Received June 23, 2018, accepted August 15, 2018, date of publication August 20, 2018, date of current version September 7, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2866133

Optimizing the Low-Carbon Flexible Job Shop Scheduling Problem Considering Energy Consumption

TIANHUA JIANG[1] AND GUANLONG DENG[2]
[1]School of Transportation, Ludong University, Yantai 264025, China
[2]School of Information and Electrical Engineering, Ludong University, Yantai 264025, China

Corresponding author: Tianhua Jiang (jth1127@163.com)

This work was supported in part by the Training Foundation of Shandong Natural Science Foundation of China under Grant ZR2016GP02 and Grant ZR2017PEE025, in part by the National Natural Science Foundation Project of China under Grant 61403180, in part by the Project of Shandong Province Higher Educational Science and Technology Program under Grant J17KA199, in part by the Special Research and Promotion Program of Henan Province under Grant 182102210257, and in part by the Project of Henan Province Higher Educational Key Research Program under Grant 16A120011.

ABSTRACT Flexible job shop scheduling problem (FJSP) is a typical discrete combinatorial optimization problem, which can be viewed as an extended version of the classical job shop scheduling problem. In previous researches, the scheduling problem has historically emphasized the production efficiency. Recently, scheduling problems with green criterion have been paid great attention by researchers. In this paper, the mathematical model of the low-carbon flexible job shop scheduling problem is established with the objective of minimizing the sum of the energy consumption cost and the earliness/tardiness cost. For solving the model, a kind of bi-population based discrete cat swarm optimization algorithm (BDCSO) is presented to obtain the optimal scheduling scheme in the workshop. In the framework of the BDCSO, two sub-populations are used to adjust the machine assignment and operation sequence respectively. At the initialization stage, a two-component discrete encoding mechanism is first employed to represent each individual, and then a heuristic method is adopted to generate the initial solutions with good quality and diversity. By considering the discrete characteristics of the scheduling problem, the modified updating methods are developed for the seeking and tracing modes to ensure the algorithm work directly in a discrete domain. To coordinate the global and local search in each sub-population, six adjustment curves are used to change the number of cats in the seeking and tracing modes, based on which six algorithms are developed, i.e., LBDCSO, SinBDCSO, CosBDCSO, TanBDCSO, LnBDCSO, and SquareBDCSO. In addition, the information exchanging strategy is introduced to implement the cooperation of the two sub-populations. Finally, extensive simulation based on random instances and benchmark instances is carried out. The comparisons results demonstrate the effectiveness of the proposed algorithms in solving the FJSP under study.

INDEX TERMS Flexible job shop, low-carbon production scheduling, energy consumption, bi-population based discrete cat swarm optimization algorithm.

I. INTRODUCTION

Nowadays, energy consumption has become a major issue in the world. The manufacturing industry has been viewed as an intensive energy consumer, which is responsible for about one-third of the total energy consumption. Therefore, many companies are urged or even forced to adopt effective energy-saving measures to control the energy consumption both for economic and environmental reasons. In the manufacturing field, production scheduling is crucial for the manufacturing performance, by which available resources in the workshop can be properly allocated to tasks. In previous researches, the scheduling problem has historically emphasized the production efficiency, such as makespan, tardiness, lateness, etc. However, green metrics were seldomly considered, such as energy consumption, CO_2 emission, etc. Until recent ten years, some researchers have realized the importance of scheduling policies in reducing the energy consumption. Then some environmental metrics are considered alongside traditional production indicators in the energy-conscious optimization scheduling problems.

Mouzon and Yildirim [1] proposed a greedy randomised multi-objective adaptive search algorithm to minimize total energy consumption and total tardiness on a single machine. Yildirim and Mouzon [2] developed a multi-objective genetic algorithm to minimize the energy consumption and the completion time of a single machine system. Shrouf *et al.* [3] adopted a genetic algorithm to optimize the energy consumption costs on a single machine by considering variable energy prices. Che *et al.* [4] considered an energy-conscious single machine scheduling problem under time-of-use (TOU) strategy, in which electricity prices varies during a day. Dai *et al.* [5] built an energy-efficient scheduling model in a flexible flow shop and proposed a genetic-simulated annealing algorithm to make a trade-off between makespan and energy consumption. Ding *et al.* [6] addressed a carbon-efficient scheduling problem in a permutation flow shop with the criterion to minimize the total carbon emission and makespan. Mansouri and Aktas [7] developed multi-objective genetic algorithms for a two-machine flow shop scheduling problem to make a trade-off between energy consumption and makespan. Luo *et al.* [8] proposed an any colony optimization algorithm to minimize the production efficiency and electric power cost under the time-of-use policy. Li *et al.* [9] developed an energy-aware multi-objective optimization algorithm for solving the hybrid flow shop scheduling problem by considering the setup energy consumptions. Regarding the reviewed literature, most of the energy-conscious optimization scheduling researches concentrate on the simple system, e.g., single machine or flow shop. Therefore, further researches are needed to be carried out on this issue, and more complex manufacturing system should be considered.

In the real-life production, many problems can be viewed as a job-shop scheduling problem (JSP), such as workshop scheduling in the industry, departure and arrival times of logistic problems, the delivery times of orders in a company, etc. [10]. As the extended version of the JSP, the flexible job shop problem (FJSP) provides a more closer approximation to the actual production. Compared with the JSP, the complexity of the FJSP lies on the addition of the machine selection for operations. In this paper, the manufacturing system of a flexible job shop type is selected as the research object to study the energy consumption optimization scheduling problem. At present, there are some literature reported by researchers about this type of problem. Jiang *et al.* [11] considered a multi-objective flexible job shop scheduling problem and proposed a modified non-dominated sorting genetic algorithm to minimize the makespan, processing cost, energy consumption and cost-weighted processing quality. Zhang *et al.* [12] presented a model of low-carbon flexible job shop scheduling problem and developed a hybrid non-dominated sorting genetic algorithm II to optimize both production factors (makespan and machine workload) and environmental influence (carbon emission). Lei *et al.* [13] considered a FJSP and proposed a shuffled frog-leaping algorithm to minimize the workload balance and energy consumption. Yin *et al.* [14] proposed a new low-carbon mathematical model to optimize productivity, energy efficiency and noise reduction in the flexible job shop. A multi-objective genetic algorithm based on a simplex lattice design is develpoed to solve it. Piroozfard *et al.* [15] considered a multi-objective flexible job shop scheduling problem to minimize total carbon footprint and total late work criterion and presented an improved multi-objective genetic algorithm for solving it. Mokhtari and Hasani [16] designed an energy-efficient scheduling in a flexible job shop and proposed an enhanced evolutionary algorithm to optimize total completion time, total availability of the system and total energy cost. For a energy-conscious optimization scheduling problem, the introduction of energy consumption increases the number of variables and constraints, which makes the flexible job shop scheduling problem more complex than the original one. It is well-known that meta-heuristics is effective for solving various optimization problems [17], [18]. Especially for production scheduling problems, the application of meta-heuristic algorithm has been the research hot spot in the manufacturing filed.

