algorithms

MDPI

Article

Modified Migrating Birds Optimization for Energy-Aware Flexible Job Shop Scheduling Problem

Hongchan Li [1], Haodong Zhu [1,*] and Tianhua Jiang [2]

[1] School of Computer and Communication Engineering, Zhengzhou University of Light Industry, Zhengzhou 450002, China; 2011017@zzuli.edu.cn
[2] School of Transportation, Ludong University, Yantai 264025, China; jth1127@163.com
* Correspondence: 2011016@zzuli.edu.cn

Received: 1 February 2020; Accepted: 18 February 2020; Published: 20 February 2020

Abstract: In recent decades, workshop scheduling has excessively focused on time-related indicators, while ignoring environmental metrics. With the advent of sustainable manufacturing, the energy-aware scheduling problem has been attracting more and more attention from scholars and researchers. In this study, we investigate an energy-aware flexible job shop scheduling problem to reduce the total energy consumption in the workshop. For the considered problem, the energy consumption model is first built to formulate the energy consumption, such as processing energy consumption, idle energy consumption, setup energy consumption and common energy consumption. Then, a mathematical model is established with the criterion to minimize the total energy consumption. Secondly, a modified migrating birds optimization (MMBO) algorithm is proposed to solve the model. In the proposed MMBO, a population initialization scheme is presented to ensure the initial solutions with a certain quality and diversity. Five neighborhood structures are employed to create neighborhood solutions according to the characteristics of the problem. Furthermore, both a local search method and an aging-based re-initialization mechanism are developed to avoid premature convergence. Finally, the experimental results validate that the proposed algorithm is effective for the problem under study.

Keywords: flexible job shop; energy-aware scheduling; energy consumption; modified migrating birds optimization

1. Introduction

Nowadays, with the increasing emphasis on the environmental protection and sustainable development, manufacturing enterprises are facing not only economic pressure but also environmental challenges. It is very important to take some measurements to control energy consumption. Several different directions are being pursued by researchers in academia and industry area, which concentrate on the machine level, the product level and the management level. Due to the considerable investment, it may be only appropriate for large-size enterprises to purchase energy-saving machines and develop energy-efficient products in the perspective of the machine level and product level. Therefore, the existing research has mainly focused on the management level. Production scheduling is one of the most important factors in production management, which allocates limited resources to tasks in order to reach expected targets during the whole manufacturing process. In recent years, production scheduling has proved to be an effective way of reducing energy consumption [1–3]. There has been an increasing number of studies on energy-aware scheduling problems.

The flexible job shop scheduling problem (FJSP) is a well-known combinatorial optimization problem, which is extended from the classical job shop scheduling problem (JSP) [4,5]. Compared with the JSP, FJSP considers not only the operation permutation of each machine but also the machine

assignment to each operation, which makes it closer to practical production. Due to the essential complexity and the wide applications, FJSP has been paid a lot of attention by researchers at home and abroad. However, most of the published literature mainly concentrates on time-related metrics, such as makespan, earliness/tardiness, workload and flow time, while ignoring the indicators closely related to energy and environment. With the increasing promotion of sustainable manufacturing, energy-aware flexible job shop scheduling (EFJSP) has attracted the interest of scholars. Mokhtari and Hasani [6] presented a mathematical model to optimize the total completion time, the system availability and the total energy cost. For such a multi-objective model, an enhanced evolutionary algorithm was proposed to obtain the optimal scheduling solutions. Lei et al. [7] investigated a FJSP with the consideration of workload balance and energy consumption. A shuffled frog leaping algorithm was developed to get the trade-off between the two indicators. Wu and Sun [8] established a mathematical model to save energy in a flexible job shop by determining when to turn machines on/off and which speed level to select. A non-dominated sorted genetic algorithm was designed to solve this complicated problem. Wang et al. [9] developed a two-stage energy-saving optimization approach for the flexible job shop scheduling problem. Lei et al. [10] investigated a flexible job shop scheduling problem with the objective of minimizing makespan and total tardiness under an energy consumption threshold. A two-phase evolutionary algorithm was proposed based on an imperialist competitive algorithm and a variable neighborhood search algorithm. Meng et al. [11] addressed the flexible job shop scheduling problem with a criterion to reduce the total energy consumption. Six mixed-integer linear programming models were presented in the study, whose correctness and effectiveness were tested by using CPLEX solver through numerical experiments. Jiang and Deng [12] established a mathematical model for the energy-aware flexible job shop scheduling problem to minimize the energy cost and the earliness/tardiness cost. A bi-population based discrete cat swarm optimization algorithm was developed to solve the problem. Yin et al. [13] proposed a mathematical model for the flexible job shop with a variable machining speed to optimize productivity, energy efficiency and noise reduction. A multi-objective genetic algorithm was proposed to deal with the model. Song et al. [14] presented a mathematical model of a flexible job shop, considering energy consumption and preventive maintenance and proposed a non-dominated sorting genetic algorithm II (NSGA-II) to optimize the maximum completion time, the total processing energy cost and the total maintenance energy cost. Liu et al. [15] introduced an integrated green flexible job shop scheduling problem with the consideration of crane transportation. A hybrid algorithm, called as GA–GSO–GTHS, was developed based on a genetic algorithm (GA), a glowworm swarm optimization (GSO) algorithm and a green transport heuristic strategy (GTHS). Zhang et al. [16] built a new mathematical model of the flexible job shop under a time of use strategy to minimize the makespan and electricity consumption cost. A hybrid algorithm based on the biogeography-based optimization algorithm and variable neighborhood search was proposed to solve the bi-objective problem. Zhang et al. [17] studied an energy-saving flexible job shop scheduling problem to minimize the makespan and the total energy consumption. A modified shuffled frog-leaping algorithm (SFLA) was employed to solve the model. Lu et al. [18] proposed a discrete water wave optimization algorithm to solve an energy-conscious FJSP with various machining speeds.

With regards to the above literature, the relevant research on the energy-aware FJSP is still in the initial stage of exploration. Extensive work is yet to be carried out to narrow the gap between theoretical research and practical production. In the practical production, each machine can be operated by different workers belonging to an eligible worker set. In other words, for each machine, an appropriate worker needs to be chosen from the eligible worker set before processing jobs on it. In this case, the processing time of each job is dependent on both the assigned machine and the selected worker. Therefore, the worker can be also viewed as a kind of limited resource to be scheduled in the workshop. Such a problem is always categorized as a dual-resource constrained flexible job shop scheduling problem (DRCFJSP) [19]. However, in the previous research, the environmental metrics were not considered in the DRCFJSP. Nowadays, with increasing wage costs, the rational use of limited workers

has become more and more important. Therefore, the integration of energy-aware scheduling and dual-resource constraints in the FJSP, named the energy-aware dual-resource constrained flexible job shop scheduling problem (EDRCFJSP), is a new issue which deserves to be studied.

