IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 14, 2018, accepted January 9, 2019, date of publication January 14, 2019, date of current version February 8, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2892826

Bi-Population Based Discrete Bat Algorithm for the Low-Carbon Job Shop Scheduling Problem

YI LU[1] AND TIANHUA JIANG[2]
[1]Henan Institute of Technology, Xinxiang 453003, China
[2]School of Transportation, Ludong University, Yantai 264025, China

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

This work was supported in part by the Special Research and Promotion Program of Henan Province under Grant 182102210257, in part by the Project of Henan Province Higher Educational Key Research Program under Grant 16A120011, in part by the High-level Talent Research Foundation of Henan Institute of Technology, and in part by the Shandong Provincial Natural Science Foundation of China under Grant ZR2016GP02.

ABSTRACT Job shop scheduling problem (JSP) is a combinatorial optimization problem, which has been widely studied due to its strong theoretical and background for application. However, in previous studies on the traditional JSP, the optimization objective is mainly relative to time, such as makespan, flow time, tardiness, earliness, and workload. With the advent of green manufacturing, energy consumption should be considered in the JSP. Therefore, a low-carbon JSP is studied in this paper. Due to the NP-hard nature, a meta-heuristic algorithm, bat algorithm (BA), is considered in this paper. According to the characteristics of the problem, a kind of bi-population-based discrete BA (BDBA) is proposed to minimize the sum of the energy consumption cost and the completion-time cost. A parallel searching mechanism is first introduced to the algorithm, by which the population is divided into two sub-populations to, respectively, adjust the job permutation and the processing speed of each machine. Three communication strategies are used to implement the cooperation between the sub-populations. In addition, due to the fact that the original BA was developed to deal with the continuous problems, a modified discrete updating approach is proposed to make the BA algorithm directly work in a discrete domain. Finally, extensive simulations have been conducted to test the effectiveness of the proposed BDBA algorithm. The experimental data demonstrate that the proposed BDBA is effective in solving the low-carbon JSP under study.

INDEX TERMS Job shop, low-carbon production scheduling, energy consumption, bi-population based discrete bat algorithm.

I. INTRODUCTION

In recent years, environmental deterioration (such as climate change and global warming) has been paid more and more attention. As a dominant energy consumer, the manufacturing industry is accountable for about 33% of the global energy consumption [1]. Hence, effective energy-saving measures are increasingly needed to balance the economic development and environmental protection, which is the key challenge for the long-term sustainable manufacturing. To achieve this goal, some researchers focus on designing innovative manufacturing machines, equipment or processes [2], [3]. However, due to the considerable financial requirements, this method may not be very appropriate for small to medium sized companies. On the other hand, from the system-level perspective, some operational approaches (such as production scheduling and planning) can be significant drivers in energy consumption reduction [4]. These approaches have been proven to be more efficient in energy consumption reduction with a modest investment, and more easily to be applied to the existing manufacturing systems [5].

Since the 1950s, many production scheduling problems have been studied. However, these previous researches have primarily focused on the production efficiency with the consideration of some traditional time-related objectives (e.g., makespan, flow time, tardiness, earliness and workload, etc.), while ignoring the environmental metrics (e.g., energy consumption, carbon footprint and CO_2 emission, etc.) [6]. Among these studies, the job shop scheduling problem (JSP) has been widely concerned due to its strong theoretical and background for application. In the real life, many problems can be considered as a job shop scheduling problem, such as production scheduling in workshop, departure/arrival times in logistic systems, the delivery times of orders in a company, and so on. However, the low-carbon job shop scheduling problem with the consideration of energy consumption is not fully studied at present. Therefore, in this study, job shop is chosen as a manufacturing type for the low-carbon scheduling problem.

After the promotion of green manufacturing, the low-carbon scheduling with the consideration of environmental metrics has gradually caught many scholars' attention. Compared with the traditional production scheduling problem, there are relatively few literature about the low-carbon scheduling problem. Mouzon and Yildirim [7] considered a single machine scheduling problem to minimize the total energy consumption and the total tardiness. The decision-maker has to determine the timing and length of turn off/turn on operation and obtain a sequence of jobs. To achieve this goal, a new greedy randomized adaptive search algorithm was used to obtain an approximate Pareto front. Liu *et al.* [8] established a multi-objective optimization model to minimize the total CO_2 emissions and the total completion time in a single-machine system. Then a non-dominated sorting genetic algorithm II (NSGA-II) was designed to solve the model. Shrouf *et al.* [9] built a mathematical model to minimize the energy consumption cost in a single machine system by considering variable energy prices. Che *et al.* [10] addressed a single-machine scheduling problem with time-dependent electricity tariffs to minimize the total electricity cost. A greedy insertion heuristic was developed for the established continuous-time mixed-integer linear programming model. Rubaiee *et al.* [11] studied a non-preemptive scheduling problem in a single-machine system to minimize the total tardiness and the total energy cost under time-of-use (TOU) electricity tariffs. Several genetic algorithms were developed to solve a mixed-integer multi-objective mathematical programming model. Dai *et al.* [12] modeled a flexible flow shop scheduling problem based on the energy-efficient mechanism. An improved genetic-simulated annealing algorithm was adopted to optimize the makespan and the total energy consumption. Luo *et al.* [13] proposed an ant colony optimization algorithm (ACO) considering both production efficiency and electric power cost (EPC) under time-of-use (TOU) strategy. Zhang *et al.* [14] developed a time-indexed integer programming model to minimize the electricity cost and the carbon footprint under time-of-use tariffs. Lin *et al.* [15] developed a multi-objective teaching-learning-based optimization algorithm (TLBO) for processing parameter optimization and flow-shop scheduling with the objective to minimize the makespan and the carbon footprint. Tang *et al.* [16] addressed the dynamic flexible flow shop scheduling problem with the objective of reducing the makespan and the energy consumption. An improved particle swarm optimization was developed to obtain the Pareto optimal solution. Lu *et al.* [17] investigated an energy-efficient permutation flow shop scheduling problem (PFSP) with sequence-dependent setup and controllable transportation time. A hybrid multi-objective backtracking search algorithm was presented to optimize the makespan and the energy consumption. Wang and Wang [18] investigated an energy-efficient scheduling of a distributed permutation flow shop with the criterion to minimize the makespan and the total energy consumption. By considering the complexity of the problem, a knowledge-based cooperative algorithm was designed to solve the problem. Lei *et al.* [19] proposed a teachers' teaching-learning-based optimization (TTLBO) to minimize the total energy consumption and the total tardiness in a hybrid flow shop.

As observed from the reviewed literature above, the low-carbon scheduling problems mainly focus on the single-machine or flow shop systems. Due to the importance of JSP, it is more practical for considering the problem with environmental metrics. However, by contrast, researches on the low-carbon job shop scheduling problem are not enough. Liu *et al.* [4] established a bi-objective model to minimize the total electricity consumption and the total weighted tardiness in a job shop environment. A non-dominant sorting genetic algorithm was employed to obtain the Pareto front. Jiang *et al.* [20] investigated an energy-efficient job shop scheduling problem with the objective of minimizing the total cost of energy-consumption and tardiness and proposed a grey wolf optimization algorithm with double-searching mode. May *et al.* [21] studied the effects of production scheduling aimed towards improving productive and environmental performances in a job shop environment. A green genetic algorithm was designed for the assessment of multi-objective problems. Kawaguchi and Fukuyama [22] considered a job shop scheduling problem with the objective of minimizing the makespan and the total secondary energy costs. An improved parallel reactive tabu search was proposed to obtain not only the optimal production scheduling but also the optimal operation of energy plants. In these researches, the addition of environmental factors increases the number of variables and constraints and makes the problem more complex than the traditional JSP. More researches need be carried out on the low-carbon JSP, and some realistic constraints should be added to the problem. In some real-life manufacturing systems, such as the CNC machines for mechanical processing, machines can work at different speeds. When machine works at a higher speed, the processing time decreases but the amount of energy consumption increases, and when machine works at a lower speed, the processing time increases while the amount of energy consumption decreases [5]. This provides an opportunity to control the energy consumption in the workshop. In such a problem, the speed of machine affects the processing time and energy consumption. Therefore, it should be taken as an independent decision-making variable, which is not considered in the above low-carbon JSP. As far as the authors' knowledge, the low-carbon job shop scheduling problem with variable machine speed is not fully investigated. Jiang *et al.* [5] considered a job shop scheduling problem with adjustable speeds of machines to minimize the sum of the energy consumption cost and the completion-time cost. An improved whale optimization algorithm (IWOA) was developed to solve the problem. Escamilla *et al.* [23] addressed an energy-efficiency job shop scheduling problem where the machine can work at different speed rates with consuming different amounts of energy. A genetic algorithm was developed to solve this scheduling problem. Since the traditional JSP is an NP-hard problem, the low-carbon JSP is also very difficult to be solved. It is well-known that meta-heuristics are effective in solving the production scheduling problems. Therefore, more effective and efficient meta-heuristic algorithms for the low-carbon JSP are highly desirable.

