IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received August 25, 2021, accepted September 19, 2021, date of publication September 22, 2021, date of current version October 4, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3114712

A New Job Shop Scheduling Method for Remanufacturing Systems Using Extended Artificial Bee Colony Algorithm

XIANGQI LIU[1,2], JIE CHEN[3], XIAOLING HUANG[4], SHANSHAN GUO[4], SHUAI ZHANG[3], AND MENGJIAO CHEN[3]

[1]Hangzhou Dianzi University Information Engineering School, Hangzhou 311305, China
[2]School of Mechanical Engineering, Hangzhou Dianzi University, Hangzhou 310018, China
[3]School of Information Management and Artificial Intelligence, Zhejiang University of Finance and Economics, Hangzhou 310018, China
[4]Library, Zhejiang University of Finance and Economics, Hangzhou 310018, China

Corresponding author: Jie Chen (chenjzufe103@zufe.edu.cn)

This work was supported in part by the Humanities and Social Sciences Research Project of Ministry of Education under Grant 20YJC870003, in part by the National Natural Science Foundation of China under Grant 51975512 and Grant 51875503, in part by Zhejiang Key Research and Development Project of China under Grant 2021C03153, and in part by Zhejiang Natural Science Foundation of China under Grant LZ20E050001.

ABSTRACT With the emergence of the remanufacturing industry, scheduling problems related to remanufacturing systems have drawn considerable interest from scholars. There are usually two execution modes of remanufacturing for reprocessing components, i.e., replacement mode and repair mode. However, few studies have focused on the job shop scheduling problem that considers the trade-off between the two execution modes. Thus, a new job shop scheduling method with job families (JSS-JF) is proposed, which can handle the selection of appropriate execution modes for reprocessing components with different damaged conditions. It decomposes the scheduling problem into three sub-problems: (1) execution mode selection; (2) replacement job assignment and sequencing; and (3) repair machine assignment and operation sequencing. The scheduling objective is to minimize the total completion time and total cost of job families. To solve the proposed JSS-JF model, an extended artificial bee colony algorithm with a new three-dimensional encoding scheme is presented to find a near-optimal solution of the proposed JSS-JF model. The crossover and mutation operators, local search, and elite replacement strategy are also integrated in the proposed algorithm. The experiments are conducted for verifying the practicality and effectiveness of the presented algorithm by comparing with six baseline algorithms.

INDEX TERMS Job shop scheduling, job families, remanufacturing, execution mode, extended artificial bee colony algorithm.

I. INTRODUCTION

Remanufacturing is a process that end-of-life (EOL) products are disassembled into components, reprocessed through advanced technology, and reassembled to fabricate new products [4]. By means of innovative surface engineering technology and dedicated devices, the remanufactured products will have the same quality as the new ones, consequently realizing the objectives of green, circular, and low-carbon consumptions. Compared with new products, remanufactured products can be obtained for only 50% cost, 40% energy, 30% raw materials, and pollution discharge not exceeding 20% [23]. Thus, the remanufacturing industry is highly appreciated worldwide because of its considerable contribution to minimizing resource consumption and carbon emissions [23].

Remanufacturing can be considered as an industrial process covering a wide range of industries, such as those in the production of ballpoints, personal computers, and automotive clutch segments [30], [17], [3]. The remanufacturing system mainly includes three sub-systems: disassembly, reprocessing, and reassembly shops. Among them, the reprocessing shops serve as a link between disassembly and reassembly shops, through which the defective components of an

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

EOL product are reprocessed into high-quality components. The quality of the new and reprocessed components should essentially be the same in appearance, reliability, and performance. The components of an EOL product must be matched together when they are reprocessed and reassembled into the corresponding remanufactured product, hence the components of an EOL product constitutes a job family [36]. The job shop scheduling with job families (JSS-JF) has been widely recognized as a critical research issue. It aims to minimize the completion time of job families by allocating jobs to machines and sequencing the operations on corresponding machines for components reprocessing.

Although there are available literatures on the JSS-JF, most literatures only consider the time factor, such as total completion time of job families or total tardiness of job families on basis of due-dates [38], [16], [36], [37]. On the other hand, components obtained from disassembly shops in the remanufacturing environment are of different damaged conditions and the appropriate execution mode needs to be selected for each component. Gharbi *et al.* [9] and Berthaut *et al.* [1] considered that remanufacturing operations can be implemented on different components with either replacement or repair mode. Thus, the execution mode selection can be integrated with remanufacturing scheduling to achieve a more optimal solution. However, few studies have focused on JSS-JF that considers different damaged conditions of components and selects the appropriate execution modes for reprocessing components.

In this study, a new JSS-JF model is proposed. In contrast to existing studies on JSS-JF, this study considers not only the total completion time of job families but also the two execution modes (i.e., replacement mode and repair mode) for reprocessing components. Gharbi *et al.* [9] mentioned that the remanufacturing process can be accelerated by replacing parts rather than repairing them. However, repairing a component usually costs less than replacement but takes more time. Thus, the appropriate execution mode should be selected for each component, depending on the comprehensive evaluation of the total completion time and total cost of job families. The scheduling problem in this study is decomposed into three sub-problems: (1) execution mode selection; (2) replacement job assignment and sequencing; and (3) repair machine assignment and operation sequencing. Once the execution modes are determined, the jobs with replacement mode are handled through replacement job assignment and sequencing, and the jobs with repair mode are handled through repair machine assignment and operation sequencing. The objective of the proposed JSS-JF model is to minimize the total completion time and total cost of job families.

The job shop scheduling problem is an NP-hard problem and has been widely solved using population-based algorithms, including genetic algorithm (GA) [27], [41] and particle swarm optimization (PSO) [40], [32]. An artificial bee colony (ABC) algorithm based on the intelligent foraging behavior of a honey bee swarm was first proposed by Karaboga [13] for solving numerical optimization problems and had been used to solve job shop scheduling problems successfully in recent years. Furthermore, Karaboga and Akay [14] compared the ABC algorithm with other population-based optimization algorithms and summarized its advantages, such as fewer control parameters, shorter search time, and better global optimization ability. In view of the successful application of the ABC algorithm in solving job shop scheduling problems, it is extended in the present study to solve the proposed JSS-JF model for remanufacturing systems.

Compared with the classical job shop scheduling problem, the presented JSS-JF problem in this study is a more complex multi-variable and multi-dimensional optimization problem. However, the one-dimensional encoding scheme of the basic ABC algorithm cannot be directly used to represent the complex structure of a JSS-JF solution. Therefore, a new three-dimensional encoding scheme is proposed to describe the JSS-JF solution. In addition, to improve the search ability and accelerate the convergence of the ABC algorithm, it is extended with crossover and mutation operators, local search, and elite replacement strategy. Finally, simulation experiments are performed for comparison with six baseline population-based algorithms. The experimental results confirm the practicability and effectiveness of the proposed extended ABC, i.e., EABC algorithm in solving the proposed JSS-JF model for remanufacturing systems.

The remainder of this paper is organized as follows. Section 2 reviews the related work on the job shop scheduling problem and ABC algorithm. Section 3 presents the proposed JSS-JF model in more detail with mathematical representation. Section 4 introduces the basic concept of the ABC algorithm and proposes the EABC algorithm to solve the proposed JSS-JF model. Section 5 presents the simulation experiments and discusses the experimental results to illustrate the superiority of the proposed EABC algorithm. Section 6 summarizes the main contributions of this study and directions for future work.

II. RELATED WORK

In this section, previous literatures that are closely related to this study including the job shop scheduling problem and ABC algorithm are reviewed.

A. JOB SHOP SCHEDULING PROBLEM

Various classical job shop scheduling problems and their extensions, such as job shop scheduling with alternative machines, flexible job shop scheduling, and stochastic job shop scheduling, have been studied. With the emergence of the remanufacturing industry, scheduling problems in remanufacturing systems have also drawn the interest of scholars. In the past decade, the JSS-JF in the remanufacturing system has been investigated as a variant of job shop scheduling.

In manufacturing systems, classical job shop scheduling problems, including machine assignment and operation sequencing, have been extensively studied [34], [42], [41]. In contrast to manufacturing, remanufacturing is more

complex because of its high uncertainty [44]. For example, the conditions of products obtained in remanufacturing facilities are highly variable [25]. Moreover, component matching requirements has to be considered in the remanufacturing system, i.e., the components for reassembling the product to be remanufactured must be matched [36].