To the best of the authors' knowledge, there are few studies related to the EDRCFJSP that are more complex and that require more hard work to solve than the traditional FJSP [20]. Thus, an effective scheduling algorithm is highly desirable for the solving of the problem under study. In recent years, swarm intelligence algorithms have been developed and widely used for solving various optimization problems [21–28]. In this paper, a swarm intelligence algorithm, namely the migrating birds optimization (MBO), is introduced to deal with the addressed problem, which was originally proposed for quadratic assignment problems [29]. Since then, the MBO algorithm has been successfully applied to various optimization problems, such as the production scheduling problem [30], closed loop layout [31], knapsack problem [32], travelling salesman problem [33] and task allocation problem [34]. However, as far as the we know, there is no application of the MBO algorithm on the EDRCFJSP. In addition, the MBO is a kind of neighborhood search algorithm, which makes it easy to adopt for the production scheduling problem. Thus, according to the characteristics of the addressed problem, we present a modified MBO algorithm (MMBO) for solving the EDRCFJSP. Some effective technologies are included alongside the original MBO, such as population initialization, problem-based neighborhood structures, local search strategy and an age-based re-initialization mechanism. Experimental data demonstrate the effectiveness of the proposed algorithm for solving the EDRCFJSP.

The rest of this paper is organized as follows. In Section 2, mathematical model of EDRCFJSP is established. In Section 3, the proposed MMBO algorithm is described in detail. Section 4 shows the related experimental data and Section 5 provides conclusions and future works.

2. Mathematical Model of EDRCFJSP

2.1. Problem Description

In the workshop, n independent jobs are supposed to be processed on m machines with w workers. For each job i, J_i operations need to be processed following a given processing order, i.e., the precedence order between operations of the same job is fixed and known. Each operation must be processed by a machine selected from the operation's eligible machine set. In addition, each machine must be operated by a worker selected from the machine's eligible worker set. The processing time of each operation is dependent on the assigned machine and the selected worker. Moreover, different energy consumption may be needed by different machines and workers. Thus, the EDRCFJSP problem can be decomposed into three sub-problems: machine assignment (MA), worker selection (WS) and operation permutation (OP). The optimization goal is to minimize the total energy consumption in the workshop. Some assumptions should be considered as follows:

(1) Jobs, machines and workers are ready at the time zero;
(2) Each machine can process, at most, one operation at a time;
(3) Each worker can operate, at most, one machine at a time;
(4) For each operation, preemption is not permitted;
(5) Any two operations belonging to different jobs are independent of each other;
(6) A worker cannot be changed when he/she is processing jobs;
(7) Each machine cannot be completely turned off unless all jobs assigned to it are finished;
(8) The transportation times of jobs and moving times of the workers between different machines are ignored.

Before formulating the problem, some necessary symbols are listed as follows:
n: Number of jobs, $i = 1, 2, 3, \ldots, n$;
m: Number of machines, $k = 1, 2, 3, \ldots, m$;

w: Number of workers, $l = 1, 2, 3, \ldots, w$;
J_i: Number of operations in job i, $j = 1, 2, 3, \ldots, J_i$;
O_{ij}: The jth operation of job i;
F: The objective function;
E_1: The processing energy consumption;
E_2: The idle energy consumption;
E_3: The setup energy consumption;
E_4: The common energy consumption;
S_{ij}: Start time of O_{ij};
C_{ij}: Completion time of O_{ij};
p_{ijkl}: Processing time of O_{ij} on machine k operated by worker l;
PE_{ijkl}: Processing energy consumption coefficient of O_{ij} on machine k operated by worker l;
SE_k: Idle energy consumption coefficient of machine k when it is running in the idle state;
M: A big positive constant number;
S'_k: Start time of machine k;
C'_k: Completion time of machine k;
W_k: Workload of machine k, which equals the sum of the processing times of jobs assigned to machine k;
TST_k: Total setup time of machine k;
TU_k: Setup energy consumption coefficient of machine k;
CE: Common energy consumption coefficient;
$C_{\max}$: Final completion time (makspan) of the workshop;
$SU_{i'j'ijk}$: Setup time of machine k when O_{ij} is processed immediately after $O_{i'j'}$
y_{ijkl}: A binary variable, if O_{ij} is processed on machine k operated by worker l, $y_{ijkl} = 1$; otherwise, $y_{ijkl} = 0$;
$z_{iji'j'k}$: A binary variable, if O_{ij} is processed on machine k prior to $O_{i'j'}$, $z_{iji'j'k} = 1$; otherwise, $z_{iji'j'k} = 0$.

2.2. Energy Consumption Model

2.2.1. Processing Energy Consumption

The processing energy consumption (E_1) denotes the energy consumed by machines for processing operations, which can be formulated by Equation (1).

$$E_1 = \sum_{i=1}^{n} \sum_{j=1}^{J_i} \sum_{k=1}^{m} \sum_{l=1}^{w} PE_{ijkl} y_{ijkl} p_{ijkl} \tag{1}$$

2.2.2. Idle Energy Consumption

The idle energy consumption (E_2) represents the energy consumed by machines during the time interval between each pair of consecutive jobs, which can be represented by Equation (2).

$$E_2 = \sum_{k=1}^{m} SE_k (C_k - S_k - W_k) \tag{2}$$

2.2.3. Setup Energy Consumption

The energy consumption (E_3) defines the energy consumed by machines during the setup process for each pair of consecutive jobs assigned to the same machine.

$$E_3 = \sum_{k=1}^{m} TU_k TST_k \tag{3}$$

2.2.4. Common Energy Consumption

The common energy consumption (E_4) is the energy consumed for maintaining the daily operation of the workshop, such as lighting and air conditioning, which can be calculated by Equation (4).

$$E_4 = CE \times C_{\max} \tag{4}$$

2.2.5. Total Energy Consumption

The total energy consumption (F) is the sum of the processing energy consumption, the idle energy consumption, the setup energy consumption and the common energy consumption, which is shown in Equation (5).

$$F = E_1 + E_2 + E_3 + E_4 \tag{5}$$

2.3. Problem Modelling

In our previous work, an energy-aware flexible job shop scheduling problem is studied without the consideration of the dual-resource constraints [12]. Based on the existing model, a new mathematical model of the EDRCFJSP problem is modeled as below.

