symmetry

MDPI

Article

A Novel Multi-Population Artificial Bee Colony Algorithm for Energy-Efficient Hybrid Flow Shop Scheduling Problem

Yandi Zuo [1], Zhun Fan [2,*], Tierui Zou [3] and Pan Wang [1,*]

[1] School of Automation, Wuhan University of Technology, Wuhan 430062, China; yandizuo@whut.edu.cn
[2] Department of Electronic and Information Engineering, Shantou University, Shantou 515063, China
[3] Department of Electrical & Computer Engineering, University of Florida, Gainesville, FL 32611, USA; tieruizou@ufl.edu
* Correspondence: zfan@stu.edu.cn (Z.F.); wangpan@whut.edu.cn (P.W.)

Abstract: Considering green scheduling and sustainable manufacturing, the energy-efficient hybrid flow shop scheduling problem (EHFSP) with a variable speed constraint is investigated, and a novel multi-population artificial bee colony algorithm (MPABC) is developed to minimize makespan, total tardiness and total energy consumption (TEC), simultaneously. It is necessary for manufacturers to fully understand the notion of symmetry in balancing economic and environmental indicators. To improve the search efficiency, the population was randomly categorized into a number of subpopulations, then several groups were constructed based on the quality of subpopulations. A different search strategy was executed in each group to maintain the population diversity. The historical optimization data were also used to enhance the quality of solutions. Finally, extensive experiments were conducted. The results demonstrate that MPABC can achieve an outstanding performance on three metrics DI_R, c and nd for the considered EHFSP.

Keywords: energy-efficient; hybrid flow shop scheduling; artificial bee colony; multi-population

check for updates

Citation: Zuo, Y.; Fan, Z.; Zou, T.; Wang, P. A Novel Multi-Population Artificial Bee Colony Algorithm for Energy-Efficient Hybrid Flow Shop Scheduling Problem. *Symmetry* **2021**, *13*, 2421. https://doi.org/10.3390/sym13122421

Academic Editors: Ming Li and Deming Lei

Received: 17 November 2021
Accepted: 6 December 2021
Published: 14 December 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Shop scheduling is an essential subject and an effective way to improve resource utilization [1]. With the increase of energy demand and the emergence of various environmental issues, reducing energy consumption has become particularly urgent for the manufacturing industry. Many scholars devote themselves to the research of energy-efficient scheduling problems, related to single machine [2], unrelated parallel machines [3], job shop [4], permutation flow shop [5] and hybrid flow shop [6].

In the typical flow shop scheduling problem, sequencing jobs are processed in predetermined orders, and each stage contains only one machine, which has been proved to be NP-hard [7]. Regarding the energy-efficient flow shop scheduling problem, Ding et al. [8] investigated a carbon-efficient scheduling of flow shops. Foumani et al. [9] considered the impact of various carbon reduction policies. Mansouri et al. [10] addressed a two-machine green flow shop scheduling. Jiang et al. [11] studied an energy-efficient permutation flow shop scheduling problem.

As an extension of flow shop, hybrid flow shop employs multiple parallel machines to enlarge the capacity of stage and eliminate the restriction of the bottleneck stage [12]. The hybrid flow shop scheduling problem (HFSP) extensively exists in various real industrial scenarios, such as steel [13], textile [14], glass [15], paper [16] and electronics [17].

EHFSP often consists of green constraints, green objectives and three sub-problems including job permutation, machine assignment and speed selection, which is more complicated than energy-efficient flow shop scheduling problems and apparently NP-hard. In recent years, EHFSP has attracted much attention from researchers and manufacturers. Bruzzone et al. [18] utilized a heuristic method to obtain an energy-aware scheduling in a sustainable manufacturing process. Dai et al. [19] designed a novel genetic-simulated

annealing algorithm to compromise between makespan and TEC, where makespan is the total time length of the schedule, and TEC is the sum of energy consumption in the production process. Meng et al. [20] developed an improved genetic algorithm to solve the energy-related problem with unrelated parallel machines. Gong et al. [21] incorporated a hybrid evolutionary algorithm into an EHFSP reckoning in the worker flexibility. Wu et al. [22] applied a hybrid non-dominated sorting genetic algorithm (NSGA-II) with variable local search to EHFSP considering renewable energy. Afterwards, the optimization algorithms based on hybrid NSGA-II were also raised for EHFSP in Zeng et al. [23] and Liu et al. [24]. To investigate EHFSP in a real-life case, a multi-level optimization method [25] and problem-tailored constructive heuristic method [26] were proposed. Work conducted by Zhou and Liu [27] investigated a multi-objective algorithm based on the Nawaz–Enscore–Ham heuristic for EHFSP with fuzzy processing time. Schulz et al. [28] designed a novel iterated local search algorithm to consider peak load, total energy costs and makespan in energy-aware scheduling.

The previous work also considered EHFSP through meta-heuristic optimization algorithms. Luo et al. [29] developed an improved ant colony algorithm for HFSP considering the time-of-use electricity prices. A teaching−learning-based optimization algorithm was provided by Lin et al. [30] to minimize carbon footprint and makespan. Zhou et al. [31] utilized an imperialist competitive algorithm for minimizing makespan and TEC simultaneously. Afterwards, Zhang et al. [32] suggested a three-stage optimization framework based on decomposition to minimize makespan and TEC in EHFSP. Li et al. [33] constructed another version of MOEA/D in the lot-streaming hybrid flow shop scheduling problem. Besides, other meta-heuristic optimization algorithms were also applied in EHFSP, including particle swarm optimization algorithm [34], shuffled frog-leaping algorithm [35] and artificial fish swarm algorithm [36].

In the literature above, speed constraint in EHFSP is not considered. At present, machines are able to operate at variable processing speeds. In general, a higher speed level leads to less processing time and more energy consumption. The speed selection subproblem in EHFSP is used to choose a proper speed for the selected processing machine, which can establish a balance between productivity and energy consumption. Meanwhile, there is an available speed set for the machines of each stage, and the machine speed needs to be regarded as an independent decision variable. Three sub-problems in EHFSP should be optimized simultaneously. Therefore, it is necessary to investigate the speed constraint in EHFSP.

In recent years, only several works have been studied on EHFSP with a variable speed constraint. Liu et al. [37] introduced the speed selection into a special two-stage EHFSP in the iron and steel industry. Tang et al. [38] studied an improved particle swarm optimization method to minimize makespan and TEC. Zhang et al. [39] suggested a discrete ABC based on decomposition to address the green scheduling problem with sequence dependent setup operations. A multi-objective genetic algorithm was developed for EHFSP with lot streaming in Chen et al. [40]. Lei et al. [41] proposed a teaching-learning-based optimization algorithm for the bi-objective EHFSP. Li et al. [42] presented an imperialist competitive algorithm with the relative importance of objectives. Afterwards, Oztop et al. [43] suggested an ensemble of metaheuristics to optimize makespan and energy consumption simultaneously. In conclusion, research on EHFSP with a variable speed constraint is significant in theory and application level.

Artificial bee colony (ABC) is a promising metaheuristic algorithm motivated by the collective behavior of the bee colony [44]. Since ABC was proposed, it was mainly used to solve unconstraint and constraint continuous function optimization problems [45,46]. Compared with other metaheuristic algorithms, there are some remarkable characteristics in ABC such as its effective global exploration capability, excellent local exploitation ability and fast convergence speed. Due to its simple structure and convenient implementation, it has also been applied to solve combinatorial optimization problems, especially in the field of production scheduling problems [47–54]. Its successful applications on scheduling prove

that ABC possesses an excellent search ability and convergence rate. In addition, ABC is seldom adopted in EHFSP. These factors motivated our idea to use ABC to solve EHFSP. However, ABC cannot be directly used to solve EHFSP because it is a typical discrete optimization problem and EHFSP is completely different from other scheduling problems. The algorithm needs to be modified according to the features of EHFSP. Therefore, we are committed to make improvements on ABC for solving EHFSP in this paper.

The contributions of this study can be summarized as follows: (1) EHFSP with a variable speed constraint is addressed to optimize makespan, total tardiness and TEC simultaneously, where total tardiness is a vital index for manufactures and equals the sum of tardiness of all jobs. (2) A novel multi-population artificial bee colony algorithm is proposed, in which a completely different multi-population division method is adopted. After randomly dividing the population into a few subpopulations, several groups are constructed. Each group implements different search approaches. (3) The quality of subpopulation is newly defined and mainly applied in group division. (4) The historical optimization data were stored in a memory set and fully utilized to update the weakest group.

The remainder of this paper is organized as follows. The considered problem is described in Section 2. The introduction of the original ABC is presented in Section 3, followed by the description of MPABC for EHFSP in Section 4. Extensive numerical experiments are revealed in Section 5. Conclusions and future work are provided in Section 6.

2. Problem Description

The notations used in this paper are shown in Table 1.

Table 1. Notations and descriptions.

Notation	Description
n	The number of jobs
m	The number of stages
s	The number of available unrelated parallel machines for S_j
d	The number of available processing speeds for M_{jk}
M_{jk}	The kth machine of stage j
η_{ijk}	The basic processing requirement of job J_i on machine M_{jk}
t_{ijkl}	The processing time of job J_i on machine M_{jk} at speed v_l
$z_{jk}(t)$	$z_{jk}(t) = 1$, if the machine M_{jk} is in stand-by mode at time t; otherwise, $z_{jk}(t) = 0$
$y_{jkl}(t)$	$y_{jkl}(t) = 1$, if the stage j is processed on kth machine with speed v_l at time t; otherwise, $y_{jkl}(t) = 0$
y_{ijkl}	$y_{ijkl} = 1$, if the jth stage in job i is processed on kth machine with speed v_l otherwise, $y_{ijkl} = 0$
$x_{ii'jk}$	$x_{ii'jk} = 1$, if the job i is prior to i' processed on machine k at stage j; otherwise, $x_{ii'jk} = 0$
a_{ij}	The starting processing time of job i at stage j
b_{ij}	The finishing processing time of job i at stage j
E_{jkl}	The energy consumption per unit time when machine M_{jk} runs at speed v_l
SE_{jk}	The energy consumption per unit time when machine M_{jk} runs in stand-by mode
C_i	The completion time of job J_i
D_i	The due date of job J_i
c_{max}	The maximum completion time of all jobs

In EHFSP, n jobs $J = \{J_i\}_{1 \le i \le n}$ is processed through m stages $O = \{1, 2, \ldots, j, \ldots, m\}$, consecutively. At stage j, there is a set of unrelated parallel machines, $S_j = \left\{M_{jk}\right\}_{1 \le k \le s}$.

