdi [14] investigated six heuristics based on simulated annealing (SA) and genetic algorithms (GA). Pan et al., respectively, used the discrete particle swarm optimization algorithm [15], differential evolution algorithm [16], hybrid discrete particle swarm algorithm [17], artificial bee colony algorithm [18], and harmony search algorithm [19] to solve the no-wait flow shop scheduling problem. The goal was to minimize the maximum accomplishment time. For the same criterion, Fink and Voß [20] employed the simulated annealing algorithm and Tabu search algorithm, and Liu et al. [21] introduced a hybrid particle swarm optimization algorithm based on local search and self-adaptive learning mechanism. An ant colony optimization algorithm that improved the ant colony algorithm based on local search was applied separately to solutions for NWFSP [22, 23]. Nagano et al. [24, 25] designed an algorithm based on the hybrid meta-heuristic evolutionary clustering search (ECS\_NSL). By testing the standard instances and comparison with algorithms provided by other literature, ECS\_NSL performance is better. Qi et al. [26] presented a fast local neighborhood search algorithm (FLNS) with makespan criterion. The experimental results show that FLNS outperformed IHA, IBHLS, GA-VNS, and DHS in the solution quality and robustness. Deng et al. [27] proposed a cooperative evolutionary quantum genetic algorithm based on the competition mechanism to solve NWFSP. By maintaining population diversity and a competition mechanism to balance the development and exploit algorithm abilities, good solution results were obtained. Davendra et al. [28] introduced a new discrete self-organizing migrating algorithm. Through standard testing instances, it was shown in the statistical results that the algorithm is better than the heuristic algorithm in the literature.

FPA is a swarm intelligent bionic algorithm [29, 30] proposed by Yang et al. in 2012. The algorithm is highly regarded by scholars due to its simple structure, few parameter settings, relatively stronger global search performance, and easy implementation. Many scholars put forward an improvement method based on the FPA and popularized into industrial and agricultural production. Based on the basic FPA, Wang and Zhou [31] added a neighborhood search strategy and dimension-by-dimension greedy search method to improve cross-pollination. In comparison with other intelligent algorithms, the proposed algorithm is better. Abdel-Raouf and Abdel-Baset [32] proposed a hybrid FPA based on a combination particle swarm optimization algorithm and FPA to solve the constrained optimization problem. FPA is applied to solve the economic load dispatch problem [33–35]. Bekdaş et al. [36] used FPA to optimize the sizing truss structure. Abdelaziz et al. [37, 38] used FPA to obtain the optimal capacitor placement and sizing in distribution systems. They obtained the solution using combined economic and emission dispatch. In order to improve the accuracy and stability of cluster analysis, Wang et al. [39] proposed an FPA cluster analysis method with a bee pollinator. As shown in the experimental statistical results, the algorithm is better than DE, CS, ABC, PSO, FPA, and k-means algorithms in convergence, cluster accuracy, and stability performance. In order to solve the image segmentation problem via multilevel thresholding, Quadfel and Taleb-Ahmed [40] presented an algorithm that combined social spider optimization (SSO) and FPA. The results showed that the proposed algorithm outperforms the PSO and BAT algorithms. Flower pollen gamete cross-pollination relies on animals like bees and butterflies; however, this is ignored in information sharing among the flowers in the basic FPA. Thus, the algorithm is inclined to engage in local optimum.

Aiming at minimizing the NWFAP accomplishment time, the hormone modulation factor is introduced based on the FPA to realize information sharing and improve the global search performance. A variable neighborhood search strategy based on dynamic self-adaptive variable work piece blocks is constructed to improve the local search quality. The computational results based on benchmark instances show the effectiveness of the proposed algorithm.

## 2. No-Wait Flow Shop Scheduling Problem Model and Description

Based on the traditional flow shop scheduling problem, the no-wait flow shop scheduling problem is described as follows: Assuming that  $n$  work pieces need to be processed on  $m$  machines in the same sequences (without any preemption and interruption). There is a continuous process through  $m$  machines without interruption when a work piece is started on the first machine. Here, the processing time  $t_{i,j}$  of work piece  $i$  on machine  $j$  is provided.

At any time, one work piece can be processed on only one machine and one machine can process only one work piece. To satisfy the no-wait constraints, on a given

purpose of minimizing the maximum accomplishment time, NWFSP can be described by the mathematics model below:

$$\begin{aligned}
 C(\pi_1, k) &= \sum_{j=1}^k t(\pi_1, j), \quad k = 1, 2, \dots, m, \\
 C(\pi_i, k) &= C(\pi_{i-1}, 1) + \sum_{j=1}^k t(\pi_i, j) + \Delta(\pi_i), \quad i = 2, 3, \dots, n; \quad k = 1, 2, \dots, m, \\
 \Delta(\pi_i) &= \max \left\{ 0, \max \left\{ C(\pi_{i-1}, k) - C(\pi_{i-1}, 1) - \sum_{j=1}^{k-1} t(\pi_i, j) \right\} \right\}, \quad 2 \leq i \leq n, \quad 2 \leq k \leq m, \\
 \pi_* &= \arg \{C(\pi_n, m)\} \rightarrow \min,
 \end{aligned} \tag{1}$$

machine, the work piece completion time should be equal to the work piece start time on the next machine. For the where  $\pi = (\pi_1, \pi_2, \dots, \pi_n)$  is a scheduling sequence of  $n$  work pieces,  $t(\pi_i, k)$  is the processing time required by the job  $\pi_i$  on the machine  $k$ ,  $C(\pi_i, k)$  is the accomplishment time of the job  $\pi_i$  on the machine  $k$ , and  $\pi_*$  is the scheduling sequence aiming at the maximum accomplishment time minimum.

### 3. Flower Pollination Algorithm

FPA is a new swarm optimization algorithm based on flower pollination behavior. The basic thought is as follows: (i) Each flower is mapped as an individual in the population. (ii) Cross-pollination is conducted by the switch probability  $p \in [0, 1]$ . (iii) Self-pollination is conducted at the probability  $(1 - p)$ .

- (1) Cross-pollination. Cross-pollination refers to animals like bees and butterflies pollinate among different kinds of flowers using the *levy* flight mode. The flower update mode is shown as

$$x_i^{t+1} = x_i^t + \delta \cdot \text{levy}(\lambda) \cdot (x_{\text{best}}^t - x_i^t), \tag{2}$$

where  $\delta$  is the scaling factor,  $x_i^t$  is the flower individual  $x_i$  at generation  $t$ ,  $x_{\text{best}}^t$  is the current optimal individual found among all flower individuals at generation  $t$ , and  $\text{levy}(\lambda)$  is the random number subordinated to *Levy* distribution.

$$\text{levy}(\lambda) \sim \frac{\lambda \Gamma(\lambda) \sin(\pi \lambda / 2)}{\pi} \cdot \frac{1}{s^{1+\lambda}} \quad (s \gg s_0 > 0). \tag{3}$$

For calculation convenience, Yang et al. [30] used the method proposed by Mantegna to calculate the step size  $s$ . The calculation method is shown below:

$$\begin{aligned}
 s &= \frac{u}{|v|^{1/\lambda}}, \quad u \sim N(0, \sigma^2), \quad v \sim N(0, 1), \\
 \sigma &= \left( \frac{\Gamma(1 + \lambda) \times \sin(\pi \times \lambda / 2)}{\Gamma((1 + \lambda) / 2) \cdot \lambda \cdot 2^{(\lambda-1)/2}} \right)^{1/\lambda},
 \end{aligned} \tag{4}$$

where  $\lambda = 3/2$  and  $\Gamma(\lambda)$  is a standard gamma function.

- (2) Self-pollination. Self-pollination simulates close-distance pollination among the same species of flowers. The pollination method is shown below:

$$x_i^{t+1} = x_i^t + \varepsilon \cdot (x_j^t - x_k^t), \tag{5}$$

where  $x_i^t$  is the flower individual  $x_i$  at generation  $t$ ,  $\varepsilon$  is the random number subordinated to uniform distribution in  $[0, 1]$ , and  $x_j^t$  and  $x_k^t$  are the different flower individuals.

### 4. FPA Based on the Hormone Modulation Mechanism

4.1. *Coding of the Algorithm.* A job-permutation-based encoding scheme has been widely used in solving the no-wait flow shop scheduling problem. The job scheduling problem is discrete; it is impossible to use the standard FPA encoding scheme to directly express the sequence of work pieces. A random key coding principle based on the work piece ascending order is applied in this paper to realize the reflecting relationship between individual flowers and the

TABLE 1: Transformation from a flower individual  $x_i$  to sequence of work piece  $\pi_i$ .

Dimension $j$	1	2	3	4	5
$x_i^j$	3.56	2.23	-0.29	1.48	-2.97
$\theta_i^k$	-2.97	-0.29	1.48	2.23	3.56
$\pi_{i,\theta_i^k}$	5	3	4	2	1

scheduling sequence. Let the solution vector be  $x_i = (x_i^1, x_i^2, \dots, x_i^j, \dots, x_i^n)$ , where there are two meanings for each vector  $x_i^j$  including its own value  $x_i^j$  and the sequence number  $j$  representing the value of the new generating sequence. The process from a real vector of flower individual to the work pieces' sequence is as follows: Each dimension vector value of  $x_i$  is sequenced according to the ascending sequence principle; thus,  $\theta_i = (\theta_i^1, \theta_i^2, \dots, \theta_i^k, \dots, \theta_i^n)$ . The sequence of work piece  $\pi_{i,\theta_i^k}$  can be acquired through (6). A case on transformation from flower individual to sequence of work piece is provided in Table 1.

$$\pi_{i,\theta_i^k} = j. \quad (6)$$

**4.2. Population Initialization.** The quality of the initial population is significantly affected by the intelligent swarm optimization algorithm solution result. In order to ensure initial population diversity and improve the algorithm convergence velocity, the NEH heuristic algorithm is applied to acquire the first solution to the initial population. The remaining  $ns$  size - 1 ( $ns$  is the size of population) solutions to the initial population randomly emerge.