The JSS-JF for remanufacturing systems has generally been solved from a local or systematic perspective. For example, Yu *et al.* [38] elaborated the job shop scheduling problem in which jobs are grouped into job families to minimize the total family flow time. Kim *et al.* [16] explored the job shop scheduling problem with job families to minimize the total family flow time with iterative greedy algorithms. Yu and Lee [26] focused on a scheduling problem with job families for remanufacturing systems with three sub-systems and separately solved the disassembly, reprocessing, and reassembly scheduling problems. Yu and Lee [37] considered a due-date-based objective to minimize the total family tardiness in the job shop scheduling problem with job families. Shi *et al.* [28] proposed an energy-aware remanufacturing scheduling method for the entire remanufacturing system with non-dedicated reprocessing lines. Fu *et al.* [5] focused on a stochastic multi-objective remanufacturing scheduling problem that integrates disassembly, reprocessing, and reassembly sub-systems.

In contrast, to leverage the advantages of different execution modes in various contexts, multiple execution modes have been studied in the domains of project scheduling [11], [24] and remanufacturing [22], [26], [2]. Linton and Jayaraman [22] indicated that the inherent full values of products cannot be fully exploited without recognizing their differences between the products, and proposed different modes of product life extension. Pellerin and Gharbi [26] and Berthaut *et al.* [2] considered two forms of execution modes: component repair and component replacement with new parts depending on the product to be remanufactured. In the domain of remanufacturing scheduling, Lin *et al.* [21] considered four execution modes (i.e., replacement, repair, refurbish, and recondition) and solved the multi-plant remanufacturing scheduling problem by exploring the relationship between execution mode and rework possibility.

Existing studies have explored the advantages of the execution mode in the remanufacturing industry; however, the trade-off among the execution modes and the impact of different execution mode selections on scheduling methods have not been considered. Accordingly, the current study proposes a new job shop scheduling method that integrates the execution mode selection, replacement job assignment and sequencing, and repair machine assignment and operation sequencing.

B. ARTIFICIAL BEE COLONY ALGORITHM

The ABC algorithm was originally proposed by Karaboga [13] for solving continuous optimization problems. It is a population-based algorithm that simulates the intelligent foraging behavior of a honey bee swarm to optimize the objective function. Karaboga and Basturk [15] stated that the ABC algorithm outperformed many other intelligent algorithms and can be employed to solve complex optimization problems efficiently. Consequently, the ABC algorithm has been widely employed for practical applications, and its variations have also been developed. Singh [29] proposed an ABC algorithm for solving the leaf-constrained minimum spanning tree problem; their experiments demonstrated that the algorithm could yield improved solutions over a short period. Li *et al.* [18] presented a discrete ABC algorithm to solve the steelmaking scheduling problems with multiple constrained resources. Zhu *et al.* [45] proposed an ABC algorithm based on multiple improvement strategies to address the cloud manufacturing service composition. Zou *et al.* [46] utilized an effective discrete ABC algorithm to deal with the automatic guided vehicle scheduling problem in a linear manufacturing workshop. Gao *et al.* [6] surveyed the literatures that utilized ABC algorithm to solve the complicated discrete optimization problems in remanufacturing scheduling. In our previous work [43], an extended ABC algorithm was proposed to solve the networked correlation-aware manufacturing service composition. It was also demonstrated that the proposed algorithm outperformed other population-based algorithms.

Existing literature have also shown that ABC algorithms have been widely employed to solve job shop scheduling problems. For example, Yin *et al.* [35] proposed a discrete ABC algorithm to solve the job shop scheduling problem. Zhang and Wu [42] utilized the ABC algorithm to solve the stochastic job shop scheduling problem with the objective of minimizing the maximum lateness. Gao *et al.* [7] proposed a two-stage ABC algorithm for flexible job scheduling and re-scheduling with new job insertion. Gao *et al.* [8] proposed a two-stage ABC algorithm to solve the flexible job shop scheduling problem with fuzzy processing time.

Because of the good performance of ABC algorithm in solving optimization problems, it is extended for solving the proposed JSS-JF model in this study. The proposed JSS-JF model handles a multi-variable and multi-dimensional optimization problem. Accordingly, a new three-dimensional encoding scheme is proposed to describe the JSS-JF solution effectively. Moreover, three improvements, specifically the crossover and mutation operators, local search, and elite replacement strategy, are integrated in the EABC algorithm to solve the proposed model.

III. PROBLEM DEFINITION AND MODELING

In remanufacturing, the job shop scheduling problem is defined through the proposed JSS-JF model. The illustrative example in Figure 1 shows that products 1 and 2 are disassembled on either one of the parallel disassembly workstations (DW_1, DW_2, DW_3, ...) in the disassembly shop. Let C_p be the set of components of product p; for example, $C_1 = \{C_{11}, C_{12}, C_{13}\}$ and $C_2 = \{C_{21}, C_{22}, C_{23}\}$ for products 1 and 2, respectively. The components of C_1 and

C_2 are individually reprocessed by selecting an appropriate execution mode (i.e., replacement mode or repair mode). Finally, the components are reassembled into corresponding remanufactured products 1 and 2 in either one of the parallel reassembly workstations (RW_1, RW_2, RW_3, ...) in the reassembly shop.

As shown in the lower part of Figure 1, each component corresponds to a job in the reprocessing shop. The jobs are grouped into job families but are processed individually. Let JF_m be the mth job family containing a set of jobs, e.g., $JF_1 = \{J_1, J_2, J_3\}$ and $JF_2 = \{J_4, J_5, J_6\}$ for the 1st and 2nd job families, respectively. In the current illustrative example, jobs J_1, J_2, J_3, and J_6 are executed with replacement mode, and human operators replace the defective components, i.e., C_{11}, C_{12}, C_{13}, and C_{23}, with new components, i.e., C'_{11}, C'_{12}, C'_{13}, and C'_{23}, respectively. By contrast, jobs J_4 and J_5, which reprocess components C_{21} and C_{22}, are executed with repair mode and processed by the repair machines for sequential operations (O_{41}, O_{42}, ...) and (O_{51}, O_{52}, ...), respectively.

In the proposed JSS-JF model, the selection of different execution modes for jobs necessitates different scheduling methods, depending on the comprehensive evaluation of the total completion time and total cost of job families. If jobs are executed with replacement mode, the defective components are replaced with new components by human operators. If jobs are executed with repair mode, a set of jobs is processed by repair machines for sequential operations. Each repair job may involve multiple operations, and each of these is processed on one of the candidate repair machines.

For a more focused problem modeling, the following assumptions were made in this study: (1) the job description, such as the processing times of replacement jobs and repair operations, are determined and given in advance; (2) there is no priority relationship among job families or jobs; (3) replacement jobs and repair operations cannot be split and preempted; (4) setup and transportation times are negligible; (5) the human operators for processing replacement jobs are available; and (6) repair machine breakdowns are neglected.

A. NOTATIONS

To describe the proposed model explicitly, the following notations are introduced.

1) INDICES AND SETS

e	execution mode index; $e = 1$ and $e = 2$ indicate replacement and repair modes, respectively.
n	human operator index, $n = 1, 2, \ldots, N$.
J_i	the i th job, $i = 1, 2, \ldots, I$.
O_{il}	the l th operation of job J_i with repair mode, $l = 1, 2, \ldots, L_i$.
JF_m	the m th job family containing a set of jobs, $m = 1, 2, \ldots, M$.
MA_k	the kth repair machine, $k = 1, 2, \ldots, K$.
MS_{il}	set of candidate repair machines for performing O_{il}.

2) PARAMETERS

JCT_i^e	completion time of J_i with the eth execution mode.
HST_i^n	starting time of J_i with replacement mode by the nth human operator.
HPT_i^n	processing time of J_i with replacement mode by the nth human operator.
HCT_i^n	completion time of J_i with replacement mode by the nth human operator.
HRC_i^n	replacement cost of J_i with replacement mode by the nth human operator.
OST_{il}^k	starting time of O_{il} with repair mode on MA_k.
OPT_{il}^k	processing time of O_{il} with repair mode on MA_k.
OCT_{il}^k	completion time of O_{il} with repair mode on MA_k.
ORC_{il}^k	repair cost of O_{il} with repair mode on MA_k.
TT	total completion time of all job families.
TC	total cost of all job families.
M	a large number.

3) DECISION VARIABLES

α_i^e	binary variable; if J_i is processed with the eth execution mode, then, $\alpha_i^e = 1$; otherwise, $\alpha_i^e = 0$.
χ_i^n	binary variable; if J_i with replacement mode is processed by the nth human operator, then, $\chi_i^n = 1$; otherwise, $\chi_i^n = 0$.
β_{il}^k	binary variable; if O_{il} with repair mode is processed on MA_k, then, $\beta_{il}^k = 1$; otherwise, $\beta_{il}^k = 0$.
$x_{ii'}^n$	binary variable; if J_i with replacement mode is immediately followed by $J_{i'}$ with replacement mode by the nth human operator, then, $x_{ii'}^n = 1$; otherwise, $x_{ii'}^n = 0$.
$y_{ili'l'}^k$	binary variable; if O_{il} with repair mode is immediately followed by $O_{i'l'}$ with repair mode on MA_k, then, $y_{ili'l'}^k = 1$; otherwise, $y_{ili'l'}^k = 0$.