For a meta-heuristic algorithm, the balance between exploration and exploitation is very important for its searching ability. As a bio-inspired intelligent algorithm, cat swarm optimization (CSO) was originally developed by Chu and Tsai [19], which has been used to solve different optimization problems [20]–[23]. The main characteristics of the algorithm is that it consists of two types of searching modes (seeking and tracing) corresonding to the global and local seach respectively. In the algorith, the global search and local search can be conducted simultaneously during the evolutionary process, which provides an opportunity to implement the balance between the exploration and exploitation abilities. Therefore, we attempt to design an effective algorithm for solving the low-carbon FJSP based on the searching mechnism of CSO. However, in the original CSO, the evolutionary process is conducted based on the continuous updating of individual positions, which means that the original CSO cannot be directly used to solve the discrete production scheduling problem. By considering the feature of the low-carbon FJSP, we proposed a bi-population based discrete cat swarm optimization algorithm (BDCSO). The main contribution of this study are as follows: (1) a parallel searching mechanism is proposed to divide the population into two sub-popualtions, which can exchange their information to implement the cooperation during the evolutionary process; (2) a modified discrete updating approaches are proposed to make the algorithm work directly in a discrete domain; and (3) six adaptive adjustment curves are employed to balance the ability of global and local search in each sub-population. Extensive experimental data demonstrate the effectiveness of our algorithm for the problem under study.

The rest of this paper is organized as follows: the low-carbon FJSP scheduling problem is described in Section II. The overview of the original CSO algorithm is described in Section III. The implementation of the bi-population based discrete cat swarm optimization is addressed in Section IV. In Section V, we report the experimental results, summarize the findings of this study.

II. PROBLEM DESCRIPTION

The definition of the classical FJSP can be described that n jobs to be processed on m machines. Each job consists of a sequence of operations with known processing times. The problem aims to achieve the optimal scheduling scheme by determining the appropriate machine assignment and operation permutation. The traditional objectives of FJSP involve the makespan, machine workload and due date, and so on. Among these indicators, the due date may be more important for many companies under the just-in-time (JIT) production environment. Therefore, total earliness/tardiness cost is considered as one of the objectives in this study. Besides that, the energy consumption cost is also considered during the production process. Here, two types of energy consumption are considered: the useful and the wasted. The former defines the energy consumption required for processing jobs. The latter represents the energy consumption when machines keep no-load running within the time interval between two successive jobs.

The scheduling objective is aiming to minimize the sum of energy consumption cost and total earliness/tardiness cost. For such a scheduling problem, the machine selection should be considered because the energy consumption per unit time will be different when an operation is processed a different machine. In addition, the operation permutation on each machine need to be optimized to reduce the energy consumption of machines for no-load running. To make the problem more concise, some assumptions should be satisfied as below.

(1) All machines and jobs are available at zero time.

(2) Each machine can process only one operation at a time.

(3) Each job cannot be interrupted once it is started.

(4) Each operation must be processed after its predecessor is completed.

(5) Setup times of the machines are negligible.

(6) A machine cannot be stopped until all jobs assigned to it are finished.

To facilitate the understanding of the model, some symbols and variables are shown as follows:

n : the number of jobs in the workshop;

m : the number of machines in the workshop;

J_i : the number of operations of job i;

O_{ij} : the jth operation of job i;

p_{ijk} : the processing time of O_{ij} on machine k;

λ_{ijk} : the energy consumption cost per unit time of O_{ij}on machine k;

θ_k : the energy consumption cost per unit time of machine k on the standby mode;

κ_i : the earliness/tardiness cost per unit time of job i;

ST_{ij} : the start time of O_{ij};

CT_{ij} : the completion time of O_{ij};

ξ : a big constant;

CS : the total cost in the workshop;

C_k : the completion time of machine k;

W_k : the workload of machine k;

y_{ijk} : 0-1 variable, if O_{ij} is processed on machine k, $y_{ijk}=1$; otherwise, $y_{ijk}=0$;

$z_{iji'j'k}$: 0-1 variable, if O_{ij} is processed on machine k prior to $O_{i'j'}$, $z_{iji'j'k}=1$; otherwise, $z_{iji'j'k}=0$.

$$\min\ CS=\sum_{i=1}^{n}\sum_{j=1}^{J_i}\sum_{k=1}^{m}\lambda_{ijk}y_{ijk}p_{ijk}+\sum_{k=1}^{m}\theta_k(CT_k-W_k)+\sum_{i=1}^{n}\kappa_i|c_i-d_i| \quad (1)$$

$$\text{s.t. } CT_{ij}-ST_{ij}=\sum_{k=1}^{m}y_{ijk}p_{ijk},\quad i=1,2,\cdots,n;\ j=1,2,\cdots,J_i \quad (2)$$

$$ST_{i(j+1)}\geq CT_{ij},\quad i=1,2,\cdots,n;\ j=1,2,\cdots,J_i-1 \quad (3)$$

$$ST_{i'j'}+\xi(1-z_{iji'j'k})\geq CT_{ij},\quad i,i'=1,2,\cdots,n;\ j,j'=1,2,\cdots,J_i;\quad k=1,2,\cdots,m \quad (4)$$

$$ST_{ij}+\xi z_{iji'j'k}\geq CT_{i'j'},\quad i,i'=1,2,\cdots,n;\quad j,j'=1,2,\cdots,J_i;\quad k=1,2,\cdots,m \quad (5)$$

$$\sum_{k=1}^{m}y_{ijk}=1,\quad i=1,2,\cdots,n;\quad j=1,2,\cdots,J_i \quad (6)$$

$$y_{ijk}\in\{0,1\},\quad i=1,2,\cdots,n;\quad j=1,2,\cdots,J_i;\quad k=1,2,\cdots,m \quad (7)$$

$$z_{iji'j'k}\in\{0,1\},\quad i,i'=1,2,\cdots,n;\quad j,j'=1,2,\cdots,J_i;\quad k=1,2,\cdots,m \quad (8)$$

(1) addresses the objective of the problem, where the first item represents the total useful energy consumption cost for processing all jobs, the second defines the total wasted energy consumption cost when machines are no-loading running, and the third is the total earliness/tardiness cost; (2) means that no preemption is allowed; (3) represents that the operations of each job have priority constraints; (4) and (5) ensure that each machine can process only one operation simultaneously; (6) means that each operation cannot be assigned to another machine once it starts; (7) and (8) presents 0-1 variables.

III. OVERVIEW OF CAT SWARM OPTIMIZATION

Cat swarm optimization (CSO) was first proposed by Chu and Tsai [19], which was inspired from the behavior of natural cats. In the algorithm, each individual can be viewed as a cat, whose behavior can be classified into two modes: seeking and tracing. In the seeking mode, each cat stays in the rest position but being alert-looking around its surrounding environment for its next move, which corresponds the global search of the algorithm. In the tracing mode, the cat moves with a high speed to chase a prey or any moving target, which corresponds the local search of the algorithm. To combine these two modes, during the searching process, cats are randomly selected from the population according to a mixed radio (MR) inside [0,1]. The steps of the CSO algorithm can be described as below.

Step 1: Create a number of cats in the process.

Step 2: Randomly sprinkle the cats into the solution space, and set the velocity of each cat. Then haphazardly pick number of cats and set them into seeking and tracing mode according to the value of MR.