For M_{jk}, variable processing speeds $V = \{v_l\}_{1 \le l \le d}$ are available. Moreover, if a job is processed at one stage, an assigned machine and a selected speed must be arranged for its processing. The arranged machine and selected speed cannot be changed during the whole processing. When job J_i is processed on machine M_{jk} at speed v_l, the processing time t_{ijkl} is denoted as η_{ijk}/v_l.

Several assumptions are presented as follows:

- Buffer size between stages is unlimited;
- Each machine can only process one operation;
- Some stages can be skipped, but each job must be processed at one stage at least;
- Once the process starts, interruption and preemption are not allowed.

In EHFSP, economic benefit and environmental impact are symmetrical and equally important. In this paper, the aim is to arrange jobs to the corresponding machines of each stage, decide the job sequences on each machine and choose the suitable speed for machines of each stage in order to minimize the following three objectives simultaneously when all constraints are met.

$$\min f_1 = C_{max} = \max_{i \in J} C_i \tag{1}$$

$$\min f_2 = TT = \sum_{i=1}^{n} max\{C_i - D_i, 0\} \tag{2}$$

$$\min f_3 = TEC = \sum_{j=1}^{m} \sum_{k=1}^{|s_j|} \int_0^{C_{max}} \left(\sum_{l=1}^{d} E_{jkl} y_{jkl}(t) + SE_{jk} z_{jk}(t) \right) dt \tag{3}$$

Constraints:

$$\sum_{k=1}^{s} \sum_{l=1}^{d} y_{ijkl} = 1, \forall i \in J, j \in O \tag{4}$$

$$b_{ij} - a_{ij} = \sum_{k=1}^{s} \sum_{l=1}^{d} t_{\mathrm{ijkl}} \cdot y_{ijkl}, \forall i \in J, j \in O \tag{5}$$

$$a_{ij} - b_{i,j-1} \ge 0, \forall i \in I, j \in \{2, \ldots, m\} \tag{6}$$

$$x_{ii'jk} + x_{i'ijk} \le 1, \forall i, i' \in J, j \in O, k \in S_j \tag{7}$$

$$y_{ijkl}, x_{ii'jk}, y_{jkl(t)}, z_{jk(t)} \in \{0, 1\}, \forall i, i' \in I, j \in O, k \in S_j, l \in V \tag{8}$$

$$a_{ij} \ge 0, b_{ij} > 0, \forall i \in J, j \in O \tag{9}$$

where Equation (1) is makespan, Equation (2) indicates the total tardiness (TT), and Equation (3) is total energy consumption. Constraint (4) illustrates that only one machine with a single speed level is available for each stage of jobs at a time. Constraint (5) ensures that interruption is not allowed. Constraints (6) and (7) show the sequential constraints of job processing. Constraints (8) and (9) describe the range of variables.

There are some definitions in the multi-objective optimization problem with G objectives.

Definition 1. *Pareto dominance. x Pareto dominates y, denoted as $x \succ y$, if $f_i(x) \le f_i(y)$ for $\forall i \in \{1, 2, \ldots, G\}$, and $f_j(x) < f_j(y), \exists j \in \{1, 2, \ldots, G\}$*

Definition 2. *Pareto optimal. Solution $x^* \in \Theta$ is a Pareto optimal solution if and only if $\nexists x \in \Theta$ such that $x \succ x^*$. Solution x^* is also called a non-dominated solution.*

Definition 3. *Pareto front (PF). Pareto optimal set (PS) is a set of Pareto optimal solutions. PF is a set of Pareto optimal vectors, $PF = \{F(x) \in R^m | x \in PS\}$.*

Table 2 gives an example, including ten jobs with three stages, two machines for each stage and three processing speed levels available for each machine.

Table 2. An example of EHFSP.

Job	***Stage*1**		***Stage*2**		***Stage*3**	
	M_{11}	M_{12}	M_{21}	M_{22}	M_{31}	M_{32}
J_1	9	$9(v_1)$	$6(v_1)$	6	9	$6(v_2)$
J_2	$3(v_2)$	3	6	$9(v_2)$	$9(v_2)$	9
J_3	$6(v_1)$	3	6	$9(v_3)$	$6(v_3)$	9
J_4	$3(v_1)$	3	$3(v_2)$	3	9	$3(v_1)$
J_5	6	$6(v_2)$	$3(v_2)$	9	9	$9(v_1)$
J_6	$9(v_1)$	3	$9(v_2)$	3	3	$6(v_1)$
J_7	$9(v_1)$	6	6	$3(v_1)$	9	$3(v_2)$
J_8	9	$3(v_2)$	3	$9(v_2)$	$3(v_2)$	6
J_9	9	$6(v_2)$	$9(v_3)$	9	$6(v_2)$	3
J_{10}	3	$6(v_1)$	9	$3(v_2)$	$6(v_2)$	6

3. Introduction to ABC

Bionic computing is a comprehensive interdisciplinary subject in which problems are solved by discovering mechanisms in biology [55–57]. At present, few existing bionic computing methods can be used to simulate biological characteristics. Bees are intelligent insects. From bee dancing and optimizing honeycomb design to flying, human beings have gained a lot of enlightenment from these features. Karl von Frisch won the Nobel Prize in 1973 for his research on bee behavior. ABC is a potential bionic computing method inspired by their biological characteristics [44].

In ABC, food sources represent solutions in search space, and the nectar density corresponds to the fitness value of a feasible solution. Three main phases are included in ABC: employed bee phase, onlooker bee phase and scout bee phase. Each bee colony plays a different role in the optimization process. Employed bees fly to exploit food source information around the assigned solutions. Onlooker bees wait and choose a selected solution from the employed bees. Scout bees help to jump out of the local optima.

The basic framework of ABC is shown in Algorithm 1.

Algorithm 1 The basic framework of ABC.

1. **begin**
2. initialization
3. **while** the terminal criterion is not met **do**
4. employed bee phase
5. onlooker bee phase
6. scout bee phase
7. **end while**
8. output best solutions found so far
9. **end**

In initialization, related parameters are set and initial population P with N solutions is randomly generated.

In the employed bee phase, food sources x_m are all allocated to employed bees. A new source y_m is obtained by

$$y_m = x_m + \phi(x_m - x_n) \tag{10}$$

where x_n is another randomly selected solution from P, $\phi \in [-1, 1]$. Greedy selection based on fitness value is conducted to determine whether x_m can be updated by y_m. Afterwards, employed bees dance and share information with onlooker bees.

In the onlooker bee phase, the food source is chosen through the roulette selection probability

$$p_m = fitness(x_m) / \sum_{l=1}^{N} xl \tag{11}$$

where p_m is the probability for x_m. A new source y_m is acquired and compared with x_m in a similar way to that in the employed bee phase.

If a solution has not been updated after *Limit* trails, the corresponding employed bee will become a scout bee and exploit a new source to replace the old one, where *Limit* is the maximum number of attempts.

4. MPABC for EHFSP

In previous ABCs [45–53], the whole population is seldom categorized into several groups, and the food sources of onlooker bees are completely from employed bees. In this study, a novel multi-population method is proposed, where the divided groups are different from the three kinds of bee colonies in the classical ABC algorithm. When the population is divided into several groups, ABC can be considered as a multi-population metaheuristic algorithm. Multi-population division is useful to maintain the population diversity and avoid premature convergence [58]; on the other hand, different search strategies can be implemented in each group simultaneously, which significantly contributes to improving the search efficiency. However, these approaches are seldom introduced into ABC. In this paper, a novel MPABC is proposed based on above ideas.

4.1. Encoding and Decodinga

For EHFSP, a feasible solution contains three strings: the job permutation string (JS), machine assignment string (MS) and speed selection string (SS). A three-string encoding method [41,42] was used to represent three sub-problems in this paper. For example,

$$JS = \{\pi_1, \ldots, \pi_i, \ldots, \pi_n\}, MS = \{q_{11}, q_{12}, \ldots, q_{1m}, \ldots, q_{n1}, \ldots, q_{nm}\}, SS = \{z_{11}, z_{12}, \ldots, z_{1m}, \ldots, z_{n1}, \ldots, z_{nm}\},$$

where $\pi_i \in J$ is the processing sequence, $q_{ij} \in S_j$ denotes the selected machine at stage j for job J_i, $z_{ij} \in V$ indicates the processing speed of machine $q_{ij} \in S_j$.

As the inverse procedure of encoding, decoding translates the solution into a feasible schedule. Generally, decoding starts from the first element of JS. For machine π_i, the corresponding machine $q_{\pi_i j}$ and processing speed $z_{\pi_i j}$ are selected according to MS and SS. Subsequent stages are arranged in sequence until all processes are completed.

For the instance in Table 2, a feasible solution can be described as follows: job permutation string [1–10], machine assignment string [M_{12}, M_{22}, M_{31}, M_{11}, M_{22}, M_{32}, M_{12}, M_{21}, M_{31}, M_{11}, M_{22}, M_{31}, M_{11}, M_{21}, M_{32}, M_{12}, M_{21}, M_{32}, M_{12}, M_{22}, M_{31}, M_{11}, M_{22}, M_{31}, M_{11}, M_{21}, M_{32}, M_{12}, M_{21}, M_{32}] and speed selection string [v_2, v_2, v_2, v_1, v_1, v_2, v_2, v_3, v_1, v_1, v_3, v_3, v_1, v_2, v_1, v_2, v_2, v_1, v_1, v_2, v_1, v_2, v_2, v_2, v_1, v_2, v_1, v_1, v_1, v_2]. $V = \{1.0, 1.5, 2.0\}$ and $E_{kjl} = 4 \times v_l^2$, which are expressed in [41]. Figure 1 shows a feasible schedule. $f_1 = 43$. Suppose that $D_1 = 40$, $D_2 = 30$, $D_3 = 25$, $D_4 = 40$, $D_5 = 25$, $D_6 = 40$, $D_7 = 20$, $D_8 = 15$, $D_9 = 20, D_{10} = 30$, then $f_2 = 33.5$. For M_{11}, $EC_{11} = E_{111} \times 9 + E_{111} \times 6 + E_{111} \times 9 + E_{112} \times 2 + E_{111} \times 3 = 126$, $EC_{12} = 88$, $EC_{21} = 203.5$, $EC_{22} = 202.5$, $EC_{31} = 207$, $EC_{32} = 145$. $f_3 = 972$.