In recent years, more and more swarm-based intelligence optimization algorithms are gradually developed [24]. Bat algorithm (BA) is a relatively new paradigm firstly proposed by Yang [25] to optimize continuous problems. As a swarm-based intelligence algorithm, BA has been adopted to solve various optimization problems, such as global engineering optimization [26], feature selection [27], traveling salesman problem [28], maximum power point tracking for photo-voltaic systems [29], and flexible job shop scheduling problem [30], and so on. Nevertheless, to the best of the authors' knowledge, the BA algorithm has never been applied to the low-carbon JSP. This is one of the motivations behind using the BA algorithm in this study. The important reasons which have motivated the using of BA are its own merits like fast execution, few parameters and ease of implementation. In this study, to solve the low-carbon JSP, a bi-population based discrete bat algorithm (BDBA) is proposed in this paper. The main contribution of this study are as follows: (1) a parallel searching mechanism is proposed to divide the population into two sub-populations, which can exchange their information to implement the cooperation during the evolutionary process; (2) a modified discrete updating approach is proposed to make the BA algorithm directly search in a discrete domain. Finally, extensive experimental data demonstrate the effectiveness of the proposed algorithm for the problem under study.

The rest of this paper is organized as follows: Problem description of the low-carbon JSP is addressed in Section II. Overview of the original BA algorithm is described in Section III. Implementation of the BDBA algorithm is shown in Section IV. Experimental results and the findings of this study are reported in Section V.

II. PROBLEM DESCRIPTION

In this study, the low-carbon JSP can be explained as follows: n jobs are assumed to be processed on m machines in a job shop. Here, the main difference of the low-carbon JSP from the traditional JSP is that the adjustable speed level of each machine is considered. That is, one machine processes a job with a finite and discrete speed set $\boldsymbol{v} = \{v_1, v_2, \cdots, v_d\}$. The higher the speed selected for a machine, the shorter the processing time of the job processed on it. There exists a basic processing time q_{jk} when job j is processed on machine k. If v_d is selected for machine k, the processing time of job j, p_{jkd}, can be calculated by $p_{jkd} = q_{jk} \big/ v_d$. The energy consumption cost per unit time can be represented by E_{kd}. For any two speeds v_d and $v_{d'}$, if $v_{d'} > v_d$, $E_{kd'}p_{jkd'} > E_{kd}p_{jkd}$ holds. In other words, a machine processing at a higher speed will reduce the processing time, but increase the energy consumption cost.

Some additional assumptions are listed as follows: Machines and jobs are available at zero time; Each machine can not process more than one operation at the same time; Any job can only be processed on one machine at a time; Non-preemption is allowed once a job starts on a machine; Setup time and breakdown of machines are neglected; The speed of a machine can not be adjusted when a job is being processed on it; For a machine, it will not be stopped until all jobs on it are completed. When the machine is waiting for process, it runs on a stand-by mode with energy consumption cost per unit time SE_k. Here, the optimization objective is aiming to minimize the sum of energy-consumption cost and completion-time cost.

$$\min F = \sum_{j=1}^{n}\sum_{k=1}^{m}\sum_{d=1}^{D_k} E_{kd}p_{jkd}x_{jkd} + \sum_{k=1}^{m} SE_k(C_k - W_k) + \rho C_{\max} \tag{1}$$

In Equation (1), F means the optimization objective. x_{jkd} is a 0-1 variable, if job j is processed on machine k with speed-level d, $x_{jkd} = 1$; otherwise, $x_{jkd} = 0$. D_k means the number of adjustable speed levels of machine k. C_k denotes the final completion time of machine k. W_k represents the total workload of machine k. $C_{\max}$ defines the makespan of the workshop. ρ is the cost coefficient relevant to the makespan. With regard to the mathematical model, the readers may consult the study of Jiang *et al.* [5].

III. OVERVIEW OF ORIGINAL BAT ALGORITHM

In the nature, micro-bats use the echolocation system to detect prey, avoid obstacles and locate the roosting places [25]. BA is a nature-inspired evolutionary algorithm on the basis of the echolocation behavior of micro-bats.

In the search process of the original BA algorithm, each bat updates its position and velocity according to Equations (2)-(4), where f_i is the pulse frequency of bat i, $f_{\min}$ and $f_{\max}$ represent the minimum and maximum values, $\boldsymbol{x}_i^t$ and $\boldsymbol{v}_i^t$ define the individual position and the velocity of bat i at generation t, $\boldsymbol{x}^*$ is the global best position found so far.

$$f_i = f_{\min} + (f_{\max} - f_{\min}) \times rand \tag{2}$$

$$\boldsymbol{v}_i^t = \boldsymbol{v}_i^{t-1} + (\boldsymbol{x}_i^{t-1} - \boldsymbol{x}^*)f_i \tag{3}$$

$$\boldsymbol{x}_i^t = \boldsymbol{x}_i^{t-1} + \boldsymbol{v}_i^t \tag{4}$$

After updating the position, a random number *rand* is generated. If it is greater than the pulse emission rate r_i, a new position will be generated around the current best solutions following Equation (5).

$$\boldsymbol{x}_{new} = \boldsymbol{x}_{old} + \varepsilon A^t \tag{5}$$

where $\varepsilon \in [-1, 1]$ is a random number, while $A^t = \langle A_i^t \rangle$ is the average loudness at generate t.

If a random number *rand* is less than A_i and $F(\boldsymbol{x}_i) < F(\boldsymbol{x}^*)$, the new solution will be accepted, and the pulse emission rate r_i and loudness A_i are varied following Equations (6) and (7), i.e., the loudness A_i is decreased, and the pulse emission rate r_i is increased. λ $(0 < \lambda < 1)$ and γ $(\gamma > 0)$ are constant.

$$A_i^{t+1} = \lambda A_i^t \tag{6}$$

$$r_i^{t+1} = r_i^0(1 - \exp(1 - \gamma t)) \tag{7}$$

The detailed steps of the original bat algorithm [24] can be described as follows:

Step 1: Randomly initialize the bat population with a predefined size.

Step 2: For each bat $\boldsymbol{x}_i$, initialize the velocity $\boldsymbol{v}_i$, the pulse emission rate r_i and loudness A_i.

Step 3: Generate new solutions following Equations (2) ~(4).

Step 4: If *rand* $> r_i$, select a solution among the best solutions and generate a new individual around the best one.

Step 5: If *rand* $< A_i$ and $F(\boldsymbol{x}_i) < F(\boldsymbol{x}^*)$ are met, accept the new solution, and increase r_i and reduce A_i.

Step 6: Rank the bats and find out the best individual.

Step 7: Check the stopping criterion. If met, output the optimum and end the procedure; otherwise, go to Step 3.

IV. IMPLEMENTATION OF THE PROPOSED BDBA

A. ENCODING AND DECODING APPROACH

Like other meta-heuristic algorithms, the first step of designing the algorithm is to define an encoding approach. According to the description above, there are two sub-problems involved in the problem under study, i.e., job sequencing and speed-level selection. The decision-maker has to select a processing speed for each job and determine a job permutation on each machine. Thus, a job-speed-based encoding method can be used to represent the scheduling solution. The solution can be divided into two parts. The first part defines the job permutation on each machine, and the second represents the speed level selected for processing each job.

Taking a $3 \times 2 \times 3$ low-carbon JSP into account, three jobs are supposed to be processed on two machines, and three speed-levels are considered for each job on machines. The scheduling solution can be illustrated by Fig.1. In the job sequencing part, elements with the same values represent different operations of the same job. In the speed-level selection part, elements represent the selected speed level for processing jobs. O_{ik} represents the kth operation of job i.

