Received October 4, 2020, accepted October 16, 2020, date of publication October 22, 2020,
date of current version December 3, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3033002

A Bottleneck Degree-Based Migrating Birds Optimization Algorithm for the PCB Production Scheduling

JUN CAO[1], ZAILIN GUAN[1], LEI YUE[1], SAIF ULLAH[1,2], AND REHAN AHMAD KHAN SHERWANI[3]
[1]State Key Laboratory of Digital Manufacturing Equipment and Technology, HUST–SANY Joint Laboratory of Advanced Manufacturing Technology, Huazhong University of Science and Technology, Wuhan 430074, China
[2]Department of Industrial Engineering, University of Engineering and Technology, Taxila 47080, Pakistan
[3]College of Statistical and Actuarial Sciences, University of the Punjab, Lahore 54590, Pakistan

Corresponding author: Saif Ullah (cao96qc@hust.edu.cn)

This work was supported in part by the China Postdoctoral Science Foundation under Grant 2019M652665, and in part by the National Natural Science Foundation of China under Grant 51905196.

ABSTRACT To address the printed circuit board shop scheduling problem, a dynamic based on multi-indicator bottleneck degree is proposed to formulate this complex flexible job shop scheduling problem (FJSP). Then, a bottleneck degree-based migrating birds optimization algorithm (MBO) is proposed to tackle this NP-hard problem. Specifically, a multi-indicator assessment model is first developed to obtain the bottleneck degree, and then a two-layer code-decoding structure based on the variable domain structure and the addition of the improved migratory bird algorithm with competitive mechanisms are employed to improve the effectiveness. Finally, the computational experiments demonstrate that the proposed dynamic scheduling approach based on bottleneck degrees and improved migrating bird algorithms can obtain promising results and it outperforms several other algorithms.

INDEX TERMS Bottleneck degree, dynamic scheduling, migrating bird algorithm, printed circuit board shop.

I. INTRODUCTION

A printed circuit board (PCB) is an interconnected electronic component utilized in many electronic appliances ranging from simple routers to sophisticated computer systems. Due to a wide variety of manufactured products, most of the PCB factories are operated in make-to-order production mode [1], [2]. For instance, a PCB manufacturing factory located in south China mainly handles orders involving a wide range of diversified and customized products. Each product may have a unique processing route for tens of procedures and there is more than one homogeneous machine for each procedure. All those make the production scheduling within the PCB factory become into a typical flexible job shop scheduling problem [3]–[5].

Due to the wide variety of products, capacity demand for machines changes all the time. Even worse, owing to the shortened production cycle, most products need to be manufactured simultaneously, which adds an extra burden to the core machines. As a result, the current bottleneck of the workshop keeps changing. The number of bottlenecks may change continuously and the degree of bottlenecks is always unknown. To improve the overall efficiency of the workshop, it is necessary to make clear the status of all related machines. On the other hand, limiting the production capacity according to the bottleneck process may lead to overload or fluctuating delays in the execution of some processes, resulting in some order delays. However, to the best of our knowledge, rare literature has been focused on the production scheduling with bottlenecks within PCB factories and hence this paper aims to solve this problem to achieve satisfactory performance [6], [7].

Till now, exhaustive literature has been published on PCB scheduling and they may be partitioned into two categories. The first focuses on the improvement of specific processes via sequencing. For example, Rahmani and Mahdavi [8]

The associate editor coordinating the review of this manuscript and approving it for publication was Muhammad Zakarya.

developed and validated a three-step scheduling method for inserting or placing components on a PCB by an automated inserter. Sabouni and Logendran [9] considered to minimize the completion time of an important machine in the PCB process using a single-machine scheduling method. Pollett [10] solved the problem of nozzle distribution, part pick and place order. The second category considers the production scheduling in PCB factories. Li [11] proposed an integrated optimization of process planning and scheduling for PCB assembly process. Yan *et al.* [12] provided a better estimator for production planning and control of PCB assembly, including PCB job grouping, PCB batch-to-production line distribution, batch sequencing, and line load balancing. Obviously, bottlenecks are not considered in the above literature on PCB scheduling.

Although there is now considerable research in the manufacturing field, there is no current research on PCB scheduling that considers bottlenecks. There are three types of bottleneck-oriented production scheduling methods applied in manufacturing.

(1) When there is only one bottleneck in the production system, scheduling is done around the bottleneck with the bottleneck in the system as the core in order to improve the production efficiency of the bottleneck as much as possible [8].

(2) When there are multiple bottlenecks in the production system, usually the multiple bottlenecks are decomposed into multiple single bottleneck scheduling problems, and then each sub-bottleneck problem is solved separately to form a feasible scheduling scheme [13].

(3) Using a dynamic monitoring production system to dynamically evaluate the bottleneck conditions and scheduling according to the dynamic changes in the bottleneck often results in better scheduling results [14].

Traditional bottleneck scheduling methods (the first and second methods above) typically identify bottlenecks based on the production data generated after the production system has been run. After identifying the bottleneck, schedule is done around the bottleneck to maximize the production capacity of the production line. The disadvantage is that the bottleneck situation cannot be grasped in real time, and the resultant value is based on the optimal solution of the identified bottleneck, which is often not the optimal solution [11], [15]. However, regarding the third method: dynamic monitoring production system, it depends on the on-site data and does not bring extra production costs. Meanwhile, since this method considers the dispatch front-end state, it can ensure the schedule feasibility and improve productivity. Therefore, it is necessary to dynamically grasp the real time dynamics of the bottleneck and dynamically schedule the production process to avoid triggering oscillations and instability in the manufacturing system. Accordingly, this paper aims to use the third method to deal with printed circuit board shop scheduling problems.

To tackle the job shop scheduling problem, four types of procedures are proposed: exact, heuristic and meta-heuristic algorithms. Among them, exact algorithms can solve the problems to optimality but they are limited to the small-scaled [16]–[18]. Heuristics are simple to implement and easy to explain but their strongly problem-specific attribute may bring about huge fluctuations following tiny change in production [19], [20]. Meta-heuristic methods, the main research methods of flexible job shop scheduling involve particle swarm optimization (PSO) [21], genetic algorithm (GA) [22], [23], ant colony algorithm (ACO) [24], [25] and other hybrid algorithms. Hasani *et al.* [26] got genetic algorithms to the problem of scheduling a set of jobs on two parallel machines to minimize the makespan Gao *et al.* [27] developed a new genetic algorithm hybridized with an innovative local search procedure (bottleneck shifting) for the Flexible job shop scheduling problem. Wang *et al.* [28] adopted an improved ant colony algorithm to solve the scheduling problem of flexible manufacturing workshop.

Ding and Gu [29] proposed an improved particle swarm optimization algorithm for solving FJSP. Li *et al.* [30] developed a hybrid artificial bee colony algorithm based tabu search to solve flexible job shop scheduling problems in a modern manufacturing enterprise. Zhu *et al.* [31] adopted a memetic algorithm for a new low carbon flexible job shop scheduling problem by considering worker learning. Li *et al.* [32] proposed an improved Jaya algorithm to solve the flexible job shop scheduling problem based on time constraints and energy efficiency.

More recently, among the metaheuristics, migrating birds optimization algorithm (MBO), as a new meta-heuristic algorithm inspired by the V-shaped flight formation of migrating birds, has proved to be effective on energy conservation. This algorithm is unique where the benefit mechanism is utilized to replace the poor-quality solution and accelerate the evolution process greatly. Meanwhile, this algorithm has successfully applied in related combination optimization problems. For instance, Pan applied MBO to mixed flow shop scheduling [33]; Zhang *et al.* [34] studied the problem of hybrid flow shop hybridizing with lot streaming (HLFS) with the objective of minimizing the total flow time with Migratory bird optimization algorithm; Zhang *et al.* [35] solved the u-shaped assembly line balance problem and worker assignment problem with MBO; Li *et al.* [36] applied improved MBO to the mathematical model of robot U-shaped assembly Line Balance problem; Janardhanan *et al.* [37] solved the problem of assignment and balance of two-sided assembly line workers with migratory bird optimization algorithm. Hence this work aims to extend a new version of MBO to tackle this proposed problem.

In sum, this paper presents two contributions for solving the PCB as follows. 1) This study introduces the concept of the core bottleneck degree of shop scheduling, which effectively solves the problem of rational selection of machines and load distribution. 2) This study proposes a bottleneck degree-based migrating birds algorithm to tackle this PCB problem. In which, a two-layer coding and decoding mechanism, a variable neighborhood structure and a competition mechanism are employed to achieve the proper balance between exploration and exploitation. Computational study demonstrates that the proposed bottleneck degree-based migrating birds optimization (BMBO) has a greater performance compared with six other state-of-the-art algorithms.

The remainder of this paper is organized as follows. Section II presents the problem description and mathematical formulation. Subsequently, the proposed bottleneck degree-based migrating birds optimization algorithm is introduced in Section III. Section IV presents the experimental results and finally Section V concludes this study and suggests several future research directions.

II. PROBLEM STATEMENT AND FORMULATION

Recognizing real-time bottlenecks and correspondingly producing a schedule based on the recognized bottleneck are important for workshop to increase the machine utilization rate and production efficiency simultaneously. Hence, a novel approach is first proposed to discover the potential bottlenecks and to evaluate their bottleneck degree. Then, based on the evaluation, a dynamical scheduling mathematical model is formulated.

As the most popular dynamic scheduling, periodic rescheduling is featured on relative simplicity and reliability with no response to emergencies. Continuous rescheduling can deal with them with future events not foreseen in the overall concept. For the periodic and event-driven scheduling strategy, a scheduling event will be triggered to reschedule immediately when the bottleneck degree of one or more equipment in a production system is greater than 80 (see Table 7). Otherwise, wait until a production cycle is completed before scheduling.

A. EXPLICIT REPRESENTATION OF BOTTLENECKS

In the production system, the production equipment with the largest load is considered as the bottleneck [12]. Pollett [10] defined the equipment with the longest average processing queue wait time as a bottleneck. Roser *et al.* [38] used equipment duration active time as an indicator to identify instantaneous and average bottlenecks in discrete processing systems. They carried out bottleneck identification with machine load rate as indicator. Pegels and Watrous [39] proposed a method for seeking dynamic identification of bottlenecks with the number of work-in-progress as the object of study. Muthiah and Huang [40] integrated equipment productivity, utilization and quality efficiency trio as an indicator of system bottleneck identification.