**4.3. The Cross-Pollination Operator Based on Hormone Modulation Mechanism.** In standard flower pollination algorithm cross-pollination, the update of flower individuals is determined only by the current best solution. However, the solution is greatly influenced by the other flowers around it. Meanwhile, the flowers can be responsive to the surrounding flowers and adjusts itself in real time. Thus, inspired by hormone modulation mechanism, FARHY [41] discovered a general changing principle of biological hormone secretion: monotonicity and nonnegativity. The ascending and declining principles of hormone secretion abide by the Hill characteristic function, which is shown in

$$f_{\text{up}}(g) = \frac{g^n}{t^n + g^n}, \quad (7)$$

$$f_{\text{down}}(g) = \frac{t^n}{t^n + g^n}, \quad (8)$$

where  $t(t > 0)$  and  $g$  refer to the independent variable and threshold of the function, respectively.  $n(n \geq 1)$  is the coefficient of the Hill function. Setting the hormone and regulatory hormone as  $x$  and  $y$ , respectively, the relationship between

the velocity  $v_x$  of hormone secretion  $x$  and the concentration  $c_y$  of regulatory hormone  $y$  is shown below:

$$v_x = \alpha \cdot f_{\text{up,down}}(c_y) + v_{x0}, \quad (9)$$

where  $v_{x0}$  and  $\alpha$  are the basic secretion velocity and common hormone coefficient, respectively.

Gu et al. [42] designed a hormone modulation mechanism and achieved great effect. In this paper, the hormone modulation mechanism is introduced into the FPA algorithm on the original cross-pollination to give the flower update method better global pollination performance. The flower update method is shown below:

$$x_i^{t+1} = x_i^t + \delta \cdot \text{levy}(\lambda) \cdot (x_{\text{best}}^t - x_i^t) + H_f \cdot \text{rand} \cdot n(0, \text{dim}), \quad (10)$$

where  $\text{rand} \cdot n(0, \text{dim})$  can satisfy the  $\text{dim}$  dimension random number in standard Gaussian distribution and  $H_f$  is the hormone regulatory function as shown in

$$H_f = \begin{cases} \arctan \left( \frac{(f_{\text{avg}} - f_i)^2}{(f_{\text{max}} - f_{\text{min}})^2 + (f_{\text{avg}} - f_i)^2} \right), & f_i < f_{\text{avg}} \\ \arctan \left( \frac{(f_{\text{max}} - f_{\text{min}})^2}{(f_{\text{max}} - f_{\text{min}})^2 + f_i^2} \right), & f_i \geq f_{\text{avg}} \end{cases} \quad (11)$$

where  $f_i$  is the flower fitness value  $x_i$ , and  $f_{\text{max}}$ ,  $f_{\text{min}}$ , and  $f_{\text{avg}}$  are the maximum fitness value, the minimum fitness value, and the average fitness value, respectively, of  $n_i$  flowers that are closest to the flower  $x_i$  in the current population. The solution method is shown in Algorithm 1.

According to the regulation in (11), the local location of flower  $x_i$  is better when  $f_i \geq f_{\text{avg}}$ . So fewer adjustment ranges should be conducted to the original flower. The local location of flower  $x_i$  is worse when  $f_i < f_{\text{avg}}$ . The endocrine systems make the flower  $x_i$  move to a better location by distributing more hormones.

**4.4. The Variable Neighborhood Search of Dynamic Self-Adaptive Variable Work Piece Blocks.** As a specific scheduling problem, the no-wait flow shop scheduling problem solutions are difficult and easily trapped into local optimum. The local search improvement can enhance the search performance for the swarm intelligent algorithm [43]. Hansen and Mladenović [44] proved that the probability of obtaining the optimal solution through systematic transformation of the neighborhood structure is higher than that from search results using a single neighborhood structure. A variable neighborhood search based on dynamic self-adaptive variable work piece blocks for local search is constructed in this paper. For a solution  $r = (r(1), r(2), \dots, r(n))$ , the local search process in this paper is described in Algorithm 2.

```

Step1: for  $k = 1:\text{size}$ 
Step2:    $\text{dist}(k) = \|x_k - x_i\|$    % $x_i$  is the current solution
Step3: end for
Step4: Set  $\text{dist}$  array in order according to the ascending sequence.
Step5: Select the minimum  $n_t$  flower locations, and record them as  $\text{index}_1, \text{index}_2, \dots, \text{index}_{n_t}$ .
Step6:  $f_{\max} = \max \{ \text{fit}(x_{\text{index}_j}), j \in [1, n_t] \}$ 
Step7:  $f_{\min} = \min \{ \text{fit}(x_{\text{index}_j}), j \in [1, n_t] \}$ 
Step8:  $f_{\text{avg}} = 1/n_t \cdot \sum_{j=1}^{n_t} \text{fit}(x_{\text{index}_j})$ 
Step9: if  $f_i < f_{\text{avg}}$ 
Step10:   $x'_i \leftarrow x_i + \delta \cdot \text{levy}(\lambda) \cdot (x_{\text{best}} - x_i) + ar \tan \left( \frac{(f_{\text{avg}} - f_i)^2 / (f_{\max} - f_{\min})^2 + (f_{\text{avg}} - f_i)^2}{(f_{\max} - f_{\min})^2 + f_i^2} \right) \cdot \text{rand } n(0, \text{dim})$ 
Step11: else
Step12:   $x'_i \leftarrow x_i + \delta \cdot \text{levy}(\lambda) \cdot (x_{\text{best}} - x_i) + ar \tan \left( \frac{(f_{\max} - f_{\min})^2 / (f_{\max} - f_{\min})^2 + f_i^2}{(f_{\max} - f_{\min})^2 + f_i^2} \right) \cdot \text{rand } n(0, \text{dim})$ 
Step13: end if.

```

ALGORITHM 1: The pseudo code of the cross-pollination operator based on hormone modulation mechanism.

```

Step1: Calculate the accomplishment time  $C_{\max}(r_{\text{best}})$  of the current optimal solution ( $r_{\text{best}}$ ).
Step2: Calculate the  $\text{step}$ .

```

$$\text{step} \leftarrow \left[ \text{max\_step} - \frac{\text{iter}}{\text{iter max}} \cdot \text{max\_step} \right],$$

where  $\text{iter}$  is the iteration,  $\text{iter max}$  is the largest iteration,  $\text{max\_step}$  is the maximum of work piece blocks, and  $\text{step}$  is the size of current work piece blocks.

```

Step3: Generate  $s \in (1, n)$  (random location), and delete consecutive  $\text{step}$  work pieces in  $r_{\text{best}}$  from location  $s$  to form a module. The deleted module is set to be  $r = (r(1), r(2), \dots, r(n))$ , and the remaining work pieces form the sequence  $r'(r'(1), r'(2), \dots, r'(n - \text{step}))$ .

```

```

Step4: for  $k = 1:\text{step}$ 

```

```

Step5: Successively insert  $r(k)$  into corresponding location in  $r'$ .

```

```

Step6: keep the result of the best location.

```

```

Step7: end.

```

```

Step8: Acquire a new solution  $r'$ . If  $C_{\max}(r_{\text{best}}) > C_{\max}(r')$ ,  $r_{\text{best}} = r'$ .

```

```

Step9: If  $\text{iter} < \text{iter max}$ ,  $\text{iter} = \text{iter} + 1$ , then turn to Step 3. Or else, stop the algorithm.

```

ALGORITHM 2: The pseudo code of variable neighborhood search of dynamic self-adaptive variable work piece blocks.

Compared with the traditional swap and insert neighborhood search methods, the variable neighborhood search strategy with dynamic self-adaptive variable work piece blocks can make the algorithm able to search in a broader search space. This increases the probability of acquiring better solutions. In early iterations, the algorithm has good global search ability due to the relatively large size of the work piece blocks. With increasing iterations, the size of the work piece blocks  $\text{step}$  is adjusted dynamically, which makes the algorithm possess better local search ability.

4.5. *Process of Solutions.* The detailed procedure for the proposed algorithm is shown in Algorithm 3.

## 5. Algorithm Simulation and Experimental Test

5.1. *Test Settings.* In order to testify the performance of the proposed algorithm for solving NWFSF, we conducted

experiments using three common benchmark problems: (i) 8 small-scale instances provided by Carlier [45]: Car1 to Car8 that consist of eight problem instances with small job and machine sizes but large processing time; (ii) 21 instances provided by Reeves [46]: Rec01 to Rec41 that have 21 different problems with 20~75 work pieces and 5~20 machines; and (iii) 120 large-scale instances provided by Taillard [47]: Ta001 to Ta120 including 120 problem instances with 12 subsets of different sizes, ranging from 20 work pieces and 5 machines to 500 work pieces and 20 machines. In order to avoid the influence of random factors, each test instance runs independently 20 times. The algorithm in this paper was tested using Matlab 2016a coding on the platform with Win 10, Intel Core i5-4210U 2.4 GHz, and 4 GB memory.