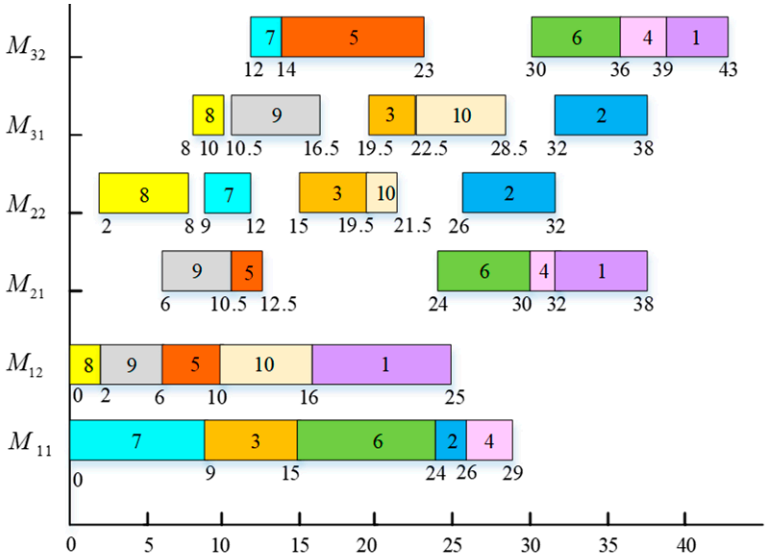


Figure 1. The Gantt chart for example.

4.2. Initialization and Group Division

Initial population P with N solutions is stochastically generated.

For each solution i, the fitness value F_i is calculated as follows [42].

$$F_i = \max_i\{rank_i\} - rank_i + \frac{dist_i}{\sum_{k\in\theta_{rank_i}} dist_k}, i = 1, 2, \ldots, N \tag{12}$$

where $rank_i$ is a non-dominated sorting value [59], $dist_i$ is the crowding distance defined in [60] and θ_{rank_i} represents a set of solutions with $rank_i$.

After that, N/s solutions are randomly allocated into s subpopulations.

For each subpopulation $subpop_j (j = 1, 2 \ldots s)$, the quality Q_j is measured as follows.

$$TF_j = \sum_{x\in subpop_j} F_j^x \tag{13}$$

$$Q_j = \delta \times \frac{TF_j - min(TF_i)}{max(TF_i) - min(TF_i)} + (1-\delta) \times \frac{Num_j^{Iter} - min\left(Num_i^{Iter}\right)}{max\left(Num_i^{Iter}\right) - min\left(Num_i^{Iter}\right) + \varepsilon}, i = 1, 2, \ldots, s \tag{14}$$

where TF_j is the total fitness value of $subpop_j$, Num_j^{Iter} represents the update quantity of solutions for $subpop_j$ in each iteration, Q_j expresses the quality of $subpop_j$, δ denotes the weight coefficient and ε is a real number. The larger Q_j is, the better the subpopulation j is. δ is set to 0.6, and ε is set to 0.001 by experiments.

Unlike the previous division methods [61–64], a new group division method according to the quality of the subpopulation is described in this paper. When the population is randomly divided into s subpopulations, several groups are constructed based on Q_j. The detailed division process is as follows. Sort $Q_j (j = 1, 2, \cdots s)$ in descending order, suppose that $Q_1 > Q_2 > \ldots > Q_{s-1} > Q_s$, if $Q_k > \bar{Q}$, $subpop_k (k = 1, 2, \cdots, p)$ is assigned into $Group_1$, the last subpopulation $subpop_s$ is allocated to $Group_3$ and the remaining $s - p - 1$ subpopulations are distributed to $Group_2$. $\bar{Q}$ represents the value of average quality.

Members of $Group_1$ are assigned to act as food sources of employed bees, solutions in $Group_2$ are used for sources of onlooker bees, and individuals in $Group_3$ are not directly involved in evolutionary operations and wait to be updated by historical optimization data.

4.3. Employed Bee Phase

Employed bees often search for new sources through crossover. In this phase, we propose a new method to execute an employed bee search. p subpopulations in $Group_1$ are totally regarded as the food sources of employed bees. These employed bees obtain computing resources through competition, that is, a higher quality solution can compete for more computing resources and opportunities to conduct global and multi-neighborhood searches. It is important to accelerate the search process and improve search efficiency.

The search procedures are shown in Algorithm 2.

Algorithm 2 Employed bee phase.

if $p > 1$ **then**
  for $j = 1$ to p **do**
    for each $x \in subpop_j$ **do**
     stochastically select a solution $y \in subpop_k (j \neq k)$ from $Group_1$
    randomly choose a solution $z \in \Omega$
     if $x \succ y$ **then**
       conduct global search between x and z
       execute multi-neighborhood search for x
     else
         perform global search between y and z
         apply multi-neighborhood search to y
      end if
    end for
   end for
else
    for each $x \in subpop_j$ **do**
        stochastically select $y \in subpop_k (j = k,\ x \neq y)$ from $Group_1, z \in \Omega$
        execute above search strategies for the higher quality solution between x and y
    end for
end if

Global search is implemented based on three crossover operations [41,42], which is described as follows. For two solutions, if a random number $\theta < \alpha$, the crossover operation on the scheduling string is carried out; otherwise, another two crossover operations on the machine assignment string and speed selection string are performed with the same probability.

In general, the scheduling sub-problem is much more complicated than the machine assignment and speed selection sub-problems, so $\alpha > 0.5$ is set to guarantee that more computing resources can be allocated to the scheduling sub-problem. In this paper, α is equal to 0.6 in experiments.

Three neighborhood structures *insert*, *change* and *speed* [41,42] are adopted, which are donated as N_1, N_2 and N_3, respectively. For solution λ, N_1 is conducted on the scheduling string. A new scheduling plan can be generated by randomly selecting a job and inserting it into a random position.

N_2 is executed on the machine assignment string and used to choose a new processing machine. The detailed steps are as follows: A machine set $\Psi = \{q_{ik} || S_k | > 1, 1 \leq i \leq n, 1 \leq k \leq m\}$ is firstly built, then randomly choose some elements from Ψ. If q_{ik} is selected, a randomly selected machine from S_k is used to replace q_{ik}.

N_3 is held on speed selection string and used for a new speed choice for the given machines. A new solution is obtained by randomly selecting some elements from the speed selection string and distributing a new speed to substitute for the selected elements stochastically. An example of three neighborhood structures is shown in Figure 2.

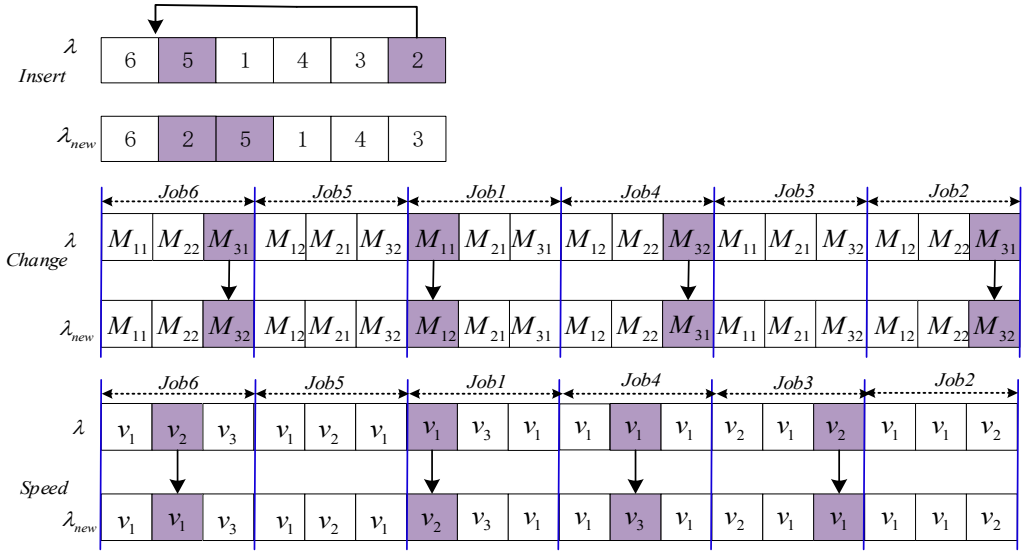


Figure 2. An example of three neighborhood structures.

Multi-neighborhood search is described in Algorithm 3, where $N_t(\lambda)$ is a neighborhood set after the N_t neighborhood search for X_i, and R is the number of neighborhood searches.

The initial archive Ω consists of non-dominated solutions generated by MPABC. If a new solution g is used to update Ω, it is firstly added to Ω; then a comparison according to the Pareto dominance relationship is conducted, and all the dominated solutions are deleted.

Memory set Λ is used to store historical optimization data, and initial Λ is empty. In the evolution procedure, the dominated solutions can be regarded as historical optimization data. When a solution g is used to renew Λ, it is directly included in Λ. If the number of solutions in Λ is greater than $|\Lambda|_{max}$, the worst solution is removed. $|\Lambda|_{max}$ is set to 20 after many experiments.

Algorithm 3 Multi-neighborhood search.

```
t =1
for l = 1 to R do
   generate a new solution g ∈ N_t(λ)
      if g is non-dominated by λ then
           replace λ with g and updated Ω with g
           update Λ with λ
      else
           updated Λ with g, t = t+1 and let t =1 if t =4
      end if
end for
```

4.4. Onlooker Bee Phase

Generally, sources of onlooker bees are all from employed bees. In this phase, solutions in $Group_2$ belong to the food sources of onlooker bees. As the elite solutions in $Group_1$ have better fitness than members in $Group_1$, best solutions from $Group_1$ are chosen as the following object of most onlooker bees, which is important to generate better scheduling plans. Meanwhile, a small number of onlooker bees execute multi-neighborhood search. Diversified search strategies performed on onlooker bees are beneficial to balance exploitation and exploration.