$$\min F = \sum_{i=1}^{n}\sum_{j=1}^{J_i}\sum_{k=1}^{m}\sum_{l=1}^{w} PE_{ijkl} y_{ijkl} p_{ijkl} + \sum_{k=1}^{m} SE_k (C_k - S_k - W_k) + \sum_{k=1}^{m} TU_k TST_k + CE \times C_{\max} \tag{6}$$

$$\text{s.t. } C_{ij} - S_{ij} = \sum_{k=1}^{m}\sum_{l=1}^{w} y_{ijkl} p_{ijkl},\ i = 1,2,\ldots,n;\ j = 1,2,\ldots,J_i \tag{7}$$

$$S_{i(j+1)} - C_{ij} \geq 0,\ i = 1,2,\ldots,n; j = 1,2,\ldots,J_i - 1 \tag{8}$$

$$S_{i'j'} + \mathrm{M}(1 - z_{iji'j'k}) \geq C_{ij} + SU_{iji'j'k},\ i,i' = 1,2,\ldots,n; j,j' = 1,2,\ldots,J_i; k = 1,2,\ldots,m \tag{9}$$

$$S_{ij} + Mz_{iji'j'k} \geq C_{i'j'} + SU_{i'j'ijk},\ i,i' = 1,2,\ldots,n; j,j' = 1,2,\ldots,J_i; k = 1,2,\ldots,m \tag{10}$$

$$\sum_{k=1}^{m}\sum_{l=1}^{w} y_{ijkl} = 1,\ i = 1,2,\ldots,n;\ j = 1,2,\ldots,J_i \tag{11}$$

$$W_k = \sum_{i=1}^{n}\sum_{j=1}^{J_i}\sum_{l=1}^{w} p_{ijkl} y_{ijkl},\ k = 1,2,\ldots,m \tag{12}$$

$$TST_k = \sum_{i=1}^{n}\sum_{j=1}^{J_i}\sum_{i'=1}^{n}\sum_{j'=1}^{J_{i'}} SU_{i'j'ijk} z_{i'j'ijk},\ k = 1,2,\ldots,m \tag{13}$$

$$C'_k = \max\left\{C_{ij} y_{ijkl}\right\},\ i = 1,2,\ldots,n; j = 1,2,\ldots,J_i;\ k = 1,2,\ldots,m; l = 1,2,\ldots,w \tag{14}$$

$$S'_k = \min\left\{S_{ij} y_{ijkl}\right\},\ i = 1,2,\ldots,n; j = 1,2,\ldots,J_i;\ k = 1,2,\ldots,m; l = 1,2,\ldots,w \tag{15}$$

$$y_{ijkl} \in \{0,1\},\ i = 1,2,\ldots,n; j = 1,2,\ldots,J_i; k = 1,2,\ldots,m;\ l = 1,2,\ldots,w \tag{16}$$

$$z_{iji'j'k} \in \{0,1\},\ i,i' = 1,2,\ldots,n; j,j' = 1,2,\ldots,J_i; k = 1,2,\ldots,m \quad (17)$$

Equation (6) is the objective function; Constraint (7) indicates that the processing of each operation must not be interrupted; Constraint (8) defines the precedence order between operations in a job; Constraints (9) and (10) ensure that each machine cannot process more than one operation at a time. For any two operations on the same machine, the processing of the operation cannot be started until its preceding operation is completed and the setup of the machine is finished; Constraint (11) represents that each operation is processed by one machine and the assigned machine is operated by one worker. Constraint (12) shows the workload of each machine; Constraint (13) gives the total setup time of each machine; Constraint (14) denotes the completion time of each machine; Constraint (15) defines the start time of each machine; Constraints (16) and (17) show the binary variables.

3. Modified Migrating Birds Optimization

A modified migrating birds optimization (MMBO) algorithm is implemented in this section for solving the problem. Firstly, a three-vector encoding method is employed to represent the scheduling solution. Secondly, a population initialization method is developed to generate the initial solutions. Thirdly, five neighborhood structures are presented to create neighborhood solutions according to the characteristics of the problem. In addition, a local search algorithm and an aging-based re-initialization mechanism are developed to avoid premature convergence.

3.1. The Basic MBO Algorithm

MBO is a swarm intelligence algorithm which originates from birds' migration behavior when they form a V-shaped formation [29]. In the MBO, each solution corresponds to a bird. In the flock, one bird is viewed as the leader and the others follow in a line on the right and left sides of the leader bird. The MBO algorithm consists of four steps: population initialization, improvement of the leader bird, improvement of the following birds and selection of a new leader bird.

First, a given number of birds are randomly generated and one bird is selected as the leader bird. In order to improve the leading solution, the MBO generates several neighboring solutions by exploring the leader bird's neighborhood. If the best solution among these neighbors is better than the leading solution, the leading solution is replaced by the best neighbor. Then, for each following bird, it evaluates its own neighboring solutions and a certain number of the best unused neighboring solutions of its previous solution in the line. The best solution will be the new solution if it is better than the current one. This updating process progresses from the leader towards the tail of the left or right lines. Once all solutions in the flock are considered, this updating process will be repeated. After several tours, the leader bird will be moved to one of the tails, and a solution behind it will take up the position of the leading solution. Then, another loop starts. The evolutionary procedure will not stop unless a termination condition is satisfied. The detailed steps of the basic MBO are shown as follows:

Step 1: Set the parameters of the MBO algorithm, such as the population size *popsize*, the number of neighboring solutions k', the number of shared neighboring solutions x, the number of tours G, the predefined lifespan ls, and the maximum iteration $K_{\max}$.

Step 2: Generate the initial population in a random manner.

Step 3: Set the iteration number $K \leftarrow 1$, the tour number $g \leftarrow 1$, the flag numer $flag \leftarrow 1$.

Step 4: Randomly generate k' neighboring solutions of the leader bird. Improve the leader solution and fill the shared neighboring sets S_{L} and S_{R}, each of which has x elements.

Step 5: For each solution π_{L} in the left line L, randomly generate $k' - x$ neighboring solutions. N_{L} represents the set of the $k' - x$ neighbors. The best solution in $N_{\mathrm{L}} \cup S_{\mathrm{L}}$ is used to replace the original solution. Empty S_{L} and refill it by using x best unused solutions in $N_{\mathrm{L}} \cup S_{\mathrm{L}}$.

Step 6: For each solution π_R in the right line R, randomly generate $k' - x$ neighboring solutions. N_R represents the set of the $k' - x$ neighbors. The best solution in $N_R \cup S_R$ is used to replace the original solution. Empty S_R and refill it by using x best unused solutions in $N_R \cup S_R$.

Step 7: Evaluate the fitness value of each individual and update the current best solution.

Step 8: Set $g \leftarrow g + 1$. If the number of tours G is met, go to Step 9; otherwise, go to Step 4.

Step 9: If $flag = 1$, move the leader to the end of L, and set the first solution of L as the new leader, and let $flag = 0$; otherwise, move the leader to the end of R, and set the first solution of R as the new leader, and let $flag = 1$.

Step 10: Set $K \leftarrow K + 1$ and check the terminate condition. If $K > K_{max}$ is not met, then set $g \leftarrow 1$, and go to Step 4; otherwise, go to Step 11.

Step 11: End the procedure.

3.2. Solution Encoding

To implement the application of the algorithm in solving a problem, one of the key tasks is to adopt an appropriate encoding method, which can represent the necessary information about the considered problem. The EDRCFJSP is made up of three sub-problems: machine assignment (MA), worker selection (WS) and operation permutation (OP). Therefore, an encoding method with three vectors is adopted to represent the scheduling solution, namely the MA vector, the WS vector and the OP vector, which represent the assignment of operations to machines, the selection of workers for machines and the processing sequence of operations on machines. The size of each vector is equal to the numbers of all operations.