To compare the results obtained from the experiments, the relative deviation (RD) between solutions from relative algorithms and the best known results up to now were collected. BRD, ARD, and WRD, respectively, refer to the

Step 1.	Objective function $f(x_i) = C(x_i)$ , $x = (x_1, x_2, \dots, x_n)$ .
Step 2.	Initialize the parameters of $nsize$ , $p$ , $iter$ max et al.
Step 3.	Generate the first flower individual by NEH heuristic algorithm, and generate $(nsize - 1)$ flower individuals randomly to construct the initial population.
Step 4.	Evaluate its fitness $fit(x_i) = f(x_i)$ , and find the best solution $x_{best}$ .
Step 5.	while (stopping criterion is not satisfied).
Step 6.	for $i = 1$ to $nsize$
Step 7.	Generate a random number $r$ that obey the uniformly distribution.
Step 8.	if $r < p$ .
Step 9.	Perform the cross-pollination operator based on hormone modulation mechanism (described in Algorithm 1).
Step 10.	else
Step 11.	Perform the self-pollination operator.
	$x'_i \leftarrow x_i + \varepsilon \cdot (x_j - x_k) \quad (j \neq k)$
Step 12.	end if.
Step 13.	Calculate the fitness $fit(x'_i)$ of the new flower gamete $x'_i$ .
Step 14.	if $fit(x'_i) > fit(x_i)$ , update $x_i$ with $x'_i$ .
Step 15.	if $fit(x'_i) > fit(x_{best})$ , update $x_{best}$ with $x'_i$ .
Step 16.	end for
Step 17.	Perform the variable domain search of dynamic self-adaptive variable work piece blocks operator for the best flower individual $x_{best}$ (described in Algorithm 2).
Step 18.	end while

ALGORITHM 3: The pseudo code of HMM-FPA.

optimal relative deviation, average relative deviation, and the worst relative deviation. The RD, BRD, ARD and WRD are calculated as follows:

$$\begin{aligned}
RD_i &= \frac{C_i - C^*}{C^*} * 100\%, \\
BRD &= \frac{C_{best} - C^*}{C^*} * 100\%, \\
ARD &= \frac{1}{R} \sum_{i=1}^R \frac{C_i - C^*}{C^*} * 100\%, \\
WRD &= \frac{C_{worst} - C^*}{C^*} * 100\%,
\end{aligned} \tag{12}$$

where  $C_i$  is the solution generated by a specific algorithm,  $R$  is the run times,  $C^*$  is the best solution found,  $C_{best}$  is the best solution over  $R$  runs, and  $C_{worst}$  is the worst solution over  $R$  runs. Obviously, the smaller the ARD value, the better the algorithm's performance is. In addition, ACT [48, 49] and ARPT [48, 49] are also recorded to indicate the CPU time efforts.

$$\begin{aligned}
ACT_i &= \frac{1}{I} \sum_{h=1}^I T_{i,h}, \quad \forall i = 1, \dots, I, \\
ARPT_h &= \frac{1}{I} \sum_{i=1}^I \left( \frac{T_{i,h} - ACT_i}{ACT} + 1 \right), \quad \forall i = 1, \dots, I, \quad h = 1, \dots, H,
\end{aligned} \tag{13}$$

where  $T_{i,h}$  is the CPU time required by the algorithm  $h$  in instance  $i$ ,  $I$  refers to the number of instances, and  $H$  is the number of swarm intelligent algorithms.

This section includes four subsections. The first part discusses the influence of the work piece block size. Secondly, comparison of particle swarm optimization algorithm (PSO), cuckoo search algorithm (CS), flower pollination algorithm (FPA), and the proposed algorithm will be introduced. Third is the comparison of HMM-FPA with some other existing intelligent algorithms. Finally, for solving the large-scale problems, 120 Taillard instances are executed to identify the effectiveness of the proposed algorithm.

## 5.2. Analysis on Simulation Results

**5.2.1. Discussion on the Settings of the Size of Work Piece Blocks.** In order to discuss the influence of the work piece block size in the neighborhood search of the proposed algorithm, seven Rec instances (Rec01, Rec07, Rec13, Rec19, Rec25, Rec31, and Rec37) and eight Car instances (Car1~Car8) are used to test. For the Car instances, the size of work piece blocks  $step$  is, respectively, set as 1, 2, 3, 4, 6, and the dynamic self-adaptive variable work piece blocks (DSVWB). The  $step$  is, respectively, set as 1, 3, 5, 8, 10, and DSVWB for the seven Rec instances. Here, the values of  $C^*$  for Car instances and Rec instances are the optimal solution found so far.

It is clear from Tables 2 and 3 that the proposed algorithm with DSVWB is the winner, since the ARD and WRD obtained by the DSVWB are better than or equal to those obtained by other  $step$ . The larger the work piece block size is, the better the algorithm search performance is. However, when the size of the work piece blocks surpasses a certain value, the larger the size is, the worse the search performance of the algorithm is.

As for the causes, the neighborhood search in this paper can be essentially regarded as a disturbance operation that

TABLE 2.: Performance comparison of the tested work piece blocks with different size for Cars.

Problem	$n \times m$	$C^*$	step = 1			step = 2			step = 3			step = 4			step = 6			DACWB	
			BRD	WRD	ARD	BRD	WRD	ARD	BRD	WRD	ARD	BRD	WRD	ARD	BRD	WRD	ARD	BRD	WRD
Car1	11 × 5	8142	0.00	0.15	0.26	0.00	0.03	0.28	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Car2	13 × 4	8242	0.00	0.08	0.18	0.00	0.03	0.16	0.00	0.02	0.18	0.00	0.04	0.18	0.00	0.00	0.00	0.00	0.15
Car3	12 × 5	8866	0.00	0.03	0.07	0.00	0.03	0.09	0.00	0.03	0.08	0.00	0.01	0.07	0.00	0.00	0.00	0.00	0.08
Car4	14 × 4	9195	0.00	0.45	0.80	0.00	0.29	0.69	0.00	0.22	0.71	0.00	0.12	0.71	0.00	0.01	0.05	0.00	0.06
Car5	10 × 6	9159	0.00	0.40	1.87	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.45	2.05	0.00	0.00
Car6	8 × 9	9690	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Car7	7 × 7	7705	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Car8	8 × 8	9372	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.16	0.00	0.00
Average			0	0.139	0.398	0	0.048	0.153	0	0.034	0.121	0	0.0212	0.12	0	0.06	0.283	0	0.016

$C^*$  is the best solution found.

TABLE 3: Performance comparison of the tested work piece blocks with different size for Recs.

Problem	$n \times m$	$C^*$	step = 1			step = 3			step = 5			step = 8			step = 10			DACWB		
			BRD	ARD	WRD	BRD	ARD	WRD	BRD	ARD	WRD	BRD	ARD	WRD	BRD	ARD	WRD	BRD	ARD	
Rec01	$20 \times 5$	1526	0.00	0.15	0.41	0.00	0.04	0.25	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Rec07	$20 \times 10$	2042	0.00	1.07	2.27	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Rec13	$20 \times 15$	2545	0.00	0.45	1.98	0.00	0.18	0.33	0.00	0.05	0.25	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Rec19	$30 \times 10$	2850	0.00	0.95	1.85	0.00	0.45	1.20	0.00	0.58	1.14	0.00	0.30	1.17	0.00	0.23	0.50	0.00	0.00	0.00
Rec25	$30 \times 15$	3593	0.13	0.78	1.65	0.00	0.41	1.22	0.00	0.32	0.91	0.00	0.21	0.85	0.00	0.40	0.88	0.00	0.09	0.65
Rec31	$50 \times 10$	4311	1.41	2.71	3.29	0.15	1.07	2.11	0.14	0.75	1.41	0.32	0.49	1.04	0.31	0.52	1.03	0.13	0.20	0.49
Rec37	$75 \times 20$	8014	3.50	4.17	4.91	1.01	2.37	3.58	0.97	1.60	2.05	0.95	1.55	1.99	1.02	1.70	2.23	0.25	0.67	1.40
Average			0.720	1.469	2.337	0.166	0.646	1.241	0.159	0.471	0.823	0.181	0.364	0.721	0.190	0.407	0.663	0.0543	0.137	0.363

$C^*$  is the best solution found.



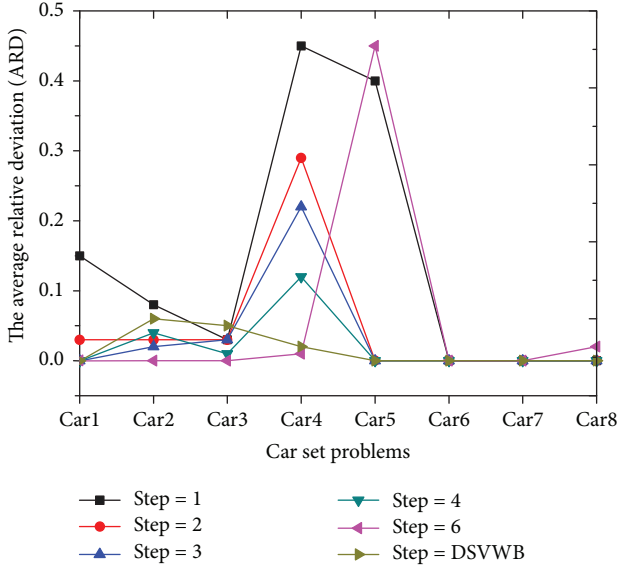


FIGURE 1: The ARD line chart for Car set problems with different step.