The search process in the onlooker bee phase is shown in Algorithm 4, where μ and β are real numbers, which are set to 0.1 and 0.3 in experiments.

In the above mentioned search process, onlooker bees can own the exclusive food sources, randomly select a solution from the upper employed bees to follow and employ the multiple search strategies. Therefore, the activities of onlooker bees differ from other ABCs.

Algorithm 4 Onlooker bee phase.

1. select $\mu * p * \frac{N}{s}$ best solutions from $Group_1$ and construct a set Θ
2. **for** each $x \in Group_2$ **do**
3. **if** a random number $\eta < \beta$ **then**
4. perform multi-neighborhood search on x
5. **else**
6. randomly choose a solution y from set Θ
7. execute global search between x and y
8. **end if**
9. **end for**

4.5. Scout Bee Phase

In the original ABC, if a solution x remains unchanged after *Limit* trials, it will be replaced by a randomly produced solution x'. In general, x is of lower quality, that is, it is probably an inefficient scheduling scheme. In this study, a solution $y \in \Lambda$ was chosen to implement multi-neighborhood search and if a new solution z is not dominated by x, solution z replaces x.

For members in $Group_3$, the worst quality is owned, and the search efficiency is lower than other ones. In this paper, the historical optimization data stored in memory set Λ are used to update $Group_3$. Firstly, the solutions in Λ and $Group_3$ are combined, then $|Group_3|$ solutions with the best solutions are selected to replace all the ones in $Group_3$. In this way, $Group_3$ is updated and the population quality is also enhanced.

After all groups are updated, they are regrouped according to Q_j. The division process is the same as that described in Section 4.2.

4.6. Algorithm Description

The whole procedure of MPABC is shown in Algorithm 5, where the terminal criterion is the maximum elapsed time.

Algorithm 5 MPABC.

1. randomly generate an initial population P, construct Ω and Λ
2. construct initial $subpop_j (j = 1, 2, \cdots, s)$ and divide them into $Group_k (i = 1, 2, \cdots, 3)$
3. **while** the terminal criterion is not met **do**
4. **for** each solution $x \in Group_1$ **do**
5. employed bee phase
6. **end for**
7. **for** each solution $y \in Group_2$ **do**
8. onlooker bee phase
9. **end for**
10. scout bee phase
11. divide all subpopulations into three groups
12. **end while**
13. output solutions in Ω

The flow chart of MPABC is shown in Figure 3.

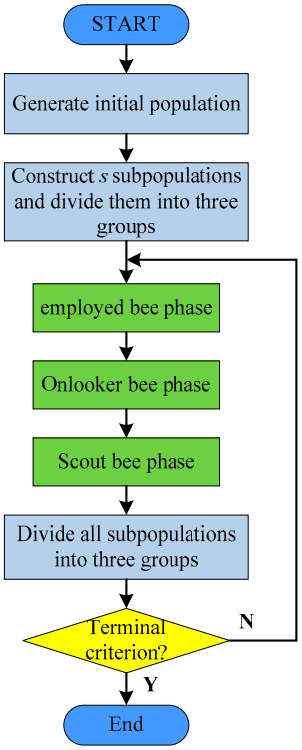


Figure 3. Flow chart of MPABC.

MPABC has the following characteristics. The whole population is divided into some subpopulations, and several groups are constructed according to the quality of subpopulations. Different groups execute different search strategies. The historical optimization data are used to update the worst group. Global search and multi-neighborhood searches are also kept balanced. These characteristics are beneficial to maintain population diversity and improve search efficiency in searching for an optimal scheduling scheme taking into account makespan, total tardiness and TEC.

5. Experiments and Results

Extensive computational experiments were implemented to verify the performance of MPABC. All experiments were coded in Visual Studio 2015 and run in the same environment: Intel(R) Core(TM) i5-6200U CPU @ 2.30 GHz 2.40 GHz, 8.00 GB RAM.

5.1. Instances, Metrics and Comparative Algorithms

In this paper, we used the instances in the literature [41,42] to test the performance of MPABC on EHFSP with a variable speed constraint. A total of 44 instances were adopted, consisting of jobs $n \in \{20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120\}$ and stages $m \in \{2, 4, 6, 8\}$. Other detailed descriptions such as S_j and V are described in [41]. Three performance metrics were utilized to evaluate the results of different algorithms.

Metric DI_R [65] is often applied to measure the convergence performance, in which the distance of non-dominated set Ω_l relative to a reference set Ω^* is computed.

$$DI_R(\Omega_l) = \frac{1}{|\Omega^*|} \sum_{y \in \Omega^*} min\{d_{xy} | x \in \Omega_l\} \tag{15}$$

where d_{xy} represents the distance between a non-dominated solution y and a reference point x in the normalized objective space. Ω_l is an approximation set to PF, Ω^* is a reference set that consists of the whole non-dominated solutions gained by each algorithm. The smaller the DI_R value is, the better the convergence is.

Metric c [66] is often adopted to compare approximate Pareto optimal set acquired by algorithms. $c(A, B)$ indicates that the proportion of individuals in
B are dominated by members in A.

$$c(A, B) = \frac{|\{y \in B: x \in A, x \succ y\}|}{|B|} \tag{16}$$

Metric *nd* is usually used to denote the number of non-dominated solutions. A larger *nd* signifies more non-dominated solutions in the approximate PF.

Zhang et al. [39] proposed a multi-objective discrete artificial bee colony algorithm based on decomposition called MDABC for EHFSP. It can be directly applied to solve our EHFSP through adding a speed selection string and neighborhood structure speed. Thus, we selected it as the first comparative algorithm. Another algorithm used was the energy-aware multi-objective optimization algorithm (EAMOA) [67], which is highly effective in solving EHFSP with makespan and energy consumption.

5.2. Parameter Settings

MPABC contains four main parameters: N, s, R and *Limit*. Taguchi method is conducted on instance 80×6. Four levels for each parameter are shown in Table 3. The orthogonal array $L_{16}(4^4)$ is given in Table 4.

Table 3. Parameters and their levels.

Parameter	Factor Level			
	1	**2**	**3**	**4**
N	60	80	100	120
S	4	6	8	10
R	8	10	12	14
Limit	5	10	15	20

Table 4. The orthogonal array $L_{16}(4^4)$.

Test	Factor Level				DI_R
	N	***S***	***R***	***Limit***	
1	1	1	1	1	11.40
2	1	2	2	2	9.184
3	1	3	3	3	10.50
4	1	4	4	4	7.429
5	2	1	2	3	8.650
6	2	2	1	4	9.530
7	2	3	4	1	10.43
8	2	4	3	2	5.987
9	3	1	3	4	8.996
10	3	2	4	3	6.512
11	3	3	1	2	7.119
12	3	4	2	1	9.038
13	4	1	4	2	10.62
14	4	2	3	1	8.815
15	4	3	2	4	10.96
16	4	4	1	3	9.610

MPABC with each combination ran 20 times independently. The metric DI_R was calculated and shown in Table 3. The trend of factor levels is demonstrated in Figure 4, and the rank of each parameter is displayed in Table 5. As shown in Figure 4, the best settings are N =100, $S = 10$, $R = 12$, $Limit = 10$.

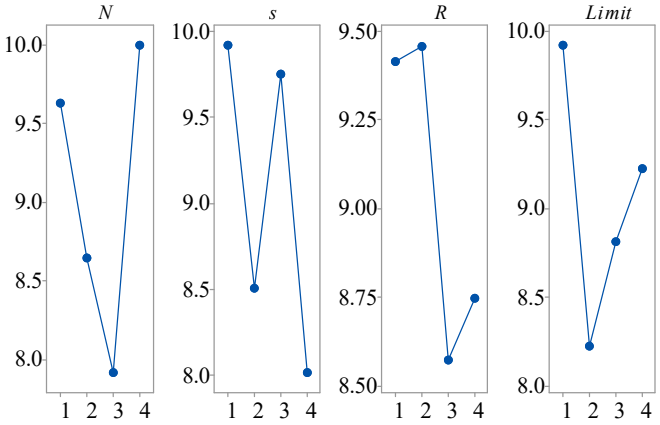


Figure 4. The trend of factor levels.

Table 5. The rank of each parameter.

Level	*N*	*S*	*R*	*Limit*
1	9.628	9.916	9.415	9.921
2	8.649	8.510	9.458	8.227
3	7.916	9.752	8.575	8.818
4	10.001	8.016	8.748	9.229
Delta	2.085	1.900	0.883	1.693
Rank	1	2	4	3

All parameter settings of MDABC, ABC and EAMOA were acquired from the data of the reference.

5.3. Results and Discussion

MPABC was compared with MDABC, ABC and EAMOA. For each instance, all algorithms independently ran 20 times and took an average of 20 times for the final result. Each algorithm ran under the same stopping criterion: the maximum CPU elapsed time $n \times m \times t$, where t was set to 100 for all instances.

The computational results about three metrics are, respectively reported in Tables 6–8. The reference set Ω^* consists of the non-dominated solutions of $\Omega_1 \cup \Omega_2 \cup \Omega_3 \cup \Omega_4$, where Ω_1,Ω_2, Ω_3 and Ω_4 are non-dominated solutions gained by MPABC, ABC, MDABC and EAMOA, respectively. In Table 7, "*P*" denotes MPABC, "*D*" indicates MDABC, "*A*" means ABC and "*E*" represents EAMOA. Figure 5 provides the box plot of all algorithms on metrics DI_R, c and nd. Figure 6 exhibits the distribution curves of non-dominated solutions produced by four algorithms in instances 60×8 and 90×6.

Table 6. Results of MPABC, MDABC, ABC and EAMOA on metric DI_R.