Taking a $3 \times 3 \times 2$ (three jobs, three machines and two workers) problem for example, it is assumed that each job consists of three operations. Therefore, the length of each vector is equal to nine. The encoding method can be illustrated by Figure 1. For the OP vector, the operation-based encoding method is used. In this vector, each element with an integer value represents the job index. The elements with the same values represent different operations in the same job. For the MA vector, each element with an integer value denotes the assigned machine for an operation. For the WS vector, each element with an integer value indicates the selected worker for a machine. The elements in the WS and MA vectors are stored in a given order according to the identification number of the jobs and operations.

	O_{21}	O_{31}	O_{11}	O_{22}	O_{23}	O_{32}	O_{12}	O_{33}	O_{13}
OP	2	3	1	2	2	3	1	3	1
	O_{11}	O_{12}	O_{13}	O_{21}	O_{22}	O_{23}	O_{31}	O_{32}	O_{33}
MA	2	1	1	1	3	3	2	3	1
WS	1	2	2	2	1	1	2	2	2

Figure 1. A scheduling scheme for a $3 \times 3 \times 2$ energy-aware dual-resource constrained flexible job shop scheduling problem (EDRCFJSP) with three jobs, three machines and two workers.

3.3. Population Initialization

As a swarm intelligence algorithm, the quality of initial solutions is crucial for the performance of the algorithm. Here, a two-phase initialization method is proposed to obtain the initial population by some dispatching rules. In the first phase, two dispatching rules in [35] are employed to generate the OP vector. Most work remaining (MWR) means that the job with the maximal total processing time has the highest priority to be scheduled. Random rule (RR) means that the operation permutation is obtained in a random manner.

In the second phase, two rules are adopted to generate the MA and WS vectors: the modified assignment rule (MAR) and random rule (RR). The MAR is modified from assignment rule number one

in [35], whose pseudo-code is illustrated in Figure 2. The approach considers both the processing times and the workload of the machines. For each operation, the procedure includes finding the combination of the machine and the worker with the minimum processing time, fixing that assignment, and then adding this time to the entries in the columns with the selected machine. The RR means that the worker and the machine are randomly selected for the corresponding operation.

Create an array D with the size of $c \times b$ and two arrays A and B with the same length of c. b is the number of the combinations of machines and workers ($b=m \times w$), and c is the number of operations in the workshop ($c = \sum_{i=1}^{n} J_i$).
If (Operation c_1 can be processed by the combination b_1) **then** // $c_1 = 1, 2, \cdots, c$, $b_1 = 1, 2, \cdots, b$
  Set $D(c_1, b_1)$ to be the processing time of operation c_1 processed by the combination b_1
Else
  Set $D(c_1, b_1)$ to be a big number Λ
Endif
Set $i = 1$
While ($i \leq c$)
  Obtain the minimum value d in D
  If (The number of d is larger than 1) **then**
    Randomly select one of these elements
  End if
  Record the row c_1 and column b_1 of the selected element
  Set $A(c_1)$ to be the machine in the combination b_1
  Set $B(c_1)$ to be the worker in the combination b_1
  Set all the elements in the row c_1 to be Λ in order to avoid the operation being selected repeatedly
  Find out all the columns that have the same machine with the one in the combination b_1
  Add the processing time of operation c_1 machined by the combination b_1 to the selected columns
  $i = i + 1$
End while

Figure 2. Pseudo-code of the modified assignment rule (MAR).

3.4. Neighborhood Structure

In MBO, the solutions are updated by searching neighborhoods. Thus, six neighborhood structures are employed in this section to generate neighboring solutions. The first structure attempts to change the operation sequencing, and the second structure aims to change the machine assignment and the worker selection. The third structure is used to change the worker selection. The fourth and the sixth structures are the combination of other structures. In the algorithm, the neighborhood structures are randomly selected to generate the neighboring solutions.

NB_1: Randomly select two elements with different values from the OP vector, and then the selected elements are exchanged.

NB_2: Randomly select an element from the MA vector, and a different machine is randomly selected from the eligible machine set of the corresponding operation to replace the original one. Then a new worker is randomly selected from the eligible worker set of the selected machine.

NB_3: Randomly select an element from the WS vector and find out the corresponding machine for the operation. Then, a different worker is randomly selected from the eligible worker set of the corresponding machine to replace the original one.

NB_4: It is a combination of neighborhood structures NB_1 and NB_2. The neighboring solution is obtained by performing the two neighborhood structures simultaneously.

NB_5: It is a combination of neighborhood structures NB_1 and NB_3. The neighboring solution is obtained by performing the two neighborhood structures simultaneously.

NB_6: It is a combination of neighborhood structures NB_1, NB_2 and NB_3. The neighboring solution is obtained by performing the three neighborhood structures simultaneously.

3.5. Aging-Based Re-Initialization Mechanism

As the evolutionary process of the algorithm proceeds, the population may achieve a low diversity which makes the algorithm stall around a local optimum. To overcome this drawback, an aging-based re-initialization mechanism is adopted to increase the possibility of jumping out of the local optima and improving the search ability of the algorithm. In the aging-based re-initialization mechanism, the 'age' is used to describe the updating process of each individual bird. For a newly generated individual, the age is initially set to one. If there is no improvement after one iteration, the age will be increased by one. If it is larger than the predefined lifespan *ls*, the re-initialization procedure is invoked to reinitialize the individual following the initialization method in Section 3.3.

3.6. Local Search Strategy

To further enhance the search ability, the local search strategy is always embedded into various intelligence algorithms [36,37]. Here, the local search strategy starts from a given solution and stops when the predefined maximum number of iterations is met. The detailed steps of the local search are shown as follow:

Step 1: Obtain the initial solution π, and set $ct = 1$ and determine the maximum iteration number $\rho_{\max}$.

Step 2: Randomly select a neighborhood structure to obtain a new solution π'.

Step 3: If $TEC(\pi') < TEC(\pi)$, set $\pi = \pi'$.

Step 3: Set $ct = ct + 1$.

Step 4: Judge whether $ct > \rho_{\max}$ is satisfied. If yes, go to Step 5, otherwise, go to Step 2.

Step 5: End the procedure.

3.7. Procedure of the Proposed Algorithm

To implement the proposed MMBO, some items are designed according to the characteristics of the problem, such as encoding, population initialization, neighborhood structure, aging-based re-initialization and local search. Based on these items, the detailed steps of the proposed MMBO are shown as follows:

Step 1: Set the related parameters of the MMBO algorithm, such as the population size *popsize*, the number of neighboring solutions k', the number of shared neighboring solutions x, the number of tours G, the predefined lifespan *ls*, and the maximum iteration $K_{\max}$.

Step 2: Generate the initial population following the method in Section 3.3.

Step 3: Set the iteration number $K \leftarrow 1$, the tour number $g \leftarrow 1$, the flag numer $flag \leftarrow 1$.

Step 4: Randomly generate k' neighboring solutions of the leader bird. Improve the leader solution and fill the shared neighboring sets S_{L} and S_{R}, each of which has x elements.

Step 5: For each solution π_L in the left line L, randomly generate $k' - x$ neighboring solutions. N_L represents the set of the $k' - x$ neighbors. The best solution in $N_L \cup S_L$ is used to replace the original solution. Empty S_L and refill it by using x best unused solutions in $N_L \cup S_L$.