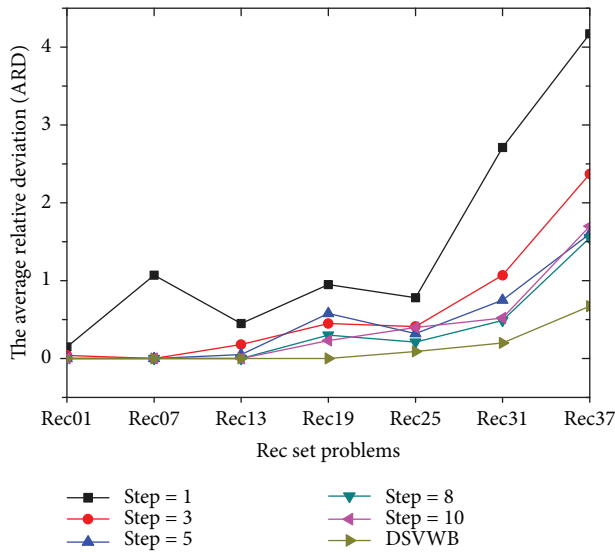


FIGURE 2: The ARD line chart for Rec set problems with different step.

can easily remove the current solution to other neighboring areas with partial characteristics of the current solution.

The size of disturbance step exerts greater effects on algorithm performance. An excessively long disturbance step can easily lead to loss of great characteristics in the current solution, similar to the random emergence process. If the disturbance step is excessively short and the movement range of the current solution is excessively small, it is easier to fall into the local optimum position.

Figures 1 and 2 are the line charts of the average relative error (ARE) for solving Car instances and Rec instances. It can be indicated from Figures 1 and 2 that ARE solved by DSVWB is equivalent to or better than the results solved by

other fixed steps. The size of oversized or undersize work piece blocks will decrease the local search ability of the neighborhood search. Apparently, taking the time complexity of the algorithm, it is reasonable to adopt the dynamic self-adaptive variable work piece block size, which not only balances the local and global search ability of the algorithm but also reduces the algorithm calculation time.

**5.2.2. Comparison of PSO, CS, FPA, and HMM-FPA.** Four algorithms were selected as contrast algorithms in the simulation experiment to evaluate the proposed algorithm's performance, that is, particle swarm optimization algorithm (PSO) [15], cuckoo search algorithm (CS) [50], and flower pollination algorithm (FPA) [29], respectively. Eight Car instances (Car1~Car8) were used in this test.

The same coding method is applied to the four algorithms as shown in Section 4.1. The algorithm parameter settings are shown below:

The flower pollination algorithm (FPA) has the size of the population  $nsize = 50$ , selection probability of pollination method  $p = 0.8$ , and the largest iterations  $iter\ max = 5000$ .

The particle swarm optimization algorithm (PSO) has the population size  $nsize = 50$ , the iterations  $iter\ max = 5000$ , linear inertia  $w_{max} = 0.9$  and  $w_{min} = 0.4$ , learning factor  $C1 = C2 = 1.4692$ , and search range of particles  $x_{max} = 4$  and  $x_{min} = -4$ .

The cuckoo search algorithm (CS) has the population size  $nsize = 50$ ,  $p_a = 0.25$ ,  $\lambda = 1.5$ , and the largest iterations  $iter\ max = 5000$ .

The flower pollination algorithm based on hormone modulation mechanism (HMM-FPA) has the population size  $nsize = 50$ , pollination method selection probability  $p = 0.8$ , the largest work piece blocks  $max\_step = 4$ , and the largest iterations  $iter\ max = 5000$ .

The statistical results of 20 independent runs for the four algorithms are listed in Table 4, including the best relative deviation (BRD), the average relative deviation (ARD), and the average CPU time ( $T_{avg}$ ) for finding the optimal solutions in the iterations. Here, the values of  $C^*$  for Car1~Car8 are the optimal solution found so far. In order to be able to perform a fair comparison among swarm intelligent algorithms, we use ARPT as a measure of the computational effort. Results in terms of average CPU time and the ARPT are shown in Table 4 (the last two rows represent the average CPU time and the ARPT, resp.).

It can be indicated from Table 4 that PSO can only acquire the optimal solution in four instances including Car 1, Car 6, Car 7, and Car 8. The optimal solutions to test instances of Car 2, Car 4, and Car 5 cannot be acquired by basic cuckoo algorithm. The basic FPA contributes nothing to test instances of Car 2, Car 3, and Car 4. The proposed algorithm can acquire all the optimal solutions to Car instances and is better in BRD and ARD than the other three swarm optimization algorithms.

The average CPU time and APRT of the HMM-FPA algorithm is less than that of the CS algorithm, but more than that of the PSO and the standard FPA algorithm. However, its solution accuracy is apparently better than that of the other three swarm intelligent algorithms. This demonstrates

TABLE 4: The comparative statistical results between FPA, PSO, CS, and HMM-FPA.

Problem	$n \times m$	$C^*$	PSO			CS			FPA			HMM-FPA		
			BRD	ARD	$T_{avg}$	BRD	ARD	$T_{avg}$	BRD	ARD	$T_{avg}$	BRD	ARD	$T_{avg}$
Car1	$11 \times 5$	8142	0.00	0.00	0.01	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.01
Car2	$13 \times 4$	8242	0.60	1.24	0.01	0.18	2.75	0.01	0.64	1.85	0.00	0.00	0.06	0.01
Car3	$12 \times 5$	8866	0.73	0.85	0.01	0.00	0.89	0.01	0.70	2.61	0.00	0.00	0.05	0.01
Car4	$14 \times 4$	9195	3.54	4.46	0.02	0.65	3.79	0.015	3.49	6.45	0.00	0.00	0.02	0.011
Car5	$10 \times 6$	9159	2.35	3.78	0.01	0.54	2.20	0.01	0.00	3.96	0.00	0.00	0.00	0.01
Car6	$8 \times 9$	9690	0.00	0.00	0.004	0.00	0.00	0.008	0.00	0.00	0.00	0.00	0.00	0.006
Car7	$7 \times 7$	7705	0.00	0.50	0.004	0.00	0.15	0.008	0.00	3.60	0.00	0.00	0.00	0.005
Car8	$8 \times 8$	9372	0.00	0.15	0.004	0.00	0.20	0.008	0.00	0.19	0.00	0.00	0.00	0.006
Average			0.90	1.37	0.01	0.17	1.25	0.01	0.60	2.33	0.01	0.00	0.02	0.01
APRT			0.9707	1.1952						0.8331			1.0011	

$C^*$  is the best solution.

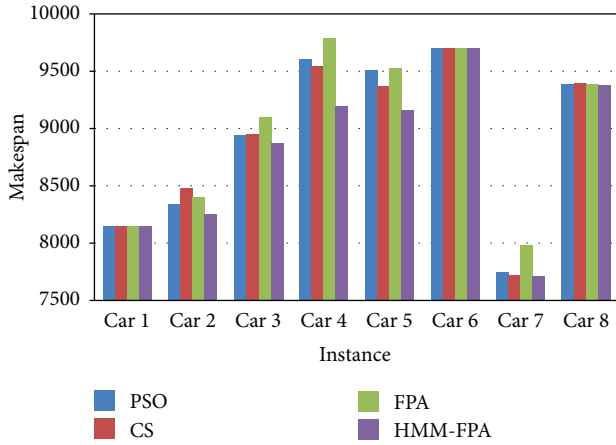


FIGURE 3: The makespan of the statistical results for the different algorithms.

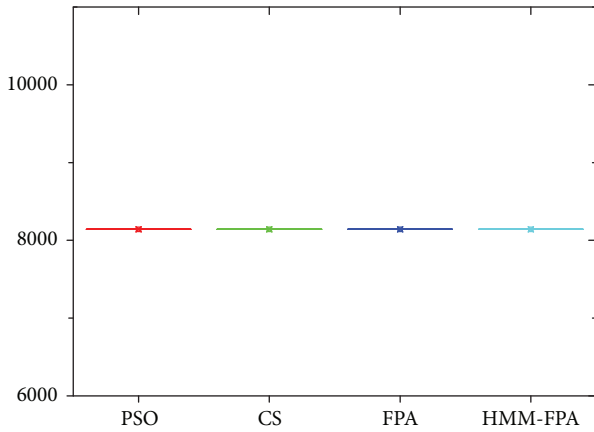


FIGURE 4: ANOVA tests of Car 1.

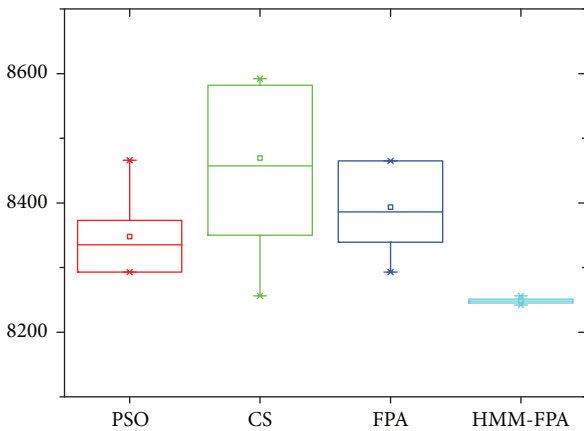


FIGURE 5: ANOVA tests of Car 2.

that the global searching ability of HMM-FPA is effective and HMM-FPA is suitable for solving NWFSP.