Instance	MPABC	ABC	MDABC	EAMOA	Instance	MPABC	ABC	MDABC	EAMOA
20 × 2	1.553	14.14	5.020	6.182	70 × 6	4.190	65.62	6.942	8.484
20 × 4	4.740	29.94	2.881	5.624	70 × 8	3.191	71.57	15.47	12.47
20 × 6	6.843	39.42	1.310	4.417	80 × 2	6.399	48.04	2.114	4.567
20 × 8	0.000	46.06	11.94	15.28	80 × 4	0.000	77.92	27.73	16.87
30 × 2	10.82	27.97	3.381	1.107	80 × 6	2.244	77.07	6.043	10.84
30 × 4	2.459	31.81	8.658	8.019	80 × 8	0.000	96.04	29.47	34.82
30 × 6	1.295	44.56	11.86	8.923	90 × 2	1.078	57.20	4.659	3.673
30 × 8	0.000	41.22	13.33	15.04	90 × 4	3.367	39.16	14.68	10.43
40 × 2	12.37	35.63	6.864	0.405	90 × 6	1.557	85.01	43.44	13.80
40 × 4	2.020	27.68	5.752	9.707	90 × 8	0.000	101.8	39.80	33.22
40 × 6	1.993	52.07	1.743	3.484	100 × 2	1.598	53.43	3.583	2.850
40 × 8	0.000	68.30	15.53	11.75	100 × 4	4.236	54.75	11.03	13.86

Table 6. *Cont.*

Instance	MPABC	ABC	MDABC	EAMOA	Instance	MPABC	ABC	MDABC	EAMOA
50 × 2	3.575	34.68	6.725	11.39	100 × 6	0.863	96.38	23.26	28.31
50 × 4	2.703	26.54	12.18	10.90	100 × 8	0.000	77.93	32.41	29.69
50 × 6	1.110	66.73	16.00	9.244	110 × 2	5.300	62.46	8.325	5.494
50 × 8	0.298	62.55	5.921	11.99	110 × 4	8.568	49.24	8.781	7.596
60 × 2	2.303	49.10	7.310	4.680	110 × 6	0.000	95.69	29.71	37.99
60 × 4	4.234	45.45	11.24	7.678	110 × 8	0.000	96.78	31.57	33.42
60 × 6	0.180	73.62	4.070	9.416	120 × 2	1.011	52.29	7.371	4.061
60 × 8	5.854	70.56	15.15	19.34	120 × 4	0.349	62.83	18.70	13.23
70 × 2	2.740	27.31	6.779	5.565	120 × 6	7.649	66.71	10.52	12.07
70 × 4	4.606	45.68	14.15	14.69	120 × 8	0.000	105.1	51.27	48.80

Table 7. Computational results of MPABC, MDABC, ABC and EAMOA on metric *c*.

Instance	*c*(*P*,*A*)	*c*(*A*,*P*)	*c*(*P*,*D*)	*c*(*D*,*P*)	*c*(*P*,*E*)	*c*(*E*,*P*)
20 × 2	1.000	0.000	0.466	0.317	0.823	0.079
20 × 4	1.000	0.000	0.000	0.875	0.000	0.750
20 × 6	1.000	0.000	0.000	0.878	0.024	0.622
20 × 8	1.000	0.000	1.000	0.000	1.000	0.000
30 × 2	0.989	0.000	0.000	0.980	0.000	1.000
30 × 4	1.000	0.000	0.770	0.000	0.216	0.226
30 × 6	1.000	0.000	0.163	0.177	0.111	0.142
30 × 8	1.000	0.000	1.000	0.000	1.000	0.000
40 × 2	1.000	0.000	0.043	0.867	0.000	1.000
40 × 4	1.000	0.000	0.320	0.219	0.611	0.000
40 × 6	1.000	0.000	0.325	0.269	0.286	0.179
40 × 8	1.000	0.000	1.000	0.000	1.000	0.000
50 × 2	1.000	0.000	0.023	0.821	0.846	0.021
50 × 4	1.000	0.000	0.713	0.000	0.049	0.387
50 × 6	1.000	0.000	0.971	0.000	0.990	0.000
50 × 8	1.000	0.000	0.952	0.000	1.000	0.000
60 × 2	1.000	0.000	0.803	0.013	0.083	0.558
60 × 4	1.000	0.000	0.000	0.667	0.028	0.686
60 × 6	1.000	0.000	0.563	0.260	1.000	0.000
60 × 8	1.000	0.000	0.664	0.000	0.820	0.000
70 × 2	1.000	0.000	0.754	0.012	0.730	0.012
70 × 4	1.000	0.000	0.105	0.015	0.000	0.169
70 × 6	1.000	0.000	0.193	0.000	0.115	0.033
70 × 8	1.000	0.000	0.640	0.000	0.212	0.000
80 × 2	1.000	0.000	0.065	0.738	0.034	0.767
80 × 4	1.000	0.000	1.000	0.000	1.000	0.000
80 × 6	1.000	0.000	0.045	0.297	0.086	0.034
80 × 8	1.000	0.000	1.000	0.000	1.000	0.000
90 × 2	1.000	0.000	0.586	0.036	0.512	0.095
90 × 4	1.000	0.000	0.717	0.000	0.916	0.000
90 × 6	1.000	0.000	0.776	0.000	0.074	0.016
90 × 8	1.000	0.000	1.000	0.000	1.000	0.000
100 × 2	1.000	0.000	0.432	0.283	0.405	0.161
100 × 4	1.000	0.000	0.180	0.028	0.014	0.063
100 × 6	1.000	0.000	1.000	0.000	0.978	0.000
100 × 8	1.000	0.000	1.000	0.000	1.000	0.000
110 × 2	1.000	0.000	0.161	0.204	0.059	0.476
110 × 4	1.000	0.000	0.433	0.000	0.088	0.000
110 × 6	1.000	0.000	1.000	0.000	1.000	0.000

Table 7. *Cont.*

Instance	*c*(*P*,*A*)	*c*(*A*,*P*)	*c*(*P*,*D*)	*c*(*D*,*P*)	*c*(*P*,*E*)	*c*(*E*,*P*)
110 × 8	1.000	0.000	1.000	0.000	1.000	0.000
120 × 2	1.000	0.000	0.815	0.029	0.456	0.233
120 × 4	1.000	0.000	1.000	0.000	0.985	0.000
120 × 6	1.000	0.000	0.082	0.000	0.066	0.000
120 × 8	1.000	0.000	1.000	0.000	1.000	0.000

Table 8. Results of MPABC, MDABC, ABC and EAMOA on metric *nd*.

Instance	MPABC	ABC	MDABC	EAMOA	Instance	MPABC	ABC	MDABC	EAMOA
20 × 2	0	0	28	71	70 × 6	114	0	22	39
20 × 4	7	0	28	21	70 × 8	117	0	0	52
20 × 6	11	0	91	8	80 × 2	11	0	67	21
20 × 8	73	0	0	0	80 × 4	51	0	0	0
30 × 2	0	0	32	92	80 × 6	102	0	33	20
30 × 4	113	0	9	85	80 × 8	54	0	0	0
30 × 6	93	0	33	9	90 × 2	123	0	18	24
30 × 8	67	0	0	0	90 × 4	185	0	0	92
40 × 2	0	0	3	82	90 × 6	124	0	0	25
40 × 4	75	0	51	5	90 × 8	62	0	0	0
40 × 6	49	0	27	5	100 × 2	129	0	32	51
40 × 8	64	0	0	0	100 × 4	133	0	10	62
50 × 2	25	0	42	0	100 × 6	73	0	0	2
50 × 4	114	0	0	77	100 × 8	151	0	0	0
50 × 6	111	0	3	1	110 × 2	49	0	35	30
50 × 8	76	0	3	0	110 × 4	74	0	10	100
60 × 2	69	0	5	55	110 × 6	36	0	0	0
60 × 4	31	0	13	62	110 × 8	200	0	0	0
60 × 6	37	0	7	0	120 × 2	155	0	7	49
60 × 8	136	0	41	19	120 × 4	116	0	0	2
70 × 2	82	0	12	31	120 × 6	90	0	56	33
70 × 4	113	0	0	92	120 × 8	138	0	0	0

As shown in Tables 6–8, MPABC generated smaller DI_R than ABC in 44 instances. It also had higher $c(P, A)$ than $c(A, P)$ in 44 instances; moreover, $c(P, A)$ was equal to 1 in 43 instances, that is, the total solutions of ABC were dominated by the obtained solutions of MPABC. Similarly, the metric *nd* of MPABC was far greater than that of ABC in all instances, that is, ABC could not make any contribution to the reference set Ω^*. MPABC displayed a better performance than ABC. This conclusion can also be reached by analyzing Figures 5 and 6. The significant difference between MPABC and ABC discloses that the multi-population scheme, different search strategies and the utilization of historical optimization data are extremely beneficial to the performance of MPABC.

In addition, results of MPABC are distinctly different from MDABC in most instances within the specified time. DI_R of MPABC is smaller than that of MDABC in 38 instances and larger than that of MDABC in only 6 instances. Moreover, $c(P, D)$ was superior to $c(D, P)$ in 34 instances, and $c(P, D)$ was equal to 1 in 12 instances. That is, all solutions of MDABC were dominated by those of MPABC in 12 instances. On the other hand, the *nd* metric produced by MDABC was much smaller than that of MPABC in 37 instances.

Compared with EAMOA, MPABC generated lower DI_R in 39 instances and obtained a larger $c(P, E)$ than $c(E, P)$ in 31 instances. Moreover, the non-dominated solutions of MPABC dominated those of EAMOA in 12 instances. On the other hand, the *nd* metric generated by EAMOA was less than that of MPABC in 37 instances. Obviously, MPABC has more significant advantages over three metrics than ABC, MPABC and EAMOA.