Step 6: For each solution π_R in the right line R, randomly generate $k' - x$ neighboring solutions. N_R represents the set of the $k' - x$ neighbors. The best solution in $N_R \cup S_R$ is used to replace the original solution. Empty S_R and refill it by using x best unused solutions in $N_R \cup S_R$.

Step 7: Update the current best solution and the age of each individual.

Step 8: Set $g \leftarrow g + 1$. If the number of tours G is met, go to Step 9; otherwise, go to Step 4.

Step 9: Check the age of each individual and perform the re-initialization mechanism when the age is larger than ls.

Step 10: Perform the local search to the current best individual.

Step 11: If $flag = 1$, move the leader to the end of L, and set the first solution of L as the new leader, and let $flag = 0$; otherwise, move the leader to the end of R, and set the first solution of R as the new leader, and let $flag = 1$.

Step 12: Set $K \leftarrow K + 1$ and check the terminate condition. If $K > K_{max}$ is not met, then set $g \leftarrow 1$, and go to Step 4; otherwise, go to Step 13.

Step 13: End the procedure.

4. Computational Results and Discussion

This section reports the computational results to evaluate the performance of the proposed algorithm. All experiments are implemented by FORTRAN language and run on VMware Workstation with 2GB main memory under WinXP. To this end, some testing data need to be generated. Here, a set of instances with the number of jobs $n \in \{10, 20, 30, 50, 80\}$ and the number of machines $m \in \{10, 15, 20, 25\}$ are considered. Twenty instances are generated for each combination of n and m. In addition, some other parameters are randomly generated in the given range following a discrete uniform distribution in Table 1. For each instance, ten independent replications are conducted to get statistical results.

Table 1. Some parameters for the EDRCFJSP.

J_i	nop	w	nwk	p_{ijkl}	E_{ijkl}	SE_k	TU_l	ST_{hzijl}	CE
[1, 5]	[2, m]	$\lceil m \times 0.6 \rceil$	[2, w]	[15, 30]	[10, 20]	[6, 12]	[5, 10]	[1, 3]	[12, 20]

nop represents the number of eligible machines for each operation, nwk denotes the number of eligible workers for each machine.

4.1. Effectiveness of the Improvement Strategy

In this paper, three improvement strategies are adopted to enhance the performance of the proposed algorithm, such as population initialization method, aging-based re-initialization mechanism and local search strategy. Here, we first test the effectiveness of the three improvement strategies. In Table 2, the first column shows the names of different instances, other columns report the computational data. 'MMBO' is our proposed algorithm. 'MBO1' is the algorithm where the random rule is only used to generate the initial population. 'MBO2' represents the algorithm where the aging-based re-initialization mechanism is excluded from the MMBO. 'MBO3' is the algorithm where the local search strategy is excluded from the MMBO. In the table, 'Best' represents the best value in the ten runs. 'Avg.' denotes the average result in the ten runs. The average relative percent deviation (ARPD) is measured by Equation (18).

$$ARPD = \sum_{nr=1}^{NR} \frac{100 \times (Alg_{nr} - Min)}{Min} / NR \quad (18)$$

where 'Min' is the minimum value obtained by all compared algorithms. 'Time' is the average time (in seconds) of the ten runs. For each instance, boldface represents the best value obtained by all compared algorithms.

Table 2. Effectiveness analysis of improvement strategies.

Instance	$m \times n \times w$	MMBO				MBO1			
		Best	**Avg.**	**ARPD**	**Time**	**Best**	**Avg.**	**ARPD**	**Time**
RM01	10 × 10 × 6	9588	9865.5	3.53	23.8	9696	10,089.8	5.89	24.0
RM02	10 × 20 × 6	**22,347**	**22,970.0**	**2.79**	66.7	24,080	24,336.4	8.90	64.3
RM03	10 × 30 × 6	33,351	34,018.9	2.88	130.5	34,999	35,830.6	8.36	121.5
RM04	10 × 50 × 6	57,298	58,542.5	2.48	303.5	62,475	63,914.8	11.89	279.2
RM05	10 × 80 × 6	**113,813**	**116,054.4**	**1.97**	667.5	126,105	129,280.6	13.59	658.6
RM06	15 × 10 × 9	**8324**	**8598.8**	**3.30**	30.5	8421	8708.4	4.62	29.0
RM07	15 × 20 × 9	**18,334**	**19,149.9**	**4.45**	84.7	19,437	19,767.8	7.82	72.5
RM08	15 × 30 × 9	29,063	29,906.4	2.98	150.7	30,338	31,387.4	8.08	130.6
RM09	15 × 50 × 9	54,172	55,361.1	2.51	355.9	58,186	58,710.6	8.71	300.1
RM10	15 × 80 × 9	**103,491**	**104,707.1**	**1.18**	820.9	115,624	117,196.6	13.24	729.4
RM11	20 × 10 × 12	**8259**	8457.5	2.40	37.4	8370	8493.4	2.84	30.0
RM12	20 × 20 × 12	**17,736**	18,447.0	4.01	103.0	18,863	19106.0	7.72	78.0
RM13	20 × 30 × 12	**27,732**	28,833.5	3.97	194.9	28,961	29,667.2	6.98	146.5
RM14	20 × 50 × 12	**52,213**	**53,483.0**	**2.43**	426.4	54,483	55,211.8	5.74	344.6
RM15	20 × 80 × 12	**98,071**	100,441.3	2.42	944.4	107,788	110,569.8	12.74	758.8
RM16	25 × 10 × 15	7704	7828.3	2.12	42.4	**7666**	**7782.4**	**1.52**	32.7
RM17	25 × 20 × 15	**16,621**	**16,964.3**	**2.07**	114.3	16,888	17,551.4	5.60	90.3
RM18	25 × 30 × 15	25,708	26,074.5	2.83	203.6	26,442	26,712.8	5.35	162.1
RM19	25 × 50 × 15	49,020	50,020.3	2.89	436.2	50,874	52,262.4	7.50	374.5
RM20	25 × 80 × 15	**96,029**	**97,181.2**	**1.20**	1099.2	105,785	106,591.2	11.00	823.2
Instance	$m \times n \times w$	**MBO2**				**MBO3**			
		Best	**Avg.**	**ARPD**	**Time**	**Best**	**Avg.**	**ARPD**	**Time**
RM01	10 × 10 × 6	**9529**	**9850.8**	**3.38**	23.5	9706	9918.8	4.09	27.4
RM02	10 × 20 × 6	22,627	23,807.2	6.53	64.5	22,868	23,185.2	3.75	74.0
RM03	10 × 30 × 6	**33,065**	**33,810.7**	**2.26**	117.2	33,583	33,970.8	2.74	132.0
RM04	10 × 50 × 6	**57,123**	**58,310.0**	**2.08**	276.0	57,971	58,585.2	2.56	304.3
RM05	10 × 80 × 6	115,800	117,652.8	3.37	632.8	115,076	117,048.8	2.84	690.7
RM06	15 × 10 × 9	8470	8633.7	3.72	25.9	8407	8603.4	3.36	30.2
RM07	15 × 20 × 9	19,035	19,387.7	5.75	74.2	18,868	19,362.2	5.61	86.5
RM08	15 × 30 × 9	**29,042**	**29,796.7**	**2.60**	122.9	29,379	29,854.4	2.80	152.7
RM09	15 × 50 × 9	**54,007**	**55,111.9**	**2.05**	293.8	54,253	55,768.8	3.26	341.0
RM10	15 × 80 × 9	103,888	104,957.7	1.42	719.6	104,286	105,955.2	2.38	780.8
RM11	20 × 10 × 12	8327	8489.9	2.80	28.7	8287	**8404.6**	**1.76**	35.5
RM12	20 × 20 × 12	17,864	18,519.7	4.42	85.3	17,803	**18,283.8**	**3.09**	102.5
RM13	20 × 30 × 12	27,916	**28,451.3**	**2.59**	147.1	28,247	28,491.6	2.74	165.1
RM14	20 × 50 × 12	53,014	53,582.4	2.62	338.2	52,355	53,707.6	2.86	419.2
RM15	20 × 80 × 12	98,552	100,016.5	1.98	788.4	98,248	**99,705.2**	**1.67**	859.5
RM16	25 × 10 × 15	7672	7796.3	1.70	29.1	**7666**	7816.0	1.96	42.0
RM17	25 × 20 × 15	16,770	16,994.2	2.25	79.8	16,685	17,266.4	3.88	109.3
RM18	25 × 30 × 15	25,478	25,818.9	1.83	140.4	**25,356**	**25,763.2**	**1.61**	193.3
RM19	25 × 50 × 15	**48,616**	**49,959.3**	**2.76**	319.1	49,870	50,074.0	3.00	466.1
RM20	25 × 80 × 15	97,474	98,483.8	2.56	777.5	96,966	97,951.4	2.00	1037.6