In the basic particle swarm optimization algorithm, the particle position update is determined by the particle at the

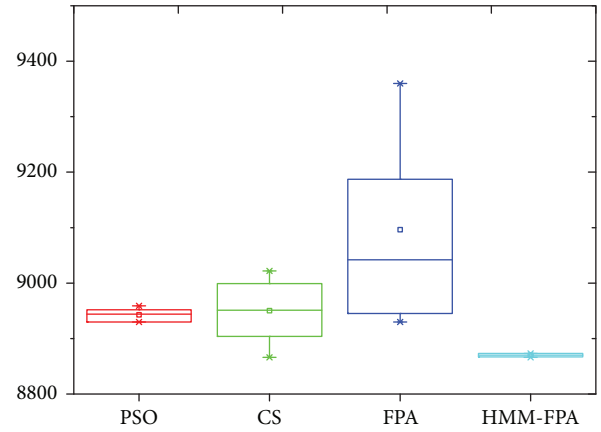


FIGURE 6: ANOVA tests of Car 3.

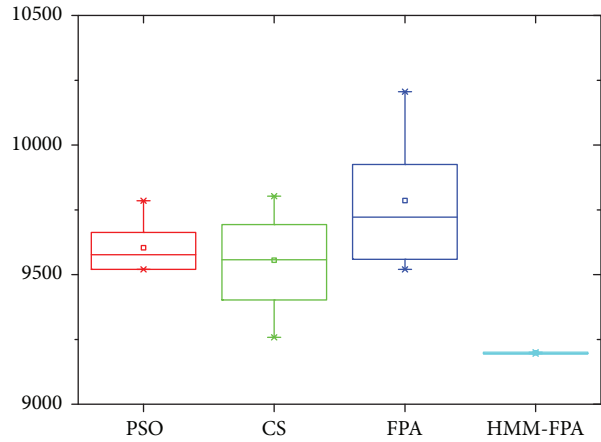


FIGURE 7: ANOVA tests of Car 4.

current optimal position and the historical optimum position that the particle has experienced, neglecting the influence of the particles at the common nonoptimal position (the particles may be near the global optimum position). The algorithm is easily trapped into the local optimum position. In selecting the host bird nest position, a kind of random search method with levy distribution is applied in the cuckoo algorithm. It has been proven that the cuckoo algorithm is better than the basic particle swarm optimization algorithm [50] in searching for the global optimum solution. Thus, in solutions to NWFSP, the cuckoo algorithm is better than the basic particle swarm optimization algorithm. However, similarly, the two algorithms ignore the influence of other neighboring particle positions on themselves.

In the HMM-FPA algorithm mentioned in this paper, when a single individual flower conducts cross-pollination with the help of animal vectors like bees and butterflies, the influence of neighboring flowers on itself and the irritability of the individual towards the flower's status at other positions are both considered. Furthermore, certain adjustment is appropriately conducted on its position. For the proposed algorithm, the flower positions are updated by (10), and the neighboring individual flower factors are sufficiently

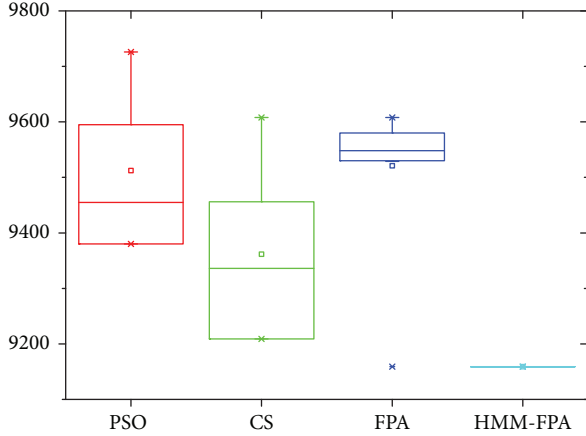


FIGURE 8: ANOVA tests of Car5.

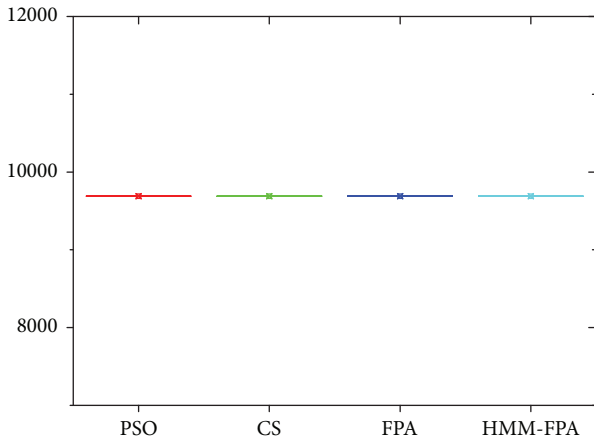


FIGURE 9: ANOVA tests of Car6.

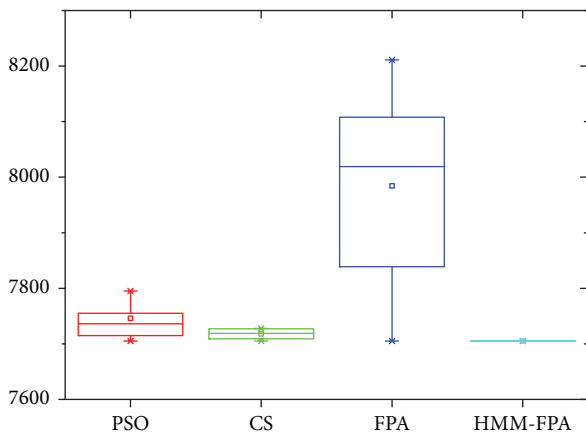


FIGURE 10: ANOVA tests of Car7.

considered. When  $f_i < f_{avg}$ , the flower performance of flower  $i$  is better (in a better position), leading to a smaller adjustment range in the flower position. Conversely, the performance of flower  $i$  is worse (in a worse position), leading to a larger adjustment range in the flower position.

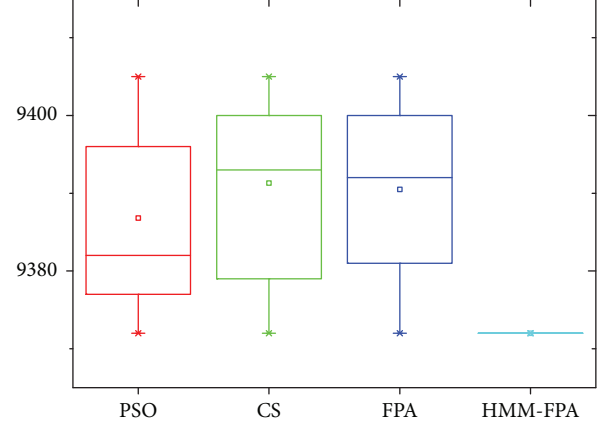


FIGURE 11: ANOVA tests of Car8.

The HMM-FPA sufficiently considers the flower information at the optimal position and neighboring flower positions, which combines the global expansion ability and local development ability, so the solution performance of the proposed algorithm is apparently better than that of other swarm intelligent algorithms.

Figure 3 is the statistical result from the makespan for PSO, CS, FPA, and HMM-FPA. It is shown that the makespan of the HMM-FPA is the least except Car1 and Car6 (the four algorithms can search the global optimal theoretical value).

Figures 4–11 are the statistical results from box diagrams for PSO, CS, FPA, and HMM-FPA. It can be seen from Figures 4–11 that the stability of HMM-FPA is better than the compared algorithm for solving Car instances, which illustrates the good robustness.

**5.2.3. Comparison of HMM-FPA and Existing Intelligent Algorithm.** In order to further verify the algorithm performance, the proposed algorithm is compared with other intelligent algorithms. Comparisons are carried out with six typical methods from the literatures, including the discrete particle swarm optimization algorithm (DPSO) [51], improved iterated greedy algorithm with a Tabu-based reconstruction strategy (TMIIG) [52], improved iterated greedy algorithm (IIGA) [53], DE-based approach (HDE) [54], effective hybrid particle swarm optimization (HPSO) [21], and GA [55]. Results of DPSO, TMIIG, IIGA, HDE, HPSO, and GA come from the corresponding literatures. The 21 Rec problems (Rec01~Rec41) are used as test instances. Comparison results are shown in Table 5. Table 5 shows that IMIIG only conducts statistics on the ARD perspective without statistics on BRD and  $T_{avg}$ . GA only conducts statistics on BRD.

For BRD, the accuracy of results acquired from HMM-PFA is generally better than that from other algorithms. The advantages of some instances are more obvious, such as rec 31, rec 33, and rec 41. The average BRD value acquired from HMM-PFA is, respectively, fewer than that of DPSO, IIGA, HPSO, HDE, and GA by 0.03, 0.03, 0.55, 0.04, and 4.54. It can be concluded that the BRD performance of the proposed algorithm is



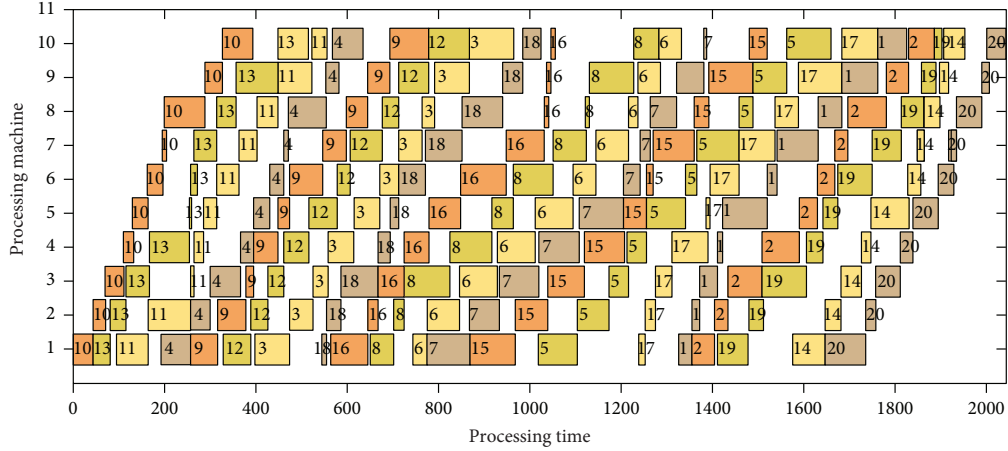


FIGURE 12: Gantt chart of the solution for Rec07.