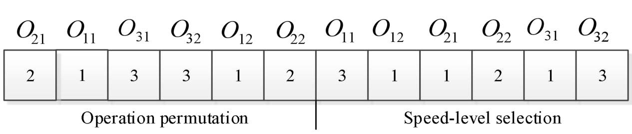

FIGURE 1. **Encoding approach for the 3 × 2 × 3 low-carbon JSP.**

In this study, the initial population is randomly generated according to the encoding method. To acquire a feasible scheduling scheme, the decoding process is adopted as follows:

(1) Scanning the job permutation from left to right, and find the speed level for processing each operation on machines.

(2) The first operation in the job permutation part is first scheduled, and then the second one is done and so on; each operation is processed in the earliest available time on its assigned machine. This procedure is repeated until a scheduling scheme is obtained.

B. PARALLEL SEARCHING MECHANISM

In a parallel structure, several groups are generated by dividing the population into sub-populations, which evolve independently and exchange the information during the evolutionary process. This mechanism results in the reduction of the population size for each sub-population and the implementation of cooperation between sub-populations [6], [31].

As mentioned above, the low-carbon JSP consists of two sub-problems. The optimization function can be optimized by adjusting the job permutation and speed-level selection. However, it may be unnecessary to modify the two vectors of a certain solution simultaneously in every iteration, especially for the local search [31]. Therefore, in the proposed BDBA, the population is divided into two sub-populations (*PJ* and *PS*) of the same size to adjust the job permutation and the speed-level selection respectively. Before dividing the population, all the individuals are sequenced in an ascending order according to their fitness values. Then, the first individual is assigned to *PJ*, the second is assigned to *PS*, the third is assigned to *PJ*, and so on. In regular iterations, a commutation strategy will be conducted to implement the cooperation between the sub-populations after every fixed number of iterations *iter*. The commutation strategy can be classified into three types as below.

W-B: the worst agent of a sub-population is replaced by the best agent of the another sub-population.

R-R: a randomly selected agent of a sub-population is replaced by a randomly selected agent of the another sub-population.

R-B: a randomly selected agent of a sub-population is replaced by the best agent of the another sub-population.

C. DISCRETE INDIVIDUAL UPDATING METHOD

The original BA was designed to deal with continuous optimization problems. However, the low-carbon JSP is a discrete optimization problem. Therefore, Equations (2)-(4) cannot be directly adopted to the problem under study. By considering this situation, some modifications are designed for the BA algorithm.

Seen from Equation (3), $\boldsymbol{v}_i^t$ depends on the $\boldsymbol{v}_i$ in the time step $t - 1$, the difference between the bat i and the current best position and the pulse frequency f_i. It is obvious that this parameter cannot be directly adopted in our algorithm. Here, we modify the velocity based on a crossover operator between each individual and the current best solution, which can be represented by Equation (8).

$$v_i^t = c \otimes CR(x_i^{t-1}, x^*) \tag{8}$$

where CR defines the crossover operation between x_i^{t-1} and x^*, and c is the crossover rate. If a random number $rand < c$, two different crossover methods are respectively performed to PJ and PS. Here, the job-based crossover (JBX) is applied to the job permutation in PJ, and the multi-point crossover (MPX) is adopted to the speed-level selection in PS.

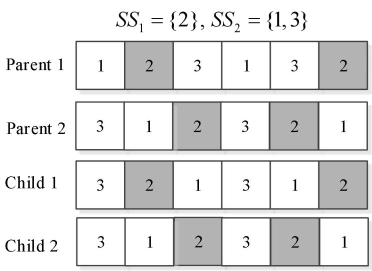

FIGURE 2. **JBX crossover operation.**

The steps of the JBX is shown in Fig.2 and described as follows:

Step 1: Create two job subsets SS_1 and SS_2.

*Step 2:*Some jobs are randomly selected into SS_1, and the others are filled into SS_2.

Step 3: Copy the jobs in SS_1 from Parent 1 to Child 1, and copy the jobs in SS_2 from Parent 2 to Child 2, and maintain their original positions.

Step 4: Copy the jobs in SS_2 from Parent 2 to Child 1 and copy the jobs in SS_1 from Parent 1 to Child 2, and keep their positions.

The steps of the MPX is shown in Fig.3 and described as follows:

Step 1: Randomly create a 0-1 set BL.

Step 2: Copy the speed level in the same place with '1' in set BL from Parent 1 to Child 1 and from Parent 2 to Child 2.

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

Furthermore, in the original BA, new solutions are generated following Equation (4). As previously mentioned, it cannot be applied directly to the low-carbon JSP. For this reason, we present a modification of it in Equation (9).

$$x_i^t = w \otimes MU(v_i^t) \tag{9}$$

where MU defines a mutation operator, and w is the mutation rate. If a random number $rand < w$, two different mutation methods are respectively performed to PJ and PS. In this study, a swap mutation is conducted to the job permutation in PJ, and a single-point mutation is applied to the speed-level selection in PS.

For the swap mutation, two operations belonging to different jobs are randomly selected from the job permutation scheme, and then the position of the selected operations are swapped to obtain a new solution.

For the single-point mutation, an operation with more than one alternative speed level is randomly selected from the speed level selection scheme, and then a different speed level is randomly selected to replace the original one.

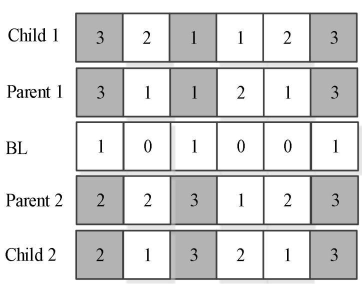

FIGURE 3. **MPX crossover operation.**

D. NEIGHBORHOOD STRUCTHRE

In every iteration of the original BA, a random number $rand$ is generated after updating the position. If $rand$ is greater than the pulse emission rate r_i, a new position is generated around the current best solution. According to the characteristics of the low-carbon JSP, two types of neighborhood structures are respectively adopted in PJ and PS. When $rand < r_i$, a new solution is obtained by randomly performing one of the neighborhood structures to the current best solution.

(1) **Neighborhood structures for job permutation**

Swap: A new neighboring solution is obtained according to the above swap mutation method.

Insert: Randomly choose two operations O_1 and O_2, and then insert O_2 before O_1.

Inverse: Randomly choose two positions, and then invert the order of the elements between the two selected positions.

(2) **Neighborhood structures for speed level selection**

Random selection: A new neighboring solution is obtained according to the above single-point mutation method.

Slow down: Randomly choose an operation with more than one alternative speed level. Then the lowest speed level is chosen to take the place of the original one.

Speed up: Randomly choose an operation with more than one alternative speed level. Then the highest speed level is chosen to take the place of the original one.

E. UPDATING APPROACH OF PULSE EMISSION RATE AND LOUDNESS

In the original BA, the pulse emission rate r_i and loudness A_i are used to control the intensive local search [32]. Each individual has its own pulse emission rate r_i, which is initially set be a positive and small value and will increase to 1. In this study, the method proposed by Luo *et al.* [32] is used to update the value of r_i, which is shown in Equation (10). Following this method, the algorithm can not only quickly exploit near the current optimal solution in the early iteration to accelerate the convergence speed, but also can mainly focus on diversity in later stage and can avoid the premature. For the loudness A_i, the updating method is expressed by Equation (11), where $F_{\max}$ and $F_{\min}$ are the maximum and the minimum fitness values in the current population. In Equation (11), the loudness is relative to the solution quality of individual. The individual with a good fitness will be reserved for the next iteration.

$$r_i(t) = (1 + \exp(-\frac{10}{t_{\max}} \times (t - \frac{t_{\max}}{2}) + r_i^0))^{-1} \quad (10)$$

$$A_i = \frac{F_i - F_{\min}}{F_{\max} - F_{\min}} \quad (11)$$

F. STEPS OF THE PROPOSED BDBA

The pseudo-code of the proposed BDBA is shown in Fig.4.

```
(1) Randomly generate the initial population with a predefined population size.
(2) Initialize the pulse emission rate r_i and loudness A_i .
(3) Divide the population into two sub-populations ( PJ and PS ).
(4) For each sub-population, perform the procedure below.
    Generate new solutions following Equations (6) and (7).
    if ( rand > r_i ) then
        Select a solution among the best solutions
        Generate a new individual around x* according to the neighborhood structures
    end if
    if ( rand < A_i and F(x_i) < F(x*) ) then
        Accept the new solution
        Increase r_i and reduce A_i
    end if
    Rank the bats and find out the best individual.
(5) Check the information exchanging condition. If it is met, perform the exchanging procedure.
(6) Check the stopping criterion. If it is met, output the optimum and end the procedure; otherwise, go to (4).
```

FIGURE 4. Pseudo-code of the proposed BDBA.