B. OBJECTIVES

This study aims to minimize TT and TC to obtain the optimal schedule.

TT is an important indicator of production efficiency. It can be calculated as the maximal completion time using Equation (1) which is obtained by comparing the completion time of each job family.

$$TT = \max_{1 \le m \le M} \left\{ \max_{J_i \in JF_m} \left\{ \sum_{e=1}^{2} \alpha_i^e JCT_i^e \right\} \right\} \tag{1}$$

The total job cost, TC, which includes the replacement and repair costs for replacement jobs and repair operations, can be calculated using Equation (2). If the job is executed with

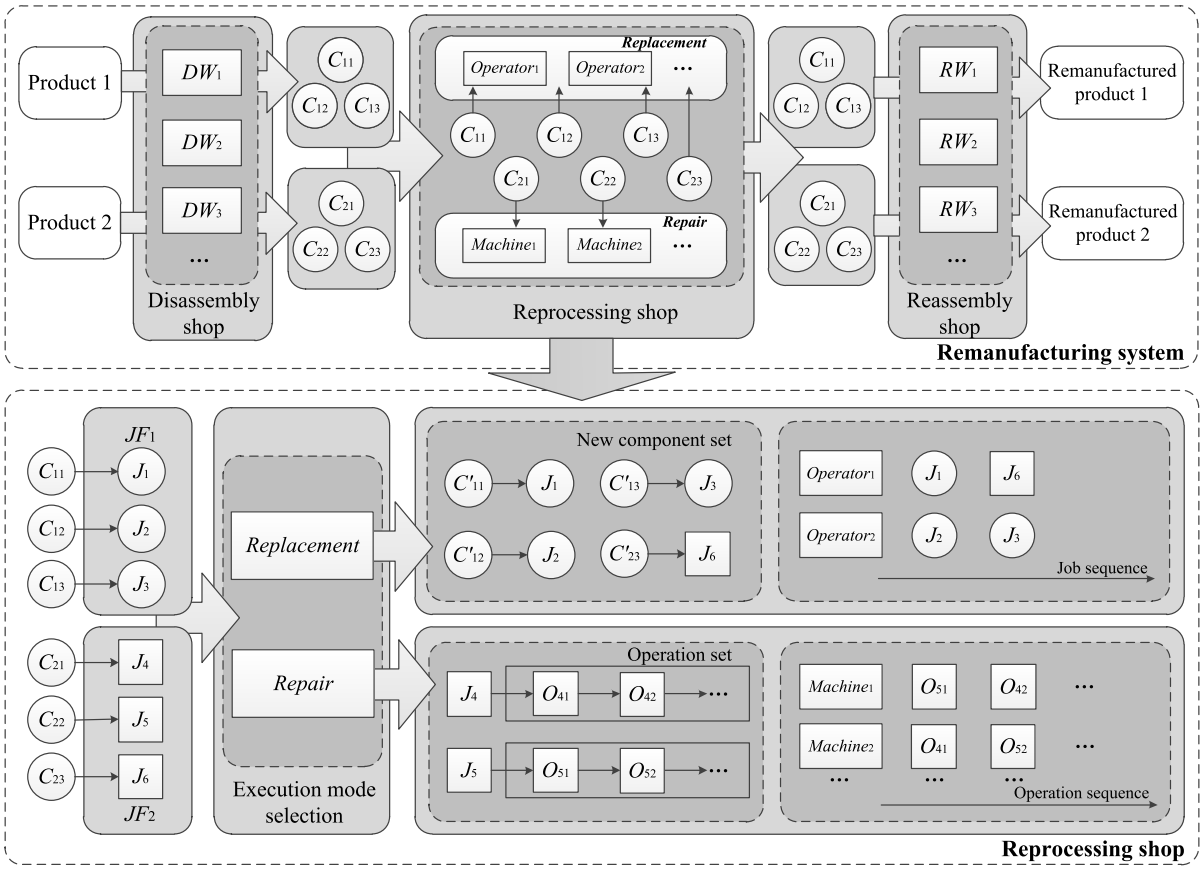

FIGURE 1. An example: system configuration of the proposed JSS-JF model for remanufacturing systems.

replacement mode, the human operator replaces the defective component with a new part, incurring replacement cost. If the job is executed with the repair mode, the operations of the job are processed on repair machines, incurring repair cost.

$$TC = \sum_{i=1}^{I}\sum_{n=1}^{N}\alpha_i^1\chi_i^n HRC_i^n + \sum_{i=1}^{I}\sum_{l=1}^{L_i}\sum_{MA_k\in MS_{il}}\alpha_i^2\beta_{il}^k ORC_{il}^k \quad (2)$$

The comprehensive evaluation measure of schedule f is calculated by setting appropriate weights to TT and TC. Accordingly, the objective of the proposed JSS-JF model is formulated using Equation (3).

$$max\, f = \omega_1\frac{TT_{max} - TT}{TT_{max} - TT_{min}} + \omega_2\frac{TC_{max} - TC}{TC_{max} - TC_{min}} \quad (3)$$

where TT_{max} and TC_{max} denote the maximal TT and TC, respectively; TT_{min} and TC_{min} denote the minimal TT and TC, respectively; and ω_1 and ω_2 that add up to 1 represent the weights of TT and TC, respectively.

C. TOTAL COMPLETION TIME OF JOB FAMILIES

In the proposed JSS-JF model, the total completion time of job families is calculated according to the completion time of the jobs; JCT_i^e is calculated using Equation (4). If the job is executed with replacement mode, then, its completion time is equal to that of the human operator who performed the replacement. If the job is executed with repair mode, then, its completion time is equal to that of the last operation of the job.

$$JCT_i^e = \begin{cases} \sum_{n=1}^{N}\chi_i^n HCT_i^n & e = 1 \\ \sum_{MA_k\in MS_{iL_i}}\beta_{iL_i}^k OCT_{iL_i}^k & e = 2. \end{cases} \quad (4)$$

The starting and completion times of processed jobs with replacement mode (i.e., HST_i^n and HCT_i^n) are calculated using Equations (5) and (6), respectively.

$$HST_i^n = EAH_n \quad (5)$$

$$HCT_i^n = HST_i^n + HPT_i^n \quad (6)$$

where EAH_n represents the earliest available time of the nth human operator.

The starting and completion times of processed operations with repair mode (i.e., OST_{il}^k and OCT_{il}^k) are obtained by Equations (7) and (8), respectively.

$$OST_{il}^k = max\left\{EAM_k, OCT_{i(l-1)}^{k'}\right\} \quad (7)$$

$$OCT_{il}^k = OST_{il}^k + OPT_{il}^k \quad (8)$$

where EAM_k represents the earliest available time of MA_k, and k' is the index of repair machine that processes the previous operation of O_{il} (i.e., $O_{i(l-1)}$).

D. CONSTRAINTS OF THE PROPOSED MODEL

The given objective of the JSS-JF model is subject to the constraints expressed by Equations (9) – (17).

$$\sum_{e=1}^{2} \alpha_i^e = 1, \quad \forall i$$

$$HCT_i^n \le HST_{i'}^n + \left(1 - x_{ii'}^n\right) \cdot M, \tag{9}$$

$$\alpha_i^1 = 1, \alpha_{i'}^1 = 1, i \ne i', \quad \forall n \tag{10}$$

$$\sum_{i'=1, i' \ne i}^{I} \alpha_{i'}^1 x_{ii'}^n \le 1, \alpha_i^1 = 1, \quad \forall n \tag{11}$$

$$\sum_{i=1, i \ne i'}^{I} \alpha_i^1 x_{ii'}^n \le 1, \alpha_{i'}^1 = 1, \quad \forall n \tag{12}$$

$$\sum_{n=1}^{N} \chi_i^n = 1, \alpha_i^1 = 1, \quad \forall i$$

$$OCT_{il}^k \le OST_{i'l'}^k + \left(1 - y_{ili'l'}^k\right) \cdot M, \tag{13}$$

$$\alpha_i^2 = 1, \alpha_{i'}^2 = 1, l \in [1, L_i], l' \in [1, L_{i'}], i \ne i', l \ne l', \ \forall k \tag{14}$$

$$\sum_{\substack{i'=1, i' \ne i \\ 1 \le l' \le L_{i'}, l' \ne l}}^{I} \alpha_{i'}^2 y_{ili'l'}^k \le 1, \alpha_i^2 = 1, l \in [1, L_i], \quad \forall k \tag{15}$$

$$\sum_{\substack{i=1, i \ne i' \\ 1 \le l \le L_i, l \ne l'}}^{I} \alpha_i^2 y_{ili'l'}^k \le 1, \alpha_{i'}^2 = 1, l' \in [1, L_{i'}], \quad \forall k \tag{16}$$

$$\sum_{MA_k \in MS_{il}} \beta_{il}^k = 1, \alpha_i^2 = 1, l \in [1, L_i] \tag{17}$$

Constraint (9) guarantees that each job is executed with one execution mode. Constraints (10) – (13) are related to the replacement jobs. Constraint (10) specifies that no two jobs can be simultaneously processed by one human operator. Constraint (11) ensures that at most one another job is selected to be processed following J_i by each human operator. Constraint (12) ensures that at most one another job is selected to be processed preceding $J_{i'}$ by each human operator. Constraint (13) guarantees that each replacement job is processed by one human operator.