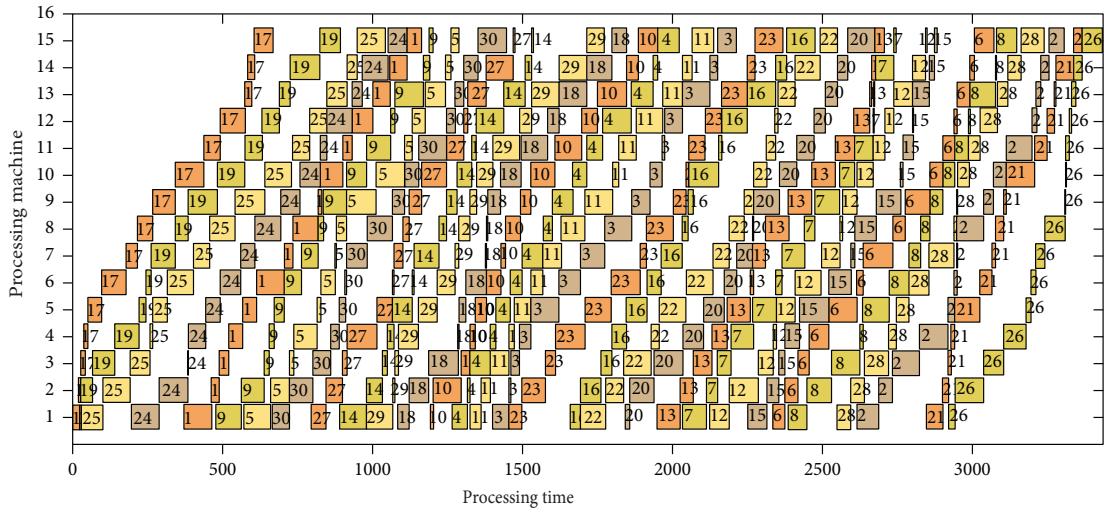


FIGURE 13: Gantt chart of the solution for Rec27.

TABLE 6: Comparison of results based on Taillard’s benchmark instances.

$n \times m$	ACO-SA			IIGA			HMM-PFA		
	BRD	ARD	$T_{avg}$	BRD	ARD	$T_{avg}$	BRD	ARD	$T_{avg}$
$20 \times 5$	0	0	0.28	—	0	—	0	0	0.0161
$20 \times 10$	0	0.320	0.44	—	0.01	—	0	0	0.021
$20 \times 20$	0	0.493	0.56	—	0.02	—	0	0	0.026
$50 \times 5$	0.346	0.852	1.79	—	1.34	—	0.22	0.37	0.888
$50 \times 10$	0.145	0.885	2.96	—	0.71	—	0.04	0.18	1.584
$50 \times 20$	0.095	1.770	3.18	—	0.58	—	0.05	0.37	2.202
$100 \times 5$	0.847	2.899	30.20	—	2.78	—	0.62	1.13	25.974
$100 \times 10$	0.607	1.781	57.23	—	1.7	—	0.47	0.97	48.559
$100 \times 20$	0.599	1.546	61.26	—	1.46	—	0.41	0.69	73.573
$200 \times 10$	2.009	2.695	189.17	—	2.49	—	1.26	1.49	162.453
$200 \times 20$	1.633	2.190	200.82	—	2.1	—	1.04	1.35	221.214
$500 \times 20$	—	—	—	—	2.94	—	2.10	2.31	883.166
Average	0.571	1.403	49.808	—	1.199 (1.344)*	—	0.374 (0.518)*	0.595 (0.738)*	48.774 (118.306)*

\* is the statistical results for the 120 instances in  $500 \times 20$  scale.

TABLE 7: The HMM-PFA statistical result to solve 120 Taillard's benchmark instances.

$n \times m$	Instance	Opt	BRD	ARD	$T_{avg}$	$n \times m$	Instance	Opt	BRD	ARD	$T_{avg}$
$20 \times 5$	Ta001	1486	0	0	0.016	$100 \times 5$	Ta061	6361	0.57	1.07	25.557
	Ta002	1528	0	0	0.017		Ta062	6212	0.90	1.13	26.021
	Ta003	1460	0	0	0.016		Ta063	6104	0.48	0.87	25.998
	Ta004	1588	0	0	0.016		Ta064	5999	0.93	1.50	26.59
	Ta005	1449	0	0	0.017		Ta065	6179	0.44	0.82	25.709
	Ta006	1481	0	0	0.016		Ta066	6056	0.18	0.76	26.054
	Ta007	1483	0	0	0.016		Ta067	6221	0.37	1.21	26.121
	Ta008	1482	0	0	0.016		Ta068	6109	1.34	1.99	25.993
	Ta009	1469	0	0	0.016		Ta069	6355	0.90	1.34	25.578
	Ta010	1377	0	0	0.015		Ta070	6365	0.06	0.61	26.121
$20 \times 10$	Ta011	2044	0	0	0.021	$100 \times 10$	Ta071	8055	0.62	0.81	47.664
	Ta012	2166	0	0	0.02		Ta072	7853	0.45	1.88	50.001
	Ta013	1940	0	0	0.02		Ta073	8016	0.41	0.95	49.654
	Ta014	1811	0	0	0.021		Ta074	8328	0.41	0.75	49.721
	Ta015	1933	0	0	0.022		Ta075	7936	0.38	0.93	48.332
	Ta016	1892	0	0	0.021		Ta076	7773	0.68	0.86	49.024
	Ta017	1963	0	0	0.021		Ta077	7846	0.43	0.75	47.238
	Ta018	2057	0	0	0.021		Ta078	7880	0.56	1.15	47.967
	Ta019	1973	0	0	0.022		Ta079	8131	0.30	0.85	48.054
	Ta020	2051	0	0	0.021		Ta080	8092	0.46	0.74	47.937
$20 \times 20$	Ta021	2973	0	0	0.025	$100 \times 20$	Ta081	10,675	0.09	0.21	75.345
	Ta022	2852	0	0	0.026		Ta082	10,562	0.94	1.32	72.49
	Ta023	3013	0	0	0.026		Ta083	10,587	0.73	0.89	70.376
	Ta024	3001	0	0	0.026		Ta084	10,588	0.30	0.63	73.298
	Ta025	3003	0	0	0.025		Ta085	10,506	0.10	0.42	71.439
	Ta026	2998	0	0	0.025		Ta086	10,623	0.57	0.76	76.296
	Ta027	3052	0	0	0.027		Ta087	10,793	0.29	0.53	74.387
	Ta028	2839	0	0	0.027		Ta088	10,801	0.45	0.91	72.363
	Ta029	3009	0	0	0.027		Ta089	10,703	0.14	0.53	75.832
	Ta030	2979	0	0	0.026		Ta090	10,747	0.47	0.73	73.901
$50 \times 5$	Ta031	3160	0.28	0.36	0.892	$200 \times 10$	Ta091	15,225	1.22	1.48	159.072
	Ta032	3432	0.26	0.53	0.902		Ta092	14,990	0.75	0.95	165.921
	Ta033	3210	0.09	0.27	0.912		Ta093	15,257	1.07	1.40	162.209
	Ta034	3338	0.30	0.50	0.89		Ta094	15,103	0.70	0.89	167.011
	Ta035	3356	0.60	0.72	0.89		Ta095	15,088	1.47	1.79	158.001
	Ta036	3346	0.00	0.04	0.914		Ta096	14,976	1.65	1.85	159.928
	Ta037	3231	0.09	0.29	0.911		Ta097	15,277	1.07	1.23	163.785
	Ta038	3235	0.12	0.17	0.799		Ta098	15,133	1.37	1.62	164.902
	Ta039	3070	0.33	0.52	0.85		Ta099	14,985	1.77	1.98	158.397
	Ta040	3317	0.15	0.26	0.922		Ta100	15,213	1.57	1.75	165.299
$50 \times 10$	Ta041	4274	0.00	0.04	1.568	$200 \times 20$	Ta101	19,531	0.65	0.77	220.632
	Ta042	4177	0.05	0.21	1.57		Ta102	19,942	1.44	1.79	217.997
	Ta043	4099	0.00	0.09	1.587		Ta103	19,759	1.16	1.31	218.318
	Ta044	4399	0.05	0.14	1.563		Ta104	19,759	1.22	1.66	225.638
	Ta045	4322	0.05	0.37	1.569		Ta105	19,697	0.90	1.05	237.689
	Ta046	4289	0.02	0.18	1.601		Ta106	19,826	0.94	1.31	209.396
	Ta047	4420	0.00	0.13	1.602		Ta107	19,946	0.87	1.63	217.332
	Ta048	4318	0.07	0.11	1.59		Ta108	19,872	1.37	1.62	219.744

TABLE 7: Continued.