Based on the above studies, this study identifies the following four indicators for the establishment of bottleneck degrees: machine utilization, machine capacity load ratio, machine uninterrupted activity time and machine down time.

B. MULTI-INDEX BOTTLENECK RECOGNITION

Provided that all machines in the production system may be bottlenecked, bottleneck degree is coined as an indicator to evaluate quantitatively the workload degree of these machines. The bigger the bottleneck degree, the greater the probability is for the corresponding machine to be bottlenecked. This study selects four attributes for evaluation: machine utilization (X_1); machine capacity load rate (X_2); uninterrupted activity time (X_3); machine down time (X_4).

Based on the attributes of a machine, the corresponding values of each machine for all the four attributes are indicated as, $X_l^k = (X_1^k, X_2^k, X_3^k, X_4^k)$, in which l denotes the evaluation index type and $l = 1, 2, 3, 4$; k denotes the serial number of machines for processing. The bottleneck degree of each machine is calculated by expressions (1-6) by using technique for order performance by similarity to ideal solution (TOPSIS) [41]–[43]. The equation (1) normalizes the given data of all the evaluation indexes to eliminate dimensional differences in different attributes.

$$Y_l^k = X_l^k \Big/ \sqrt{\sum_{k=1}^{K} (X_l^k)^2} \quad \forall k \in \{1, 2, \ldots, K\}, l = 1, 2, 3, 4 \tag{1}$$

Equations (2-3) screen the maximum and minimum scenarios for each evaluation index from obtaining data sets.

$$Y_l^+ = \max_k Y_l^k, \forall l \tag{2}$$

$$Y_l^- = \min_k Y_l^k, \forall l \tag{3}$$

Equations (4-5) use the Euclidean norm as a measure of distance, and calculate the distance from any feasible solution y_l^k to the maximum/minimum values.

$$s_k^+ = \sqrt{\sum_{l=1}^{L} \left(Y_l^k - Y_l^+\right)^2} \tag{4}$$

$$s_k^- = \sqrt{\sum_{l=1}^{L} \left(Y_l^k - Y_l^-\right)^2} \tag{5}$$

Based on the distance from positive and negative ideal solutions, the equation (6) is utilized to obtain the bottleneck degree of each machine w_k.

$$w_k = 100 \left(\frac{s_k^+}{s_k^+ + s_k^-} \right). \tag{6}$$

Using the equations (1-6), we can obtain the bottleneck degree of each machine $\{w_k\}$. Then, the values of bottleneck degrees will be regarded as an input into the scheduling system.

C. MATHEMATICAL FORMULATION

The dynamical PCB scheduling problem based on bottleneck degrees can be characterized by jobs, machines, and bottleneck degrees of these machines. Particularly, each job has one or more operations to be processed according to the given processing route. Each operation can be assigned to more than one optional machine with specific processing time. The goal of dynamical scheduling is to select a suitable machine from the optional set for each operation and determine the sequence of operations on each machine to eliminate potential bottlenecks and achieve higher production efficiency. The main assumptions of this model are first presented as follows.

(1) Each machine can perform at most one operation at a time.

(2) Each operation of a job should be performed exactly once while at most one of them can be performed at a time.

(3) All operations of a job should be performed subject to process route.

(4) Each job can be processed at zero.

(5) Interruption is not allowed once an operation of a job starts.

(6) Setup time before performing an operation is included in the processing time.

Indices and Sets:

l	The evaluation index type, (l = 1,2,3,4)
$k(k')$:	Machines, ($k = 1, 2, \ldots, K$).
$i(i')$:	Jobs, $\mathrm{I} = \{i = 1, 2, \ldots, I\}$.
$j(j'), vs_k(ve_k)$:	Operations, $\mathrm{J} = \{j = 1, 2, \ldots, J\}$, and virtual start / end operations on each machine.
M_{ij} :	Set of available machines for the jth operation of job i and $\lvert M_{ij}\rvert \geq 1$.
P_{ijk} :	Nominal processing time of the jth operation of job i on machine k.
w_k	Bottleneck degree of machine k.
U :	A large real number.
ε	Weight coefficient of objective.
z_1^*/z_2^*	The best result achieved for the objective function.

Variables:

X_{ijk}	Binary variable. Takes value 1 if the j^{th} operation of job i is allocated to machine k.
$R_{iji'j'k}$	Binary variable. Takes value 1 if on machine k, the j^{th} operation of job i is processed before the j'^{th} operation of job i'; and 0 otherwise.
S_{ijk} :	Continuous variable, start time of the j^{th} operation of job i on machine k.
E_{ijk} :	Continuous variable, completion time of the j^{th} operation of job i on machine k.
C_i :	Completion time of job i.
z_1, z_2, z_{Lp}	Free variable, objective functions corresponding to bottleneck degree, makespan and linearized weighted objective.

Utilizing the notations above, the dynamical scheduling problem based on bottleneck degrees can be described as follows:

$$z_1 = \min\,(\max_k w_k - \min_k w_k) \tag{7}$$

$$z_2 = \min\left\{\max_{\mathrm{i}}(C_{\mathrm{i}})\right\} \tag{8}$$

$$z_{Lp} = (1-\varepsilon)(\frac{z_1 - z_1^*}{z_1^*}) + \varepsilon(\frac{z_2 - z_2^*}{z_2^*}) \tag{9}$$

With regard to objectives, equation (7) tries to reduce the difference between the maximum and minimum bottleneck degrees to increase the smoothness of work allocation among machines and decrease the severity degree of bottlenecked machines. Equation (8) minimizes the manufacturing cycle time of a batch of jobs, also called makespan. Based on this, equation (10) calculates the normalized combination of these two objective values.

s.t.

$$\sum_{k=1}^{M_{ij}} X_{ijk} = 1 \;\; \forall i \in \mathrm{I}, j \in \mathrm{J} \tag{10}$$

$$\sum_{i'=1}^{\mathrm{I}}\sum_{j'=1}^{\mathrm{J}} R_{iji'j'k} = \mathrm{X}_{ijk} \;\; \forall i \in \mathrm{I}, j \in (\mathrm{J} \cup vs_k), k \in \mathrm{M}_{ij} \tag{11}$$

$$\sum_{i=1}^{\mathrm{I}}\sum_{j=1}^{\mathrm{J}} R_{iji'j'k} = \mathrm{X}_{i'j'k} \;\; \forall i' \in \mathrm{I}, j' \in (\mathrm{J} \cup ve_k) k \in \mathrm{M}_{i'j'} \tag{12}$$

$$E_{i'j'k} - E_{ijk} \geq P_{i'j'k} - U \times (3 - X_{ijk} - X_{i'j'k} - R_{iji'j'k})$$
$$\forall (i, i') \in \mathrm{I}, (j, j') \in \mathrm{J}, k \in M_{ij}, k \in M_{i'j'} \tag{13}$$

$$E_{i,j+1,k'} - E_{ijk} \geq P_{i,j+1,k'} - U \times (2 - X_{ijk} - X_{i,j+1,k'})$$
$$\forall i \in \mathrm{I}, j < \mathrm{J}, k \in M_{ij}, k' \in M_{i,j+1} \tag{14}$$

$$E_{ijk} - S_{ijk} \geq P_{ijk} - U \times (1 - X_{ijk})$$
$$\forall i \in \mathrm{I}, j \in \mathrm{J}, k \in M_{ij} \tag{15}$$

$$E_{ijk} - S_{ijk} \leq P_{ijk} + U \times (1 - X_{ijk})$$
$$\forall i \in \mathrm{I}, j \in \mathrm{J}, k \in M_{ij} \tag{16}$$

$$C_i = E_{ijk}.\forall i \in \mathrm{I}, k \in M_{ij}, j = \mathrm{J} \tag{17}$$

As for constraints, equation (10) requires that each operation of a job is performed exactly once by a machine. Equations (11-12) determine the sequence of all operations allocated to a machine with the help of its virtual start / end operations. Specifically, except for the virtual end operation, each operation has an immediately following one and similarly, except the virtual start operation, each operation has an immediately preceding one. Equations (13-16) determine the start times of all operations. Among them, equation (13) declares that if two operations are allocated to a machine, the next one can start after the completion of the previous one. Equation (14) demands the next operation of a job can start after the completion of its previous operation. Equations (15-16) calculate the completion time of an operation according to the specific processing time by the allocated machine. Equation (17) indicates that a job is completed at the moment of the completion time of its last operation.

The dynamical scheduling model based on the bottleneck degrees is formulated with equations (1-17). This model has been linearized and solved to optimality by the small-scaled instances in section IV. Essentially, scheduling of PCB production is a resource-constrained multi-objective flexible job shop, the goal of PCB is to determine the distribution of machine loads and to minimize the maximum completion time of all operations (i.e., makespan), with resource constraints.

D. AN ILLUSTRATED EXAMPLE

Here, this section gives an example to further illustrate PCB. In this example, this study gives a 5∗5 case with 5 jobs and 5 processes. Each process has at least 2 optional machines for processing. The Table 1 below gives the relevant data of the case. The table gives the jobs, the optional machines corresponding to the jobs in each process and the processing time on the machines, respectively. As per the problem description, Figure 1 gives the scheduling Gantt chart for the case and Figure 2 gives the scheduling Gantt chart for the case based on the bottleneck degree.

TABLE 1. Data on cases.

Job ID	Process	Processing time on the machine			
		M1	M2	M3	M4
	1	8		12	
1	2		15	10	
	3	6			16
	1		10	8	
2	2	8	4		
	3	15	10		
	1		14	10	
3	2	10	8		
	3			10	6
	1			8	10
4	2	15	18		
	3		6	16	
	1	8	20		
5	2		15	12	
	3			8	16

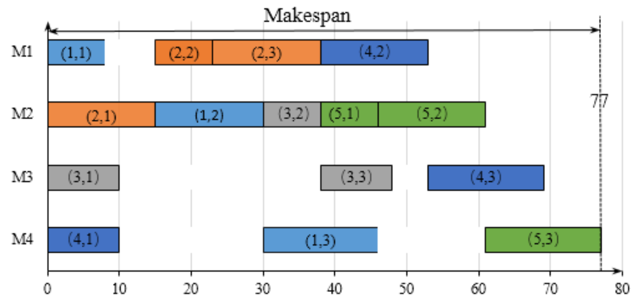

FIGURE 1. Case dispatch Gantt chart.