Step 3: Calculate the fitness value of each cat, and find out the cat with the best one.

Step 4: If the pth cat is in the seeking mode, apply it to the seeking process; otherwise, apply it to the tracing process.

Step 5: Re-pick number of cats and set them into seeking and tracing mode according to MR.

Step 6: Check the stopping condition. If met, terminate the procedure; otherwise, go to Step 3.

The detailed description of the algorithm can be found in [19].

IV. IMPLEMENT OF THE PROPOSED BDCSO

A. ENCODING APPROACH

Like other intelligence algorithms, the first step of constructing a CSO is to define an appropriate encoding method. As mentioned above, the energy consumption optimization scheduling problem can be also made up of two sub-problems, i.e., machine selection and operation permutation. Thus, a two-component discrete encoding mechanism is adopted in our study. The first component attempts to select appropriate machines to process operations, and the second tries to obtain an operation permutation on each machine.

Taking a 3 × 2 FJSP (3 jobs, 2 machines) for example, the encoding scheme can be illustrated in Fig.1, where each job contains two operations to be processed. In Fig.1, the first component denotes the machine selection scheme, whose values are stored in a fixed order. The second component represents the operation permutation. The scheduling scheme in Fig.1 can be represented by (O_{11}^2, O_{31}^1, O_{21}^1, O_{22}^2, O_{12}^1, O_{32}^2), where O_{11}^2 represents the first operation of Job 1 on Machine 2, and so on. Based on the scheduling scheme, the start time of each operation could be determined according to the earliest allowable time.

O_{11}	O_{12}	O_{21}	O_{22}	O_{31}	O_{32}	O_{11}	O_{21}	O_{31}	O_{12}	O_{22}	O_{32}
2	1	1	2	1	2	1	3	2	2	1	3
Machine selection						Operation permutation					

FIGURE 1. Encoding scheme for a 3 × 2 low-carbon FJSP.

B. POPULATION INITIALIZATION

For an intelligent algorithm, the quality of the initial solutions is crucial for the performance of the algorithm. Some different approach are used to generate the initial solutions in order to ensure the initial population with good quality and diversity. According to the characteristics of the problem, the initialization process can be divided into two phases. In the first phase, the machine selection is acquired by using the global selection (GS), local selection (LS) and random selection (RS) in [24]. In this paper, 40% of initial population could be generated by GS, 40% by LS, and 20% by RS. Once the machine selection is determined, all the operations should be sequenced in the second phase. For each machine selection, a predefined number of operation permutations are generated at random. The combination of the two components with the best fitness value will be selected as an initial solution. This procedure will be repeated until all the initial scheduling solutions are generated.

C. PARALLEL SEARCHING MECHANISM

The parallel searching means that the population of the algorithm should be divided into several sub-populations, each of which evolves independently and exchanging their information during the evolutionary process. This mechanism may result in the reduction of the population size of each sub-population, but the cooperation can be implemented. As mentioned before, the low-carbon FJSP under study consists of two sub-problems. Therefore, the objective can be optimized by adjusting the operation permutation and machine selection, respectively. However, it may be unnecessary to modify the two vectors of a certain solution simultaneously in every iteration, especially for the local search. In the BDCSO, the initial population is randomly divided into two sub-populations (P_1 and P_2) of the same size to adjust the machine selection and operation permutation respectively. During the iteration, once the predefined iteration *IES* is achieved, an exchanging strategy should be performed to exchange the information between P_1 and P_2, which can be described as follows:

Step 1: Respectively find the best individuals I_1 and I_2in the two sub-populations.

Step 2: Apply $I_1(I_2)$ to replace the worst individual in the different sub-population $I_2(I_1)$.

Step 3: End the procedure.

D. DISCRETE SEEKING MODE

In the original CSO, cats search for the solution by updating their individual positions in a continuous domain. However, the solution representation in Fig.1 implies that the problem possesses the typical discrete characteristics. Therefore, it is very important to develop a discrete updating mechanism for individuals in order to make the algorithm directly search in a discrete domain. Here, four neighborhood structures are used to serve as the discrete search operators in the seeking mode to implement the global search of the algorithm.

1) NEIGHBORHOOD STRUCTURES FOR MACHINE SELECTION

MS1: Randomly select an operation with more than one alternative machine in the first component of a scheduling solution. Then a different machine is randomly selected to replace the original one.

MS2: Randomly select an operation with more than one alternative machine in the first component of a scheduling solution. Then a machine with the smallest processing time is selected to replace the original one.

2) NEIGHBORHOOD STRUCTURES FOR OPERATION PERMUTATION

OP1: Randomly select two operations e_1 and e_2 belonging to different jobs in the second component of a scheduling solution, and then exchange the positions of e_1 and e_2.

OP2: Randomly select two operations $e_1$and e_2 belonging to different jobs in the second component of a scheduling solution, and then insert e_2 before e_1.

For each sub-population, the updating method in the discrete seeking mode can be described as follows:

Step 1: Set p =1.

Step 2: Create η copies of cat p.

Step 3: Randomly perform the neighborhood structures for machine selection (operation permutation) to each copies in P_1 (P_2).

Step 4: Evaluate the fitness values of all copies and find the copy with the best fitness.

Step 5: If the best copy is better than the original one, update cat p; otherwise, remain the value of cat p.

Step 6: Set $p = p + 1$. If p is greater than the size of the sub-population, end the procedure; Otherwise, go to Step 2.

E. DISCRETE TRACING MODE

For the original CSO, cats move towards the best solution according to the velocities of each dimension in the tracing mode. However, this method cannot be directly used to generate a discrete scheduling. Thus, we proposed a modified discrete updating method based on the crossover operator between each individual and the current best solution. If 'rand' is smaller than the crossover rate, the different crossover methods are performed to $P_1$and P_2, where 'rand' represents a random number inside [0,1]. Here, the multi-point crossover (MPX) is adopted for the machine selection in P_1 and the precedence preserving order-based crossover (POX) is employed for the operation permutation in P_2.

The detailed steps of the MPX can be illustrated by Fig.2 and described as follows:

Step 1: Randomly generate a 0-1 set *BL*.

Step 2: Copy the machine number in the same place with '1' in set *BL* from Parent 1 to Child 1and from Parent 2 to Child 2.

Step 3: Exchange the rest machines in Parent 1 and Parent 2 to obtain Child 1 and Child 2.

The detailed steps of the POX can be illustrated by Fig.3 and described as follows:

Step 1: Generate two subsets SS_1 and SS_2.

Step 2: Randomly choose jobs into SS_1, others are filled into SS_2.

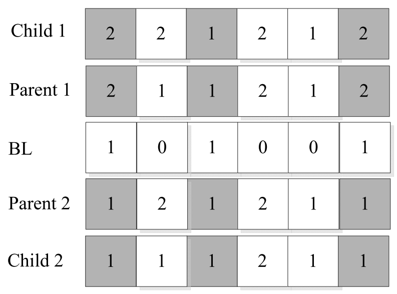

FIGURE 2. MPX crossover operation.