V. RESULTS AND DISCUSSION

In this section, extensive experiments are conducted to test the performance of the proposed BDBA algorithm. The algorithm is implemented in FORTRAN language and run on a VMware Workstation Pro 14 with 2GB main memory under WinXP. Here, 68 instances are used to validate the effectiveness of the BDBA in Tables 1 and 2. These instances include two classes:

(1) The benchmark instances FT06, FT10, FT20 and LA01-LA40 were respectively designed by Fisher and Thompson [33] and Lawrence [34].
(2) The instances RM01-RM25 are randomly generated, where processing times of operations are drawn from a [5, 100] uniform distribution, and the processing routing of each job is generated at random.

For each instance, 10 independent replications are conducted by different algorithms. Here, the original processing times are taken as the basic processing times. The speed of a machine for processing an operation can be selected from $\boldsymbol{v} = \{v_1, v_2, v_3, v_4, v_5\} = \{1.0, 1.2, 1.5, 2.0, 2.5\}$. Energy consumption cost per unit time of machine k can be calculated according to $E_{kd} = \xi_k \times v_d^2, d = 1, 2, 3, 4, 5$, where ξ_k is drown from a [2, 4] uniform distribution. The stand-by energy consumption cost per unit time of machine k can be obtained by $SE_k = \xi_k/4$. In addition, the completion time cost per unit time ρ is set to be 15.0.

To validate the effectiveness, the proposed BDBA algorithm with different communication strategies i.e., BDBA(W-B), BDBA(R-R) and BDBA(R-B), are compared with genetic algorithm (GA) [23], improved whale optimization algorithm (IWOA) [5], and single-population discrete bat algorithm (SDBA). In this paper, the parallel searching strategy is used to improve the performance of the algorithm, where the population is divided into two sub-populations. Here, the SDBA represents the discrete bat algorithm, where only a single-population is involved during the evolutionary process. In Tables 1 and 2, '*Best*' is the best value obtained by each algorithm in the ten runs. '*Avg*' defines the average results of the ten runs. '*SD*' is the standard deviation of computational results obtained by ten runs. '*ARPD*' represents the average relative percent deviation, which is represented by $ARPD = \sum_{l=1}^{L} \frac{100 \times (Alg_l - Min)}{Min} \Big/ L$, where '$L$' is the number of runs, '*Min*' is the minimum result among all the conducted experiments, Alg_l is the obtained value in the lth run by an algorithm. '*Time*' is the average computational time (in seconds) in the ten runs. In addition, '*Mean*' defines the average results obtained by each algorithm for all the instances. Boldface represents the optimal value obtained by all the compared algorithms.

For the parameters of the compared algorithms, GA and IWOA are set as the same values in [5]. In the GA, the population size is 200, the maximum iteration is 2000, the crossover rate is 0.8, and the mutation rate is 0.1. In the IWOA, the population size is 200, and the maximum iteration is 2000. To facilitate the comparison, parameters of BDBA and SDBA are set as follows: In the BDBA, the size of each sub-population is 100, the maximum iteration is 2000, the crossover rate is 0.8, the mutation rate is 0.1, and the information exchanging parameter *iter* is 20; In the SDBA, the population size is 200, and the maximum iteration is 2000, the crossover rate is 0.8, and the mutation rate is 0.1.

Seen from the computational results in Table 1, it can be easily concluded that: (1) In comparisons of the '*Best*' value, the BDBA(R-R) algorithm yields 23 optimal values out of 68 instances, which is better than the other compared algorithms. The second best algorithm, namely BDBA(R-B), obtains 20 optimal values. According to the last row of Table 1, the BDBA(R-R) algorithm obtains the lowest mean value of *Best* among all the compared algorithms. (2) In comparisons of the '*Avg*' value, the BDBA(R-R) algorithm obtains 25 optimal values out of 68 instances. The second best algorithm, namely BDBA(R-B), only obtained 17 optimal values. According to the last row of Table 1, the BDBA(R-R)

TABLE 1. Comparison with the published algorithms.