Constraints (14) – (17) are related to the repair operations. Constraint (14) ensures that no two repair operations can be simultaneously processed on one repair machine. Constraint (15) ensures that at most one another repair operation is selected to be processed following O_{il} on each repair machine. Constraint (16) ensures that at most one another repair operation is selected to be processed preceding $O_{i'l'}$ on each repair machine. Constraint (17) guarantees that each repair operation is processed on one of the candidate repair machines.

IV. EXTENDED ARTIFICIAL BEE COLONY ALGORITHM FOR SOLVING PROPOSED MODEL

In this section, the basic ABC algorithm is briefly introduced. Then, the EABC algorithm is presented to solve the proposed JSS-JF model. Two improvements are also introduced: (1) using a new three-dimensional encoding scheme to represent the JSS-JF solution and (2) integrating the crossover and mutation operators, local search, and elite replacement strategy to enhance the exploration and convergence abilities of the ABC algorithm.

A. INTRODUCTION OF THE BASIC ARTIFICIAL BEE COLONY ALGORITHM

The main steps of the basic ABC algorithm [13], which is inspired by the intelligent foraging behavior of a honey bee swarm, are summarized as follows. The numbers of employed bees and onlooker bees are both equal to the number of the solutions.

(1) Initialization phase: A solution with multiple decision variables is analogized as a food source. The initial solutions are generated in this phase.

(2) Employed bee phase: Each employed bee explores a new solution from an existing solution, and the greedy selection is used to retain the better solution. When the employed bees complete new solutions, they share important solution information with other bees.

(3) Onlooker bee phase: Each onlooker bee selects an existing solution depending on the solution's probability and explores a better solution.

(4) Scout bee phase: If a solution fails to improve after *limit* trials, the solution is abandoned. The employed bee becomes a scouter and randomly generates a new solution through initialization to replace the abandoned solution.

Phases (2) – (4) are repeated until the termination condition is satisfied, and the near-optimal solution is returned.

B. EXTENDED ARTIFICIAL BEE COLONY ALGORITHM

The EABC algorithm is proposed in this study to effectively solve the JSS-JF model. First, a new three-dimensional encoding scheme is proposed to describe the complex structure of the JSS-JF solution. This is because the traditional one-dimensional encoding scheme of the basic ABC algorithm cannot be used to represent the JSS-JF solution. Then, the crossover and mutation operators, local search, and elite replacement strategy are integrated into the EABC algorithm to enhance its search ability and accelerate convergence. The framework of the EABC algorithm is shown in Figure 2.

1) SOLUTION REPRESENTATION

The representation of a reasonable JSS-JF solution is crucial to enable the EABC algorithm to adapt to the proposed JSS-JF model. To describe the complex structure of the JSS-JF solution accurately, a new three-dimensional encoding scheme for the EABC algorithm is proposed. It is composed of three rows, representing jobs, execution modes, and human operators/repair machines. The first row represents the sorted permutation of the reprocessed jobs. The second row denotes the execution modes of the corresponding jobs. The third row indicates the human operators or repair machines depending on the corresponding execution modes for processing the jobs. If the job is processed with the

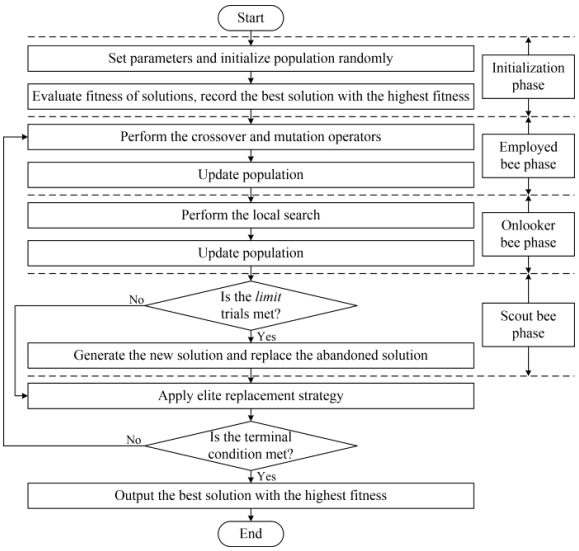

FIGURE 2. The framework of the EABC algorithm.

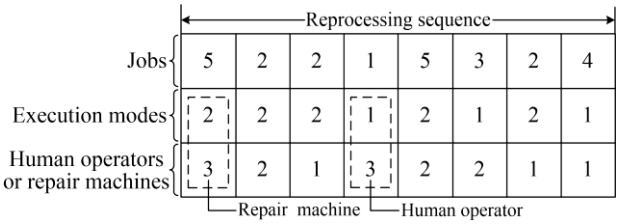

FIGURE 3. An example: the three-dimensional encoding scheme of a JSS-JF solution.

replacement mode, the value in the third row denotes the human operator. If the job is processed with the repair mode, the value in the third row denotes the repair machine.

A simple example of the solution (involving five jobs, three human operators, and three repair machines) is shown in Figure 3. It can be seen from Figure 3 that jobs J_1, J_3 and J_4 are executed with replacement mode, while jobs J_2 and J_5 are executed with repair mode. Each repair job may consist of multiple operations, and the occurrence times of the repair job indicate its operation index. The first element "5" in the first row indicates the first operation (O_{51}) of repair job J_5, and the second occurrence of "5" indicates the second operation (O_{52}) of repair job J_5. The first column indicates that the first operation (O_{51}) of repair job J_5 is processed on repair machine MA_3. The fourth column indicates that replacement job J_1 is processed by the 3*rd* human operator.

The solution in Figure 3 can be decoded from left to right. First, each job and the corresponding execution mode must be identified. Then, the human operator or repair machine for the job is determined according to the corresponding execution mode. Finally, the sequence of reprocessed jobs depends on the left-to-right positions of the encoding scheme. Based on the decoding scheme, a schedule can be determined.

2) EMPLOYED BEE PHASE

In the employed bee phase, each employed bee seeks a new solution from an existing solution. In the EABC algorithm, the crossover and mutation operators are utilized to enhance the diversity of solutions. Based on the three-dimensional encoding scheme, crossover and mutation operators are designed to evolve the population. The details of the crossover and mutation operators are as follows.

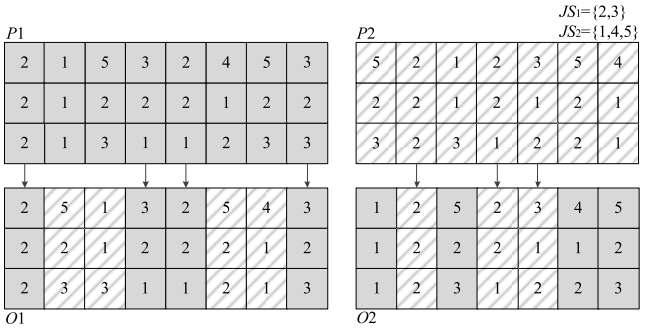

FIGURE 4. An illustrative example of the precedence operation crossover.

(1) Crossover operator

The precedence operation crossover [39] is employed to seek a new solution. It is implemented through the illustrative example shown in Figure 4, according to the following steps.

Step 1: Randomly select two parent solutions $P1$ and $P2$. Randomly divide the jobs into two sets of jobs JS_1 and JS_2.

Step 2: Copy the columns of $P1$ with the jobs in JS_1 to the offspring solution $O1$ in the same positions, and copy the columns of $P2$ with the jobs in JS_2 to the offspring solution $O1$ in the same order.

Step 3: Copy the columns of $P2$ with the jobs in JS_1 to the offspring solution $O2$ in the same positions, and copy the columns of $P1$ with the jobs in JS_2 to the offspring solution $O2$ in the same order.

(2) Mutation operator

The mutation operator selects several positions in the third row and replaces the existing human operators or repair machines with candidate ones.

The crossover and mutation operators are executed under certain probabilities including crossover rate and mutation rate respectively. Then, new solutions are generated, and the greedy selection is used to retain the solution with a higher fitness value, i.e., the objective function that is calculated by Equation (3).

When all employed bees complete the exploration of new solutions, they return to the hive and share important solution information with the onlooker and scout bees.