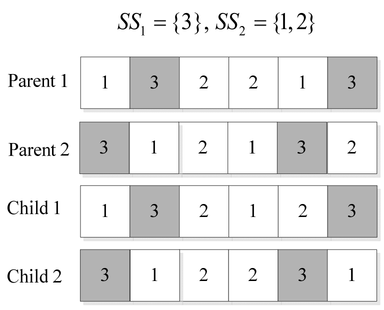

FIGURE 3. POX crossover operation.

TABLE 1. Related parameters for the low-carbon scheduling problems.

nop	meq	p_{ijk}	λ_{ijk}	θ_k	κ_i
[1,5]	[1,m]	[1,20]	[10,15]	[10,15]	[1,3]

Step 3: Copy the jobs in SS_1 from Parent 1 to Child 1 and from Parent 2 to Child 2, and keep their positions unchanged.

Step 4: Copy the jobs in SS_2 from Parent 2 to Child 1 and from Parent 1 to Child 2, and keep their positions unchanged.

F. DYNAMIC ADJUSTMENT METHOD OF MR

As mentioned above, the numbers of cats in the two modes is determined by MR, which represents the degree of emphasis on global and local search. It is well-known that the effective coordination between global search and local search can help the algorithm avoid the premature and obtain the rapid convergence. However, in the original CSO, MR is preset and fixed during the whole evolutionary process. Therefore, we developed a dynamic adjustment method of MR, by which cats are encouraged to explore the global search space at the early stage of the optimization, cluster around the local optimum and exploit information to converge on the global optimum at the latter stage. To implement it, six dynamic adjustment curves of MR are adopted in (9)-(14), where $MR_{\max}$and $MR_{\min}$define the maximum and minimum values of MR, $t_{\max}$ is the maximum iteration. The corresponding algorithms are named as LBDCSO, SinBDCSO, CosBDCSO, TanBDCSO, LnBDCSO and SquareBDCSO.

$$MR = MR_{\max} - (MR_{\max} - MR_{\min})t/t_{\max} \tag{9}$$

TABLE 2. Comparison results of the six proposed algorithms for low-carbon FJSP.

Instance	$m \times n$	LBDCSO				SinBDCSO				TanBDCSO			
		Best	Avg	ARPD	Time	Best	Avg	ARPD	Time	Best	Avg	ARPD	Time
RM01	5×20	6209.8	6320.3	2.6	20.4	6167.0	6394.8	3.8	16.0	6195.6	6319.9	2.6	22.0
RM02	5×50	22432.0	22593.9	6.3	85.2	22327.0	23158.3	9.0	65.2	21790.0	22647.3	6.6	95.6
RM03	5×80	47805.0	49367.5	7.8	194.1	49025.2	50272.3	9.8	152.3	47283.2	49095.4	7.2	220.5
RM04	5×100	68521.0	69062.5	8.6	282.6	68933.0	70865.8	11.4	216.0	67640.0	68385.2	7.5	317.4
RM05	10×20	4342.8	4350.6	2.6	27.1	4346.0	4399.1	3.7	17.7	**4241.6**	4331.6	2.1	23.2
RM06	10×50	12891.0	13022.4	4.5	91.6	12955.5	13156.3	5.5	69.5	12555.0	12824.1	2.9	94.2
RM07	10×80	25030.0	25730.7	7.3	246.7	25525.4	26328.1	9.8	159.5	24519.4	25632.5	6.9	218.6
RM08	10×100	34480.0	35586.1	4.5	305.8	35680.0	36694.9	7.8	233.0	34344.0	35396.2	4.0	316.6
RM09	15×20	**3682.2**	3795.4	3.1	24.6	3739.0	3812.6	3.5	19.5	3740.8	3793.9	3.0	26.8
RM10	15×50	8722.0	8939.0	3.6	98.8	8873.0	9048.5	4.9	73.7	8656.0	8884.2	3.0	98.0
RM11	15×80	19425.8	20085.7	6.4	219.3	19665.2	20323.3	7.7	167.9	19418.6	19884.2	5.4	230.3
RM12	15×100	25361.0	26117.9	5.8	307.2	26040.0	26607.2	7.8	236.2	25414.0	25786.7	4.5	341.4
RM13	20×20	3340.4	3383.3	2.9	27.4	3321.6	3415.6	3.9	21.4	3313.8	3380.3	2.8	29.8
RM14	20×50	8838.8	8990.6	4.2	101.5	8988.8	9120.1	5.7	79.4	8867.8	8928.0	3.5	108.9
RM15	20×80	15237.0	15774.0	6.4	223.7	15635.0	16048.0	8.2	179.3	15536.2	15644.2	5.5	252.8
RM16	20×100	21151.0	21400.7	7.4	337.0	21465.0	21907.9	10.0	248.8	20762.0	21503.3	8.0	355.6
Mean	-	20466.9	20907.5	5.3	162.1	20792.9	21347.1	7.0	122.2	20267.4	20777.3	4.7	172.0
Instance	$m \times n$	LnBDCSO				CosBDCSO				SquareBDCSO			
		Best	Avg	ARPD	Time	Best	Avg	ARPD	Time	Best	Avg	ARPD	Time
RM01	5×20	6170.6	6349.7	3.1	17.3	**6160.8**	6261.6	1.6	29.6	6170.6	6321.2	2.6	25.6
RM02	5×50	22000.0	22974.1	8.1	72.6	**21248.0**	22055.8	3.8	124.8	21762.0	22381.9	5.3	109.3
RM03	5×80	48016.2	49938.8	9.0	173.4	**45799.6**	47380.3	3.5	300.2	46776.0	48182.3	5.2	250.0
RM04	5×100	67332.0	70053.3	10.2	251.3	**63591.0**	66102.1	3.9	435.9	65645.0	67360.8	5.9	370.1
RM05	10×20	4267.4	4406.9	3.9	20.2	4257.6	4317.7	1.8	31.9	4243.0	4343.1	2.4	28.3
RM06	10×50	12688.0	13168.6	5.6	76.3	**12465.5**	12738.4	2.2	128.1	12565.5	12753.1	2.3	122.8
RM07	10×80	24773.2	25505.5	6.4	173.2	**23978.8**	25091.3	4.6	312.6	24823.2	25335.6	5.7	269.8
RM08	10×100	34909.0	36002.8	5.7	243.8	**34049.0**	34708.8	1.9	451.4	34424.0	35032.3	2.9	371.4
RM09	15×20	3684.6	3761.5	2.2	21.4	3701.4	3778.7	2.6	37.8	3733.6	3761.9	2.2	30.5
RM10	15×50	8823.0	8918.8	3.4	55.4	**8629.0**	8825.0	2.3	144.5	8760.0	8893.8	3.1	121.4
RM11	15×80	19691.8	20155.4	6.8	198.9	**18874.0**	19192.0	1.7	348.4	19057.4	19511.9	3.4	271.3
RM12	15×100	25664.0	26228.7	6.3	272.7	**24684.0**	25404.6	2.9	500.6	24822.0	25716.2	4.2	400.6
RM13	20×20	3337.3	3393.9	3.2	23.5	**3287.6**	3343.9	1.7	41.5	3310.2	3355.2	2.1	35.1
RM14	20×50	8844.5	9027.5	4.7	87.3	8657.0	8892.6	3.1	155.4	**8625.5**	8907.4	3.3	128.3
RM15	20×80	15529.8	15881.0	7.1	192.2	**14829.4**	15627.7	5.4	329.6	15088.4	15443.4	4.1	306.4
RM16	20×100	20717.0	21784.6	9.4	274.2	20399.0	21002.6	5.4	496.1	**19918.5**	21001.4	5.4	440.3
Mean	-	20403.0	21096.9	5.9	134.6	19663.2	20295.2	3.0	241.8	19982.8	20518.8	3.8	205.1