The rectangular part of the graph shows the machining operation of the job, which can only be sent to the next machining process after each machining is completed. Both use the same case and the first-come, first-served scheduling rule. From Figure 1, it can be seen that the value of makespan is 77 while that of Figure 2 is 70. This shows that under the same scheduling strategy, the shop floor scheduling based on bottleneck degree can effectively shorten the machining time and improve the delivery delay of the order. From Table 1, the bottleneck degree difference of the machine in Figure 2 is 25 and the bottleneck degree difference of the machine in Figure 1 is 29, indicating that the load of the machine in Figure 2 is more uniform.

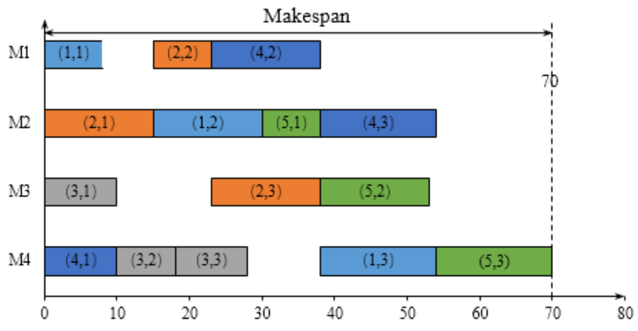

FIGURE 2. Gantt chart of case scheduling based on bottleneck degree.

III. A BOTTLENECK DEGREE-BASED SOLUTION ALGORITHM

This section introduces the basic migrating birds optimization (MBO), and later describes the main components of the bottleneck degree-based MBO algorithm for the PCB problem under consideration.

A. BASIC MIGRATING BIRDS OPTIMIZATION

The migrating birds optimization (MBO) algorithm is a recent high-performing meta-heuristic algorithm inspired by the migrating birds' flight in a V-shaped formation [35]. It shows competing performances in different kinds of combinatorial optimization problems. In MBO, each individual bird in a migrating flock searches for the best solution in its neighborhood and even in the shared neighborhood of its previous individuals. In this way, MBO can ensure that the current solution can be updated by locating a satisfactory solution in the shared neighborhood solution. MBO starts with a flock of birds flying in a hypothetical "V" formation. In this formation, the head bird provides leadership while others follow on the right and left lines. When the leader is tired, it may move to the tail of the following birds and another bird will take its place to lead the population. Generally, this algorithm mainly contains four steps: initialization, improvement on leader bird, improvement on following birds and selection of leader bird. Note that, the MBO sets four parameters: the number of birds in the population α, the number of neighbors to be considered β, the number of neighbors to be shared with next bird χ and the number of tours γ (the number of iterations spent by bird as a leader). And then, α birds are generated randomly, in which the best bird is selected as the leader and other birds are randomly put on the left and right following lines. Hence the original MBO is depicted in Figure 3 as the following.

B. THE PROPOSED BMBO

In this section, a bottleneck degree-based migrating birds optimization (BMBO) is proposed for solving the considered problem. We first determine the two-level encoding and present a decoding rule based on bottleneck-degrees. Then, we make several modifications to modify the performance of the basic MBO, including the population initialization based

```
1:   Initialize $\alpha$ birds;   (% $\alpha$ is the number of swarm
     size)
2:   Put birds on a hypothetical ‘V’ formation;
3:   **While** Termination criterion is not satisfied **do**
4:     **For** $d$=0 to $\gamma$ **do**   (% $\gamma$ is the parameter)
5:       Improve the leader bird;
6:       Improve the following birds;
7:     **End For**
8:     Update the best bird;
9:     Select leader bird;
10:  **End While**
```

FIGURE 3. **Procedure of the original MBO algorithm.**

on bottleneck-degree, neighborhood search strategies, and a competitive mechanism. With these modifications, the proposed algorithm is expected to capture the balance between the exploration and exploitation abilities, and it performs well in solving this problem. The framework of BMBO is depicted in Figure 4, and the details are described as follows.

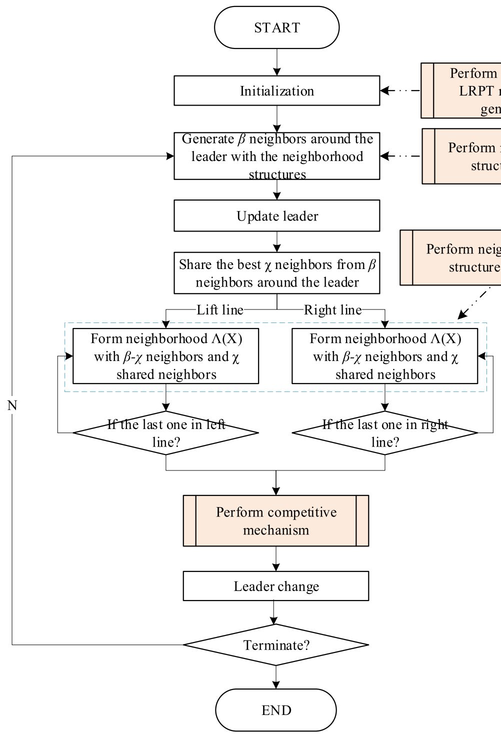
START
Initialization
Perform MOPNR and LRPT rules for the generations
Generate β neighbors around the leader with the neighborhood structures
Perform neighborhood structures (S/I)
Update leader
Share the best χ neighbors from β neighbors around the leader
Perform neighborhood structures (S/I)
Lift line
Right line
Form neighborhood Λ(X) with β-χ neighbors and χ shared neighbors
Form neighborhood Λ(X) with β-χ neighbors and χ shared neighbors
If the last one in left line?
If the last one in right line?
N
Perform competitive mechanism
Leader change
Terminate?
END

FIGURE 4. **The main procedure of the proposed BMBO.**

1) ENCODING AND DECODING

Since the scheduling process for a flexible job shop with recognized bottlenecks involves operation sequence and machine allocation, the bird representation includes two vectors: operation vector and machine vector. Accordingly, a two-layer encoding approach is adopted to describe the above two types of information respectively as depicted in Figure 5. Specifically, in the operation sequence code, each number represents the index of the processed job and the

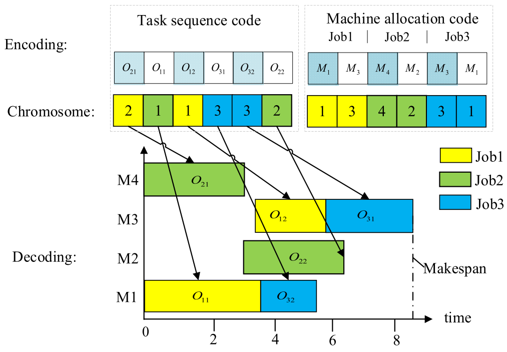

FIGURE 5. **Decoding process of double-layer encoding.**

cumulative occurrence time of the job index indicates the index of the processed operation of the job. The machine allocation code contains a queue in the increasing order of jobs and associated operations. And the allocated machine for each operation must be from the optional machine set. Take Figure 5 as an example. The task sequence code (2, 1, 1, 3, 3, 2) indicates that all operations must be started in the order of ($O_{21}, O_{11}, O_{12}, O_{31}, O_{32}, O_{22}$), and the machine allocation code (1,3,4,2,3,1) shows two operations of jobs (1–3) are assigned to machines (M_1, M_3), (M_4, M_2), and (M_3, M_1) respectively.

Decoding procedure

Input: A chromosome presenting the task sequence and machine allocation, initial set $\boldsymbol{\pi}$ for the bottleneck-degree for machines $\{W_k\}$, set the available time of machines CT_k =0.

Step 1: Go through each operation in the task sequence and confirm the task o_{ij} at the x position in the task sequence;

Step 2: Determine the corresponding machine $k*$ assigned for this task according to the ascending sequence of the values in set $\boldsymbol{\pi}$.

Step 3: Calculate the earliest start time of this task with $S_{ijk^*} = \max(E_{i(j-1)k}, CT_{k^*})$.

Step 4: Update the available time of machine, $CT_{k^*} = E_{ijk^*} = S_{ijk^*} + P_{ijk^*}$

Step 5: Update the bottleneck-degree w_{k^*} of machine $k*$ according to the Equation (4-6). And then, update W_k and set $\boldsymbol{\pi}$.

Step 6: Check if all the task have been allocated on a machine, then go to step 7, otherwise, turn back to step 1.

Step 7: Return $C\max = \max_j(\mathrm{E}_{ijk^*})$

Output: A feasible solution.

FIGURE 6. **The decoding procedure.**

The decoding process transforms the chromosome with the above representation into a feasible solution. The procedure of the proposed decoding mechanism is shown in Figure 6.

It is worth noting that an operation can start only after that its previous one has completed and the machine to perform this operation is available.

2) POPULATION INITIALIZATION

Initial solutions show significant effect on the performance of the population and a good initial population possesses the traits of better quality of best solutions and higher diversity among solutions [44]. Moreover, as described above, the flexible job shop with recognized bottlenecks involves task allocation and machine assignment. Hence, two effective heuristic rules and a bottleneck degree-based initialization mechanism are designed for the initialization of task assignment and machine assignment respectively to ensure the population diversity.

For task allocation, a rule-based initialization approach is employed to allocate tasks to promote the performance of initial birds. In this approach, two existing effective rules and their combinative operators explained as follows are applied randomly [45], [46].

MOPNR: give higher priority to the job that has a bigger number of remaining operations;

LRPT: endow higher priority to the job with the larger remaining processing time.

MOPNR + LRPT: sequence the job with the MOPNR rule at first; if two tasks have the same priority, then employ the LRPT rule;

LRPT + MOPNR: sequence the job with the LRPT rule for initialization; if two tasks have the same priority, then employ the MOPNR rule.

For machine assignment, since the bottleneck degree cannot be provided before allocating these jobs to the machines, the historical bottleneck degree of these machines is adopted. When allocating a machine to an operation, the following principle should be taken into account.

(1) All optional machines are sorted in the ascending order of historical bottleneck degrees.

(2) The machine with the minimum historical bottleneck degree is allocated to perform the operation.

(3) Update the historical bottleneck degree.

3) NEIGHBORHOOD STRUCTURES

As described in part A of Section III, β neighbors around the leader bird and β-χ neighbors around each following bird are generated to improve the corresponding bird's performance. In this paper, two operations including swap and insert, are employed to generate neighborhood solutions.

Swap operation: two different elements are randomly chosen and the positions are swapped.