Instance	(m,n)	GA					IWOA					SDBA				
		Best	*Avg*	*SD*	*ARPD*	*Time(s)*	*Best*	*Avg*	*SD*	*ARPD*	*Time(s)*	*Best*	*Avg*	*SD*	*ARPD*	*Time(s)*
FT06	(6,6)	1550.6	1582.9	20.0	15.51	7.3	**1370.4**	**1382.8**	**6.2**	**0.90**	26.9	1386.2	1411.5	16.6	3.00	70.9
FT10	(10,10)	38638.7	39772.5	620.2	31.52	30.1	31619.3	32726.4	649.4	8.22	107.8	30324	31041.7	459.1	2.65	199.0
FT20	(5,20)	42777.3	43877.3	514.1	29.90	32.7	34544.2	35155.7	**376.8**	4.08	111.4	33961.4	34934.7	807.4	3.42	206.1
LA01	(5,10)	21845.8	22302.7	279.7	26.01	11.6	18096	18204.8	**88.8**	2.85	39.0	18063.5	18296.3	178.4	3.37	93.9
LA02	(5,10)	21223	21626.4	330.1	22.96	11.3	17661	17967.5	**157.8**	2.16	38.2	17703.4	18113.7	243.2	2.99	93.4
LA03	(5,10)	19269.9	19624.5	227.7	26.12	11.5	15679.6	15913.4	**219.2**	2.27	38.7	15667.5	16009.1	241.0	2.89	92.9
LA04	(5,10)	19085.3	19620.1	216.7	22.60	11.5	16039.7	**16209.3**	**136.1**	**1.29**	38.0	16131.4	16590.2	365.3	3.67	92.2
LA05	(5,10)	17424.2	17732.9	228.1	23.13	11.6	14442.6	**14608**	**97.8**	**1.43**	38.2	**14402.2**	14663.6	152.1	1.82	91.0
LA06	(5,15)	30332.6	31156.2	428.6	26.83	21.1	25198.4	25501	**134.8**	3.81	68.2	24608.2	24998.3	289.6	1.76	142.0
LA07	(5,15)	29614.2	30278.9	422.5	28.24	21.2	24172.9	24619.2	203.1	4.27	66.3	23994.2	24233.7	**201.5**	2.64	143.0
LA08	(5,15)	29442.4	30591.2	463.1	28.26	21.3	24422.8	24775.7	**202.1**	3.87	67.6	24049	24411.6	255.0	2.35	143.0
LA09	(5,15)	32652.8	33383.0	476.8	25.07	21.3	27329.5	27608.6	**122.5**	3.44	73.5	26769.5	27167.7	227.4	1.79	142.3
LA10	(5,15)	30753.2	31272.7	276.3	26.39	21.2	25081.3	25315.5	178.9	2.32	73.2	24954.6	25190.8	**171.9**	1.81	142.3
LA11	(5,20)	40691.1	42380.5	694.1	28.20	29.9	34221.7	34659.8	**258.5**	4.84	110.1	33061.5	33951.6	427.4	2.70	194.9
LA12	(5,20)	36116.2	36705.9	364.6	25.36	30.2	29905.6	30221.5	173.9	3.21	111.6	29771.8	29963	**138.2**	2.33	194.4
LA13	(5,20)	40046	41036.5	377.0	27.30	30.8	33198.1	33490.7	216.0	3.89	110.2	32480.4	32863.3	331.5	1.95	194.3
LA14	(5,20)	42035.5	42668.4	307.0	29.70	30.3	33794.5	34214.3	242.8	4.01	114.0	33320	33586.7	308.5	2.10	194.5
LA15	(5,20)	44265	44865.7	406.6	30.59	30.6	35802.4	36199.8	203.0	5.36	113.1	**34357.1**	35223.6	612.3	2.52	197.8
LA16	(10,10)	39352.3	40191.7	505.5	35.20	28.0	31516.7	32118.2	**397.7**	8.04	108.9	30132.7	**30816.4**	485.3	**3.66**	191.1
LA17	(10,10)	34319.1	35011.7	475.3	34.20	27.9	27606.6	28166	**377.8**	7.96	110.1	26369.2	27220.8	457.7	4.34	197.9
LA18	(10,10)	37326.8	38536.1	611.9	32.63	27.9	30826.6	31480.9	**525.1**	8.35	107.5	29223.3	**29910.9**	604.4	**2.94**	207.9
LA19	(10,10)	38359.7	39177.2	405.7	34.81	27.7	31167.9	31653.8	352.9	8.92	108.5	29713.3	**30155**	**304.2**	**3.76**	200.5
LA20	(10,10)	40626.2	41834.9	596.4	35.12	28.1	32675.2	33359.4	384.3	7.75	104.1	31199.9	**31902.3**	**376.7**	**3.04**	205.1
LA21	(10,15)	57276.3	58907.5	656.6	45.86	52.5	46139.7	46595.8	**256.6**	15.37	190.7	41154.1	41940.3	556.7	3.85	332.3
LA22	(10,15)	51457.2	53373.3	924.2	42.86	50.9	41378.6	42185.1	**556.5**	12.92	193.9	37762.9	38796.7	1010.0	3.85	319.3
LA23	(10,15)	56499	57683.4	682.8	45.05	52.0	44848.3	45655.5	626.4	14.81	195.5	40388.5	41631.4	764.1	4.69	316.8
LA24	(10,15)	54503	55457.0	716.7	43.86	53.2	41747.3	43487.9	869.3	12.81	191.3	38790	**39473.6**	**643.8**	**2.40**	345.4
LA25	(10,15)	54128.9	51823.4	496.6	38.05	51.8	43328	43686.9	**364.9**	16.38	185.7	39060.7	39869.8	791.8	6.21	377.7
LA26	(10,20)	75180.8	75817.0	547.6	48.22	80.9	58956.5	60361.9	792.4	18.00	292.0	51958.6	53214.7	821.9	4.03	492.7
LA27	(10,20)	76673	78085.6	807.9	48.38	86.4	60345.2	61834.3	**701.7**	17.50	287.3	54367.2	55272.2	763.5	5.03	498.7
LA28	(10,20)	74918.6	76472.9	834.1	48.68	85.3	59361.1	60933.3	**580.5**	18.47	282.0	52113.2	53437.6	776.7	3.90	492.4
LA29	(10,20)	70744.8	72150.4	1041.7	47.41	84.8	56971.4	58028.9	934.3	18.56	289.8	50192	51091.4	**713.6**	4.38	497.8
LA30	(10,20)	76737	77547.9	659.0	47.77	85.1	60849.4	61970.9	753.8	18.09	280.0	54079.1	55096.9	**560.3**	4.99	471.6
LA31	(10,30)	106691.4	108275.1	979.2	50.63	168.9	85121.3	86301.1	**551.2**	20.06	588.0	74851.4	75829.7	837.1	5.49	655.5
LA32	(10,30)	115254.6	117103.4	859.6	53.54	165.9	91552	92848.3	1208.2	21.74	570.5	79481.4	81324.8	1656.6	6.63	689.4
LA33	(10,30)	104826.8	106660.0	957.1	50.84	172.8	83406.5	84898	667.4	20.07	557.6	73743.9	74313.4	**633.4**	5.10	711.1
LA34	(10,30)	106811.5	108573.6	734.5	49.99	172.7	85921.4	87369.6	1379.5	20.69	578.7	75243	76945.8	1491.6	6.29	688.7
LA35	(10,30)	107670.7	110468.2	1210.6	51.11	190.6	87327.1	89208.7	1336.9	22.03	563.3	74189.2	76282.6	1070.9	4.35	686.4
LA36	(15,15)	87865.1	89417.9	1058.2	46.55	109.2	70307.6	71876.5	1115.8	17.80	337.9	61018.2	62523.4	**942.7**	2.47	498.9
LA37	(15,15)	94954.6	96887.6	1033.8	47.48	113.2	76600.2	78049.8	1180.2	18.81	346.1	66576.3	67606.5	**440.1**	2.91	477.6
LA38	(15,15)	84290.9	86247.6	1066.8	45.58	112.0	68805.9	69910.3	1069.2	18.00	347.2	59392.1	60532.4	**812.6**	2.17	476.5
LA39	(15,15)	86940.2	88721.3	997.1	47.76	109.0	69951.6	71414.4	940.3	18.94	348.0	61223.9	62166.7	**753.5**	3.53	492.8
LA40	(15,15)	86335.6	87432.5	815.8	44.83	108.9	69063.5	70353.9	1110.1	16.54	363.3	61273.2	62438.5	1076.8	3.43	473.0
RM01	(5,5)	11425.8	11599.2	141.7	4.82	5.3	**11065.9**	**11186.1**	**126.7**	**1.09**	16.9	11208.8	11543.3	250.7	4.31	47.2
RM02	(5,10)	18524.6	18856.6	178.2	11.86	12.8	**16856.8**	**17206.5**	154.8	**2.07**	41.0	17040.8	17246.1	**135.1**	2.31	92.2
RM03	(5,15)	28996.3	29424.6	261.7	13.93	23.8	26329.6	26627.1	195.4	3.10	73.0	26046.6	26284.5	188.3	1.78	143.3
RM04	(5,20)	40857.2	41582.4	486.2	15.64	36.1	37604	37801.4	**168.9**	5.12	112.8	36198.2	36626.9	285.3	1.86	205.5
RM05	(5,30)	62680.8	64546.7	779.4	16.47	68.4	57890.5	58627.8	395.3	5.79	208.0	56738.4	57205.3	**382.6**	3.22	328.5
RM06	(10,5)	18926.7	19167.8	160.5	6.86	13.2	**17937.4**	**18155.8**	**148.6**	**1.22**	42.0	18121.3	18735.8	477.5	4.45	94.5
RM07	(10,10)	37756.2	38578.3	594.4	22.60	32.6	33911.5	34552	657.2	9.80	112.0	32261.6	33093.6	772.4	5.17	192.2
RM08	(10,15)	57866.6	58725.2	512.7	31.00	61.0	51857.7	52543.2	518	17.21	201.9	46093.5	47046.4	748.3	4.95	324.6
RM09	(10,20)	73756	75057.9	741.8	33.12	95.7	65249.5	66822.6	819.4	18.51	328.7	**56383.4**	59223.7	1764.4	5.04	554.8
RM10	(10,30)	112312.2	114081.9	795.9	35.57	192.6	99214	101890.4	1298.5	21.08	666.6	87826.9	88696.9	851.1	5.40	738.1
RM11	(15,5)	31800.9	32296.2	486.0	7.29	22.6	30129.7	**30298.2**	**138.1**	**0.66**	77.2	30586.9	31437.2	671.6	4.44	155.3
RM12	(15,10)	60531.6	62637.4	1260	22.82	63.5	55789.3	56397.8	**511.8**	10.58	196.4	51731.8	52838.6	822.8	3.61	294.4
RM13	(15,15)	85296.1	88169.3	1469.2	30.61	115.2	78038.9	79713.7	1544.4	18.08	388.5	67616.9	69777.9	1574.3	3.37	460.8
RM14	(15,20)	115742.8	118339.6	1604.2	35.60	179.7	105564.3	107807.7	1793.2	23.53	631.8	87512.6	90517.7	1728	3.72	649.0
RM15	(15,30)	169465.3	172683.8	1892.7	42.29	359.6	156402.8	158061.8	**1214.2**	30.24	1272.1	129614	132654.9	1359.6	9.31	1030.1
RM16	(20,5)	45491.4	46721.6	577.3	7.92	36.6	**43293.6**	**43718.4**	**262.9**	**0.98**	118.2	43860.2	44674.3	549.8	3.19	199.5
RM17	(20,10)	79231.6	82457.3	1452.3	22.78	100.8	72951.6	74433.9	**693**	10.83	336.7	67770.9	70181.6	1045.2	4.50	433.4
RM18	(20,15)	120795.7	123868.0	1432.2	34.33	185.1	108791.8	112229	2600.8	21.70	618.9	95928.8	97375.4	**980.4**	5.60	665.7
RM19	(20,20)	158133.4	159393.4	1003.3	38.34	296.2	141507.4	144207.6	2339.5	25.16	1005.0	120275.5	121577.6	1529.1	5.52	905.9
RM20	(20,30)	231954.5	233093.3	696.0	42.59	588.2	212624.1	214974.4	1363.5	31.51	1993.7	176352.9	180034.2	2979.6	10.13	1437.1

TABLE 1. *(Continued.)* Comparison with the published algorithms.