3) ONLOOKER BEE PHASE

In the onlooker bee phase, each onlooker bee explores the solutions found by the employed bees. The local search is used to enhance the search ability. Based on the three-dimensional encoding scheme, four local search operators are utilized (one-swap, one-insert, two-swap, and two-insert operators), as illustrated in Figure 5.

(1) The one-swap operator is shown in Figure 5 (a), which randomly selects two columns in different positions and swaps them.

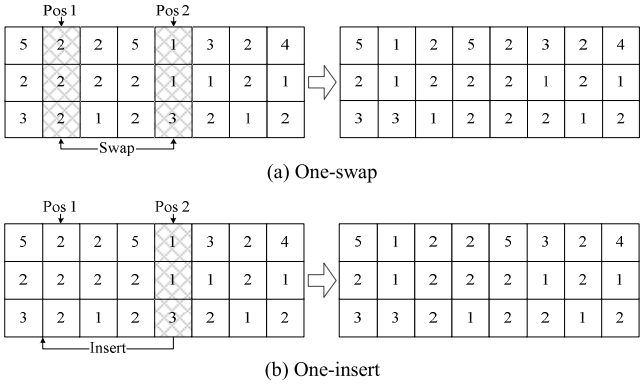

FIGURE 5. The main procedure of one-swap and one-insert operators.

(2) The one-insert operator is shown in Figure 5 (b), which randomly selects two columns in different positions, and inserts the latter column in front of the former column.

(3) The main procedure of the two-swap operator is the implementation of the one-swap process twice.

(4) The main procedure of the two-insert operator is the execution of the one-insert process twice.

Each onlooker bee randomly selects an existing solution under a certain probability and explores the neighborhood of this solution using one local search operator with a random selection. Then, new solutions are generated, and the greedy selection retains the solution with higher fitness value.

4) SCOUT BEE PHASE

The scout bee phase is used to escape from the local optima of the ABC algorithm. If a solution fails to improve after *limit* trials, the solution is abandoned. The corresponding employed bee becomes a scouter and randomly explores a new solution through initialization to replace the abandoned solution. However, it is hard to generate a new solution better than the previous solution based on single exploration in the ABC algorithm. Thus, each scout bee makes 10 explorations to increase the probability of obtaining the best solution with the highest fitness value to replace the abandoned solution.

5) ELITE REPLACEMENT STRATEGY

The elite replacement strategy [12], [13] is implemented to preserve the elite solution with the highest fitness value and replace the worst solution in the next generation. This strategy can prevent the elite solution from being lost and accelerate the convergence of the EABC algorithm.

V. SIMULATION EXPERIMENTS

This section presents the three sets of simulation experiments that are conducted to demonstrate the practicality and effectiveness of the presented EABC algorithm in solving the proposed JSS-JF model by comparing with the other six population-based algorithms, such as basic GA, basic PSO algorithm, basic ABC algorithm, hybrid genetic algorithm (HGA) [33], hybrid particle swarm optimization (HPSO) [20], and hybrid artificial bee colony algorithm (HABC) [10]. Each simulation experiment was executed 10 times to improve the robustness of the results, and the average results were used for evaluation. All experimental data of this study have been uploaded in the Figshare (https://doi.org/10.6084/m9.figshare.13530941). The simulation experiments were implemented using Python programming language on a personal computer with Windows 10 64-bits, AMD Ryzen 7 3700X 8-Core processor at 3.60 GHz and 40 GB RAM.

TABLE 1. The configurations for instances.

Scale	Instance	Number of jobs	Number of job families	Number of human operators	Number of repair machines
Small-scale	S01	6	3	3	3
	S02	8	3	3	4
	S03	10	3	4	4
	S04	12	3	4	5
Medium-scale	M01	16	5	5	5
	M02	18	5	5	6
	M03	20	5	6	6
	M04	22	5	6	6
Large-scale	L01	36	7	6	6
	L02	38	7	6	7
	L03	40	7	7	7
	L04	42	7	7	8

TABLE 2. The processing times and costs of the replacement job and repair job.

Job type	The processing time	The cost
Replacement job	$HPT_i^n \in [4, 8]$	$HRC_i^n \in [60, 100]$
Repair job	$OPT_{il}^k \in [5, 20]$, with $L_i \in [3, 6]$	$ORC_{il}^k \in [3, 10]$

A. EXPERIMENTAL SETUP

In this study, instances with three scales were used to simulate the real environment of remanufacturing systems, specifically small-scale, medium-scale, and large-scale instances. The configurations for each instance are summarized in Table 1.

The processing time and replacement cost of each replacement job, and the processing time and repair cost of each repair operation were randomly generated within certain ranges as shown in Table 2.

B. PARAMETER DESIGN

The parameters of each algorithm are summarized in Table 3.

The first set of simulation experiments was applied on a medium-scale instance M03 to determine the maximum number of the function evaluations for the seven population-based algorithms being compared. For each algorithm, the initial population size, weights of total completion time, and total cost were set as 45, 0.7, and 0.3, respectively. The evolutionary curves of the average fitness values obtained from the seven algorithms are presented in Figure 6. It can be observed that the EABC algorithm and GA are found to converge in 3020 function evaluations, and exhibit a better convergence ability than the other five baseline algorithms. The figure shows that the fitness value obtained from the EABC algorithm is better than those obtained from the six baseline algorithms when all evolutionary curves tend to be stable. It means that the EABC algorithm has a better search ability

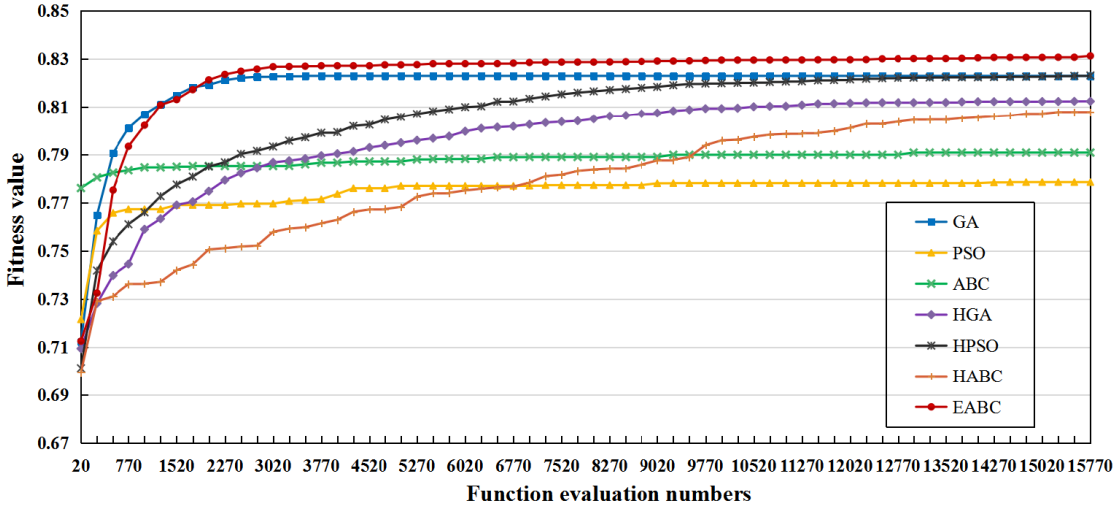

FIGURE 6. Evolutionary curves of the average fitness values obtained from seven algorithms.

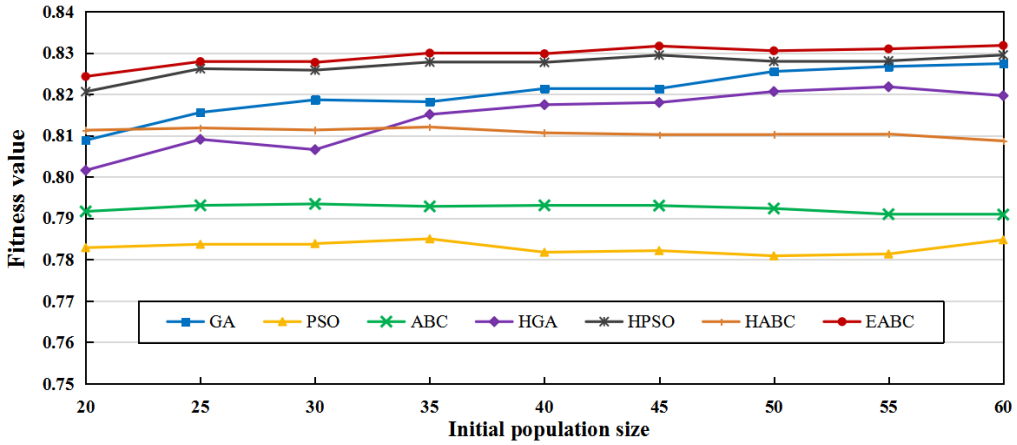

FIGURE 7. Experimental results of the average fitness values of seven algorithms with different initial population sizes.

than other algorithms. Furthermore, the HGA algorithm and HPSO algorithm converge slowly, and the PSO algorithm and ABC algorithm fall into local optima. The average fitness value obtained from the HABC algorithm is increased slowly, and tends to be stable in 15000 function evaluations. Based on the performances of the seven algorithms, the maximum number of the function evaluations is selected as 15000 in the subsequent experiments to ensure fair comparisons.