$n \times m$	Instance	Opt	BRD	ARD	$T_{avg}$	$n \times m$	Instance	Opt	BRD	ARD	$T_{avg}$
50 × 20	Ta049	4155	0.00	0.15	1.584	500 × 20	Ta109	19,784	1.08	1.20	216.655
	Ta050	4283	0.16	0.37	1.601		Ta110	19,768	0.80	1.23	228.738
	Ta051	6129	0.00	0.11	2.195		Ta111	46,121	1.96	2.20	873.905
	Ta052	5725	0.00	0.18	2.221		Ta112	46,627	1.65	1.69	878.009
	Ta053	5862	0.00	0.15	2.199		Ta113	46,013	2.12	2.54	890.675
	Ta054	5788	0.24	0.42	2.198		Ta114	46,396	1.52	1.65	897.432
	Ta055	5886	0.00	1.77	2.205		Ta115	46,251	2.07	2.33	870.002
	Ta056	5863	0.12	0.20	2.208		Ta116	46,490	2.17	2.45	896.329
	Ta057	5962	0.12	0.25	2.197		Ta117	46,043	1.96	2.15	869.428
	Ta058	5926	0.02	0.20	2.199		Ta118	46,368	1.63	1.80	877.917
	Ta059	5876	0.00	0.22	2.195		Ta119	46,240	3.76	3.92	879.013
	Ta060	5957	0.03	0.22	2.199		Ta120	46,292	2.18	2.36	898.953

superior to the comparison algorithms. For the average ARD values, the quality of solutions acquired from HMM-PFA is better than that of other algorithms, which illustrates that HMM-PFA can effectively solve NWFSP with good robustness.

The statistical results on the last two lines in Table 5 show that the average CPU time and ARPT of HMM-PFA are better than that of HPSO, HDE, and TIIIG, but slightly inferior to that of DPSO and IIGA algorithms. Comprehensive analysis shows that the dynamic self-adaptive variable work piece block (DSVWB) neighborhood search increases the time complexity in the HMM-PFA search process. However, from the BRD and ARD average, it can be concluded that HMM-PFA is significantly better than the other 6 algorithms.

Figures 12 and 13 are the Gantt charts for the optimal results obtained using HMM-FPA for Rec07 and Rec27.

All in all, by applying the hormone modulation mechanism and the dynamic self-adaptive variable work piece block strategy, the HMM-PFA search quality can be enhanced, and the global search exploration and the local search exploitation can be well balanced. HMM-PFA is another effective method for solving no-wait flow shop scheduling problems with good quality.

*5.2.4. Comparison for the Large-Scale Instances.* Taillard's benchmark instances are used to test HMM-PFA algorithm performance for solving the large-scale no-wait flow shop scheduling problem. The data files for these instances are downloaded from the website <http://mistic.heig-vd.ch/taillard/problemes.dir/ordonnancement.dir/ordonnancement.html>. Taillard's benchmark instances include 120 problems. According to the scale of the instances, they are divided into 12 groups, with 10 in each group. The proposed algorithm is compared with the ACO-SA [56] algorithm and IIGA [53] algorithm. The ACO-SA algorithm only performed the test of the first 110 instances (Ta001–Ta110) and did not test the 10 instances in 500 × 20 scale (Ta111–Ta120). Therefore, the statistical results for the 120 instances in 500×20 scale are contained in Table 6, marked with “\*”. From the statistical results in Tables 7 and 6, we can find that without the statistics for the instances in 500×20 scale, the

BRD and ARD of the HMM-PFA algorithm are better than those of the ACO-SA algorithm. Besides, the HMM-PFA algorithm has less average CPU time and ARPT than the ACO-SA algorithm. Compared with the IIGA algorithm, we can find that the ARD of the HMM-PFA algorithm is better. The results from the above analysis indicate that the HMM-PFA algorithm is effective in solving the large-scale no-wait flow shop scheduling problem.

## 6. Conclusion

A flower pollination algorithm based on the hormone modulation mechanism is designed for NWFSP. The proposed method uses a variable neighborhood search strategy based on dynamic self-adaptive variable work piece in the local search. The application of improvement on cross-pollination operators in flower pollination algorithm by hormone modulation mechanism is introduced, which considers information sharing among the flowers. The benchmark instances for NWFSP are used to conduct test experiments on the targeted algorithm. Compared with other swarm optimization algorithms, the solution quality of the proposed algorithm in this paper is apparently better than the other algorithms, which sufficiently testify the algorithm effectiveness.

Future extensions will be conducted in the following directions. First, the proposed algorithm will be used to solve other complex flow shop scheduling problems (the limited buffer permutation flow shop scheduling problem, the blocking flow shop problem, and the no-idle flow shop scheduling problem). Secondly, we can combine the hormone modulation mechanism with other intelligent algorithms for solving job (or flexible) shop scheduling problems. Furthermore, we can also use the HMM-PFA to solve optimization problems based on sequencing (e.g., vehicle routing problem and traveling salesman problem).

## Data Availability

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



## Conflicts of Interest

The authors declare no conflict of interest regarding the publication of this paper.

## Acknowledgments

This work is financially supported by the Natural Science Foundation of Guangxi Province (Grant no. 2014GXNSFB A118283), the Ability Enhancement Project of Young Teachers in Guangxi Universities (Grant no. 2018KY0579), and the Philosophy and Social Science Planning Project of Guangxi Province (Grant no. 17FJY008).

## References

- [1] H. Röck, "The three-machine no-wait flow shop is NP-complete," *Journal of the ACM (JACM)*, vol. 31, no. 2, pp. 336–345, 1984.
- [2] S. Wang, M. Liu, and C. Chu, "A branch-and-bound algorithm for two-stage no-wait hybrid flow-shop scheduling," *International Journal of Production Research*, vol. 53, no. 4, pp. 1143–1167, 2015.
- [3] M. S. Nagano and H. H. Miyata, "Review and classification of constructive heuristics mechanisms for no-wait flow shop problem," *The International Journal of Advanced Manufacturing Technology*, vol. 86, no. 5-8, pp. 2161–2174, 2016.
- [4] D. Laha and U. K. Chakraborty, "A constructive heuristic for minimizing makespan in no-wait flow shop scheduling," *The International Journal of Advanced Manufacturing Technology*, vol. 41, no. 1-2, pp. 97–109, 2009.
- [5] J. M. Framinan, M. S. Nagano, and J. V. Moccasin, "An efficient heuristic for total flowtime minimization in no-wait flowshops," *The International Journal of Advanced Manufacturing Technology*, vol. 46, no. 9-12, pp. 1049–1057, 2010.
- [6] M. S. Nagano, H. H. Miyata, and D. C. Araújo, "A constructive heuristic for total flowtime minimization in a no-wait flowshop with sequence-dependent setup times," *Journal of Manufacturing Systems*, vol. 36, pp. 224–230, 2015.
- [7] M. C. Bonney and S. W. Gundry, "Solutions to the constrained flowshop sequencing problem," *Journal of the Operational Research Society*, vol. 27, no. 4, pp. 869–883, 1976.
- [8] J. R. King and A. S. Spachis, "Heuristics for flow-shop scheduling," *International Journal of Production Research*, vol. 18, no. 3, pp. 345–357, 1980.
- [9] M. Nawaz, E. E. Enscore, and I. Ham, "A heuristic algorithm for the  $m$ -machine,  $n$ -job flow-shop sequencing problem," *Omega*, vol. 11, no. 1, pp. 91–95, 1983.
- [10] C. Rajendran, "A no-wait flowshop scheduling heuristic to minimize makespan," *Journal of the Operational Research Society*, vol. 45, no. 4, pp. 472–478, 1994.
- [11] R. Gangadharan and C. Rajendran, "Heuristic algorithms for scheduling in the no-wait flowshop," *International Journal of Production Economics*, vol. 32, no. 3, pp. 285–290, 1993.
- [12] D. P. Ronconi and V. A. Armentano, "Lower bounding schemes for flowshops with blocking in-process," *Journal of the Operational Research Society*, vol. 52, no. 11, pp. 1289–1297, 2001.
- [13] J. Grabowski and J. Pempera, "Some local search algorithms for no-wait flow-shop problem with makespan criterion," *Computers & Operations Research*, vol. 32, no. 8, pp. 2197–2212, 2005.
- [14] T. Aldowaisan and A. Allahverdi, "New heuristics for  $m$ -machine no-wait flowshop to minimize total completion time," *Omega*, vol. 32, no. 5, pp. 345–352, 2004.
- [15] Q. K. Pan, M. Fatih Tasgetiren, and Y. C. Liang, "A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem," *Computers & Operations Research*, vol. 35, no. 9, pp. 2807–2839, 2008.
- [16] Q. K. Pan, L. Wang, and B. Qian, "A novel differential evolution algorithm for bi-criteria no-wait flow shop scheduling problems," *Computers & Operations Research*, vol. 36, no. 8, pp. 2498–2511, 2008.
- [17] Q. K. Pan and L. Wang, "No-idle permutation flow shop scheduling based on a hybrid discrete particle swarm optimization algorithm," *The International Journal of Advanced Manufacturing Technology*, vol. 39, no. 7-8, pp. 796–807, 2008.
- [18] Q. K. Pan, M. F. Tasgetiren, P. N. Suganthan, and T. J. Chua, "A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem," *Information Sciences*, vol. 181, no. 12, pp. 2455–2468, 2011.
- [19] K. Gao, Q. Pan, and J. Li, "Discrete harmony search algorithm for the no-wait flow shop scheduling problem with total flow time criterion," *The International Journal of Advanced Manufacturing Technology*, vol. 56, no. 5-8, pp. 683–692, 2011.
- [20] A. Fink and S. Voß, "Solving the continuous flow-shop scheduling problem by metaheuristics," *European Journal of Operational Research*, vol. 151, no. 2, pp. 400–414, 2003.
- [21] B. Liu, L. Wang, and Y. H. Jin, "An effective hybrid particle swarm optimization for no-wait flow shop scheduling," *The International Journal of Advanced Manufacturing Technology*, vol. 31, no. 9-10, pp. 1001–1011, 2007.
- [22] S. J. Shyu, B. M. T. Lin, and P. Y. Yin, "Application of ant colony optimization for no-wait flowshop scheduling problem to minimize the total completion time," *Computers & Industrial Engineering*, vol. 47, no. 2-3, pp. 181–193, 2004.
- [23] O. A. W. Riyanto and B. Santosa, "ACO-LS algorithm for solving no-wait flow shop scheduling problem," in *Intelligence in the Era of Big Data. ICSIT 2015*, R. Intan, C. H. Chi, H. Palit, and L. Santoso, Eds., vol. 516 of Communications in Computer and Information Science, pp. 89–97, Springer, Berlin, Heidelberg, 2015.
- [24] M. S. Nagano, A. A. Da Silva, and L. A. N. Lorena, "An evolutionary clustering search for the no-wait flow shop problem with sequence dependent setup times," *Expert Systems with Applications*, vol. 41, no. 8, pp. 3628–3633, 2014.
- [25] M. S. Nagano, A. A. Da Silva, and L. A. N. Lorena, "A new evolutionary clustering search for a no-wait flow shop problem with set-up times," *Engineering Applications of Artificial Intelligence*, vol. 25, no. 6, pp. 1114–1120, 2012.
- [26] X. Qi, H. Wang, H. Zhu, J. Zhang, F. Chen, and J. Yang, "Fast local neighborhood search algorithm for the no-wait flow shop scheduling with total flow time minimization," *International Journal of Production Research*, vol. 54, no. 16, pp. 4957–4972, 2016.
- [27] G. Deng, M. Wei, Q. Su, and M. Zhao, "An effective co-evolutionary quantum genetic algorithm for the no-wait flow shop scheduling problem," *Advances in Mechanical Engineering*, vol. 7, no. 12, 2015.
- [28] D. Davendra, I. Zelinka, M. Bialic-Davendra, R. Senkerik, and R. Jasek, "Discrete self-organising migrating algorithm for