RM21	(30,5)	81395.4	82257.6	611.4	6.59	73.6	**77172.1**	**77737.6**	**497.8**	**0.73**	236.4	78913.4	80102.1	723.3	3.80	338.5
RM22	(30,10)	135785.8	140195.5	2282.9	20.84	197.2	125543.8	128826.8	2181.9	11.04	676.3	119030.4	120992.5	**1612.6**	4.29	695.7
RM23	(30,15)	196648.6	198015.2	1166.8	32.14	388.4	174621	179782.5	3416.9	19.97	1264.5	154007.1	157760.1	2765.9	5.27	1073.1
RM24	(30,20)	248795.5	250921.9	1422.1	36.34	595.6	223404	228851.2	3805.7	24.35	1997.2	192984.1	197311.3	2634.5	7.21	1703.2
RM25	(30,30)	355072.7	356940.9	1539.2	41.58	1181.7	323110.6	326742.4	1619.5	29.61	4131.5	279035.3	284158.3	3134	12.71	2437.9
Mean	-	72599.3	73852.2	733.5	32.41	111.1	62429.7	63554.7	752.5	11.84	376.8	56053.6	57199.1	805	4.03	423.9

Instance	(m,n)	BDBA(W-B)					BDBA(R-R)					BDBA(R-B)				
		Best	*Avg*	*SD*	*ARPD*	*Time(s)*	*Best*	*Avg*	*SD*	*ARPD*	*Time(s)*	*Best*	*Avg*	*SD*	*ARPD*	*Time(s)*
FT06	(6,6)	1384.7	1413.1	20	3.12	42.1	1380.7	1421	27.7	3.69	43.9	1395.7	1419.5	17.1	3.58	46.7
FT10	(10,10)	30777.8	31323.8	**393.8**	3.58	129.5	**30240.4**	**30992.5**	586.6	**2.49**	135.8	30666	31462.5	493.2	4.04	140.5
FT20	(5,20)	33867.1	35049.8	849	3.76	133.4	**33778.7**	34852	539.2	3.18	137.9	34087.9	**34811.4**	482.7	**3.06**	144.7
LA01	(5,10)	**17699.6**	**18045.9**	178.4	**1.96**	59.5	18028.4	18460	283.4	4.30	62.8	17902	18284.7	247	3.31	64.8
LA02	(5,10)	17588	17996.8	330.1	2.33	61.1	**17587.8**	17971.2	245.3	2.18	61.8	17650.4	**17939.3**	231.5	**2.00**	64.8
LA03	(5,10)	15899.5	16223	220	4.26	60.1	**15559.6**	**15868.9**	192.6	**1.99**	62.1	15893.6	16102.1	181.4	3.49	65.4
LA04	(5,10)	**16003.4**	16449.6	327.7	2.79	61.1	16084.1	16393.4	222.6	2.44	60.9	16015.3	16511.1	366.1	3.17	64.8
LA05	(5,10)	14439.7	14733	165.8	2.30	60.6	14563.2	14758	141	2.47	60.3	14590.1	14714.3	74.4	2.17	64.5
LA06	(5,15)	24610.1	**24907.7**	198.3	**1.39**	95.4	24749.2	24990.1	182.2	1.73	95.7	**24566**	24951.9	273.4	1.57	102.5
LA07	(5,15)	23675	**24063.3**	265.8	**1.92**	95.8	**23610.4**	24310.3	525.5	2.96	101.2	23870.5	24322.1	342.2	3.01	100.9
LA08	(5,15)	23920	**24306.5**	374.1	**1.91**	94.6	23902.4	24400.3	296.9	2.30	99.6	**23851.7**	24369.7	293	2.17	97.1
LA09	(5,15)	26710.4	27075.4	246.3	1.44	94.4	26788	27070.9	271.8	1.42	102.5	**26691.2**	**26967.7**	307.4	**1.04**	96.4
LA10	(5,15)	24835.6	25286.8	268.9	2.20	93.2	24823.9	25152.6	258.6	1.66	96.9	**24742.2**	**25082.1**	306.2	**1.37**	99.5
LA11	(5,20)	33193.9	33852.1	438.7	2.40	132.0	**33059.1**	33733.1	369.4	2.04	136.2	33131	**33719**	595.7	**2.00**	131.5
LA12	(5,20)	29298.3	29732.1	276.8	1.54	130.2	**29281.2**	29748.3	303.1	1.60	139.8	29494.7	**29717.8**	201	**1.49**	131.4
LA13	(5,20)	32294.5	32649.7	379.1	1.28	128.4	**32236.2**	**32609**	204.2	**1.16**	144.8	32360.7	32789	387.5	1.71	136.2
LA14	(5,20)	33023.2	33479.6	382.8	1.77	132.5	32957.5	**33352.3**	262.1	**1.39**	139.4	**32896.6**	33382.7	287.1	1.48	135.5
LA15	(5,20)	34907.6	35427.2	340.3	3.11	140.9	34585	**34917.8**	314.3	**1.63**	137.9	34831.7	35071.4	280.1	2.08	136.5
LA16	(10,10)	29826.3	30909.3	678.4	3.97	124.8	30111.9	30891.2	502.9	3.91	129.9	**29728.1**	30939	669.2	4.07	131.1
LA17	(10,10)	26773.8	27253.7	405.8	4.46	125.6	**26089.5**	**27120.3**	769.1	**3.95**	129.7	26972.4	27235	173.8	4.39	129.5
LA18	(10,10)	29692.4	30470.1	789.8	4.87	124.4	29108	29954.7	479.7	3.09	131.0	**29056**	29965	682.6	3.13	129.7
LA19	(10,10)	29273.2	30255.3	451	4.11	127.5	29275.1	30371.4	601.5	4.51	132.7	**29062.1**	30182.7	839.6	3.86	128.1
LA20	(10,10)	31427.9	32566.7	854.6	5.19	132.0	**30961.3**	32127.2	758.3	3.77	132.7	31297.1	32218.1	746.2	4.06	127.8
LA21	(10,15)	41098	42236.3	777.9	4.58	201.8	40720.8	42148.1	802.9	4.36	206.4	**40387.3**	**41887**	893.8	**3.71**	201.2
LA22	(10,15)	**37359.5**	38552.2	762.5	**3.19**	201.3	37657.4	**38505.7**	748.1	**3.07**	207.3	38385.7	39289.1	629	5.16	196.8
LA23	(10,15)	40500.5	41120.4	**476.9**	3.40	200.1	**39766.7**	**40840.1**	624.3	**2.70**	206.1	40332.2	41673	719.1	4.79	199.0
LA24	(10,15)	**38549.5**	40190.4	1305.8	**4.26**	196.7	38595.4	39608.6	669.4	2.75	208.8	38636.6	39554.4	904.2	2.61	215.9
LA25	(10,15)	38201.6	39458.4	836.5	5.11	196.7	**37538.6**	**39140**	1125.7	**4.27**	206.0	38051.8	39451.5	993.9	5.10	213.9
LA26	(10,20)	51365	52364.5	**721.4**	2.37	278.3	51272.4	**52096.4**	486.4	**1.84**	296.0	**51153.1**	52260.5	680.5	2.16	307.5
LA27	(10,20)	53390.4	54347.3	770.4	3.27	295.8	52863.3	54048.6	718.1	2.70	303.0	**52625.1**	**53931**	757.4	**2.48**	298.0
LA28	(10,20)	51880.9	53586.2	884.3	4.18	280.3	**51433.7**	**52884.9**	1051.9	**2.82**	315.4	51769.7	53287.9	978	3.61	291.3
LA29	(10,20)	49241.1	50571.7	816.8	3.32	281.7	49668.7	50795.9	604.2	3.78	307.2	**48946**	**50246.9**	1100.1	**2.66**	287.7
LA30	(10,20)	53389.4	54578.2	595.3	4.00	284.0	**52479.4**	54560	1132.1	3.96	309.3	52864.9	**54114.8**	746.2	**3.12**	291.4
LA31	(10,30)	**71882.2**	74361.4	1004.3	**3.45**	506.9	71992.8	74263.4	1321.5	3.31	506.8	72515.2	**74246.5**	1086.2	**3.29**	460.0
LA32	(10,30)	77109	78419.9	**909.3**	2.82	483.5	**76266.7**	78602.1	1445.8	3.06	499.1	77431.4	**78389.4**	1025.8	**2.78**	474.3
LA33	(10,30)	**70708.7**	72414.7	1073.6	**2.41**	505.7	71146.1	**72014.5**	585.2	**1.85**	502.5	70787.8	72171.3	1125.1	2.07	458.9
LA34	(10,30)	**72389.6**	74257.7	1042.4	**2.58**	483.9	73090.8	**73784.6**	842.2	**1.93**	511.6	73740.9	74678.7	781.3	3.16	460.4
LA35	(10,30)	73366.4	**74818.7**	**1021**	2.35	500.0	73250.8	75107.2	1352.9	2.74	520.9	**73103.8**	75245.4	1435	2.93	459.5
LA36	(15,15)	**61013.9**	**62268.3**	1085.8	**2.06**	336.6	61115.3	62737.8	1233.1	2.83	345.3	61676.1	63148.1	1007.5	3.50	322.6
LA37	(15,15)	66447.4	**67533.4**	631.5	2.80	331.0	**65694.4**	67627.9	1442.1	2.94	341.7	66177.6	67738	936.2	3.11	323.4
LA38	(15,15)	59380.8	60939.8	1226	2.86	345.8	**59244.7**	**60235.2**	647.7	**1.67**	348.9	59392.1	60535.9	1069.6	2.18	325.7
LA39	(15,15)	**60044.3**	61683.3	1047.7	**2.73**	345.0	60362.9	**61386.9**	821.1	**2.24**	334.4	60497	61867.2	1107.3	3.04	357.5
LA40	(15,15)	**60367.8**	61940.6	900.9	**2.61**	336.1	60740.8	**61926.9**	1106.1	**2.58**	335.2	60855.1	62094.7	1048.2	2.86	360.6
RM01	(5,5)	11424	11689.9	218.4	5.64	27.3	11318.5	11579.5	213.4	4.64	29.9	11307.5	11787.8	310.8	6.52	29.9
RM02	(5,10)	17084.5	17305.6	154	2.66	57.6	17030.7	17360.9	182.1	2.99	61.5	16864.5	17332.6	293.4	2.82	63.9
RM03	(5,15)	25931.8	**26122.2**	**139.1**	1.15	89.4	**25825.9**	26126.3	158	1.16	103.3	25880.7	26223.9	226.6	1.54	100.5
RM04	(5,20)	36022.7	36543.2	282.8	1.62	126.3	36171.4	36630.6	384.5	1.87	137.4	**35958.9**	**36494.9**	367.4	**1.49**	145.5
RM05	(5,30)	55986.5	56369.4	388.8	1.71	211.0	**55419.2**	**56206.6**	591	**1.42**	226.0	55865.4	56509.3	595	1.97	233.3
RM06	(10,5)	18300.8	18733.3	361.5	4.44	61.6	18522.4	18862.9	309.9	5.16	63.1	18602.3	19049.2	339.3	6.20	66.7
RM07	(10,10)	32033.4	32839.6	655.5	4.36	132.2	32078.4	32894.3	639.6	4.53	128.9	**31467.9**	**32768**	876.7	**4.13**	131.8
RM08	(10,15)	**44828.3**	46757.9	902.3	**4.30**	207.2	45760.1	46700.1	934.7	4.18	209.7	45356.3	**46481.3**	578.6	**3.69**	211.3
RM09	(10,20)	57086.9	58617.5	1000.3	3.96	309.7	56598	57961	1196.7	2.80	296.9	56835.4	**57856.3**	**727.2**	**2.61**	296.4
RM10	(10,30)	85191.2	87042	1426	3.44	527.8	**84150.6**	**85990.4**	1153.6	**2.19**	506.7	85730.5	86432.3	**704**	2.71	483.7
RM11	(15,5)	30344.5	31448.5	912.1	4.48	105.8	30300.3	31272.2	671.7	3.89	102.4	**30100.6**	31148.9	705.7	3.48	100.4
RM12	(15,10)	**50999.7**	52860.9	1276.6	**3.65**	224.7	51795.1	**52634.4**	665.3	**3.21**	214.3	51531.8	52724.6	938.6	3.38	214.5
RM13	(15,15)	**67505.9**	**68624.2**	817.4	**1.66**	356.4	67613.3	68675.3	**665.4**	1.73	352.5	68132.5	69807.4	789.1	3.41	339.0
RM14	(15,20)	88302.6	89535.8	**1044.8**	2.60	503.6	**87269.4**	**89470.2**	1283	**2.52**	503.8	87635.4	89635.5	1727.3	2.71	475.7