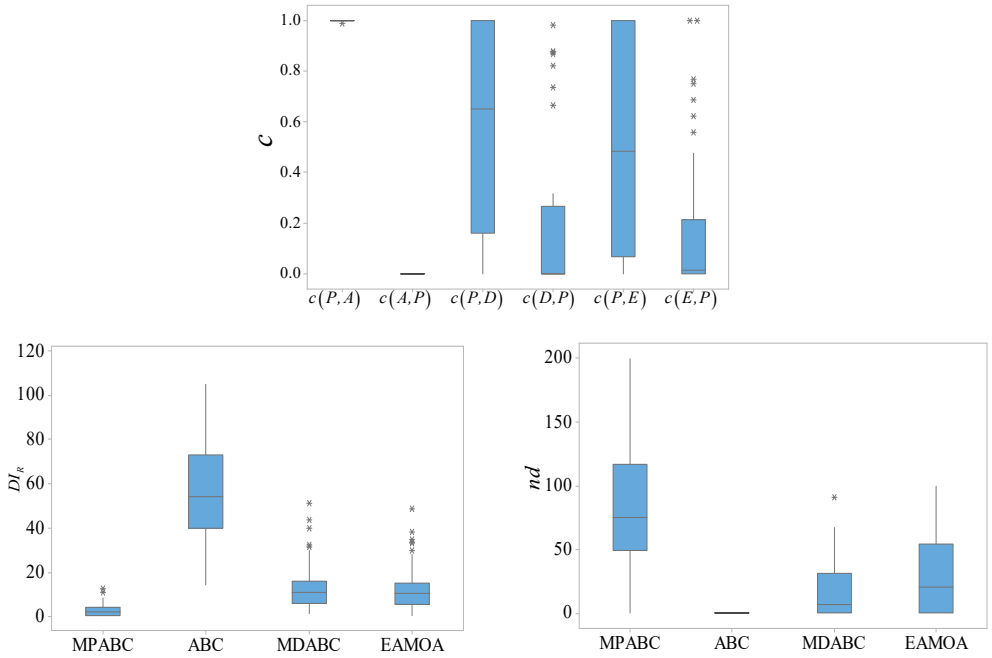


Figure 5. The box plot of four algorithms on metric DI_R, c, nd.

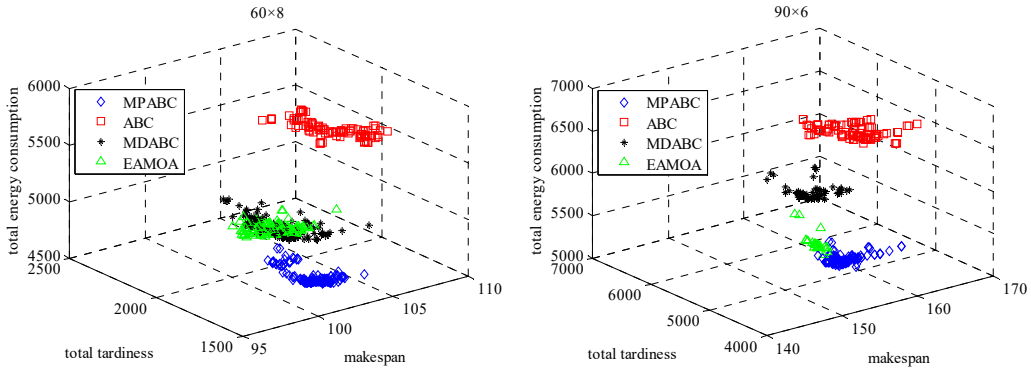


Figure 6. The distribution curve of four algorithms in instances 60×8 and 90×6.

To further verify the statistical difference in the algorithm performance, a Wilcoxon signed-rank test was conducted to compare MPABC with other algorithms. The test results are shown in Table 9. A significance level of 0.05 is assumed, which means that A is significantly different from B statistically if the asymptotically significance value is less than 0.05.

Table 9. Results of Wilcoxon signed-rank test.

Algorithm Pair	Metric	Positive Rank	Negative Rank	Asymptotically Significance
MPABC vs. ABC	DI_R	0	44	$7.6159 \times 10^{-9} < 0.05$
	c	44	0	$5.241 \times 10^{-11} < 0.05$
	nd	41 + 3 *	0	$2.421 \times 10^{-8} < 0.05$
MPABC vs. MDABC	DI_R	6	38	$1.0 \times 10^{-6} < 0.05$
	c	34	10	$3.76 \times 10^{-4} < 0.05$
	nd	37	7	$6.632 \times 10^{-7} < 0.05$
MPABC vs. EAMOA	DI_R	5	39	$9.512 \times 10^{-7} < 0.05$
	c	31	13	$1.354 \times 10^{-3} < 0.05$
	nd	37	7	$9.0 \times 10^{-6} < 0.05$

* indicates $c(P, A) = c(A, P)$.

This conclusion can also be drawn from Figures 5 and 6 and Table 9. In Figure 5, the median, upper and lower edges of MPABC on DI_R, c and nd are superior to those of ABC, MPABC and EAMOA. Figures 5 and 6 demonstrate that solutions obtained by MPABC had better convergence and distribution. On the other hand, the asymptotically significance value was smaller than 0.05. That is to say, the statistical results of MPABC are remarkably better than other three algorithms.

MPABC performs better than comparative algorithms in solving EHFSP, which results from the novel multi-population division method, different search strategies in each group and the utilization of historical optimization data. The multi-population method is effective to maintain population diversity, different search strategies in groups are important to balance exploration and exploitation, and the utilization of historical data can rapidly and easily improve search efficiency. These features are useful to maintain the population diversity and avoid falling premature. It is concluded from above results and analysis that MPABC is more competitive in EHFSP with a variable speed constraint.

5.4. A Real-Life Case Study

A real-life case study in a roll enterprise machining workshop was considered. There are three main processes in the crafting process: turning, milling and grinding, and three types of machines available for the processing. The first type is composed of three machines for turning ($M_1 \sim M_3$), the second type consists of three machines for milling ($M_4 \sim M_6$), and the third class includes four machines for grinding ($M_7 \sim M_{10}$). Its production process is a typical hybrid flow shop problem. The calculation method is shown in Section 4.

There are 30 jobs need to be processed. A speed set $V = \{1.0, 1.3, 1.55, 1.8, 2.0\}$ is available for machines. The processing data is shown in Table 10. The non-dominated solutions obtained by MPABC are shown in Table 11. An optimal schedule is presented in Figure 7, in which the makespan is 16.35, the total tardiness is 56.92 and the total energy consumption is 780.48.

Table 10. An example of EHFSP.

Job	***Stage*1**				***Stage*2**			***Stage*3**		
	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
J_1	2	3	5	1	5	5	4	4	3	5
J_2	1	1	2	3	2	2	1	3	3	2
J_3	2	5	3	4	3	3	2	2	4	1
J_4	3	2	2	4	5	3	3	5	1	5
J_5	4	2	3	4	4	5	2	2	4	4
J_6	3	5	3	3	3	5	4	2	5	4
J_7	2	1	1	3	4	2	1	3	5	4
J_8	2	1	1	5	1	1	2	2	4	4
J_9	5	4	5	5	2	1	2	2	2	4
J_{10}	5	5	2	4	3	4	4	3	5	2
J_{11}	4	1	5	4	5	1	3	2	4	2
J_{12}	2	1	5	3	1	5	3	4	3	3
J_{13}	2	1	2	2	1	3	1	5	2	3
J_{14}	1	1	2	5	2	1	3	2	3	3
J_{15}	3	3	4	4	1	5	5	4	2	4
J_{16}	3	2	2	1	4	4	1	2	3	1
J_{17}	1	5	5	3	4	4	1	2	3	1
J_{18}	2	2	5	1	2	2	4	5	4	5
J_{19}	2	5	4	5	4	4	1	4	3	3
J_{20}	4	4	4	4	3	2	1	3	4	5
J_{21}	5	2	1	2	1	5	5	2	4	4
J_{22}	5	4	5	5	5	3	1	1	4	4

Table 10. *Cont.*

Job	*Stage*1				*Stage*2			*Stage*3		
	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
J_{23}	4	3	5	4	4	1	4	4	1	2
J_{24}	4	2	5	4	5	3	3	3	4	3
J_{25}	4	5	1	3	4	2	1	4	2	2
J_{26}	3	5	3	1	4	2	3	2	1	4
J_{27}	5	3	5	2	2	4	3	2	2	5
J_{28}	3	2	3	5	1	3	1	1	4	5
J_{29}	2	4	2	5	1	4	3	1	3	1
J_{30}	4	2	3	3	3	3	5	5	2	5

Table 11. The non-dominated solutions obtained by MPABC.

Solutions	Makespan	TT	TEC
1	16.35	56.92	780.48
2	17.15	64.44	733.24
3	16.19	53.02	793.80
4	17.99	88.50	720.31
5	17.21	55.31	746.49
6	17.99	87.61	721.28
7	18.48	89.82	692.89
8	19.88	86.15	698.33
9	19.51	89.19	699.72
10	18.48	91.00	690.48
11	17.99	88.50	720.31
12	17.99	89.74	715.34
13	19.62	85.54	707.03
14	16.28	75.26	783.55
15	19.88	83.52	707.07
16	21.11	109.45	688.90
17	19.85	84.18	710.20
18	19.15	87.57	713.18
19	20.31	85.80	706.51
20	17.94	87.17	726.96
21	19.00	82.55	713.57
21	19.39	85.31	711.69
22	19.57	88.50	706.70
23	20.65	84.43	706.65
24	16.41	58.27	762.89
25	18.99	88.50	713.98
26	16.72	48.74	780.39
27	17.99	88.50	720.31
28	19.39	88.72	710.92
29	16.86	61.87	746.01
30	16.41	68.37	771.96

From above results, the manufacturers can obtain a suitable schedule scheme to make a trade-off between makespan, TT and TEC. Therefore, optimizing the speed selection sub-problem with job scheduling and machine assignment sub-problems simultaneously is meaningful; meanwhile, MPABC is of great significance to the research of EHFSP.