$$MR = MR_{\max} - (MR_{\max} - MR_{\min}) \sin(t\pi/(2t_{\max})) \quad (10)$$

$$MR = (MR_{\max} - MR_{\min}) \cos(t/t_{\max}) \quad (11)$$

$$MR = MR_{\max} - (MR_{\max} - MR_{\min}) \tan(t\pi/(4t_{\max})) \quad (12)$$

$$MR = MR_{\max} - (MR_{\max} - MR_{\min}) \ln(1 + t(e-1)/t_{\max}) \quad (13)$$

$$MR = MR_{\max} - (MR_{\max} - MR_{\min})(t/t_{\max})^2 \quad (14)$$

G. PROCEDURE OF THE BDCSO

The steps of BDCSO can be illustrated by Fig.4 and described as follows:

Step 1: Generate the initial population according to the predefined population size.

TABLE 3. ANOVA table for ARPD of six Different BDCSOs.

Source	DF	SS	MS	F	p-value
Factor	5	169.491	33.8982	8.38	1.53365e-006
Error	90	363.988	4.0443		
Total	95	533.478			

Step 2: Randomly split the population into two sub-populations (P_1 and P_2) with the same size.

Step 3: For the sub-populations, adjust the machine selection and operation permutation respectively.

Step 3.1: Separate the sub-population into two groups according to the value of MR.

TABLE 4. Effectiveness analysis of the improvement strategy.

Instanc e	$m \times n$	CosBDCSO				CosSDCSO				CosBDCSO-RR			
		Best	Avg	ARPD	Time	Best	Avg	ARPD	Time	Best	Avg	ARPD	Time
RM01	5×20	6160.8	6261.6	2.8	29.6	6258.6	6426.1	5.4	26.5	**6094.0**	6313.3	3.6	26.6
RM02	5×50	**21248.0**	22055.8	3.8	124.8	21721.0	22217.8	4.6	111.4	22777.0	23324.0	9.8	116.1
RM03	5×80	**45799.6**	47380.3	3.5	300.2	45824.0	47493.9	3.7	283.8	49071.2	51782.8	13.1	278.7
RM04	5×100	**63591.0**	66102.1	3.9	435.9	64687.0	66240.2	4.2	406.5	73641.0	76672.1	20.6	418.4
RM05	10×20	**4257.6**	4307.7	1.2	31.9	4300.8	4399.6	3.3	32.1	4263.6	4362.8	2.5	29.8
RM06	10×50	12465.5	12575.9	1.7	128.1	**12365.0**	12681.9	2.6	129.3	13088.0	13475.2	9.0	125.2
RM07	10×80	**23978.8**	25091.3	4.6	312.6	24928.0	25528.2	6.5	292.7	30399.2	32536.8	35.7	289.7
RM08	10×100	**34049.0**	34708.8	1.9	451.4	34506.0	35529.5	4.3	433.1	44594.0	48163.9	41.5	437.2
RM09	15×20	**3701.4**	3778.7	2.1	37.8	3735.4	3793.5	2.5	36.1	3719.2	3806.0	2.8	33.3
RM10	15×50	**8629.0**	8825.0	2.3	144.5	8829.0	8913.5	3.3	137.3	9271.0	9605.4	11.3	134.0
RM11	15×80	**18874.0**	19192.0	1.7	348.4	19399.0	19723.4	4.5	317.7	24763.4	26379.0	39.8	305.8
RM12	15×100	**24684.0**	25404.6	2.9	500.6	25511.0	26515.7	7.4	448.6	39080.0	40800.2	65.3	446.5
RM13	20×20	**3287.6**	3343.9	1.7	41.5	3408.7	3464.5	5.4	40.4	3303.1	3423.9	4.1	39.2
RM14	20×50	**8657.0**	8892.6	2.7	155.4	8724.5	8993.3	3.9	150.5	9354.3	9614.3	11.1	142.7
RM15	20×80	**14829.4**	15627.7	5.4	329.6	15478.6	15786.7	6.5	333.7	22073.4	23201.7	56.5	411.6
RM16	20×100	**20399.0**	21002.6	3.0	496.1	21086.5	21537.3	5.6	468.8	35056.0	36373.1	78.3	468.8
Mean	-	**19663.2**	20284.4	2.8	241.8	20047.7	20577.8	4.6	228.0	24409.3	25614.7	25.3	231.5

TABLE 5. ANOVA table for testing the effectivenss of improvement strategy.

Source	DF	SS	MS	F	p-value
Factor	2	5000.58	2500.29	12.45	4.96422e-005
Error	45	9035.4	200.79		
Total	47	14035.98			

Step 3.2: If the cat is in the seeking mode, apply it to the seeking process; otherwise, apply it to the tracing process.

Step 4: Evaluate the fitness values of cats and update the best individual.

Step 5: Check the information exchanging condition. If it is met, perform the exchanging procedure.

Step 6: Check the stopping criterion. If met, output the optimum and end the procedure; otherwise, go to Step 3.

V. RESULTS AND DISCUSSION

To test the performance of the proposed BDCSO, we coded the algorithm in FORTRAN and run it on VMware Workstation with 2GB main memory under WinXP.

To evaluate the effectiveness of our presented algorithm for the low-carbon scheduling, sixteen instances are generated with the number of machines $m \in \{5, 10, 15, 20\}$and the number of jobs $n \in \{20, 50, 80, 100\}$. Other parameters are randomly generated following a discrete uniform distribution in Table 1, where *nop* represents the number of operations of each job, and *meq* denotes the number of alternative machines for each operation. In addition, the due date data is set according to the method developed by Demirkol *et al.* [25], which can be shown by $d_i = \left(1 + \frac{0.3 \times n}{m}\right) \times \sum_{j=1}^{J_i} p_{ij}$.

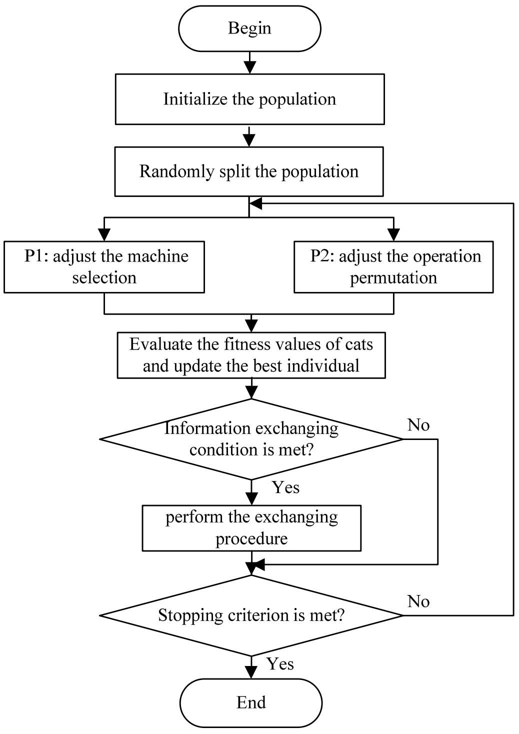
Begin
Initialize the population
Randomly split the population
P1: adjust the machine selection
P2: adjust the operation permutation
Evaluate the fitness values of cats and update the best individual
Information exchanging condition is met?
No
Yes
perform the exchanging procedure
Stopping criterion is met?
No
Yes
End

FIGURE 4. The framework of the BDCSO.