Insert operation: an element is chosen arbitrarily and then inserted into another position.

Since flexible job shop problem with recognized bottlenecks involves operation allocation and worker assignment, the neighborhood structures are applied for the task sequence code and the machine allocation code, randomly and respectively.

It should be noted that generated descendant chromosomes must be feasible because they are exchanged within the set of optional machines during the same process. If an infeasible solution is generated, a conflict resolution mechanism should be adopted.

Regarding individuals with conflicting chromosomes, it should be determined whether the number of jobs is less than the number of processes, in accordance with the front-to-back sequence of the chromosome gene position. If the number of occurrences is greater than the number of processes, then the gene position will be randomly replaced with an unassigned job.

4) COMPETITION MECHANISM

The migrating birds flight in a V-shaped formation is not only effective on saving energy, but also can share their neighbors with others. However, since the basic MBO arbitrarily puts birds on hypothetical V formation, some promising birds may emerge in the tail and have few opportunities to share their neighbors. To change this situation, a competition mechanism has been developed in the V formation, which is executed after the selection of a leader bird and aims to adjust the position of each following bird in the flock lines. For each line, the bird with the best fitness locates in the first position of the line, with the 2nd best fitness in the 2nd position while the worst bird flies at the end of the line. Consequently, this mechanism guarantees the promising birds are located in the front of the line and have more opportunities to share their neighbors.

IV. EXPERIMENTAL STUDY

The mathematical model is coded by GAMS/Cplex. The solution method based on MBO algorithm is programmed by Matlab. The above programs all run on Intel^{R} Core^{TM} i7 (4790) processor running at 3.20 GHz with 16 GBytes of RAM.

At present, three kinds of experiments are carried out: (1) small-scaled benchmark instances for a comparison study on the performance of mathematical model calculated by GAMS/Cplex and the proposed BMBO; (2) middle-scaled and large-scale benchmark instances to illustrate the performance of the proposed MBO and (3) actual data and rules used in practice for comparison. Among them, the basic configuration of the job and the number of machines in the typical flexible workshop is included in the simulation experiment, conditions for which are designed in this chapter, including the number of new jobs, machines, performance indicators, etc. The test data used in this article is from benchmark data source, download from http://www.idsia.ch/~monaldo/fjsp.html.

A. PARAMETER CALIBRATION

This section calibrates the parameters of the proposed and comparison algorithms. Since parameters have significant influence on the performance of algorithms, this study utilizes the Taguchi method to select the proper parameter values.

For BMBO, there are five parameters or controlled factors: the number of initial solutions (*Pop*), the number of neighbor solutions to be considered (*Sol*), the number of neighbor solutions to be shared (*S Sol*), the number of tours (*Loop*) and the maximum iteration number (*liter*). This study utilizes orthogonal array to arrange experiments. Based on the parameter calibration method reported in [35], the full factorial design is proposed and the multifactor analysis of variance (ANOVA) technique is applied to select the parameter values. Specifically, the largest case with 20 jobs and 15 processes is selected and is solved ten times by any combination of the parameter levels.

For all the experiments are conducted, the relative percentage deviation or *RPD* is selected as the response variable using,

$$RPD = (F_{\text{Some}} - \text{F}_{\text{Best}})/\text{F}_{\text{Best}} \times 100 \tag{18}$$

where, F_{Some} is the function value achieved by a given parameter combination and F_{Best} is the best function value obtained by all combinations for the same instance. Clearly, the algorithm with a lower *RPD* has a better performance.

Specifically, we utilize orthogonal array to arrange the experiments which then are run to obtain the corresponding response value. As shown in Table 2, there are $L_{16}(4^5)$ experiment combinations and each experiment is run 10 times. Then, the average *RPD* is regarded as the final response value.

TABLE 2. Orthogonal array of BMBO.

No.	Levels of parameters					Response value
	Pop	*Sol*	*S Sol*	*Loop*	*liter*	
1	9	6	2	5	100	0.23
2	9	8	3	10	200	0.40
3	9	10	4	15	300	0.26
4	9	12	5	20	400	0.28
5	11	6	3	15	400	0.41
6	11	8	2	20	300	0.00
7	11	10	5	5	200	0.16
8	11	12	4	10	100	0.35
9	13	6	4	20	200	0.08
10	13	8	5	15	100	0.30
11	13	10	2	10	400	0.21
12	13	12	3	5	300	0.14
13	15	6	5	10	300	0.24
14	15	8	4	5	400	0.27
15	15	10	3	20	100	0.36
16	15	12	2	15	200	0.34

After obtaining the response values, the multifactor analysis of variance (ANOVA), a powerful parametric statistical inference tool, is carried out to check the normality, homoscedasticity and independence of the residuals. Detailed ANOVA results are illustrated in Table 3. It is observed that the parameter *liter,* the maximize iteration number, has the largest *delta* 0.15, indicating that *liter* has the greatest influence on the proposed BMBO. If ranking the parameters

TABLE 3. The response value from Minitab.

Level	*Pop*	*Sol*	*S Sol*	*Loop*	*liter*
1	0.2925	0.24	0.195	0.2	0.31
2	0.23	0.2425	0.3275	0.3	0.245
3	0.1825	0.2475	0.24	0.3275	0.16
4	0.3025	0.2775	0.245	0.18	0.2925
Delta	0.12	0.0375	0.1325	0.1475	0.15
Row rank	4	5	3	2	1

in the decreasing order of the *F-ratio* values, the sequence is *liter*, *Loop, S Sol, Pop, Sol*, where the former parameter has greater influence.

Subsequently, an analysis was conducted using the data, the results of which are shown in Figure 7.

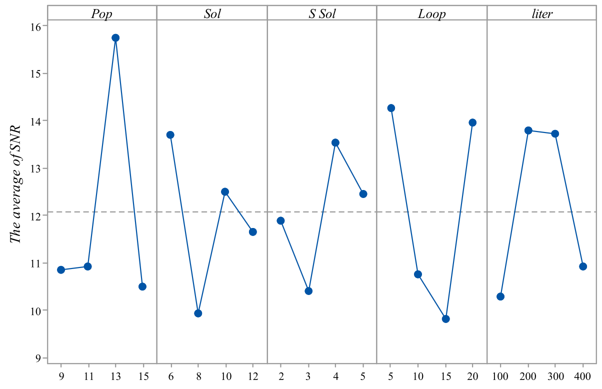

FIGURE 7. SNR main effects plot.

By comparing the slopes of the lines, the relative magnitudes of the effects of each factor can be compared. As shown above, the number of shared solutions exerts the maximum effect on the optimal solution. The maximum BMBO parameter combination at the current experiment level was obtained. According to the signal-noise ratio (SNR) main effects plot of the four parameters is illustrated in Figure 7. SNR is the ratio of the value of objective function to the variance value of objective function. The bigger the SNR, the greater robustness the parameter combination will have. As reported, the best combination of parameters is: *Pop* = 13, *Sol* = 6, *S Sol* = 4, and *Loop* = 5, *liter* = 200.

Similar to the proposed BMBO, the parameters of the six comparison algorithms are also calibrated. The recommended parameter settings of all these algorithms are summarized in Table 4.

B. COMPARISONS WITH Cplex SOLVER

We utilized the GAMS and BMBO algorithms to conduct comparative trials of examples at various scales. As revealed on the table, they share the same object without any bias. The two algorithms turn to generate consistent calculation results of the makespan and load degree. In terms of computing time, BMBO's CPU time was slightly faster than that of GAMS. The results are listed in Table 5.

TABLE 4. Parameter combination of algorithms.

Algorithm and Parameters (Abbreviation)	Range	Value
Bottleneck degree-based migrating birds' Optimization (BMBO)		
The number of initial solutions (*Pop*)	9, 11, 13, 15	13
The number of neighbor solutions to be considered (*Sol*)	6, 8, 10, 12	6
The number of neighbor solutions to be shared (*S Sol*)	2, 3, 4, 5	4
The number of tours (*Loop*)	5, 10, 15, 20	5
The maximum iteration number (*liter*)	100, 200, 300, 400	200
Genetic algorithm (GA)		
Population size (*PS*)	40, 80, 120	120
Crossover rate (*CR*)	0.95, 0.9, 0.8	0.9
Mutation rate (*MR*)	0.5, 0.4, 0.3	0.4
Genetic algorithm (MA)		
Population size (*PS*)	40, 80, 120	80
Crossover rate (*CR*)	0.9, 0.95, 0.8	0.9
Mutation rate (*MR*)	0.5, 0.4, 0.3	0.3
PSO		
Population size (*PS*)	40, 60, 80	60
Ant colony optimization (ACO)		
population size (*PS*)	3, 5, 7	5
the factor of pheromone (α)	0.25, 0.5, 0.75	0.5
the factor of heuristic information (β)	1, 2, 3	1
User-defined parameter (q_0)	0.35, 0.5, 0.85	0.35
Artificial bee colony (ABC)		
The number of scout bees (*NS*)	3, 5, 6	5
The number of following bees (*NF*)	10, 15, 18	15
The life time (*LT*)	30, 40, 60	40
JAYA		
population size (*PS*)	40, 60, 80	60

C. ALGORITHMIC PERFORMANCE

To test the performance of the proposed BMBO, the objective value of makespan and average relative percentage deviation (*RPD*) is utilized to evaluate the obtained objective values with expression (18). Here, $\overline{F}_{some}$ is the average value achieved by a given algorithm among 10 run times, and F_{best} is the best function value obtained by all algorithms for the same instance. Clearly, the algorithm with a lower *RPD* has a better performance.

$$RPD = (\overline{F}_{some} - F_{best})/F_{best} \times 100 \qquad (19)$$

1) EFFECTIVENESS OF ALGORITHM IMPROVEMENTS

In this section, the obtained results by the proposed BMBO are first compared with those solved by three variants of the BMBO with partial improvement strategies: MBO_1 represents original MBO, MBO_2 is augmented with a competitive mechanism, and MBO_3 is augmented with a new domain structure. BMBO refers to an MBO algorithm that contains all updates. The comparison results are provided in Table 6.

In this table, columns 1 presents the instances. Here, Best means the best value of makespan for each instance among all the algorithms, *AVG* means the average value of the makespan values for each instance in ten repetitions by each algorithm. The *RPD* is calculated according to "(19)" and the smaller *RPD* value denotes the better performance.