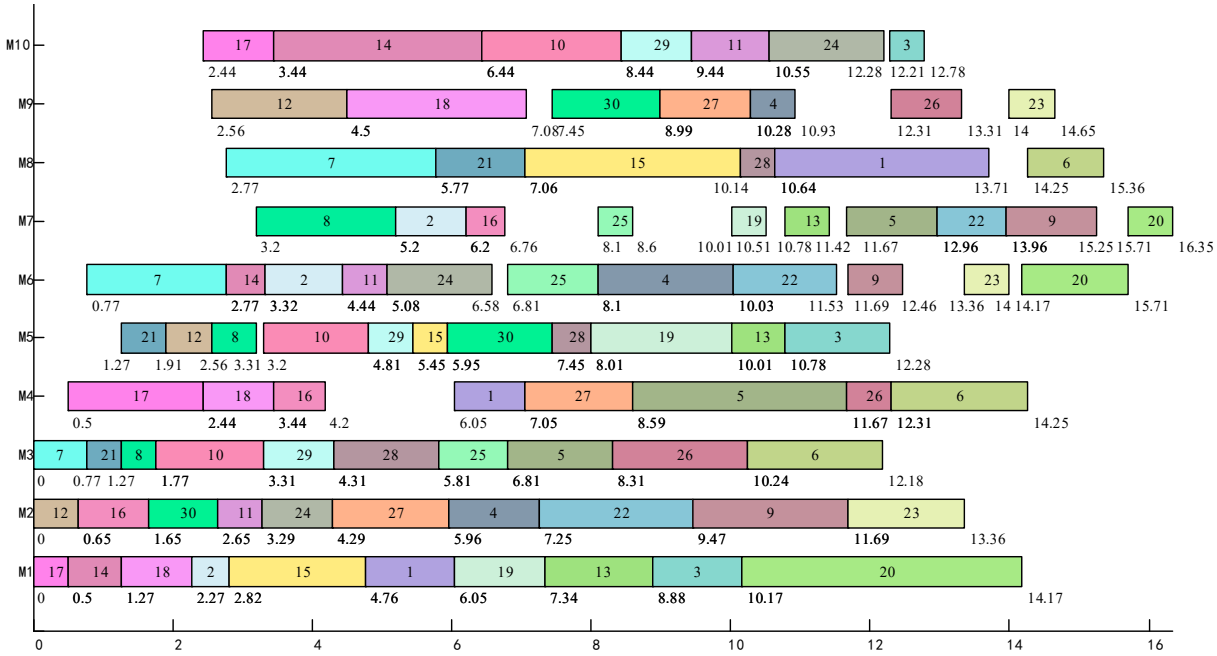


Figure 7. An optimal schedule of the real-life case.

6. Conclusions

In this paper, we proposed a novel MPABC for EHFSP with a variable speed constraint. The proposed MPABC concerns three objectives, namely makespan, total tardiness and TEC. A new multi-population division method was embedded to achieve the group division. The division of three groups made it possible for each group to own exclusive food sources and perform different search strategies, which helps to increase the population diversity and improves the search efficiency. The weakest group is excluded from the evolution and renewed through historical optimization data, which can enhance the quality of solutions. Subpopulations are measured by the newly designed evaluation index. Finally, extensive comparative experiments with ABC, MDABC and EAMOA were conducted to verify the algorithm performance. The results demonstrate that the proposed MPABC can generate more competitive solutions in solving EHFSP.

This work has not yet considered the feedback in evolution or the cooperation and competition relationship between groups. In the future research, these factors will be fully discussed, and more effective search strategies such as Q-learning can be explored to enhance the search performance. Accordingly, more intelligent optimization algorithms can be introduced to solve EHFSP. In addition, we are interested in investigating the capability of MPABC in the energy-efficient distributed scheduling problem, fuzzy flow shop scheduling problem and other engineering optimization problems.

Author Contributions: Y.Z. contributed to the conception of the study, the background research, method design and analysis of the experimental results, and wrote the manuscript; Z.F. provided the methodological guidance, writing review and funding support; T.Z. reviewed and revised the manuscript; P.W. performed the methodological guidance, writing review and funding support. All authors have read and agreed to the published version of the manuscript.

Funding: This research is funded by the National Natural Science Foundation of China (62176147, 71371148), China-Russia S&T Innovation Year 2020-2021 Action Plan (RC20200005), the Science and Technology Planning Project of Guangdong Province of China (2021A0505030072, 2021A0505070003, 2019A050520001, 2019A050519008 and 190827105585418), and the State Key Lab of Digital Manufacturing Equipment & Technology (DMETKF2019020).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Allahverdi, A.; Ng, C.T.; Cheng, T.C.E.; Kovalyov, M.Y. A survey of scheduling problems with setup times or costs. *Eur. J. Oper. Res.* **2008**, *187*, 985–1032. [CrossRef]
2. Chen, L.; Wang, J.F.; Xu, X.Y. An energy-efficient single machine scheduling problem with machine reliability constraints. *Comput. Ind. Eng.* **2019**, *137*, 106072. [CrossRef]
3. Wu, X.Q.; Che, A. A memetic differential evolution algorithm for energy-efficient parallel machines scheduling. *Omega* **2019**, *82*, 155–165. [CrossRef]
4. May, G.; Stahl, B.; Taisch, M.; Prabhu, V. Multi-objective genetic algorithm for energy-efficient job shop scheduling. *Int. J. Prod. Res.* **2015**, *53*, 7071–7089. [CrossRef]
5. Wu, X.Q.; Che, A.D. Energy-efficient no-wait permutation flow shop scheduling by adaptive multi-objective variable neighborhood search. *Omega* **2019**, *94*, 102117. [CrossRef]
6. Choi, B.C.; Yoon, S.H. Maximizing the weighted number of just-in-time jobs in flow shop scheduling. *J. Sched.* **2007**, *10*, 237–243. [CrossRef]
7. Ruiz, R.; Vázquez-Rodríguez, J.A. The hybrid flow shop scheduling problem. *Eur. J. Oper. Res.* **2010**, *205*, 1–18. [CrossRef]
8. Ding, J.Y.; Song, S.J.; Wu, C. Carbon-efficient scheduling of flow shops by multi-objective optimization. *Eur. J. Oper. Res.* **2016**, *248*, 758–771. [CrossRef]
9. Foumani, M.; Smith-Miles, K. The impact of various carbon reduction policies on green flowshop scheduling. *Appl. Energy* **2019**, *249*, 300–315. [CrossRef]
10. Mansouri, S.A.; Aktas, E.; Besikci, U. Green scheduling of a two-machine flowshop: Trade-off between makespan and energy consumption. *Eur. J. Oper. Res.* **2016**, *248*, 772–788. [CrossRef]
11. Jiang, E.D.; Wang, L. An improved multi-objective evolutionary algorithm based on decomposition for energy-efficient permutation flow shop scheduling problem with sequence-dependent setup time. *Int. J. Prod. Res.* **2019**, *57*, 1756–1771. [CrossRef]
12. Lei, D.M.; Guo, X.P. A shuffled frog-leaping algorithm for hybrid flow shop scheduling with two agents. *Expert Syst. Appl.* **2015**, *42*, 9333–9339. [CrossRef]
13. Long, J.Y.; Zheng, Z.; Gao, X.Q.; Pardalos, P.M. Scheduling a realistic hybrid flow shop with stage skipping and adjustable processing time in steel plants. *Appl. Soft. Comput.* **2018**, *64*, 536–549. [CrossRef]
14. Liao, C.J.; Tjandradjaja, E.; Chung, T.P. An approach using particle swarm optimization and bottleneck heuristic to solve hybrid flow shop scheduling problem. *Appl. Soft. Comput.* **2012**, *12*, 1755–1764. [CrossRef]
15. Cai, J.C.; Zhou, R.; Lei, D.M. Dynamic shuffled frog-leaping algorithm for distributed hybrid flow shop scheduling with multiprocessor tasks. *Eng. Appl. Artif. Intel.* **2020**, *90*, 103540. [CrossRef]
16. Li, J.Q.; Pan, Q.K.; Wang, F.T. A hybrid variable neighborhood search for solving the hybrid flow shop scheduling problem. *Appl. Soft. Comput.* **2014**, *24*, 63–77. [CrossRef]
17. Li, M.; Su, B.; Lei, D.M. A novel imperialist competitive algorithm for fuzzy distributed assembly flow shop scheduling. *J. Intell. Fuzzy Syst.* **2021**, *40*, 4545–4561. [CrossRef]
18. Bruzzone, A.A.G.; Anghinolfi, D.; Paolucci, M.; Tonelli, F. Energy-aware scheduling for improving manufacturing process sustainability: A mathematical model for flexible flow shops. *Cirp. Ann. Manuf. Techn.* **2012**, *61*, 459–462. [CrossRef]
19. Dai, M.; Tang, D.B.; Giret, A.; Salido, M.A.; Li, W.D. Energy-efficient scheduling for a flexible flow shop using an improved genetic-simulated annealing algorithm. *Robot. Cim. Int. Manuf.* **2013**, *29*, 418–429. [CrossRef]
20. Meng, L.L.; Zhang, C.Y.; Shao, X.Y.; Ren, X.Y.; Ren, C.L. Mathematical modelling and optimization of energy-conscious hybrid flow shop scheduling problem with unrelated parallel machines. *Int. J. Prod. Res.* **2019**, *57*, 1119–1145. [CrossRef]
21. Gong, G.L.; Chiong, R.; Deng, Q.W.; Han, W.W.; Zhang, L.K.; Lin, W.H.; Li, K.X. Energy-efficient flexible flow shop scheduling with worker flexibility. *Expert. Syst. Appl.* **2020**, *141*, 112902. [CrossRef]
22. Wu, X.L.; Shen, X.L.; Cui, Q. Multi-objective flexible flow shop scheduling problem considering variable processing time due to renewable energy. *Sustainability* **2018**, *10*, 841. [CrossRef]
23. Zeng, Z.Q.; Hong, M.N.; Man, Y. Multi-object optimization of flexible flow shop scheduling with batch process-Consideration total electricity consumption and material wastage. *J. Clean. Prod.* **2018**, *183*, 925–939. [CrossRef]
24. Liu, Y.; Farnsworth, M.; Tiwari, A. Energy-efficient scheduling of flexible flow shop of composite recycling. *Int. J. Adv. Manuf. Tech.* **2018**, *97*, 117–127. [CrossRef]
25. Yan, J.H.; Li, L.; Zhao, F.; Zhang, F.Y.; Zhao, Q.L. A multi-level optimization approach for energy-efficient flexible flow shop scheduling. *J. Clean. Prod.* **2016**, *137*, 1543–1552. [CrossRef]
26. Wang, S.J.; Wang, X.D.; Chu, F.; Yu, J.B. An energy-efficient two-stage hybrid flow shop scheduling problem in a glass production. *Int. J. Prod. Res.* **2020**, *58*, 2283–2314. [CrossRef]
27. Zhou, B.H.; Liu, W.L. Energy-efficient multi-objective scheduling algorithm for hybrid flow shop with fuzzy processing time. *Proc. Inst. Mech. Eng. Part I J. Syst. Control Eng.* **2019**, *233*, 1282–1297. [CrossRef]
28. Schulz, S.; Neufeld, J.S.; Buscher, U. A multi-objective iterated local search algorithm for comprehensive energy-aware hybrid flow shop scheduling. *J. Clean Prod.* **2019**, *224*, 421–434. [CrossRef]