We first compared the performance of the six algorithms with different adjustment curves of MR in Table 2. Parameters of these algorithms are set as follows: the size of each sub-population is 50, the maximum iteration $t_{\max}$ is 500, η is 15, the crossover rate is 0.8, the information exchanging parameter *IES*is 40. For each instance, ten independent runs are conducted for each algorithm. 'Best' means the best

TABLE 6. **Comparison with the published algorithms for low-carbon FJSP.**

Instance	$m\times n$	CosBDCSO				SinGWO				HGWO			
		Best	Avg	ARPD	Time	Best	Avg	ARPD	Time	Best	Avg	ARPD	Time
RM01	5×20	**6160.8**	6261.6	1.6	29.6	6326.2	6699.6	8.7	10.2	6318.8	6465.5	4.9	52.2
RM02	5×50	**21248.0**	22055.8	3.8	124.8	25514.0	25960.5	22.2	43.0	22184.0	22494.3	5.9	240.4
RM03	5×80	45799.6	47380.3	3.9	300.2	55666.6	59953.3	31.5	103.8	**45602.0**	47326.5	3.8	606.1
RM04	5×100	63591.0	66102.1	5.5	435.9	77698.0	79616.0	27.0	152.4	**62667.0**	65784.8	5.0	926.1
RM05	10×20	**4257.6**	4317.7	1.4	31.9	4511.6	4800.1	12.7	10.9	4416.2	4515.5	6.1	52.4
RM06	10×50	**12465.5**	12738.4	2.2	128.1	14009.0	14567.9	16.9	44.7	13029.5	13135.4	5.4	238.2
RM07	10×80	**23978.8**	25091.3	4.6	312.6	28858.0	30344.6	26.5	107.1	25812.0	27286.0	13.8	581.8
RM08	10×100	**34049.0**	34708.8	1.9	451.4	40266.0	40935.4	20.2	156.8	35296.0	36097.2	6.0	918.9
RM09	15×20	**3701.4**	3778.7	2.1	37.8	3970.2	4200.3	13.5	11.2	3847.8	3892.7	5.2	57.1
RM10	15×50	**8629.0**	8825.0	2.3	144.5	9960.0	10158.5	17.7	46.2	8787.0	9139.6	5.9	247.1
RM11	15×80	**18874.0**	19192.0	1.7	348.4	22618.8	23087.4	22.3	110.9	20057.6	21229.6	12.5	654.1
RM12	15×100	**24684.0**	25404.6	2.9	500.6	28438.0	29737.5	20.5	164.5	25627.0	26476.4	7.3	928.2
RM13	20×20	**3287.6**	3343.9	1.7	41.5	3504.1	3596.4	9.4	12.0	3331.3	3408.2	3.7	61.2
RM14	20×50	**8657.0**	8892.6	2.7	155.4	9789.0	10122.4	16.9	48.3	8889.8	9039.4	4.4	258.3
RM15	20×80	**14829.4**	15627.7	5.4	329.6	17228.0	18083.6	21.9	167.6	15900.6	16321.2	10.1	629.8
RM16	20×100	**20399.0**	21002.6	3.0	496.1	24483.0	25058.2	22.8	165.5	21533.5	22150.3	8.6	925.7
Mean	-	19663.2	20295.2	2.9	241.8	23302.5	24182.6	19.4	84.7	20206.3	20922.7	6.8	461.1

Instance	$m\times n$	EGA				MBA			
		Best	Avg	ARPD	Time	Best	Avg	ARPD	Time
RM01	5×20	6866.6	7397.9	20.1	9.8	6965.0	7225.0	17.3	32.2
RM02	5×50	27901.0	29796.9	40.2	40.0	27304.0	30872.5	45.3	136.3
RM03	5×80	60018.0	62950.6	38.0	95.1	63305.0	65199.9	43.0	348.7
RM04	5×100	88055.0	91927.4	46.7	138.6	90336.0	92114.8	47.0	525.2
RM05	10×20	4941.8	5088.6	19.5	10.4	4891.2	5040.1	18.4	35.9
RM06	10×50	15489.0	16573.4	33.0	43.5	15584.0	16314.1	30.9	149.3
RM07	10×80	33044.4	35645.5	48.7	100.5	33707.8	34299.6	43.0	364.3
RM08	10×100	47401.0	49329.0	44.9	147.1	49346.0	50301.4	47.7	537.1
RM09	15×20	4388.2	4600.8	24.3	12.1	4191.4	4389.5	18.6	44.7
RM10	15×50	11794.0	12009.6	39.2	45.7	10537.0	10953.6	26.9	163.6
RM11	15×80	25668.0	26838.7	42.2	107.7	24869.8	25382.3	34.5	383.3
RM12	15×100	34938.0	36538.9	48.0	153.0	34198.0	35036.0	41.9	552.9
RM13	20×20	3851.0	3977.6	21.0	12.8	3699.3	3784.4	15.1	38.9
RM14	20×50	10970.5	11723.6	35.4	49.1	10869.0	11164.0	29.0	158.1
RM15	20×80	19987.2	21932.5	47.9	111.5	18557.0	19162.8	29.2	381.3
RM16	20×100	27857.5	29173.9	43.0	158.4	25926.5	26763.2	31.2	581.6
Mean	-	26448.2	27844.1	37.0	77.2	26517.9	27375.2	32.4	277.1

value in the ten runs of each algorithm. 'Avg' denotes the average values of the ten runs. 'Time' represents the average computation time (in seconds). 'ARPD' is the average relative percent difference, which can be shown by $ARPD = \sum_{r=1}^{R} \frac{100\times(Aol_r - Min)}{Min}/R$, where '$R$' is the number of runs, 'Min' is the minimum solution among the all conducted experiments which is represented by boldface, Aol_r is the obtained value in the rth run by the algorithm for each instance. In addition, 'Mean' defines the average results obtained by each algorithm for the sixteen instances. We can see from Table 2, CosBDCSO spends a relative longer time, but it outperforms other algorithms in terms of other computational performance.

To statistically analyze the results in Table 2, an analysis of variance (ANOVA) test is conducted in Table 3, whese each algorithm is considered as a factor and ARPD is viewed as the response variaerble. The results indicate that there are significant differences among the algorithms with p-value very close to zero. In addition, the box plots of the compared algorithms are shown in Fig. 5.