Table 6 reveals that BMBO generated the lowest makespan among the four algorithms and all the variant versions outperform MBO_1, it can demonstrate the effectiveness of our modifications to the basic MBO. Specifically, the RPD of BMBO was significantly lower than that of the other three algorithms. This demonstrates that BMBO was closest to the optimal value and was a more advantageous algorithm compared with the other three algorithms. In addition, it can also be seen from Table 6 that BMBO has the lowest load degree among the four algorithms. For example, the *RPD* of BMBO was 0.06, while that of the other three algorithms was 0.20, 0.12, and 0.09, respectively. This indicates that BMBO had a more optimized *RPD* value. This finding proves statistically that the competitive mechanism and neighborhood structures can improve the performance of the algorithm much more significantly than the leader change strategy. Specifically, the process of individuals searching for their own neighborhood solutions in accordance with various neighborhood structures and evolution strategies indicates the scatter search performance of BMBO, then the process of sharing neighborhood solutions demonstrates the intensive search performance of BMBO. The combination of both can accelerate the search process of identifying the approximate solution, allowing the algorithm to quickly ascertain an approximate optimal solution in a short time.

2) EFFECTIVENESS OF BMBO

To further investigate the effectiveness of the proposed BMBO, the obtained results are first compared with those solved by the six existing algorithms.

The selection of these compared algorithms is due to them performing well on related flexible job shop problems. All the compared algorithms use our proposed decoding to assign jobs and calculate the objectives. The computational complexities of population algorithms, including GA, ABC, ACO, PSO, MA, JAYA and BMBO are $O(PS \times N)$ where *PS* is the population size. For the proposed BMBO, the average runtime of an iteration in instances MK01, MK02 MKO2, MK03, MK04, MK05, MK06, MK07, MK08, MK09 and MK10 are 1.92, 1.85, 1.88, 1.84, 1.85, 6.20, 6.00, 6.10,

TABLE 5. Comparison with GAMS.

ID	GAMS/Cplex					BMBO			
	Instance	Z_{lp}	Makespan	Bottleneck degree	CPU time (s)	Z_{lp}	Makespan	Bottleneck degree	CPU time (s)
1	2*4	8.0	35.0	5.0	5.3	8.0	35.0	5.0	5.1
2	3*4	7.2	36.0	4.0	50.1	7.2	36.0	4.0	46.2
3	4*4	13.7	56.0	9.0	66.3	13.7	56.0	9.0	58.8
4	2*5	8.0	8.0	8.0	3.9	8.0	8.0	8.0	3.2
5	3*5	14	18.0	14.0	7.0	14.0	18.0	14.0	6.5
6	4*5	16	22.0	16.0	35.5	16.0	22.0	16.0	30.4
7	5*5	24.6	24.0	18.0	727.7	24.6	24.0	18.0	680.3
8	2*6	10.0	10.0	10.0	5.4	10.0	10.0	10.0	4.9
9	3*6	6.6	30.0	4.0	43.1	6.6	30.0	4.0	37.9
10	4*6	9.0	36.0	6.0	167.7	9.0	36.0	6.0	148.4

TABLE 6. Comparison results of MBO_1, MBO_2, MBO_3 and BMBO.

Instance	Best	MBO_1			MBO_2			MBO_3			BMBO		
		C_{max}		w_d	C_{max}		w_d	C_{max}		w_d	C_{max}		w_d
		AVG	*RPD*		*AVG*	*RPD*		*AVG*	*RPD*		*AVG*	*RPD*	
MK01	45.0	56.0	0.20	11.0	49.0	0.08	6.0	48.0	0.06	8.0	48.0	0.06	6.0
MK02	45.0	57.0	0.21	7.0	49.0	0.08	8.0	49.0	0.04	6.0	47.0	0.04	4.0
MK03	47.0	58.0	0.19	12.0	55.0	0.15	10.0	52.0	0.08	9.0	49.0	0.04	9.0
MK04	45.0	56.0	0.20	8.0	51.0	0.11	11.0	50.0	0.10	10.0	48.0	0.06	10.0
MK05	42.0	52.0	0.19	7.0	49.0	0.14	4.0	48.0	0.12	6.0	46.0	0.09	4.0
MK06	45.0	50.0	0.10	8.0	50.0	0.10	6.0	49.0	0.08	9.0	48.0	0.06	8.0
MK07	43.0	54.0	0.20	10.0	42.0	0.02	7.0	45.0	0.04	10.0	46.0	0.06	8.0
MK08	40.0	53.0	0.25	4.0	53.0	0.25	5.0	47.0	0.15	6.0	45.0	0.11	4.0
MK09	44.0	59.0	0.25	10.0	48.0	0.08	8.0	48.0	0.08	7.0	47.0	0.06	7.0
MK10	41.0	55.0	0.25	6.0	50.0	0.18	7.0	49.0	0.16	7.0	45.0	0.09	6.0
Average	43.7	55.0	0.20	8.3	49.6	0.12	7.2	48.5	0.09	7.8	46.9	0.06	6.6

Note: w_d is bottleneck degree difference.

5.96, 6.11. Hence, to ensure the fairness of the comparisons, the run time or the iteration times is set the same. The comparison results are provided in Table 7 and it indicates that the improved MBO is effective. To be specific, the value of makespan obtained by the BMBO algorithm is the lowest among the seven algorithms. The *RPD* value of BMBO is 0.06, while the other six algorithms have *RPD* values of 0.10, 0.10, 0.11, 0.19, 0.10, 0.12. Based on the *RPD* values of all algorithms, BMBO produced a result closest to the optimal solution.

The results about bottleneck degree differences are provided in Table 8 and it indicates that the values of wd obtained by the BMBO algorithm are lower than the other six algorithms for all the instances. Moreover, the average bottleneck degree of BMBO is 6.6, while those of the other six algorithms are 25.9, 8.2, 8.4, 30.2, 24.7 and 9.7 respectively. This signifies that BMBO is significantly better than other algorithms in terms of machine load balance.

In addition, a comparison between the bottleneck degree curves of the compared algorithms was performed with four randomly selected procedures. It should be noted that vertical coordinates represent bottleneck degree values, while horizontal coordinates represent machine numbers. The results are depicted in Figures (8-11) as the following.

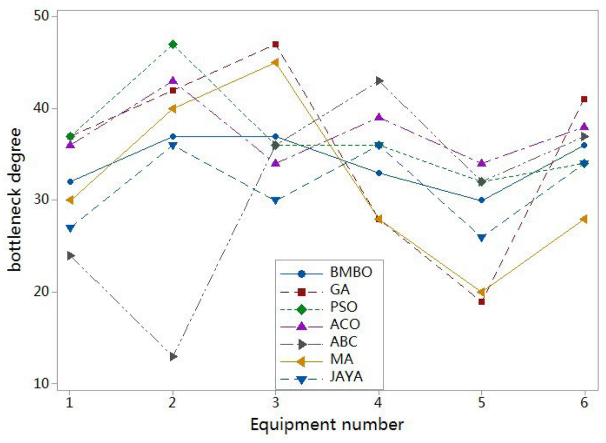

FIGURE 8. Bottleneck degree comparison of Procedure 1.

TABLE 7. Comparison results among BMBO and Other algorithms.

Instance	Best	GA		PSO		ACO		ABC		MA		JAYA		BMBO	
		Cmax		Cmax		Cmax		Cmax		Cmax		Cmax		Cmax	
		AVG	*RPD*	*AVG*	*RPD*	*AVG*	*RPD*	*AVG*	*RPD*	*AVG*	*RPD*	*AVG*	*RPD*	*AVG*	*RPD*
MK01	45.0	49.0	0.08	49.0	0.08	50.0	0.10	49.7	0.10	48.2	0.07	50.6	0.12	48.0	0.06
MK02	45.0	48.4	0.07	48.6	0.07	48.7	0.07	56.0	0.24	48.1	0.07	49.7	0.10	47.0	0.04
MK03	47.0	44.0	0.06	49.7	0.05	46.5	0.01	49.0	0.04	42.0	0.11	47.1	0.00	49.0	0.04
MK04	45.0	48.5	0.07	48.0	0.06	49.0	0.08	51.0	0.03	47.5	0.06	46.0	0.02	48.0	0.06
MK05	42.0	48.9	0.14	49.9	0.16	50.4	0.16	54.9	0.30	47.6	0.13	48.2	0.15	46.0	0.08
MK06	45.0	47.3	0.05	45.0	0.00	51.0	0.12	49.3	0.10	47.1	0.05	50.1	0.11	48.0	0.06
MK07	43.0	48.6	0.05	47.8	0.10	48.9	0.12	52.6	0.22	47.3	0.10	46.2	0.07	46.0	0.06
MK08	40.0	48.7	0.18	49.9	0.20	52.1	0.23	54.7	0.37	47.4	0.18	51.7	0.29	45.0	0.11
MK09	44.0	50.1	0.12	48.8	0.10	47.8	0.08	50.1	0.14	48.5	0.10	48.8	0.11	47.0	0.06
MK10	41.0	51.0	0.20	50.7	0.19	50.7	0.19	51.0	0.24	48.2	0.17	49.4	0.20	45.0	0.08
Average	43.7	48.5	0.10	48.7	0.10	49.5	0.11	52.1	0.19	47.2	0.10	48.8	0.12	46.9	0.06

TABLE 8. Comparison results of w_d among BMBO and Other algorithms.

Instance	GA	PSO	ACO	ABC	MA	JAYA	BMBO
MK01	23.0	7.0	7.0	23.0	23.0	9.0	6.0
MK02	28.0	6.0	7.0	39.0	24.0	7.0	5.0
MK03	22.0	10.0	12.0	28.0	21.0	14.0	6.0
MK04	24.0	11.0	8.0	29.0	24.0	9.0	6.0
MK05	26.0	7.0	6.0	26.0	24.0	7.0	6.0
MK06	28.0	10.0	11.0	31.0	28.0	13.0	7.0
MK07	28.0	11.0	9.0	35.0	24.0	8.0	6.0
MK08	29.0	7.0	7.0	29.0	29.0	10.0	7.0
MK09	23.0	6.0	8.0	27.0	23.0	9.0	8.0
MK10	28.0	8.0	9.0	28.0	27.0	11.0	6.2
Average	25.9	8.2	8.4	30.2	24.7	9.7	6.6

Note: w_d is bottleneck degree difference.