TABLE 1. *(Continued.)* **Comparison with the published algorithms.**

RM15	(15,30)	**121361.2**	125028.6	2391.6	**3.02**	859.6	122471	**124946.8**	1509.7	**2.95**	803.4	122239.7	125081.8	1495.8	3.07	782.0
RM16	(20,5)	43681.7	44495.2	409.7	2.78	144.0	43533.2	44473.6	665.8	2.73	159.0	44135.7	45121.1	988.9	4.22	148.7
RM17	(20,10)	67687.9	**69453.8**	838.7	3.42	310.5	**67158.7**	69588.4	1832	3.62	348.5	67339.6	69806.6	1357.6	3.94	312.8
RM18	(20,15)	93794.2	95978.2	1531.7	4.08	490.0	**92214.6**	**94409.3**	1412.6	**2.38**	533.7	92529.9	95542.4	2494.5	3.61	497.1
RM19	(20,20)	115455	118883.9	2100.3	3.18	679.8	115743.6	**117463.9**	1097.7	**1.95**	728.6	**115221.3**	117625.8	1602	2.09	688.0
RM20	(20,30)	165973.4	167777.1	1350.4	2.63	1108.6	163644.2	**166007.5**	1446.6	**1.55**	1164.6	**163472.2**	166905	2196.5	2.10	1089.7
RM21	(30,5)	79321	81011.4	1090.7	4.97	248.2	78649.3	80782.2	982.9	4.68	420.8	78601	79900.3	839.8	3.54	241.9
RM22	(30,10)	**116020**	**119203.9**	1970.3	**2.74**	533.0	117495.7	119332.5	1667.3	2.86	539.5	116776.2	119539.4	1695.7	3.03	504.2
RM23	(30,15)	152673.6	156058.3	3429.7	4.14	797.5	150642.8	154625.8	2385	3.18	804.4	**149858.1**	**154263.8**	2215.8	**2.94**	807.9
RM24	(30,20)	184137.3	**188172**	2147.7	2.25	1089.4	187029	189903.3	1650.1	3.19	1124.5	**184040**	188761.3	2576.3	2.57	1136.0
RM25	(30,30)	**252105.9**	259468.9	3541.8	**2.92**	1723.8	254209.8	**259267.5**	2500.6	**2.84**	1829.9	256424.9	260897.8	3347.4	3.49	1839.8
Mean	-	54736.2	56016.7	839.8	3.11	292.6	54651.6	55837.5	783.8	2.82	304.1	54735.2	55965.7	845.5	3.06	292.9

algorithm obtains the lowest mean value of *Avg* among all the compared algorithms. (3) In comparisons of the '*SD*' value, the IWOA algorithm yields 27 optimal values, which is more than the other compared algorithms. According to the last row of Table 1, the GA algorithm obtains the lowest mean value of *SD* among all the compared algorithms. For the proposed BDBAs, BDBA(R-R) outperforms the other two algorithms. (4) In comparisons of the '*ARPD*' value, the BDBA(R-R) algorithm yields 25 optimal values, which is more than the other compared algorithms. According to the last row of Table 1, the BDBA(R-R) algorithm obtains the lowest mean value of *ARPD* among all the compared algorithms. (5) In comparisons of the '*Time*' value, GA spends the shortest time among the compared algorithms. The proposed BDBAs spend the shorter time than SDBA and IWOA.

An analysis of variance (ANOVA) test is performed in Table 2, where the compared algorithms are viewed as levels and ARPD is taken as the response variable. The results show that there are significant differences among the algorithms because p-value is equal to zero.

TABLE 2. **ANOVA for ARPD of the compared algorithms.**

Source	DF	Sum of Squares	Mean Square	F	p-value
Factor	5	46732.8611	9346.57222	229.48409	0
Error	402	16372.90849	40.72863		
Total	407	63105.76959			

VI. CONCLUSIONS

In this paper, a kind of bi-population based discrete bat algorithm (BDBA) is developed to solve the low-carbon job shop scheduling problem with the consideration of energy consumption. The main contribution of this study are shown 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.

Extensive experiments are carried out to test the performance of the proposed BDBA. The comparison data show that: (1) the parallel search mechanism is effective for improving the performance of the algorithm. (2) the proposed BDBA(R-R) algorithm can obtain better results for most of the considered indicators, such as *Best*, *Avg* and *ARPD*. (3) For the three different BDBAs, the BDBA(R-R) performs better than the other two algorithms with a little longer computational time.