The second set of simulation experiments was applied on the medium-scale instance M03 to select the reasonable initial population size of the seven population-based algorithms; the weights of the total completion time and total cost were set as 0.7 and 0.3, respectively. The experimental results of the average fitness values obtained from the seven algorithms when the initial population size increases from 20 to 60 with an increment of five are shown in Figure 7. The average fitness values obtained from the EABC algorithm obviously exceed those from other algorithms, regardless of the initial population size. The average fitness values obtained from the EABC algorithm and HPSO algorithm have a slight upward trend when the initial population size increases from 20 to 45 and tend to be stable when the initial population size exceeds 45. When the initial population size increases from 20 to 60, the average fitness values obtained from GA have an upward trend. The average fitness values obtained from HGA algorithm tend to be fluctuant when the initial

TABLE 3. The parameters of each algorithm.

Algorithm	Parameters
GA	the crossover rate = 0.8, the mutation rate = 0.1
PSO	the inertia weight = 0.9, the two acceleration coefficients = 3.0
ABC	the limit = 50
HGA	the reproduction probability = 0.02, the crossover rate = 0.8, the mutation rate = 0.5
HABC	the crossover rate = 0.7, the mutation rate = 0.15
EABC	the crossover rate = 0.8, the mutation rate = 0.1, the limit = 50, the population proportion for the local search = 0.3

Note: The HPSO algorithm is parameterless. Thus, it is not listed above.

TABLE 4. Experimental results of the EABC algorithm and three basic algorithms (weight combinations set as 0.7 and 0.3 respectively).

Instance	EABC			GA			PSO			ABC		
	Max	*Ave*	*SD*	*Max*	*Ave*	*SD*	*Max*	*Ave*	*SD*	*Max*	*Ave*	*SD*
S01	**0.8099**	**0.8071**	**0.0024**	0.8091	0.8012	0.0034	0.8016	0.7990	0.0024	0.8053	0.8009	0.0024
S02	**0.8192**	**0.8149**	**0.0021**	0.8172	0.8106	0.0050	0.8107	0.7957	0.0085	0.8026	0.7886	0.0053
S03	**0.8153**	**0.8125**	**0.0026**	0.8130	0.8062	0.0045	0.7894	0.7838	0.0037	0.7917	0.7840	0.0038
S04	**0.8234**	**0.8169**	**0.0030**	0.8186	0.8126	0.0052	0.7955	0.7839	0.0060	0.7932	0.7811	0.0060
M01	**0.8286**	**0.8248**	**0.0029**	0.8253	0.8193	0.0048	0.7847	0.7751	0.0060	0.7889	0.7785	0.0064
M02	**0.8368**	**0.8317**	0.0035	0.8349	0.8285	0.0047	0.8095	0.7926	0.0080	0.8055	0.7989	0.0030
M03	**0.8395**	**0.8322**	0.0043	0.8328	0.8265	0.0046	0.7961	0.7871	0.0053	0.7976	0.7913	0.0040
M04	**0.8407**	**0.8358**	**0.0027**	0.8368	0.8303	0.0036	0.8013	0.7898	0.0071	0.8026	0.7986	0.0028
L01	**0.8326**	**0.8278**	**0.0022**	0.8318	0.8267	0.0027	0.7834	0.7610	0.0119	0.7957	0.7888	0.0041
L02	**0.8448**	**0.8363**	**0.0041**	0.8441	0.8335	0.0052	0.7740	0.7680	0.0050	0.8009	0.7947	0.0047
L03	**0.8522**	**0.8470**	0.0036	0.8507	0.8442	0.0037	0.7972	0.7768	0.0118	0.8040	0.8015	0.0025
L04	**0.8388**	**0.8338**	**0.0039**	0.8359	0.8311	0.0041	0.7876	0.7638	0.0128	0.8040	0.7955	0.0044

Note: The indicator is marked in bold if the EABC algorithm outperforms the three basic algorithms in this indicator.

TABLE 5. Experimental results of the EABC algorithm and three basic algorithms (weight combinations set as 0.5 and 0.5 respectively).

Instance	EABC			GA			PSO			ABC		
	Max	*Ave*	*SD*	*Max*	*Ave*	*SD*	*Max*	*Ave*	*SD*	*Max*	*Ave*	*SD*
S01	**0.7443**	**0.7364**	**0.0034**	0.7411	0.7306	0.0060	0.7295	0.7225	0.0034	0.7258	0.7190	0.0037
S02	**0.7557**	0.7484	0.0062	0.7531	0.7371	0.0072	0.7291	0.7184	0.0064	0.7148	0.7098	0.0050
S03	**0.7387**	**0.7328**	0.0037	0.7349	0.7306	0.0032	0.7027	0.6984	0.0030	0.7004	0.6915	0.0041
S04	**0.7612**	**0.7523**	**0.0039**	0.7563	0.7463	0.0060	0.7112	0.7024	0.0050	0.7071	0.6982	0.0061
M01	**0.7536**	**0.7493**	**0.0030**	0.7536	0.7442	0.0086	0.6949	0.6869	0.0050	0.6877	0.6805	0.0049
M02	**0.7691**	**0.7633**	0.0031	0.7660	0.7599	0.0049	0.7174	0.7135	0.0022	0.7219	0.7135	0.0053
M03	**0.7606**	**0.7555**	**0.0034**	0.7561	0.7481	0.0061	0.7021	0.6970	0.0045	0.7032	0.7004	0.0038
M04	**0.7678**	**0.7617**	**0.0031**	0.7622	0.7556	0.0049	0.7147	0.7052	0.0049	0.7268	0.7092	0.0077
L01	**0.7532**	**0.7490**	**0.0034**	0.7527	0.7474	0.0043	0.6823	0.6734	0.0063	0.7030	0.6947	0.0041
L02	**0.7745**	**0.7660**	**0.0041**	0.7719	0.7635	0.0061	0.6988	0.6887	0.0049	0.7176	0.7066	0.0047
L03	**0.7870**	**0.7732**	0.0060	0.7823	0.7717	0.0056	0.7045	0.6981	0.0047	0.7200	0.7163	0.0025
L04	**0.7694**	**0.7603**	**0.0040**	0.7669	0.7583	0.0066	0.6983	0.6819	0.0094	0.7057	0.6952	0.0049

Note: The indicator is marked in bold if the EABC algorithm outperforms the three basic algorithms in this indicator.

TABLE 6. Experimental results of the EABC algorithm and three basic algorithms (weight combinations set as 0.3 and 0.7 respectively).

Instance	EABC			GA			PSO			ABC		
	Max	*Ave*	*SD*	*Max*	*Ave*	*SD*	*Max*	*Ave*	*SD*	*Max*	*Ave*	*SD*
S01	**0.7318**	**0.7280**	**0.0023**	0.7310	0.7191	0.0068	0.6844	0.6760	0.0052	0.7190	0.6915	0.0104
S02	**0.7169**	**0.7029**	0.0062	0.7058	0.6938	0.0062	0.6736	0.6603	0.0056	0.6520	0.6458	0.0046
S03	**0.6951**	**0.6795**	0.0078	0.6848	0.6709	0.0080	0.6440	0.6300	0.0068	0.6465	0.6180	0.0124
S04	**0.7301**	**0.7236**	**0.0031**	0.7252	0.7131	0.0080	0.6617	0.6477	0.0076	0.6548	0.6359	0.0080
M01	**0.7000**	**0.6934**	**0.0048**	0.6973	0.6832	0.0075	0.6184	0.6099	0.0056	0.5994	0.5898	0.0073
M02	**0.7270**	**0.7212**	**0.0038**	0.7205	0.7109	0.0055	0.6536	0.6460	0.0054	0.6456	0.6347	0.0053
M03	**0.7076**	**0.7018**	**0.0031**	0.6951	0.6871	0.0049	0.6394	0.6250	0.0059	0.6214	0.6131	0.0043
M04	**0.7223**	**0.7123**	**0.0053**	0.7177	0.7070	0.0056	0.6392	0.6302	0.0053	0.6318	0.6238	0.0056
L01	**0.6999**	**0.6879**	0.0084	0.6954	0.6816	0.0060	0.6102	0.6006	0.0038	0.6093	0.6053	0.0032
L02	**0.7193**	**0.7098**	**0.0047**	0.7139	0.7039	0.0063	0.6279	0.6178	0.0048	0.6305	0.6208	0.0054
L03	0.7241	**0.7202**	**0.0016**	0.7283	0.7185	0.0042	0.6510	0.6362	0.0065	0.6351	0.6263	0.0049
L04	**0.7147**	**0.7083**	**0.0039**	0.7111	0.6990	0.0076	0.6185	0.6118	0.0040	0.6196	0.6055	0.0079

Note: The indicator is marked in bold if the EABC algorithm outperforms the three basic algorithms in this indicator.

population size is around 30, but have an uptrend in general. Moreover, the average fitness values obtained from GA and HGA algorithms have small changes when the initial population size exceeds 50. The tendencies of the average fitness values obtained from the HABC and ABC algorithms are both stable. The tendency of the average fitness values obtained from PSO is fluctuant when the initial population size increases. On the other hand, if the initial population size is extremely large, the computational time of the algorithms increases significantly [19]. Therefore, for a fair and efficient comparison in the subsequent experiments, the initial population sizes of the algorithms are all set as 50.