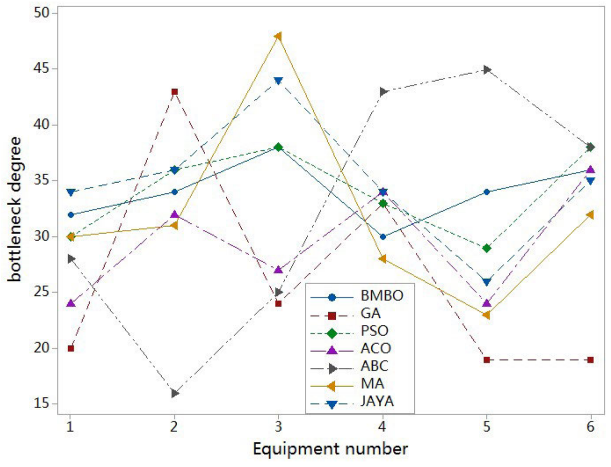

FIGURE 9. Bottleneck degree comparison of Procedure 6.

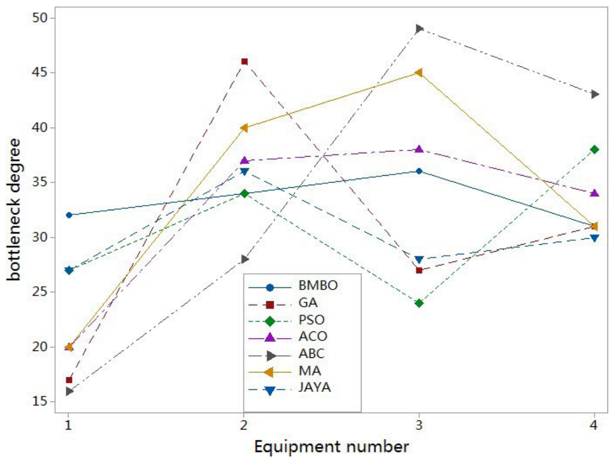

FIGURE 10. Bottleneck degree comparison of Procedure 18.

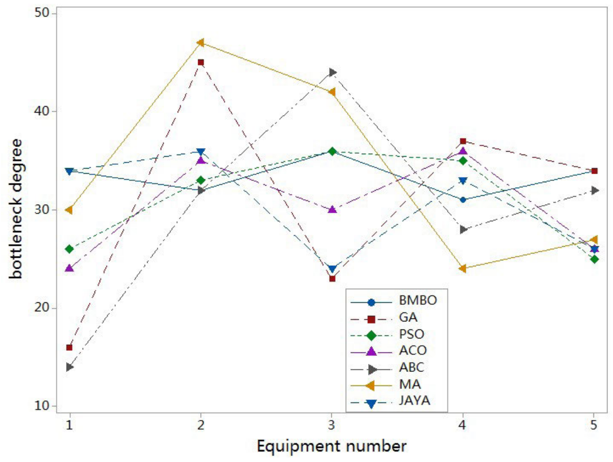

FIGURE 11. Bottleneck degree comparison of Procedure 27.

In accordance with the four load curves in Figures (8-11), the GA, ABC and MA curves have a greater curvature but insufficient smoothness. In other words, under these three algorithms, the load difference among machines remained significant, which can easily lead to a bottleneck problem. The curves of PSO, ACO and JAYA appear smoother, indicating that the machine load distribution was balanced. Upon further examination of these curves, it can be observed that the BMBO curve is smoother than the PSO, ACO and JAYA curves. This signifies that the load was distributed in a more balanced way along the BMBO curve and that the bottleneck effect on the production system was eliminated more

TABLE 9. Historical bottleneck degree.

Machine ID	Daily capacity	Real output	Bottleneck Degree	Machine ID	Daily capacity	Real output	Bottleneck Degree
1	601p	453p	75	5	354p	333p	80
2	1663p	451p	27	6	102p	34p	33
3	501p	493p	88	7	280p	266p	95
4	563p	348p	60	8	852p	359p	43

TABLE 10. Scheduling performance indicator comparison between intelligent algorithms used in P1 Factory.

ID	Example	EDD			SPT			LPT			BMBO		
		C_{max}	*RPD*	w_d	C_{max}	*RPD*	w_d	C_{max}	*RPD*	w_d	C_{max}	*RPD*	w_d
1	D01	82.0	0.20	26.0	76.4	0.12	17.0	79.0	0.15	28.0	68.5	0.00	7.0
2	D02	88.5	0.14	28.0	82.6	0.07	15.0	78.6	0.02	25.0	77.4	0.00	6.0
3	D03	86.4	0.23	21.0	85.2	0.21	18.0	89.7	0.28	24.0	70.2	0.00	4.0
4	D04	90.0	0.11	26.0	80.3	0.00	10.0	90.0	0.03	22.0	87.5	0.09	8.0
5	D05	91.2	0.25	23.0	76.4	0.05	15.0	72.9	0.00	24.0	78.4	0.08	6.0
6	D06	96.5	0.21	27.0	84.3	0.06	19.0	86.3	0.09	25.0	79.3	0.00	5.0
7	Avg	89.1	0.16	25.2	80.6	0.08	15.6	82.8	0.10	24.6	76.9	0.03	6.0

Note: w_d is bottleneck degree difference; C_{max} is makespan.

effectively. In addition, BMBO fluctuates less in the resulting values, indicating better stability of the values generated by the method. This is largely attributed to the target-directed sorting and decoding of the initial solution applied by BMBO, which ensures the generation of the optimal solution. In sum, these results demonstrate the effectiveness of the BMBO algorithm.

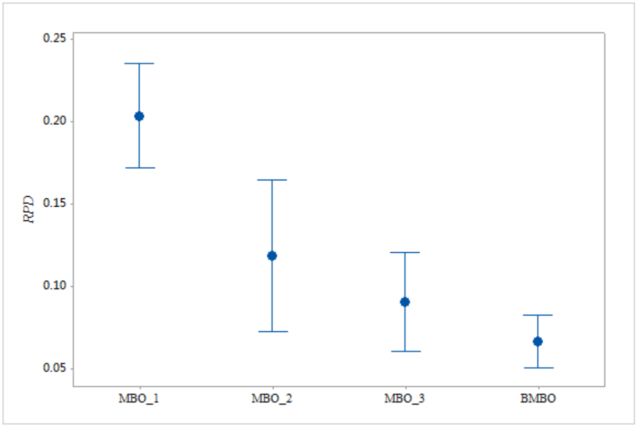

FIGURE 12. Interval plot of RPD of BMBO and variants (CI: 95%).

Based on the results of bottleneck degree differences and *RPD*, the ANOVA test is conducted to check whether BMBO outperforms other algorithms statistically. To be specific, we first compare the proposed BMBO with MBO_1, MBO_2 and MBO_3. From Figure 12, we can see clearly that the BMBO generates the best results for all the texted problems and slightly outperforms all the variant versions of original MBO, it can demonstrate the effectiveness of our modifications to the basic MBO. Moreover, Figure 13 presents the means and intervals of the and *RPD* of BMBO and other tested algorithms at 95% CI. The figure reveals that BMBO has a lower *RPD* than other algorithms.

D. CASE STUDY

When the PCB manufacturing factory scheduling, since the scheduling system is at time 0, parameters are not available at this time to calculate the bottleneck degree. Therefore, when scheduling at time 0, the bottleneck degree is calculated using historical data. Some of the historical bottleneck degrees are shown in Table 9 below.

Trials were conducted with six sets of data (D01-06) from an enterprise. In actual production, the following scheduling rules are used for simplicity: shortest process time (SPT), longest process time (LPT), and earliest due date (EDD). Under the bottleneck degree framework, we compared these three scheduling disciplines with the BMBO algorithm integrated with a bottleneck degree difference (w_d) in terms of scheduling to improve the effectiveness of these methods. The comparison results are presented in Table 10.

As illustrated in Table 10, the BMBO-generated makespan is slightly lower than that of the other three disciplines. From the data in the table, we can see that the makespan by BMBO is improved by 16.5%, 11.5% and 13.3% over EDD, SPT and LPT, respectively. This shows that the maximum completion time has been effectively improved and the on-time delivery of orders has been improved. From the bottleneck degree differences in the table, we can see that the bottleneck difference of BMBO is the smallest with the average value of 6, and the bottleneck degree differences of other rules are 25.2, 15.6 and 24.6 respectively, which shows that the bottleneck difference of each machine under BMBO is the smallest, and the load of the machine is more balanced, which can effectively improve the production efficiency. The results also indicate that when devices are first sequenced with the bottleneck-degree-integrated BMBO approach and then assigned tasks according to the bottleneck degree, their load balance can be greatly improved or ameliorated, which can help minimize the effect of a production bottleneck on production systems.

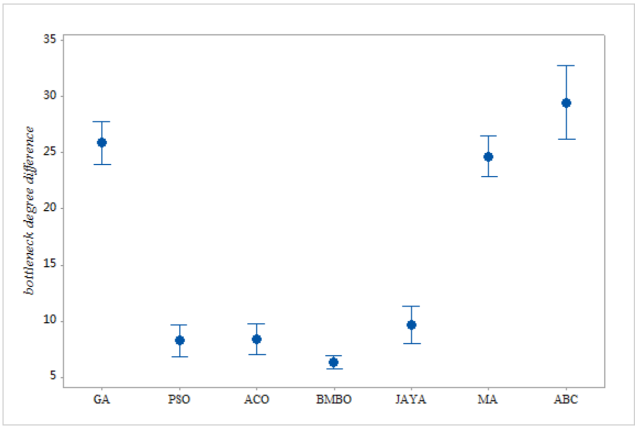

FIGURE 13. Interval plot of bottleneck degree difference of BMBO and other algorithms (CI: 95%).

Table 10 indicates that the BMBO algorithm outperforms the other three disciplines in terms of the resulting indicators, as its average *RPD* was lower. The *RPD* of BMBO is 0.03, while that of the other three algorithms are 0.16, 0.08 and 0.10. This demonstrates that BMBO is closest to the optimal value and was a more advantageous algorithm compared with the other three algorithms. This signifies that the BMBO-generated solution is closer to the optimal solution.