In the future work, some constraints will be considered in the low-carbon JSP, such as flexible processing routing, variable energy prices, and renewable energy, and so on. In addition, the low-carbon scheduling problem will be extended to some complex workshops, such as flexible job shop and assembly job shop, and so on.

REFERENCES

[1] IEA. (2008). *Worldwide Trends in Energy Use and Efficiency: Key Insights From IEA Indicator Analysis*. Accessed: Jun. 29, 2013. [Online]. Available: http://www.iea.org/Textbase/Papers/2008/indicators_2008.pdf

[2] G. May, B. Stahl, and M. Taisch, "Energy management in manufacturing: Toward eco-factories of the future—A focus group study," *Appl. Energ.*, vol. 164, pp. 628–638, Feb. 2016.

[3] T. Peng and X. Xu, "Energy-efficient machining systems: A critical review," *Int. J. Adv. Manuf. Technol.*, vol. 72, nos. 9–12, pp. 1389–1406, Jun. 2014.

[4] Y. Liu, H. B. Dong, N. Lohse, S. Petrovic, and N. Gindy, "An investigation into minimising total energy consumption and total weighted tardiness in job shops," *J. Cleaner Prod.*, vol. 65, pp. 87–96, Feb. 2014.

[5] T. Jiang, C. Zhang, H. Q. Zhu, J. Gu, and G. Deng, "Energy-efficient scheduling for a job shop using an improved whale optimization algorithm," *Mathematics*, vol. 6, no. 11, pp. 220–239, Nov. 2018.

[6] T. Jiang and G. Deng, "Optimizing the low-carbon flexible job shop scheduling problem considering energy consumption," *IEEE Access*, vol. 6, pp. 46346–46355, Aug. 2018.

[7] 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.

[8] C. Liu, J. Yang, J. Lian, W. Li, S. Evans, and Y. Yin, "Sustainable performance oriented operational decision-making of single machine systems with deterministic product arrival time," *J. Cleaner Prod.*, vol. 85, pp. 318–330, Dec. 2014.

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

[10] 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.

[11] S. Rubaiee, S. Cinar, and M. B. Yildirim, "An energy-aware multiobjective optimization framework to minimize total tardiness and energy cost on a single-machine nonpreemptive scheduling," *IEEE Trans. Eng. Manage.*, to be published, doi: 10.1109/TEM.2018.2846627.

[12] M. Dai, D. B. 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.
[13] H. Luo, B. Du, G. Q. Huang, H. 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.
[14] H. Zhang, F. Zhao, K. Fang, and J. W. Sutherland, "Energy-conscious flow shop scheduling under time-of-use electricity tariffs," *CIRP Ann.*, vol. 63, no. 1, pp. 37–40, Apr. 2014.
[15] W. Lin *et al.*, "A multi-objective teaching–learning-based optimization algorithm to scheduling in turning processes for minimizing makespan and carbon footprint," *J. Cleaner Prod.*, vol. 101, pp. 337–347, Aug. 2015.
[16] D. Tang, M. Dai, M. A. Salido, and A. Giret, "Energy-efficient dynamic scheduling for a flexible flow shop using an improved particle swarm optimization," *Comput. Ind.*, vol. 81, pp. 82–95, Sep. 2016.
[17] C. Lu, L. Gao, X. Li, Q. Pan, and Q. Wang, "Energy-efficient permutation flow shop scheduling problem using a hybrid multi-objective backtracking search algorithm," *J. Cleaner Prod.*, vol. 144, pp. 228–238, Feb. 2017.
[18] J.-J. Wang and L. Wang, "A knowledge-based cooperative algorithm for energy-efficient scheduling of distributed flow-shop," *IEEE Trans. Syst., Man, Cybern., Syst.*, to be published, doi: 10.1109/TSMC.2017.2788879.
[19] D. Lei, L. Gao, and Y. Zheng, "A novel teaching-learning-based optimization algorithm for energy-efficient scheduling in hybrid flow shop," *IEEE Trans. Eng. Manage.*, vol. 65, no. 2, pp. 330–340, May 2018.
[20] T. Jiang, C. Zhang, H. Zhu, and G. Deng, "Energy-efficient scheduling for a job shop using grey wolf optimization algorithm with double-searching mode," *Math. Problems Eng.*, vol. 2018, pp. 1–12, Oct. 2018.
[21] G. May, B. Stahl, M. Taisch, and V. Prabhu, "Multi-objective genetic algorithm for energy-efficient job shop scheduling," *Int. J. Prod. Res.*, vol. 53, no. 23, pp. 7071–7089, Jan. 2015.
[22] S. Kawaguchi and Y. Fukuyama, "Improved parallel reactive tabu search based job-shop scheduling considering minimization of secondary energy costs in factories," in *Proc. 57th Annu. Conf. Soc. Instrum. Control Eng. Jpn. (SICE)*, 2018, pp. 765–770.
[23] J. Escamilla, M. A. Salido, A. Giret, and F. Barber, "A metaheuristic technique for energy-efficiency in job-shop scheduling," *Knowl. Eng. Rev.*, vol. 31, pp. 475–485, Nov. 2016.
[24] T. Jiang and C. Zhang, "Application of grey wolf optimization for solving combinatorial problems: Job shop and flexible job shop scheduling cases," *IEEE Access*, vol. 6, pp. 26231–26240, May 2018.
[25] X.-S. Yang, "A new metaheuristic bat-inspired algorithm," in *Nature Inspired Cooperative Strategies for Optimization*. Berlin, Germany: Springer, 2010, pp. 65–74.
[26] X.-S. Yang and A. H. Gandomi, "Bat algorithm: A novel approach for global engineering optimization," *Eng. Comput.*, vol. 29, no. 5, pp. 464–483, 2012.
[27] D. Rodrigues *et al.*, "A wrapper approach for feature selection based on bat algorithm and optimum-path forest," *Expert Syst. Appl.*, vol. 41, no. 5, pp. 2250–2258, Apr. 2014.
[28] E. Osaba, X.-S. Yang, F. Diaz, P. Lopez-Garcia, and R. Carballedo, "An improved discrete bat algorithm for symmetric and asymmetric traveling Salesman problems," *Eng. Appl. Artif. Intel.*, vol. 48, pp. 59–71, Feb. 2016.
[29] M. Seyedmahmoudian *et al.*, "Maximum power point tracking for photovoltaic systems under partial shading conditions using bat algorithm," *Sustainability*, vol. 10, no. 5, pp. 1347–1363, May 2018.
[30] H. Xu and B. Cheng, "Hybrid genetic bat algorithm for the single-objective flexible job shop scheduling problem," *J. Chin. Comput. Syst.*, vol. 39, no. 5, pp. 1010–1015, May 2018.
[31] L. Wang, S. Wang, Y. Xu, G. Zhou, and M. Liu, "A bi-population based estimation of distribution algorithm for the flexible job-shop scheduling problem," *Comput. Ind. Eng.*, vol. 62., no. 4, pp. 917–926, May 2012.
[32] Q. Luo, Y. Zhou, J. Xie, M. Ma, and L. Li, "Discrete bat algorithm for optimal problem of permutation flow shop scheduling," *Sci. World J.*, vol. 2014, pp. 1–15, Aug. 2014.
[33] H. Fisher and G. L. Thompson, "Probabilistic learning combinations of local job-shop scheduling rules," *Ind. Scheduling*, vol. 3, pp. 225–251, Mar. 1963.
[34] S. Lawrence, "Resource constrained project scheduling: An experimental investigation of heuristic scheduling techniques," Graduate School Ind. Admin., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. 12, 1984.

YI LU was born in Hongqi, Xinxiang, China, in 1987. She received the B.A. degree in ecological planning from Hainan University, Haikou, China, in 2009, and the M.T.A. degree in tourism management from Guangxi Normal University, Guilin, China, in 2018.

From 2010 to 2016, she was a Teaching Assistant with the Henan Institute of Technology, Xinxiang, China, where she has been a Lecturer since 2016. She has authored two books and more than 10 articles. Her research interests include ecotourism, production scheduling, and intelligence algorithm.

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, 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 the *International Journal of Bio-Inspired Computation*, the IEEE Access, the *Journal of Intelligent and Fuzzy Systems*, the *International Journal of Industrial Engineering: Theory, Applications and Practice*, *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, *Mathematics*, and *Mathematical Problems in Engineering*.