For the MMBO, the parameters are set according to the recommendations by Duman et al. [13], which are shown as follows: the population size *popsize* = 51; the number of neighboring solutions k' = 3; the number of the shared neighboring solutions $x = 1$; and the number of tours G = 10. In addition, we set ls = 50, $\rho_{\max}$ = 10 and the maximum iteration $K_{\max} = 500$. To be fair, MBO1, MBO2 and MBO3 are set with the same parameters. According to Table 2, it can be observed that: (1) the MMBO algorithm obtains the 12 best values in comparison with the best value and outperforms other compared algorithms. The second-best algorithm, namely MBO2, can obtain the six best values; (2) the MMBO algorithm yields the eight best values in comparisons with both the average value and the ARPD value, which performs better than other algorithms. The second-best algorithm, namely MBO2, can obtain the seven best values in comparisons with both the average value and the ARPD value; (3) in comparisons with the 'time' value, the MMBO has a longer computational time than other algorithms due to the introduction of the improvement strategies.

To test whether the differences from the algorithms in Table 2 are significant or not, an analysis of variance (ANOVA) is conducted in Table 3, where all the algorithms are viewed as the factors. The results demonstrate that there is a statistically significant difference between the compared algorithms since the p value is smaller than 0.05.

Table 3. Analysis of variance (ANOVA) for the average relative percent deviation (ARPD) of the compared algorithms in Table 2.

Source	DF	Sum of Squares	Mean Square	F Value	p Value
Factor	3	383.63937	127.87979	35.40111	2.0095×10^{-14}
Error	76	274.53557	3.61231		
Total	79	658.17494			

4.2. *Effectiveness of the Proposed MMBO*

To test the effectiveness of the MMBO algorithm, it is compared with three existing algorithms, named the variable neighborhood structure (VNS) [19] and the improved whale optimization algorithm (IWOA) [38]. The VNS was developed to solve the DRCFJSP and can be directly employed to deal with the problem under study. The IWOA was developed for the energy-efficient job shop scheduling problem. The scheduling solution representation and the individual position vector are used to adapt the IWOA to the considered problem. The parameters of the compared algorithm are set as follows: For the VNS, the parameters are set as those in [19], i.e., $\theta = 0.5$, t_{rst} = 35,000 and $iter_{\max}$ = 65,000. For the IWOA, the population size is 50, and the maximum iteration is 2000, which are the same as those of the MMBO algorithm. To obtain the computational results of the VNS and the IWOA, ten independent replications are conducted with these two algorithms for each instance. As seen from Table 4, the MMBO has the longest computational time, but it can yield 20 values in comparison with the Best, Avg. and ARPD values. Figures 3 and 4 show that the proposed MMBO algorithm has a good convergence property.

Table 4. Comparison results of different algorithms.

Instance	$m \times n \times w$	MMBO				VNS				IWOA			
		Best	**Avg.**	**ARPD**	**Time**	**Best**	**Avg.**	**ARPD**	**Time**	**Best**	**Avg.**	**ARPD**	**Time**
RM01	10 × 10 × 6	**9588**	**9865.5**	**2.89**	23.8	9839	10,244.8	6.85	7.3	11,089	11,307.0	17.93	8.1
RM02	10 × 20 × 6	**22,347**	**22,970.0**	**2.79**	66.7	24,096	25,186	12.70	10.3	29,258	30,355.6	35.84	21.4
RM03	10 × 30 × 6	**33,351**	**34,018.9**	**2.00**	130.5	36,103	37,296.4	11.83	14.7	41,631	44,732.8	34.13	39.7
RM04	10 × 50 × 6	**57,298**	**58,542.5**	**2.17**	303.5	63,475	64,937.2	13.33	27.6	80,901	85,493.2	49.21	91.9
RM05	10 × 80 × 6	**113,813**	**116,054.4**	**1.97**	667.5	127,773	131,051.2	15.15	61.9	183,505	186,510.6	63.87	212.9
RM06	15 × 10 × 9	**8324**	**8598.8**	**3.30**	30.5	8822	8985.2	7.94	8.5	9851	10,147.8	21.91	8.5
RM07	15 × 20 × 9	**18,334**	**19,149.9**	**4.45**	84.7	21,079	21,417.8	16.82	11.4	23,143	24,873.4	35.67	23.3
RM08	15 × 30 × 9	**29,063**	**29,906.4**	**2.90**	150.7	31,382	32,548.0	11.99	15.6	40,030	41,550.4	42.97	41.1
RM09	15 × 50 × 9	**54,172**	**55,361.1**	**2.20**	355.9	59,499	61,390	13.32	28.0	77,961	82,297.4	51.92	100.0
RM10	15 × 80 × 9	**103,491**	**104,707.1**	**1.18**	820.9	118,048	121,022.6	16.94	65.6	167,560	172,486	66.67	213.8
RM11	20 × 10 × 12	**8259**	**8457.5**	**2.40**	37.4	8729	8939.0	8.23	7.6	9314	9851.8	19.29	8.9
RM12	20 × 20 × 12	**17,736**	**18,447.0**	**4.01**	103.0	19,958	20,384.6	14.93	11.0	24,348	25,053.4	41.26	24.9
RM13	20 × 30 × 12	**27,732**	**28,833.5**	**3.97**	194.9	30,359	31,759.2	14.52	15.6	38,457	40,512.6	46.09	43.1
RM14	20 × 50 × 12	**52,213**	**53,483.0**	**2.43**	426.4	58,905	59,957	14.83	31.2	81,699	85,436.0	63.63	105.7
RM15	20 × 80 × 12	**98,071**	**100,441.3**	**2.42**	944.4	110,985	113,107.0	15.33	67.4	155,883	168,369.8	71.68	237.1
RM16	25 × 10 × 15	**7704**	**7828.3**	**1.61**	42.4	8117	8359.0	8.50	8.0	8976	9318.2	20.95	9.6
RM17	25 × 20 × 15	**16,621**	**16,964.3**	**2.07**	114.3	18,304	19,131.0	15.10	12.4	23,126	23,744.2	42.86	25.8
RM18	25 × 30 × 15	**25,708**	**26,074.5**	**1.43**	203.6	29,204	29,598.0	15.13	16.6	37,902	38,599.6	50.15	47.2
RM19	25 × 50 × 15	**49,020**	**50,020.3**	**2.04**	436.2	55,360	56,950.6	16.18	31.3	76,848	80,789.4	64.81	106.6
RM20	25 × 80 × 15	**96,029**	**97,181.2**	**1.20**	1099.2	107,528	111,204.6	15.80	71.7	161,517	170,567.4	77.62	258.5