In this paper, some improvement strategies are employed to enhance the performance of the proposed algorithm. The heurisitic method is used to improve the quality of the initial solutions, and the parallel search mechanism is adopted to implement the cooperation between the sub-population. Here, the effectiveness of the two improvement strategies are tested in Table 4, where 'CosSDCSO' represents the algorithm where the bi-population of CosBDCSO is replaced by a single population. 'CosBDCSO-RR' represents the algorithm where the initial solutions are generated by the random rule.

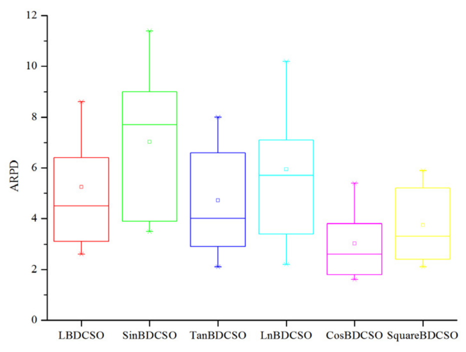

FIGURE 5. Box plot of the six diffrent BDCSOs.

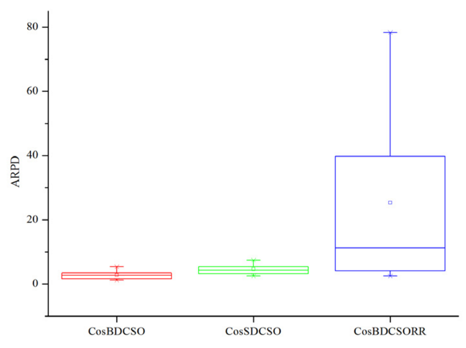

FIGURE 6. Box plot of the algorithms in Table 4.

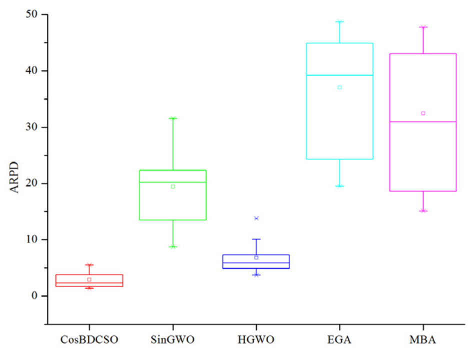

FIGURE 7. Box plot of the algorithms in Table 6.

To facilitate the comparison, the population size of the CosSDCSO is set to be 100, and other parameters are the same with the CosBDCSO. The parameters of the CosBDCSO-RR are the same with the CosBDCSO. From Table 4, it is clear that CosBDCSO yields the better results with a little longer time. Thus, it can be concluded that the heuristic method and the parallel search mechanism are both effective for improving the performance of the algorithm.

To statistically analyze the results in Table 4, an analysis of variance (ANOVA) test is conducted in Table 5. The results indicate that there are significant differences among the algorithms with p-value very close to zero. The box plots of the compared algorithms are shown in Fig. 6.

TABLE 7. ANOVA table for the comparison with publlished algorithms.

Source	DF	SS	MS	F	p-value
Factor	4	14562.7	3640.69	63.44	0.00
Error	75	4304	57.39		
Total	79	18866.7			

To further verify the effectiveness of our algorithm, CosBDCSO is compared with some published algorithms in Table 6, which are SinGWO [26], HGWO [27], EGA [24] and MBA [28]. The parameters of the compared algorithms are set as follows: for SinGWO, the population size is 200, the maximum iteration the algorithm is 500, the maximum iteration of the local search is 10; for HGWO, the population size is 100, the maximum iteration of the algorithm is 500, the crossover rate is 0.8, the mutation rate is 0.2, the maximum iterations of variable neighborhood search and local search are both 10; for EGA, the population size is 100, the maximum iteration of the algorithm is 2000, the crossover rate is 0.8, the mutation rate is 0.1; for MBA, the population size is 100, the maximum iteration of the algorithm is 500, the maximum iteration of the local search is 10. See from Table 6, it is clear that CosBDCSO performs better than other algorithms with an acceptable time.

To statistically analyze the results in Table 6, an analysis of variance (ANOVA) test is conducted in Table 7. The results indicate that there are significant differences among the algorithms with p-value equal to zero. The box plots of the compared algorithms are shown in Fig. 7.

VI. CONCLUSIONS

In this paper, a kind of bi-population based discrete cat swarm optimization algorithm (BDCSO) was presented to solve the low-carbon flexible job shop scheduling problem. In this framework, the cooperation between the two sub-populations was implemented by exchanging the information during the evolutionary process. At the initialization phase, a two-component discrete encoding mechanism was first developed, and a heuristic-based initialization method was used to ensure the quality and diversity of the initial population. By considering the discrete characteristics of the problem, the modified updating approaches were proposed in the seeking and tracing modes, by which the algorithm can work directly in a discrete domain. In addition, in order to balance the global and local search in each sub-population, six adjustment curves were employed to adjust the number of cats in the seeking and tracing modes. Consequently, we obtained six algorithms, i.e., LBDCSO, SinBDCSO, CosBDCSO, TanBDCSO, LnBDCSO and SquareBDCSO.

A number of experiments based on randomly generated instances and benchmark instances were carried out. The comparisons results demonstrate that: (1) the parallel search mechanism is effective for improving the performance of the algorithm (2) Among the six algorithms with different adjustment curves of MR, CosBDCSO can obtain the best results with an acceptable computational time, whose results is also better than the published algorithms.

In future work, the low-carbon FJSP will be further studied by considering some practical constraints, e.g., the adjustable speeds of machines, time-of-use electricity policy, etc. For the BDCSO, some novel and effective neighborhood structures for the low-carbon FJSP should be presented in the seeking mode. In addition, the application of CSO to other combination optimization problems may also be a promising direction.

REFERENCES

[1] G. Mouzon and M. B. Yildirim, "A framework to minimise total energy consumption and total tardiness on a single machine," *Int. J. Sustain. Eng.*, vol. 1, no. 2, pp. 105–116, Aug. 2008.

[2] M. B. Yildirim and G. Mouzon, "Single-machine sustainable production planning to minimize total energy consumption and total completion time using a multiple objective genetic algorithm," *IEEE Trans. Eng. Manage.*, vol. 59, no. 4, pp. 585–597, Nov. 2012.

[3] F. Shrouf, J. Ordieres-Meré, A. García-Sánchez, and M. Ortega-Mier, "Optimizing the production scheduling of a single machine to minimize total energy consumption costs," *J. Cleaner Prod.*, vol. 67, pp. 197–207, Mar. 2014.

[4] A. Che, Y. Zeng, and K. Lyu, "An efficient greedy insertion heuristic for energy-conscious single machine scheduling problem under time-of-use electricity tariffs," *J. Cleaner Prod.*, vol. 129, pp. 565–577, Aug. 2016.

[5] M. Dai, D. Tang, A. Giret, M. A. Salido, and W. D. Li, "Energy-efficient scheduling for a flexible flow shop using an improved genetic-simulated annealing algorithm," *Robot. Comput.-Integr. Manuf.*, vol. 29, no. 5, pp. 418–429, Oct. 2013.

[6] J.-Y. Ding, S. Song, and C. Wu, "Carbon-efficient scheduling of flow shops by multi-objective optimization," *Eur. J. Oper. Res.*, vol. 248, no. 3, pp. 758–771, 2016.