C. EXTENDED ARTIFICIAL BEE COLONY ALGORITHM PERFORMANCE IN SOLVING PROPOSED MODEL

To evaluate the performance of the EABC algorithm in solving the proposed JSS-JF model, the third set of simulation experiments was performed on instances with different scales and various weight combinations of total completion time and total cost. In this experiment, the weight combinations were set as 0.7 and 0.3, 0.5 and 0.5, 0.3 and 0.7 respectively. *Max*, *Ave*, and *SD* denote the best fitness value, average fitness value, and standard deviation of fitness values of the 10 runs. The indicator is marked in bold if the EABC algorithm outperforms the three baseline algorithms in this indicator.

TABLE 7. Experimental results of the EABC algorithm and three improved algorithms (weight combinations set as 0.7 and 0.3 respectively).

Instance	EABC			HGA			HPSO			HABC		
	Max	*Ave*	*SD*	*Max*	*Ave*	*SD*	*Max*	*Ave*	*SD*	*Max*	*Ave*	*SD*
S01	**0.8099**	**0.8071**	**0.0024**	0.8095	0.8006	0.0049	0.8099	0.8034	0.0033	0.8099	0.8066	0.0039
S02	**0.8192**	**0.8149**	0.0021	0.8192	0.8117	0.0056	0.8166	0.8109	0.0054	0.8136	0.8115	0.0016
S03	**0.8153**	**0.8125**	**0.0026**	0.8112	0.8037	0.0062	0.8153	0.8115	0.0037	0.8093	0.8064	0.0026
S04	**0.8234**	**0.8169**	**0.0030**	0.8188	0.8023	0.0110	0.8211	0.8158	0.0059	0.8104	0.8053	0.0040
M01	**0.8286**	**0.8248**	**0.0029**	0.8261	0.8120	0.0077	0.8281	0.8199	0.0066	0.8135	0.8066	0.0042
M02	0.8368	**0.8317**	**0.0035**	0.8358	0.8278	0.0053	0.8383	0.8306	0.0037	0.8213	0.8151	0.0035
M03	**0.8395**	**0.8322**	**0.0043**	0.8277	0.8147	0.0087	0.8380	0.8305	0.0053	0.8169	0.8084	0.0079
M04	0.8407	**0.8358**	**0.0027**	0.8401	0.8284	0.0064	0.8414	0.8356	0.0041	0.8149	0.7979	0.0145
L01	**0.8326**	**0.8278**	**0.0022**	0.8236	0.8082	0.0179	0.8318	0.8251	0.0051	0.7557	0.7330	0.0114
L02	**0.8448**	**0.8363**	**0.0041**	0.8298	0.8155	0.0091	0.8347	0.8256	0.0083	0.7577	0.7469	0.0073
L03	**0.8522**	**0.8470**	**0.0036**	0.8424	0.8264	0.0096	0.8427	0.8338	0.0068	0.7752	0.7581	0.0088
L04	0.8388	**0.8338**	**0.0039**	0.8190	0.8071	0.0130	0.8418	0.8199	0.0116	0.7844	0.7503	0.0120

Note: The indicator is marked in bold if the EABC algorithm outperforms the three improved algorithms in this indicator.

TABLE 8. Experimental results of the EABC algorithm and three improved algorithms (weight combinations set as 0.5 and 0.5 respectively).

Instance	EABC			HGA			HPSO			HABC		
	Max	*Ave*	*SD*	*Max*	*Ave*	*SD*	*Max*	*Ave*	*SD*	*Max*	*Ave*	*SD*
S01	**0.7443**	**0.7364**	0.0034	0.7431	0.7322	0.0067	0.7437	0.7362	0.0029	0.7359	0.7348	0.0016
S02	**0.7557**	**0.7484**	0.0062	0.7498	0.7408	0.0048	0.7547	0.7465	0.0060	0.7482	0.7441	0.0031
S03	0.7387	**0.7328**	0.0037	0.7357	0.7276	0.0081	0.7403	0.7321	0.0062	0.7347	0.7297	0.0029
S04	**0.7612**	**0.7523**	**0.0039**	0.7525	0.7406	0.0130	0.7610	0.7501	0.0074	0.7366	0.7227	0.0067
M01	0.7536	**0.7493**	**0.0030**	0.7488	0.7390	0.0083	0.7554	0.7486	0.0068	0.7275	0.7200	0.0061
M02	**0.7691**	**0.7633**	**0.0031**	0.7609	0.7519	0.0066	0.7680	0.7620	0.0057	0.7354	0.7300	0.0031
M03	0.7606	0.7555	**0.0034**	0.7597	0.7453	0.0104	0.7639	0.7561	0.0060	0.7260	0.7093	0.0126
M04	**0.7678**	**0.7617**	**0.0031**	0.7663	0.7468	0.0110	0.7648	0.7602	0.0041	0.7200	0.7000	0.0079
L01	**0.7532**	**0.7490**	**0.0034**	0.7392	0.7245	0.0096	0.7509	0.7360	0.0091	0.6766	0.6629	0.0079
L02	**0.7745**	**0.7660**	**0.0041**	0.7652	0.7490	0.0095	0.7636	0.7505	0.0071	0.6948	0.6799	0.0067
L03	**0.7870**	**0.7732**	0.0060	0.7614	0.7549	0.0067	0.7754	0.7656	0.0065	0.7027	0.6927	0.0041
L04	**0.7694**	**0.7603**	**0.0040**	0.7471	0.7326	0.0097	0.7524	0.7420	0.0083	0.6824	0.6740	0.0057

Note: The indicator is marked in bold if the EABC algorithm outperforms the three improved algorithms in this indicator.

TABLE 9. Experimental results of the EABC algorithm and three improved algorithms (weight combinations set as 0.3 and 0.7 respectively).

Instance	EABC			HGA			HPSO			HABC		
	Max	*Ave*	*SD*	*Max*	*Ave*	*SD*	*Max*	*Ave*	*SD*	*Max*	*Ave*	*SD*
S01	0.7318	**0.7280**	**0.0023**	0.7398	0.7197	0.0131	0.7318	0.7264	0.0028	0.7238	0.7175	0.0044
S02	**0.7169**	**0.7029**	0.0062	0.7041	0.6946	0.0076	0.7130	0.7009	0.0065	0.6960	0.6918	0.0034
S03	**0.6951**	0.6795	0.0078	0.6907	0.6736	0.0117	0.6936	0.6831	0.0083	0.6606	0.6514	0.0041
S04	0.7301	**0.7236**	**0.0031**	0.7212	0.7094	0.0103	0.7324	0.7184	0.0151	0.6892	0.6722	0.0071
M01	**0.7000**	**0.6934**	**0.0048**	0.6907	0.6775	0.0080	0.6989	0.6902	0.0071	0.6312	0.6243	0.0052
M02	**0.7270**	**0.7212**	**0.0038**	0.7073	0.6994	0.0068	0.7232	0.7126	0.0072	0.6790	0.6615	0.0083
M03	**0.7076**	**0.7018**	**0.0031**	0.6941	0.6807	0.0097	0.7044	0.6956	0.0074	0.6399	0.6276	0.0053
M04	**0.7223**	**0.7123**	**0.0053**	0.7087	0.6916	0.0108	0.7218	0.7089	0.0115	0.6443	0.6337	0.0053
L01	**0.6999**	**0.6879**	0.0084	0.6726	0.6618	0.0056	0.6845	0.6765	0.0051	0.6240	0.6026	0.0088
L02	**0.7193**	**0.7098**	**0.0047**	0.7018	0.6835	0.0097	0.7058	0.6960	0.0058	0.6314	0.6211	0.0047
L03	**0.7241**	**0.7202**	**0.0016**	0.6973	0.6881	0.0066	0.7177	0.7101	0.0043	0.6462	0.6388	0.0057
L04	**0.7147**	**0.7083**	**0.0039**	0.6890	0.6731	0.0086	0.7016	0.6909	0.0053	0.6242	0.6157	0.0044

Note: The indicator is marked in bold if the EABC algorithm outperforms the three improved algorithms in this indicator.