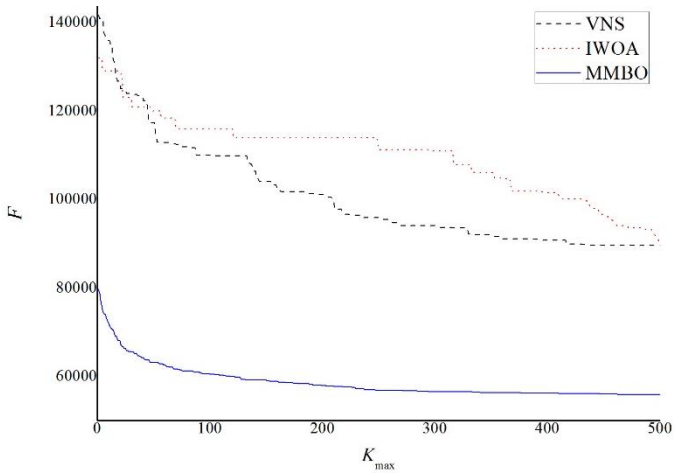

Figure 3. Convergence curve of the compared algorithms in Instance RM09.

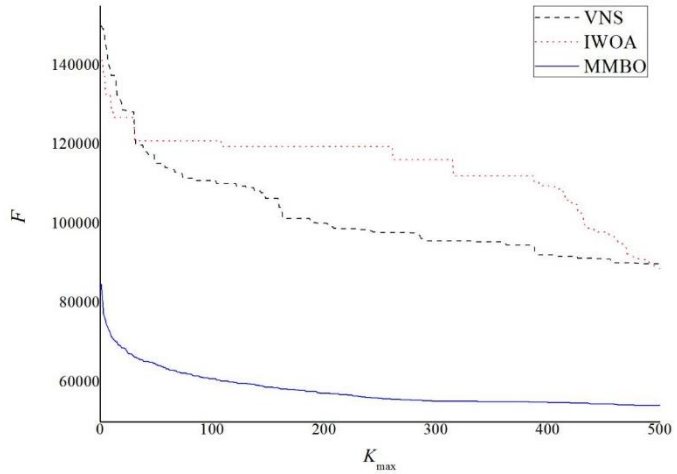

Figure 4. Convergence curve of the compared algorithms in Instance RM14.

To test whether the differences from the algorithms in Table 4 are significant or not, an analysis of variance (ANOVA) is conducted in Table 5, where all the algorithms are viewed as the factors. The results demonstrate that there is a statistically significant difference between the compared algorithms since the p value is smaller than 0.05.

Table 5. ANOVA for ARPD of the compared algorithms in Table 4.

Source	DF	Sum of Squares	Mean Square	F Value	p Value
Factor	2	1555.056	777.528	115.41936	0
Error	57	383.98322	6.73655		
Total	59	1939.03922			

5. Conclusions

This study investigated an energy-aware dual-resource constrained flexible job shop scheduling problem (EDRCFJSP). The energy consumption model is first built to represent the energy consumption in the workshop. A mathematical model is subsequently established to optimize the total energy consumption. To deal with the problem, a modified migrating birds optimization algorithm (MMBO) is proposed according to the characteristics of the considered problem. Extensive experiments are conducted to demonstrate the effectiveness of the MMBO algorithm. As seen from the comparison results, the proposed improvement strategies can effectively enhance the search ability of the algorithm.

In addition, the proposed MMBO algorithm outperforms the existing algorithms in comparison with three computational indicators, which shows the effectiveness of the algorithm in solving the problem under study. However, compared with the existing algorithms, MMBO has a longer computational time.

In a future work, the EDRCFJSP will be further studied to make it more closely resemble realistic production. Some realistic constraints will be considered, such as controllable machining speeds, time of use (TOU) electricity price policy, job deterioration effects and transportation constraints. In addition, some uncertain events should be considered, such as machine breakdown and the arrival of new jobs. Furthermore, some efforts are necessary in the design of more efficient algorithms.

Author Contributions: Methodology, H.L. and H.Z.; writing—original draft, H.L. and T.J.; Writing—review & editing, H.Z.; Funding acquisition, T.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Shandong Provincial Natural Science Foundation, grant number ZR2016GP02 and the Project of Shandong Province Higher Educational Science and Technology Program, grant number J17KA199.