[7] S. A. Mansouri and E. Aktas, "Minimizing energy consumption and makespan in a two-machine flowshop scheduling problem," *J. Oper. Res. Soc.*, vol. 67, no. 11, pp. 1382–1394, Feb. 2016.

[8] H. Luo, B. Du, G. Q. Huang, H. P. Chen, and X. Li, "Hybrid flow shop scheduling considering machine electricity consumption cost," *Int. J. Prod. Econ.*, vol. 146, no. 2, pp. 423–439, Dec. 2013.

[9] J.-Q. Li, H.-Y. Sang, Y.-Y. Han, C.-G. Wang, and K.-Z. Gao, "Efficient multi-objective optimization algorithm for hybrid flow shop scheduling problems with setup energy consumptions," *J. Cleaner Prod.*, vol. 181, pp. 584–598, Apr. 2018.

[10] M. A. Salido, J. Escamilla, A. Giret, and F. Barber, "A genetic algorithm for energy-efficiency in job-shop scheduling," *Int. J. Adv. Manuf. Technol.*, vol. 85, nos. 5–8, pp. 1303–1314, Jul. 2016.

[11] Z. Jiang, L. Zuo, and M. E, "Study on multi-objective flexible job-shop scheduling problem considering energy consumption," *J. Ind. Eng. Manage.*, vol. 7, no. 3, pp. 589–604, Jul. 2014.

[12] C. Zhang, P. Gu, and P. Jiang, "Low-carbon scheduling and estimating for a flexible job shop based on carbon footprint and carbon efficiency of multi-job processing," *Proc. Inst. Mech. Eng. B, J. Eng. Manuf.*, vol. 229, no. 2, pp. 328–342, Apr. 2015.

[13] D. Lei, Y. Zheng, and X. Guo, "A shuffled frog-leaping algorithm for flexible job shop scheduling with the consideration of energy consumption," *Int. J. Prod. Res.*, vol. 55, no. 11, pp. 3126–3140, Nov. 2017.

[14] L. Yin, X. Li, L. Gao, C. Lu, and Z. Zhang, "A novel mathematical model and multi-objective method for the low-carbon flexible job shop scheduling problem," *Sustain. Comput., Inform. Syst.*, vol. 13, pp. 15–30, Mar. 2017.

[15] H. Piroozfard, K. Y. Wong, and W. P. Wong, "Minimizing total carbon footprint and total late work criterion in flexible job shop scheduling by using an improved multi-objective genetic algorithm," *Resour., Conservation Recycling*, vol. 128, pp. 267–283, Jan. 2018.

[16] H. Mokhtari and A. Hasani, "An energy-efficient multi-objective optimization for flexible job-shop scheduling problem," *Comput. Chem. Eng.*, vol. 104, pp. 339–352, Sep. 2017.

[17] J. Li, P. Duan, H. Sang, S. Wang, Z. Liu, and P. Duan, "An efficient optimization algorithm for resource-constrained steelmaking scheduling problems," *IEEE Access*, vol. 6, pp. 33883–33894, 2018, doi: 10.1109/ACCESS.2018.2840512.

[18] Z. Zheng and J.-Q. Li, "Optimal chiller loading by improved invasive weed optimization algorithm for reducing energy consumption," *Energy Buildings*, vol. 161, pp. 80–88, Feb. 2018.

[19] S.-C. Chu and P.-W. Tsai, "Computational intelligence based on the behavior of cats," *Int. J. Innov. Comput., Inf. Control*, vol. 3, no. 1, pp. 163–173, Jan. 2007.

[20] F. Yang, M. Ding, X. Zhang, W. Hou, and C. Zhong, "Non-rigid multi-modal medical image registration by combining L-BFGS-B with cat swarm optimization," *Inf. Sci.*, vol. 316, pp. 440–456, Sep. 2015.

[21] K. C. Lin, K. Y. Zhang, Y. H. Huang, and J. C. Hung, "Feature selection based on an improved cat swarm optimization algorithm for big data classification," *J. Supercomput.*, vol. 72, no. 8, pp. 3210–3221, 2016.

[22] V. I. Skoullis, I. X. Tassopoulos, and G. N. Beligiannis, "Solving the high school timetabling problem using a hybrid cat swarm optimization based algorithm," *Appl. Soft. Comput.*, vol. 52, pp. 277–289, Mar. 2017.

[23] R. Rautray and R. C. Balabantaray, "Cat swarm optimization based evolutionary framework for multi document summarization," *Phys. A, Stat. Mech. Appl.*, vol. 477, pp. 174–186, Jul. 2017.

[24] G. Zhang, L. Gao, and Y. Shi, "An effective genetic algorithm for the flexible job-shop scheduling problem," *Expert Syst. Appl.*, vol. 38, no. 4, pp. 3563–3573, Apr. 2011.

[25] E. Demirkol, S. Mehta, and R. Uzsoy, "Benchmarks for shop scheduling problems," *Eur. J. Oper. Res.*, vol. 109, no. 1, pp. 137–141, Aug. 1998.

[26] T. H. Jiang, "Low-carbon workshop scheduling problem based on grey wolf optimization," *Comput. Integr. Manuf. Syst.*, 2017. [Online]. Available: http://kns.cnki.net/kcms/detail/11.5946.TP.20171017.1723.002.html

[27] T.-H. Jiang, "Flexible job shop scheduling problem with hybrid grey wolf optimization algorithm," *Control Decision*, vol. 33, no. 3, pp. 503–508, Mar. 2018.

[28] H. Zhu, B. He, and H. Li, "Modified Bat algorithm for the multi-objective flexible job shop scheduling problem," *Int. J. Performability Eng.*, vol. 13, no. 7, pp. 999–1012, Nov. 2017.

TIANHUA JIANG was born in Weihai, China, in 1983. He received the B.S. degree in automation from Jinan University, Jinan, China, in 2007, the M.S. degree in control science and engineering from the Sichuan University of Science and Engineering, Zigong, China, in 2010, and the Ph.D. degree from the MOE Key Laboratory of Measurement and Control of Complex Systems of Engineering and the School of Automation, Southeast University, China, in 2015.

Since 2015, he has been a Lecturer with the School of Transportation, Ludong University, Yantai, China. His research interests include production scheduling and intelligent algorithm. His recent publications have appeared in some peer-reviewed journals such as IEEE Access, the *Journal of Intelligent and Fuzzy Systems*, the *International Journal of Industrial Engineering: Theory, Applications and Practice*, and the *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*.

GUANLONG DENG was born in Chenzhou, China, in 1985. He received the B.S. degree in automation and the Ph.D. degree in control theory and control engineering from the East China University of Science and Technology, Shanghai, China, in 2008 and 2012, respectively.

Since 2018, he has been an Associate Professor with the School of Information and Electrical Engineering, Ludong University, Yantai, China. His research interests include production scheduling and intelligent algorithm. His recent publications have appeared in some peer-reviewed journals such as *Mathematical Problems in Engineering*, *Advances in Mechanical Engineering*, the *International Journal of Systems Science*, the *International Journal of Computer Integrated Manufacturing*, and *Computers and Operations Research*.