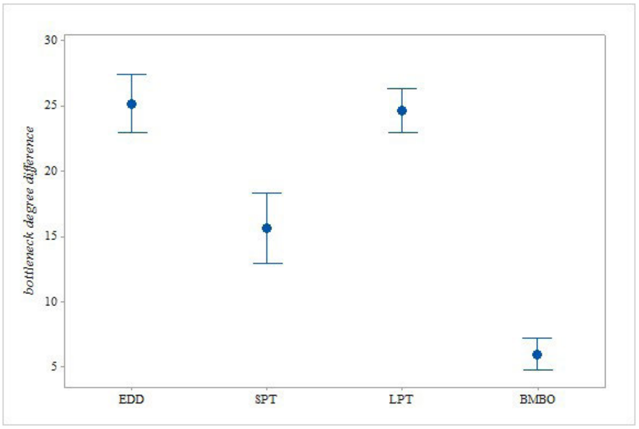

FIGURE 14. Interval plot of bottleneck degree difference of BMBO, EDD, SPT, and LPT (CI: 95%).

Figures (14-15) present the means and intervals of the bottleneck degree and *RPD* of EDD, SPT, and LPT at 95% CI. Among them, Figure 14 reveals that with respect to the bottleneck degree and *RPD*, BMBO has a more balanced load than EDD, SPT, and LPT (95% CI). Similarly, from Figure 15, we can see clearly that the BMBO generates the best results for all the texted instances. Moreover, BMBO fluctuates less in the resulting values, indicating that values generated with this approach have better stability. This can be largely attributed to the goal-directed sequencing and decoding of the initial solution applied by BMBO, which guarantees the generation of the optimal solution.

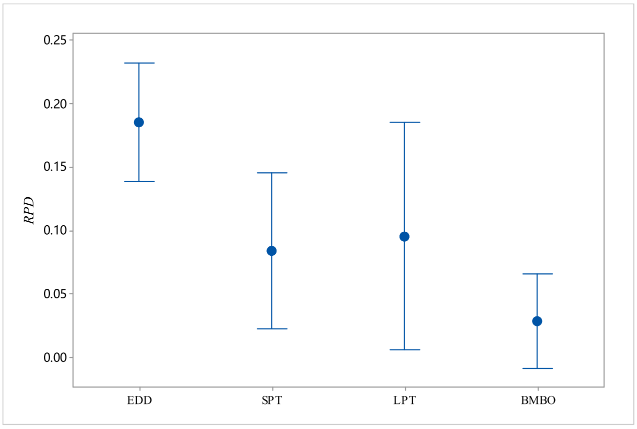

FIGURE 15. Interval plot of RPD of BMBO, EDD, SPT, and LPT (CI: 95%).

V. CONCLUSION AND FUTURE STUDIES

In this paper, an MBO algorithm is proposed to solve printed circuit board shop scheduling problem. The MBO design complexity is reduced using a double-layer chromosome coding/decoding approach. In addition, the combination of double-layer coding/decoding-based crossover and variation approaches and the variable neighborhood approach achieves a significantly higher search efficiency. Trials with standard data as well as actual data of an enterprise demonstrate the satisfactory operating efficiency of the improved MBO. These trials also demonstrate that the improved algorithm can lower the probability of bottleneck occurrence, can balance loads on different devices in the production line, and can raise the production efficiency of an enterprise.

In future work, studies will be conducted that take into account product differentiation in the flexible production workshop and additional differentiation in terms of machine distribution in the production process and each procedure. The proposed BMBO algorithm lacks memory and there is no guarantee that a new solution will be generated for each iteration. Therefore, it is suggested that some heuristic rules can be used to generate the initial solution. and another interesting direction is to incorporate exact algorithms or contraption lists into BMBO to further improve its performance.

A. IMPROVED PRODUCTIVITY

the current management model is underutilized and uneven with respect to the machines on the production line. When production managers determine the daily production tasks, they assign tasks to each line based on orders. If bottlenecks are created, they must be scheduled around the bottlenecks to maximize the utilization of the bottlenecks and ignore the utilization of other machines. In the PCB production line, to enhance the utilization of each machine is very necessary. Therefore, each machine's bottleneck degree of real-time monitoring, dynamic task allocation and adjustment, productivity has increased significantly.

B. REQUIREMENTS AND IMPLEMENTATION

Firstly, it is important to ensure that all machines are highly flexible and available and that production tasks can be assigned to them at any time. Secondly, the maintenance level is determined according to the production needs to maximize the productivity of the machines. After the above conditions are met, the manager can use the models and algorithms provided to obtain a model of the task assignment in real production. Based on the model, the manager can obtain a production task assignment plan and then select the appropriate plan for production to achieve a balance between production efficiency and order delivery time.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their helpful comments and constructive suggestions.

REFERENCES

[1] J. Mumtaz, Z. Guan, L. Yue, Z. Wang, S. Ullah, and M. Rauf, "Multi-level planning and scheduling for parallel PCB assembly lines using hybrid spider monkey optimization approach," *IEEE Access*, vol. 7, pp. 18685–18700, 2019, doi: 10.1109/Access.2019.2895954.

[2] A. Noroozi and H. Mokhtari, "Scheduling of printed circuit board (PCB) assembly systems with heterogeneous processors using simulation-based intelligent optimization methods," *Neural Comput. Appl.*, vol. 26, no. 4, pp. 857–873, May 2015, doi: 10.1007/s00521-014-1765-z.

[3] N. Xie and N. Chen, "Flexible job shop scheduling problem with interval grey processing time," *Appl. Soft Comput.*, vol. 70, pp. 513–524, Sep. 2018, doi: 10.1016/j.asoc.2018.06.004.

[4] W. Xiong and D. Fu, "A new immune multi-agent system for the flexible job shop scheduling problem," *J. Intell. Manuf.*, vol. 29, no. 4, pp. 857–873, Apr. 2018.

[5] D. Lei, M. Li, and L. Wang, "A two-phase meta-heuristic for multiobjective flexible job shop scheduling problem with total energy consumption threshold," *IEEE Trans. Cybern.*, vol. 49, no. 3, pp. 1097–1109, Mar. 2019, doi: 10.1109/TCYB.2018.2796119.

[6] K. Gao, F. Yang, M. Zhou, Q. Pan, and P. N. Suganthan, "Flexible job-shop rescheduling for new job insertion by using discrete jaya algorithm," *IEEE Trans. Cybern.*, vol. 49, no. 5, pp. 1944–1955, May 2019, doi: 10.1109/Tcyb.2018.2817240.

[7] D. Gao, G.-G. Wang, and W. Pedrycz, "Solving fuzzy job-shop scheduling problem using DE algorithm improved by a selection mechanism," *IEEE Trans. Fuzzy Syst.*, early access, Jun. 18, 2020, doi: 10.1109/TFUZZ.2020.3003506.

[8] K. Rahmani and I. Mahdavi, "A genetic algorithm for the single machine preemptive scheduling problem with linear earliness and quadratic tardiness penalties," *Int. J. Adv. Manuf. Technol.*, vol. 65, nos. 5–8, pp. 763–770, Mar. 2013, doi: 10.1007/s00170-012-4215-z.

[9] M. T. Yazdani Sabouni and R. Logendran, "A single machine carryover sequence-dependent group scheduling in PCB manufacturing," *Comput. Oper. Res.*, vol. 40, no. 1, pp. 236–247, Jan. 2013, doi: 10.1016/j.cor.2012.06.006.

[10] P. K. Pollett, "Modelling congestion in closed queueing networks," *Int. Trans. Oper. Res.*, vol. 7, nos. 4–5, pp. 319–330, Sep. 2000.

[11] L. Li, "Multi-ant colony-based sequencing method for semiconductor wafer fabrication facilities with multi-bottleneck," *Int. J. Model., Identificat. Control*, vol. 15, no. 4, pp. 259–266, 2012.

[12] Z. Yan, G. Hanyu, and X. Yugeng, "Modified bottleneck-based heuristic for large-scale job-shop scheduling problems with a single bottleneck," *J. Syst. Eng. Electron.*, vol. 18, no. 3, pp. 556–565, Sep. 2007.

[13] Y. Zhai, C. Liu, W. Chu, R. Guo, and C. Liu, "A decomposition heuristics based on multi-bottleneck machines for large-scale job shop scheduling problems," *J. Ind. Eng. Manage.*, vol. 7, no. 5, pp. 1397–1414, Dec. 2014.

[14] W.-T. Huang, P.-S. Chen, J. J. Liu, Y.-R. Chen, and Y.-H. Chen, "Dynamic configuration scheduling problem for stochastic medical resources," *J. Biomed. Informat.*, vol. 80, pp. 96–105, Apr. 2018, doi: 10.1016/j.jbi.2018.03.005.

[15] H. M. Zhou, Y. R. Chen, Z. L. Guan, L. I. Pei, Z. M. Zheng, and W. University, "A integrated workload control method for the general flow shop with bottleneck," Tech. Rep., 2015.

[16] R. Braune, G. Zäpfel, and M. Affenzeller, "An exact approach for single machine subproblems in shifting bottleneck procedures for job shops with total weighted tardiness objective," *Eur. J. Oper. Res.*, vol. 218, no. 1, pp. 76–85, Apr. 2012.

[17] A. Ozolins, "A new exact algorithm for no-wait job shop problem to minimize makespan," *Oper. Res.*, vol. 10, pp. 1–31, Jul. 2018.

[18] I. Muter, "Exact algorithms to minimize makespan on single and parallel batch processing machines," *Eur. J. Oper. Res.*, vol. 285, no. 2, pp. 470–483, Sep. 2020.

[19] O. Sobeyko and L. Mönch, "Heuristic approaches for scheduling jobs in large-scale flexible job shops," *Comput. Oper. Res.*, vol. 68, pp. 97–109, Apr. 2016.

[20] G. Ozturk, O. Bahadir, and A. Teymourifar, "Extracting priority rules for dynamic multi-objective flexible job shop scheduling problems using gene expression programming," *Int. J. Prod. Res.*, vol. 57, no. 10, pp. 3121–3137, May 2019, doi: 10.1080/00207543.2018.1543964.

[21] H. Tang, R. Chen, Y. Li, Z. Peng, S. Guo, and Y. Du, "Flexible job-shop scheduling with tolerated time interval and limited starting time interval based on hybrid discrete PSO-SA: An application from a casting workshop," *Appl. Soft Comput.*, vol. 78, pp. 176–194, May 2019, doi: 10.1016/j.asoc.2019.02.011.

[22] F. M. Defersha and S. Bayat Movahed, "Linear programming assisted (not embedded) genetic algorithm for flexible jobshop scheduling with lot streaming," *Comput. Ind. Eng.*, vol. 117, pp. 319–335, Mar. 2018.