29. Luo, H.; Du, B.; Huang, G.Q.; Chen, H.P.; Li, X.L. Hybrid flow shop scheduling considering machine electricity consumption cost. *Int. J. Prod. Econ.* **2013**, *146*, 423–439. [CrossRef]
30. Lin, W.W.; Yu, D.Y.; Zhang, C.Y.; Liu, X.; Zhang, S.Q.; Tian, Y.H.; Liu, S.Q.; Xie, Z.P. A Multi-objective teaching−learning-based optimization algorithm to scheduling in turning processes for minimizing makespan and carbon footprint. *J. Clean. Prod.* **2015**, *101*, 337–347. [CrossRef]
31. Zhou, R.; Lei, D.M.; Zhou, X.M. Multi-objective energy-efficient interval scheduling in hybrid flow shop using imperialist competitive algorithm. *IEEE Access* **2019**, *7*, 85029–85041. [CrossRef]
32. Zhang, B.; Pan, Q.K.; Gao, L.; Meng, L.L.; Li, X.Y.; Peng, K.K. A three-stage multiobjective approach based on decomposition for an energy-efficient hybrid flow shop scheduling problem. *IEEE Trans. Syst. Man Cybern. Syst.* **2020**, *50*, 4984–4999. [CrossRef]
33. Li, J.Q.; Tao, X.R.; Jia, B.X. Efficient multi-objective algorithm for the lot-streaming hybrid flowshop with variable sub-lots. *Swarm Evol. Comput.* **2020**, *52*, 100600. [CrossRef]
34. Singh, S.; Chauhan, P.; Singh, N. Capacity optimization of grid connected solar/fuel cell energy system using hybrid ABC-PSO algorithm. *Int. J. Hydrogen Energy* **2020**, *45*, 10070–10088. [CrossRef]
35. Wang, L.; Li, D. Fuzzy distributed hybrid flow shop scheduling problem with heterogeneous factory and unrelated parallel machine: A shuffled frog leaping algorithm with collaboration of multiple search strategies. *IEEE Access* **2020**, *8*, 214209–214223. [CrossRef]
36. Tirkolaee, E.B.; Goli, A.; Weber, G.W. Fuzzy mathematical programming and self-adaptive artificial fish swarm algorithm for just-in-time energy-aware flow shop scheduling problem with outsourcing option. *IEEE Trans. Fuzzy Syst.* **2020**, *28*, 2772–2783. [CrossRef]
37. Liu, C.H.; Huang, D.H. Reduction of power consumption and carbon footprints by applying multi-objective optimisation via genetic algorithms. *Int. J. Prod. Res.* **2014**, *52*, 337–352. [CrossRef]
38. Tang, D.B.; Dai, M.; Salido, M.A.; Giret, A. Energy-efficient dynamic scheduling for a flexible flow shop using an improved particle swarm optimization. *Comput. Ind.* **2016**, *81*, 82–95. [CrossRef]
39. Zhang, B.; Pan, Q.K.; Gao, L.; Li, X.Y.; Meng, L.L.; Peng, K.K. A Multiobjective Evolutionary Algorithm Based on Decomposition for Hybrid Flowshop Green Scheduling Problem. *Comput. Ind. Eng.* **2019**, *136*, 325–344. [CrossRef]
40. Chen, T.L.; Cheng, C.Y.; Chou, Y.H. Multi-objective genetic algorithm for energy-efficient hybrid flow shop scheduling with lot streaming. *Ann. Oper. Res.* **2020**, *290*, 813–836. [CrossRef]
41. Lei, D.M.; Gao, L.; Zheng, Y.L. A novel teaching-learning-based optimization algorithm for energy-efficient scheduling in hybrid flow shop. *IEEE Trans. Eng. Manag.* **2018**, *65*, 330–340. [CrossRef]
42. Li, M.; Lei, D.M.; Cai, J.C. Two-level imperialist competitive algorithm for energy-efficient hybrid flow shop scheduling problem with relative importance of objectives. *Swarm. Evol. Comput.* **2019**, *49*, 34–43. [CrossRef]
43. Oztop, H.; Tasgetiren, M.F.; Kandiller, L.; Eliiyi, D.T.; Gao, L. Ensemble of metaheuristics for energy-efficient hybrid flowshops: Makespan versus total energy consumption. *Swarm Evol. Comput.* **2020**, *54*, 100660. [CrossRef]
44. Karaboga, D.; Basturk, B. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *J. Global Optim.* **2007**, *39*, 459–471. [CrossRef]
45. Karaboga, D.; Akay, B. A modified artificial bee colony (ABC) algorithm for constrained optimization problems. *Appl. Soft Comput.* **2011**, *11*, 3021–3031. [CrossRef]
46. Aslan, S.; Badem, H.; Karaboga, D. Improved quick artificial bee colony (iqABC) algorithm for global optimization. *Soft Comput.* **2019**, *23*, 13161–13182. [CrossRef]
47. Pan, Q.K.; Wang, L.; Mao, K.; Zhao, J.H.; Zhang, M. An effective artificial bee colony algorithm for a real-world hybrid flowshop problem in steelmaking process. *IEEE Trans. Autom. Sci. Eng.* **2013**, *10*, 307–322. [CrossRef]
48. Cui, Z.; Gu, X.S. An improved discrete artificial bee colony algorithm to minimize the makespan on hybrid flow shop problems. *Neurocomputing* **2015**, *148*, 248–259. [CrossRef]
49. Pan, Q.K.; Wang, L.; Li, J.Q.; Duan, J.H. A novel discrete artificial bee colony algorithm for the hybrid flowshop scheduling problem with makespan minimisation. *Omega* **2014**, *45*, 42–56. [CrossRef]
50. Li, J.Q.; Pan, Q.K. Solving the large-scale hybrid flow shop scheduling problem with limited buffers by a hybrid artificial bee colony algorithm. *Inform. Sci.* **2015**, *316*, 487–502. [CrossRef]
51. Li, J.Q.; Pan, Q.K.; Duan, P.Y. An improved artificial bee colony algorithm for solving hybrid flexible flowshop with dynamic operation skipping. *IEEE Trans. Cybern.* **2016**, *46*, 1311–1324. [CrossRef] [PubMed]
52. Lin, C.; Xu, J.Y.; Baia, D.Y.; Chung, I.H.; Liu, S.C.; Wu, C.C. Artificial bee colony algorithms for the order scheduling with release dates. *Soft Comput.* **2019**, *23*, 8677–8688. [CrossRef]
53. Li, J.Q.; Han, Y.Q. A hybrid multi-objective artificial bee colony algorithm for flexible task scheduling problems in cloud computing system. *Cluster Comput.* **2020**, *23*, 2483–2499. [CrossRef]
54. Lei, D.M.; Yuan, Y.; Cai, J.C. An improved artificial bee colony for multi-objective distributed unrelated parallel machine scheduling. *Int. J. Prod. Res.* **2021**, *59*, 5259–5271. [CrossRef]
55. Wang, P.; Zhang, J.J.; Xu, L.; Wang, H.; Feng, S.; Zhu, H.S. How to measure adaptation complexity in evolvable systems-A new synthetic approach of constructing fitness functions. *Expert Syst. Appl.* **2011**, *38*, 10414–10419. [CrossRef]
56. Wang, P.; Zhu, H.S.; Wilamowska-Korsak, M.; Bi, Z.M.; Li, L. Determination of weights for multiobjective decision making or machine learning. *IEEE Syst. J.* **2014**, *8*, 63–72. [CrossRef]

57. Wang, P.; Wang, J.S.; Zhang, J. Methodological research for modular neural networks based on "an expert with other capabilities". *J. Glob. Inf. Manag.* **2018**, *26*, 104–126. [CrossRef]
58. Ma, H.P.; Shen, S.G.; Yu, M.; Yang, Z.L.; Fei, M.R.; Zhou, H.Y. Multi-population techniques in nature inspired optimization algorithms: A comprehensive survey. *Swarm Evol. Comput.* **2019**, *44*, 365–387. [CrossRef]
59. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Evolut. Comput.* **2002**, *6*, 182–197. [CrossRef]
60. Lei, D.M.; Li, M.; Wang, L. A two-phase meta-heuristic for multiobjective flexible job shop scheduling problem with total energy consumption threshold. *IEEE Trans. Cybern.* **2019**, *49*, 1097–1109. [CrossRef]
61. Xiang, Y.; Zhou, Y.R. A dynamic multi-colony artificial bee colony algorithm for multi-objective optimization. *Appl. Soft Comput.* **2015**, *35*, 766–785. [CrossRef]
62. Zhou, J.J.; Yao, X.F.; Chan, F.T.S.; Lin, Y.Z.; Jin, H.; Gao, L.; Wang, Y.P. An individual dependent multi-colony artificial bee colony algorithm. *Inform. Sci.* **2019**, *485*, 114–140. [CrossRef]
63. Gao, W.F.; Huang, L.L.; Liu, S.Y. Artificial bee colony algorithm based on information learning. *IEEE Trans. Cybern.* **2015**, *45*, 2827–2839. [CrossRef]
64. Tereshko, V. Reaction-Diffusion Model of a Honeybee Colony's Foraging Behavior. In *Parallel Problem Solving from Nature VI*; Lecture Notes in Computer Science; Schoenauer, M., Ed.; Springer: Berlin/Heidelberg, Germany, 2000; pp. 807–816.
65. Knowles, J.D.; Corne, D.W. On metrics for comparing nondominated sets. In Proceedings of the 2002 Congress on Evolutionary Computation, Honolulu, HI, USA, 12–17 May 2002; pp. 711–716.
66. Zitzler, E.; Thiele, L. Multi-objective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Trans. Evol. Comput.* **1999**, *3*, 257–271. [CrossRef]
67. Li, J.Q.; Sang, H.Y.; Han, Y.Y.; Wang, C.G.; Gao, K.Z. Efficient multi-objective optimization algorithm for hybrid flow shop scheduling problems with setup energy consumptions. *J. Clean. Prod.* **2018**, *181*, 584–598. [CrossRef]