- flow-shop scheduling with no-wait makespan," *Mathematical and Computer Modelling*, vol. 57, no. 1-2, pp. 100–110, 2013.
- [29] X. S. Yang, "Flower pollination algorithm for global optimization," in *Unconventional Computation and Natural Computation. UCNC 2012*, J. Durand-Lose and N. Jonoska, Eds., vol. 7445 of Lecture Notes in Computer Science, pp. 240–249, 2012.
- [30] X. S. Yang, M. Karamanoglu, and X. He, "Flower pollination algorithm: a novel approach for multiobjective optimization," *Engineering Optimization*, vol. 46, no. 9, pp. 1222–1237, 2014.
- [31] R. Wang and Y. Zhou, "Flower pollination algorithm with dimension by dimension improvement," *Mathematical Problems in Engineering*, vol. 2014, Article ID 481791, 9 pages, 2014.
- [32] O. Abdel-Raouf and M. Abdel-Baset, "A new hybrid flower pollination algorithm for solving constrained global optimization problems," *International Journal of Applied Operational Research*, vol. 4, no. 2, pp. 1–13, 2014.
- [33] R. Prathiba, M. B. Moses, and S. Sakthivel, "Flower pollination algorithm applied for different economic load dispatch problems," *International Journal of Engineering and Technology*, vol. 6, no. 2, pp. 1009–1016, 2014.
- [34] H. M. Dubey, M. Pandit, and B. K. Panigrahi, "Hybrid flower pollination algorithm with time-varying fuzzy selection mechanism for wind integrated multi-objective dynamic economic dispatch," *Renewable Energy*, vol. 83, pp. 188–202, 2015.
- [35] H. M. Dubey, M. Pandit, and B. K. Panigrahi, "A biologically inspired modified flower pollination algorithm for solving economic dispatch problems in modern power systems," *Cognitive Computation*, vol. 7, no. 5, pp. 594–608, 2015.
- [36] G. Bekdaş, S. M. Nigdeli, and X. S. Yang, "Sizing optimization of truss structures using flower pollination algorithm," *Applied Soft Computing*, vol. 37, pp. 322–331, 2015.
- [37] A. Y. Abdelaziz, E. S. Ali, and S. M. Abd Elazim, "Flower pollination algorithm for optimal capacitor placement and sizing in distribution systems," *Electric Power Components and Systems*, vol. 44, no. 5, pp. 544–555, 2016.
- [38] A. Y. Abdelaziz, E. S. Ali, and S. M. A. Elazim, "Combined economic and emission dispatch solution using flower pollination algorithm," *International Journal of Electrical Power & Energy Systems*, vol. 80, pp. 264–274, 2016.
- [39] R. Wang, Y. Zhou, S. Qiao, and K. Huang, "Flower pollination algorithm with bee pollinator for cluster analysis," *Information Processing Letters*, vol. 116, no. 1, pp. 1–14, 2016.
- [40] S. Ouadfel and A. Taleb-Ahmed, "Social spiders optimization and flower pollination algorithm for multilevel image thresholding: a performance study," *Expert Systems with Applications*, vol. 55, pp. 566–584, 2016.
- [41] L. S. FARHY, "Modeling of oscillations in endocrine networks with feedback," *Methods in Enzymology*, vol. 384, pp. 54–81, 2004.
- [42] G. Wenbin, T. Dunbing, Z. Kun, B. Shuaifu, and P. Wenxiang, "Research on permutation flow-shop scheduling problem based on improved adaptive particle swarm optimization algorithm with hormone modulation mechanism," *Journal of mechanical engineering*, vol. 48, no. 14, pp. 177–182, 2012.
- [43] H. Ishibuchi, T. Yoshida, and T. Murata, "Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 204–223, 2003.
- [44] P. Hansen and N. Mladenović, "Variable neighborhood search: principles and applications," *European Journal of Operational Research*, vol. 130, no. 3, pp. 449–467, 2001.
- [45] J. Carlier, "Ordonnancements à contraintes disjonctives," *RAIRO - Operations Research*, vol. 12, no. 4, pp. 333–350, 1978.
- [46] C. R. Reeves, "A genetic algorithm for flowshop sequencing," *Computers & Operations Research*, vol. 22, no. 1, pp. 5–13, 1995.
- [47] E. Taillard, "Benchmarks for basic scheduling problems," *European Journal of Operational Research*, vol. 64, no. 2, pp. 278–285, 1993.
- [48] V. Fernandez-Viagas and J. M. Framinan, "A beam-search-based constructive heuristic for the PFSP to minimise total flowtime," *Computers and Operations Research*, vol. 81, pp. 167–177, 2017.
- [49] E. Vallada, R. Ruiz, and J. M. Framinan, "New hard benchmark for flowshop scheduling problems minimising makespan," *European Journal of Operational Research*, vol. 240, no. 3, pp. 666–677, 2015.
- [50] X. S. Yang and S. Deb, "Engineering optimisation by cuckoo search," *International Journal of Mathematical Modelling and Numerical Optimisation*, vol. 1, no. 4, pp. 330–343, 2010.
- [51] Q.-K. Pan, M. F. Tasgetiren, and Y.-C. Liang, "A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem with makespan criterion," in *Proceedings of the 24th Annual Workshop of the UK Planning and Scheduling Special Interest Group*, pp. 31–41, London, UK, 2005.
- [52] J. Y. Ding, S. Song, J. N. D. Gupta, R. Zhang, R. Chiong, and C. Wu, "An improved iterated greedy algorithm with a Tabu-based reconstruction strategy for the no-wait flowshop scheduling problem," *Applied Soft Computing*, vol. 30, pp. 604–613, 2015.
- [53] Q. K. Pan, L. Wang, and B. H. Zhao, "An improved iterated greedy algorithm for the no-wait flow shop scheduling problem with makespan criterion," *The International Journal of Advanced Manufacturing Technology*, vol. 38, no. 7-8, pp. 778–786, 2008.
- [54] B. Qian, L. Wang, R. Hu, D. X. Huang, and X. Wang, "A DE-based approach to no-wait flow-shop scheduling," *Computers & Industrial Engineering*, vol. 57, no. 3, pp. 787–805, 2009.
- [55] I. A. Chaudhry and A. M. Khan, "Minimizing makespan for a no-wait flowshop using genetic algorithm," *Sadhana*, vol. 37, no. 6, pp. 695–707, 2012.
- [56] V. Riahi and M. Kazemi, "A new hybrid ant colony algorithm for scheduling of no-wait flowshop," *Operational Research*, vol. 18, no. 1, pp. 55–74, 2018.




**Hindawi**

Submit your manuscripts at  
[www.hindawi.com](http://www.hindawi.com)