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

References

1. Jiang, T.; Zhang, C.; Zhu, H.; Deng, G. Energy-efficient scheduling for a job shop using grey wolf optimization algorithm with double-searching mode. *Math. Probl. Eng.* **2018**, *2018*. [CrossRef]
2. Jiang, T.; Zhang, C.; Sun, Q.M. Green job shop scheduling problem with discrete whale optimization algorithm. *IEEE Access* **2019**, *7*, 43153–43166.
3. Lu, Y.; Jiang, T. Bi-population based discrete bat algorithm for the low-carbon job shop scheduling problem. *IEEE Access* **2019**, *7*, 14513–14522.
4. Jiang, T.; Zhang, C. Application of grey wolf optimization for solving combinatorial problems: Job shop and flexible job shop scheduling cases. *IEEE Access* **2018**, *6*, 26231–26240.
5. Jiang, T.; Zhang, C. Adaptive discrete cat swarm optimisation algorithm for the flexible job shop problem. *Int. J. Bio-Inspired Comput.* **2019**, *13*, 199–208.
6. Mokhtari, H.; Hasani, A. An energy-efficient multi-objective optimization for flexible job-shop scheduling problem. *Comput. Chem. Eng.* **2017**, *104*, 339–352.
7. Lei, D.; Zheng, Y.; Guo, X. A shuffled frog-leaping algorithm for flexible job shop scheduling with the consideration of energy consumption. *Int. J. Prod. Res.* **2017**, *55*, 3126–3140.
8. Wu, X.; Sun, Y. A green scheduling algorithm for flexible job shop with energy-saving measures. *J. Clean. Prod.* **2018**, *172*, 3249–3264.
9. Wang, H.; Jiang, Z.; Wang, Y.; Zhang, H.; Wang, Y.-H. A two-stage optimization method for energy-saving flexible job-shop scheduling based on energy dynamic characterization. *J. Clean. Prod.* **2018**, *188*, 575–588.
10. Lei, D.; Li, M.; Wang, L. A two-phase meta-heuristic for multiobjective flexible job shop scheduling problem with total energy consumption threshold. *IEEE Trans. Cybern.* **2018**, *49*, 1097–1109.
11. Meng, L.; Zhang, C.; Shao, X.; Ren, Y. MILP models for energy-aware flexible job shop scheduling problem. *J. Clean. Prod.* **2019**, *210*, 710–723.
12. Jiang, T.; Deng, G. Optimizing the low-carbon flexible job shop scheduling problem considering energy consumption. *IEEE Access* **2018**, *6*, 46346–46355.
13. Yin, L.; Li, X.; Gao, L.; Lu, C.; Zhang, Z. A novel mathematical model and multi-objective method for the low-carbon flexible job shop scheduling problem. *Sustain. Comput.* **2017**, *13*, 15–30.
14. Song, W.J.; Zhang, C.Y.; Lin, W.W.; Shao, X.Y. Flexible job-shop scheduling problem with maintenance activities considering energy consumption. *Appl. Mech. Mater.* **2014**, *521*, 707–713.
15. Liu, Z.; Guo, S.; Wang, L. Integrated green scheduling optimization of flexible job shop and crane transportation considering comprehensive energy consumption. *J. Clean. Prod.* **2019**, *211*, 765–786.
16. Zhang, H.; Dai, Z.; Zhang, W.; Zhang, S.; Wang, Y.; Liu, R. A new Energy-Aware flexible job shop scheduling method using modified Biogeography-Based optimization. *Math. Probl. Eng.* **2017**, *2017*, 7249876.
17. Zhang, X.; Ji, Z.; Wang, Y. An improved SFLA for flexible job shop scheduling problem considering energy consumption. *Mod. Phys. Lett. B* **2018**, *32*, 1840112.

18. Lu, Y.; Lu, J.; Jiang, T. Energy-conscious scheduling problem in a flexible job shop using a discrete water wave optimization algorithm. *IEEE Access* **2019**, *7*, 101561–101574.
19. Lei, D.; Guo, X. Variable neighbourhood search for dual-resource constrained flexible job shop scheduling. *Int. J. Prod. Res.* **2014**, *52*, 2519–2529.
20. Meng, L.; Zhang, C.; Zhang, B.; Ren, Y. Mathematical modeling and optimization of energy-conscious flexible job shop scheduling problem with worker flexibility. *IEEE Access* **2019**, *7*, 68043–68059.
21. Kalra, M.; Singh, S. A review of metaheuristic scheduling techniques in cloud computing. *Egypt. Inform. J.* **2015**, *16*, 275–295.
22. Strumberger, I.; Bacanin, N.; Tuba, M.; Tuba, E. Resource Scheduling in Cloud Computing Based on a Hybridized Whale Optimization Algorithm. *Appl. Sci.* **2019**, *9*, 4893.
23. Sreenu, K.; Sreelatha, M. W-Scheduler: Whale optimization for task scheduling in cloud computing. *Clust. Comput.* **2019**, *22*, 1087–1098.
24. Strumberger, I.; Minovic, M.; Tuba, M.; Bacanin, N. Performance of elephant herding optimization and tree growth algorithm adapted for node localization in wireless sensor networks. *Sensors* **2019**, *19*, 2515.
25. Yang, X.S. Swarm intelligence based algorithms: A critical analysis. *Evol. Intell.* **2014**, *7*, 17–28.
26. Suganuma, M.; Shirakawa, S.; Nagao, T. A genetic programming approach to designing convolutional neural network architectures. In Proceedings of the Genetic and Evolutionary Computation Conference, Berlin, Germany, 15–19 July 2017; pp. 497–504.
27. Tuba, M.; Bacanin, N. Improved seeker optimization algorithm hybridized with firefly algorithm for constrained optimization problems. *Neurocomputing* **2014**, *143*, 197–207.
28. Strumberger, I.; Tuba, E.; Bacanin, N. Dynamic tree growth algorithm for load scheduling in cloud environments. In Proceedings of the 2019 IEEE Congress on Evolutionary Computation (CEC), Wellington, New Zealand, 10–13 June 2019; pp. 65–72.
29. Duman, E.; Uysal, M.; Alkaya, A.F. Migrating Birds Optimization: A new metaheuristic approach and its performance on quadratic assignment problem. *Inf. Sci.* **2012**, *217*, 65–77.
30. Meng, T.; Pan, Q.K.; Li, J.Q.; Tuba, M. An improved migrating birds optimization for an integrated lot-streaming flow shop scheduling problem. *Swarm Evol. Comput.* **2018**, *38*, 64–78.
31. Niroomand, S.; Hadi-Vencheh, A.; Şahin, R.; Vizvári, B. Modified migrating birds optimization algorithm for closed loop layout with exact distances in flexible manufacturing systems. *Expert Syst. Appl.* **2015**, *42*, 6586–6597.
32. Ulker, E.; Tongur, V. Migrating birds optimization (MBO) algorithm to solve knapsack problem. *Procedia Comput. Sci.* **2017**, *111*, 71–76.
33. Tongur, V.; Ülker, E. The analysis of migrating birds optimization algorithm with neighborhood operator on traveling salesman problem. In *Intelligent and Evolutionary Systems*; Springer: Cham, Switzerland, 2016; pp. 227–237.
34. Oz, D. An improvement on the Migrating Birds Optimization with a problem-specific neighboring function for the multi-objective task allocation problem. *Expert Syst. Appl.* **2017**, *67*, 304–311.
35. Pezzella, F.; Morganti, G.; Ciaschetti, G. A genetic algorithm for the flexible job-shop scheduling problem. *Comput. Oper. Res.* **2008**, *35*, 3202–3212.
36. Jovanovic, R.; Voß, S. Fixed Set Search Applied to the Minimum Weighted Vertex Cover Problem. In *International Symposium on Experimental Algorithms*; Springer: Cham, Switzerland, 2019; pp. 490–504.
37. Jovanovic, R.; Tuba, M.; Voß, S. Fixed set search applied to the traveling salesman problem. In *International Workshop on Hybrid Metaheuristics*; Springer: Cham, Switzerland, 2019; pp. 63–77.
38. Jiang, T.; Zhang, C.; Zhu, H.; Deng, G. Energy-efficient scheduling for a job shop using an improved whale optimization algorithm. *Mathematics* **2018**, *6*, 220–239.