[23] X. Shi, W. Long, Y. Li, and D. Deng, "Multi-population genetic algorithm with ER network for solving flexible job shop scheduling problems," *PLoS ONE*, vol. 15, no. 5, May 2020, Art. no. e0233759.

[24] T. Meng, Q.-K. Pan, J.-Q. Li, and H.-Y. Sang, "An improved migrating birds optimization for an integrated lot-streaming flow shop scheduling problem," *Swarm Evol. Comput.*, vol. 38, pp. 64–78, Feb. 2018, doi: 10.1016/j.swevo.2017.06.003.

[25] Y. Li, W. Huang, R. Wu, and K. Guo, "An improved artificial bee colony algorithm for solving multi-objective low-carbon flexible job shop scheduling problem," *Appl. Soft Comput.*, vol. 95, Oct. 2020, Art. no. 106544.

[26] K. Hasani, S. A. Kravchenko, and F. Werner, "Simulated annealing and genetic algorithms for the two-machine scheduling problem with a single server," *Int. J. Prod. Res.*, vol. 52, no. 13, pp. 3778–3792, Jul. 2014, doi: 10.1080/00207543.2013.874607.

[27] J. Gao, M. Gen, L. Sun, and X. Zhao, "A hybrid of genetic algorithm and bottleneck shifting for multiobjective flexible job shop scheduling problems," *Comput. Ind. Eng.*, vol. 53, no. 1, pp. 149–162, Aug. 2007, doi: 10.1016/j.cie.2007.04.010.

[28] L. Wang, J. C. Cai, M. Li, and Z. H. Liu, "Flexible job shop scheduling problem using an improved ant colony optimization," *Sci. Program.*, vol. 2017, no. PT.1, pp. 9016303.1–9016303.11, 2017, doi: Artn.901630310.1155/2017/9016303.

[29] H. Ding and X. Gu, "Improved particle swarm optimization algorithm based novel encoding and decoding schemes for flexible job shop scheduling problem," *Comput. Oper. Res.*, vol. 121, Sep. 2020, Art. no. 104951.

[30] X. Li, Z. Peng, B. Du, J. Guo, W. Xu, and K. Zhuang, "Hybrid artificial bee colony algorithm with a rescheduling strategy for solving flexible job shop scheduling problems," *Comput. Ind. Eng.*, vol. 113, pp. 10–26, Nov. 2017.

[31] H. Zhu, Q. Deng, L. Zhang, X. Hu, and W. Lin, "Low carbon flexible job shop scheduling problem considering worker learning using a memetic algorithm," *Optim. Eng.*, vol. 21, no. 4, pp. 1691–1716, Dec. 2020.

[32] J.-Q. Li, J.-W. Deng, C.-Y. Li, Y.-Y. Han, J. Tian, B. Zhang, and C.-G. Wang, "An improved jaya algorithm for solving the flexible job shop scheduling problem with transportation and setup times," *Knowl.-Based Syst.*, vol. 200, Jul. 2020, Art. no. 106032.

[33] Q.-K. Pan and Y. Dong, "An improved migrating birds optimisation for a hybrid flowshop scheduling with total flowtime minimisation," *Inf. Sci.*, vol. 277, pp. 643–655, Sep. 2014, doi: 10.1016/j.ins.2014.02.152.

[34] B. Zhang, Q.-K. Pan, L. Gao, X.-L. Zhang, H.-Y. Sang, and J.-Q. Li, "An effective modified migrating birds optimization for hybrid flowshop scheduling problem with lot streaming," *Appl. Soft Comput.*, vol. 52, pp. 14–27, Mar. 2017, doi: 10.1016/j.asoc.2016.12.021.

[35] Z. Zhang, Q. Tang, D. Han, and Z. Li, "Enhanced migrating birds optimization algorithm for U-shaped assembly line balancing problems with workers assignment," *Neural Comput. Appl.*, vol. 31, no. 11, pp. 7501–7515, Nov. 2019, doi: 10.1007/s00521-018-3596-9.

[36] Z. Li, M. N. Janardhanan, A. S. Ashour, and N. Dey, "Mathematical models and migrating birds optimization for robotic U-shaped assembly line balancing problem," *Neural Comput. Appl.*, vol. 31, no. 12, pp. 9095–9111, Dec. 2019, doi: 10.1007/s00521-018-3957-4.

[37] M. N. Janardhanan, Z. Li, and P. Nielsen, "Model and migrating birds optimization algorithm for two-sided assembly line worker assignment and balancing problem," *Soft Comput.*, vol. 23, no. 21, pp. 11263–11276, Nov. 2019.
[38] C. Roser, M. Nakano, and M. Tanaka, "Comparison of bottleneck detection methods for AGV systems," in *Proc. Simulation Conf.*, 2003, pp. 1192–1198.
[39] C. C. Pegels and C. Watrous, "Application of the theory of constraints to a bottleneck operation in a manufacturing plant," *J. Manuf. Technol. Manage.*, vol. 16, no. 3, pp. 302–311, Apr. 2005.
[40] K. M. N. Muthiah and S. H. Huang, "Overall throughput effectiveness (OTE) metric for factory-level performance monitoring and bottleneck detection," *Int. J. Prod. Res.*, vol. 45, no. 20, pp. 4753–4769, Oct. 2007, doi: 10.1080/00207540600786731.
[41] J. Sarshar, S. S. Moosapour, and M. Joorabian, "Multi-objective energy management of a micro-grid considering uncertainty in wind power forecasting," *Energy*, vol. 139, pp. 680–693, Nov. 2017, doi: 10.1016/j.energy.2017.07.138.
[42] H. Maghsoudlou, B. Afshar-Nadjafi, and S. T. A. Niaki, "A multi-objective invasive weeds optimization algorithm for solving multi-skill multi-mode resource constrained project scheduling problem," *Comput. Chem. Eng.*, vol. 88, pp. 157–169, May 2016, doi: 10.1016/j.compchemeng.2016.02.018.
[43] J. Q. Wang, J. Chen, S. Wang, Y. Z. Guo, and S. D. Sun, "Interval multi-attribute bottleneck identification in job shop," *Comput. Integr. Manuf. Syst.*, vol. 19, no. 2, pp. 429–437, 2013.
[44] H. Ahn and D. Del Vecchio, "Safety verification and control for collision avoidance at road intersections," *IEEE Trans. Autom. Control*, vol. 63, no. 3, pp. 630–642, Mar. 2018.
[45] H. Zhang and U. Roy, "A semantics-based dispatching rule selection approach for job shop scheduling," *J. Intell. Manuf.*, vol. 30, no. 7, pp. 2759–2779, Oct. 2019.
[46] P. Sharma and A. Jain, "Performance analysis of dispatching rules in a stochastic dynamic job shop manufacturing system with sequence-dependent setup times: Simulation approach," *CIRP J. Manuf. Sci. Technol.*, vol. 10, pp. 110–119, Aug. 2015.

JUN CAO was born in Chongqing, China, in 1980. He received the B.S. and master's degrees in mechanical design and theory from the Wuhan University of Science and Technology. He is currently pursuing the Ph.D. degree in mechanical engineering with the Huazhong University of Technology. His research interests include production process control, flexible job shop scheduling, and intelligent algorithms.

ZAILIN GUAN graduated from the Huazhong University of Science and Technology, Wuhan, China, in 1997.

He held a postdoctoral researcher position with The Hong Kong University of Science and Technology, in 1999. He is currently a Professor with the Department of Industrial Engineering, School of Mechanical Science and Engineering, Huazhong University of Science and Technology, Wuhan. In 2001, he performed a modern industrial production management training at Rheinisch-Westfälische Technische Hochschule Aachen. He has long been engaged in the research of new model and mechanism of multi-species small batch production operation control as well as the research and application of Advanced Planning and Scheduling. He has presided over two projects of National Natural Science Foundation and one project of 863 Program. He has participated in a program jointly funded by National Natural Science Foundation and Hong Kong Research Grants Council and a program of EU FP6 cooperation. He is a member of the State Key Laboratory of Digital Manufacturing Equipment and Technology, HUST–SANY Joint Laboratory of Advanced Manufacturing, Huazhong University of Science and Technology (HUST). His research interests include advance planning and scheduling systems, constraints management, supply chain management, logistics, and line balancing.

LEI YUE received the Ph.D. degree in industrial engineering from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2017. He is currently a Postdoctoral Researcher with the State Key Laboratory of Digital Manufacturing Equipment and Technology, HUST–SANY Joint Laboratory of Advanced Manufacturing, Huazhong University of Science and Technology. His research interests include intelligent manufacturing, optimization, planning and scheduling, supply chain management, reverse logistics, and line balancing.

SAIF ULLAH received the B.Sc. degree in mechanical engineering from the University of Engineering and Technology, Taxila, Pakistan, in 2008, and the M.S. and Ph.D. degrees in industrial engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2011 and 2015, respectively.

Since 2012, he has been a Lecturer with the Department of Industrial Engineering, University of Engineering and Technology, Taxila, where he was promoted to an Assistant Professor, in 2015. He was also a Postdoctoral Researcher with the School of Management and the School of Mechanical Science and Engineering, Huazhong University of Science and Technology, from April 2016 to April 2018. He has over 30 research publications, including 13 publications in well reputed international impact factor journals. He has published one book in 2017, and holds one Chinese patent. His research interests include production planning and control, intelligent algorithms, combinatorial optimization, multi-objective optimization, robust optimization, supply chain and remanufacturing, ergonomics, neuro sensors, and headsets.

REHAN AHMAD KHAN SHERWANI was born in Lahore, Pakistan, in 1981. He received the master's degree in statistics and the Ph.D. degree from the University of the Punjab, Pakistan.

He is currently working as an Assistant Professor with the College of Statistics, University of the Punjab, Lahore. He has numerous publications in peer-reviewed national and international research journals. His areas of specialization include regression analysis, multilevel models, structural equation models, mixed models, and their applications. He is a member of Boards of Studies of University of the Punjab and GC University Faisalabad, Pakistan. He has also worked as a member of Punjab Technical Committee for Census 2017; a member of Dean of the Faculty of Science, Purchase Committee; a Focal Person of QEC Faculty of Science, University of the Punjab; a Coordinator of M.Sc. Biostatistics Programme, and a Coordinator of M.Sc. Business Statistics and Management Programme. He is also a member of National Curriculum Review Committee by HEC for the subject of Statistics.