By comparing with the basic algorithms and the improved algorithms proposed by other scholars, the superiority of the proposed algorithm is verified. Tables 4 – 6 summarize the experimental results of the four algorithms with various weight combinations, including EABC, GA, PSO, and ABC algorithms. It is observed that the EABC algorithm has a competitive performance over the basic algorithms. In terms of *Max* and *Ave*, the EABC algorithm outperforms the three basic algorithms in most instances whatever the weight combination is set. In terms of *SD*, the EABC algorithm also outperforms them in 9 out of 12 instances when the weight combination is 0.7 and 0.3; in 8 out of 12 instances when the weight combination is 0.5 and 0.5; and in 9 out of 12 instances when the weight combination is 0.3 and 0.7.

Tables 7 – 9 show the experimental results of the four algorithms with various weight combinations, including EABC, HGA, HPSO and HABC algorithms. By observing the values of *Max*, *Ave* and *SD* in various weight combinations, it can be concluded that the EABC algorithm outperforms the other improved algorithms in at least 9 out of 12 instances in terms of *Max*; in at least 11 out of 12 instances in terms of *Ave*; and in at least 8 out of 12 instances in terms of *SD*.

Tables 4 – 9 show that the times of EABC algorithm outperforming the other algorithms are slightly fluctuant under

the various weight combinations, but the EABC algorithm has generally a better performance over other algorithms in solving the proposed JSS-JF model. Therefore, it demonstrates that the various weight combinations have a minimal effect on the EABC algorithm in solving the proposed JSS-JF model. In summary, the EABC algorithm outperforms the six baseline algorithms in terms of practicality and effectiveness.

VI. CONCLUSION

JSS-JF is a critical issue in remanufacturing scheduling. In view of the various damage conditions of components encountered in the remanufacturing process, this study proposed a new JSS-JF model for remanufacturing systems that can be solved well by the EABC algorithm presented above. The main contributions of this study are summarized as follows.

(1) The proposed JSS-JF model for remanufacturing systems was presented, which decomposed the scheduling problem into three sub-problems: execution mode selection, replacement job assignment and sequencing, and repair machine assignment and operation sequencing.

(2) The EABC algorithm, which has extended the basic ABC algorithm in four aspects, including the three-dimensional encoding scheme, crossover and mutation operators, local search and elite replacement strategy, was proposed to solve the proposed JSS-JF model effectively.

Furthermore, simulation experiments based on instances with different scales were designed to illustrate the practicability and effectiveness of the proposed EABC algorithm by comparing with the six baseline algorithms. The comparative experimental results demonstrated that the EABC algorithm outperformed the other baseline algorithms in solving the proposed JSS-JF model.

With regard to future work, this study will be extended in several directions. First, more execution modes will be considered in remanufacturing systems by analyzing the characteristics of execution modes. Second, more practical factors, including none-zero ready time and machine breakdowns, will be explored. The future study will also consider the availability of the human operators to make the model more practical. Finally, the performance of the proposed EABC algorithm will be further enhanced by the integration of heuristic rules.

REFERENCES

[1] F. Berthaut, A. Gharbi, and R. Pellerin, "Joint hybrid repair and remanufacturing systems and supply control," *Int. J. Prod. Res.*, vol. 48, no. 14, pp. 4101–4121, Jul. 2010.
[2] F. Berthaut, R. Pellerin, and A. Gharbi, "Optimisation of the control policy for a stochastic remanufacturing system with an unreliable replacement parts supply," *Int. J. Simul. Process Model.*, vol. 5, no. 3, pp. 205–214, 2009.
[3] G. Ferrer, "The economics of personal computer remanufacturing," *Resour., Conservation Recycling*, vol. 21, no. 2, pp. 79–108, Oct. 1997.
[4] G. Ferrer and J. M. Swaminathan, "Managing new and differentiated remanufactured products," *Eur. J. Oper. Res.*, vol. 203, no. 2, pp. 370–379, 2010.
[5] Y. Fu, M. Zhou, X. Guo, and L. Qi, "Stochastic multi-objective integrated disassembly-reprocessing-reassembly scheduling via fruit fly optimization algorithm," *J. Cleaner Prod.*, vol. 278, Jan. 2021, Art. no. 123364.
[6] K. Z. Gao, Z. M. He, Y. Huang, P. Y. Duan, and P. N. Suganthan, "A survey on meta-heuristics for solving disassembly line balancing, planning and scheduling problems in remanufacturing," *Swarm Evol. Comput.*, vol. 57, Sep. 2020, Art. no. 100719.
[7] K. Z. Gao, P. N. Suganthan, T. J. Chua, C. S. Chong, T. X. Cai, and Q. K. Pan, "A two-stage artificial bee colony algorithm scheduling flexible job-shop scheduling problem with new job insertion," *Expert Syst. Appl.*, vol. 42, no. 21, pp. 7652–7663, Nov. 2015.
[8] K. Z. Gao, P. N. Suganthan, Q. K. Pan, M. F. Tasgetiren, and A. Sadollah, "Artificial bee colony algorithm for scheduling and rescheduling fuzzy flexible job shop problem with new job insertion," *Knowl.-Based Syst.*, vol. 109, pp. 1–16, Oct. 2016.
[9] A. Gharbi, R. Pellerin, and J. Sadr, "Production rate control for stochastic remanufacturing systems," *Int. J. Prod. Econ.*, vol. 112, no. 1, pp. 37–47, Mar. 2008.
[10] G. Gong, R. Chiong, Q. Deng, and X. Gong, "A hybrid artificial bee colony algorithm for flexible job shop scheduling with worker flexibility," *Int. J. Prod. Res.*, vol. 58, no. 14, pp. 4406–4420, Jul. 2020.
[11] S. Hartmann and A. Drexl, "Project scheduling with multiple modes: A comparison of exact algorithms," *Networks*, vol. 32, no. 4, pp. 283–297, Dec. 1998.
[12] Z.-H. Jia, L.-Y. Gao, and X.-Y. Zhang, "A new history-guided multi-objective evolutionary algorithm based on decomposition for batching scheduling," *Expert Syst. Appl.*, vol. 141, Mar. 2020, Art. no. 112920.
[13] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Dept. Comput. Eng. Fac., Erciyes Univ., Kayseri, Turkey, Tech. Rep. TR06, 2005.
[14] D. Karaboga and B. Akay, "A comparative study of artificial bee colony algorithm," *Appl. Math. Comput.*, vol. 214, no. 1, pp. 108–132, 2009.
[15] D. Karaboga and B. Basturk, "On the performance of artificial bee colony (ABC) algorithm," *Appl. Soft Comput.*, vol. 8, pp. 687–697, Jan. 2008.
[16] J.-S. Kim, J.-H. Park, and D.-H. Lee, "Iterated greedy algorithms to minimize the total family flow time for job-shop scheduling with job families and sequence-dependent set-ups," *Eng. Optim.*, vol. 49, no. 10, pp. 1719–1732, Oct. 2017.
[17] M. Lage Junior and M. Godinho Filho, "Master disassembly scheduling in a remanufacturing system with stochastic routings," *Central Eur. J. Oper. Res.*, vol. 25, no. 1, pp. 123–138, Mar. 2017.
[18] J. Li, P. Duan, H. Sang, S. Wang, Z. Liu, and P. Duan, "An efficient optimization algorithm for resource-constrained steelmaking scheduling problems," *IEEE Access*, vol. 6, pp. 33883–33894, 2018.
[19] X. Li and S. Ma, "Multiobjective discrete artificial bee colony algorithm for multiobjective permutation flow shop scheduling problem with sequence dependent setup times," *IEEE Trans. Eng. Manag.*, vol. 64, no. 2, pp. 149–165, May 2017.
[20] X. Li, L. Gao, W. Wang, C. Wang, and L. Wen, "Particle swarm optimization hybridized with genetic algorithm for uncertain integrated process planning and scheduling with interval processing time," *Comput. Ind. Eng.*, vol. 135, pp. 1036–1046, Sep. 2019.
[21] D. Lin, C. C. Teo, and C. K. M. Lee, "Heuristics for integrated job assignment and scheduling in the multi-plant remanufacturing system," *Int. J. Prod. Res.*, vol. 53, no. 9, pp. 2674–2689, May 2015.
[22] J. D. Linton and V. Jayaraman, "A framework for identifying differences and similarities in the managerial competencies associated with different modes of product life extension," *Int. J. Prod. Res.*, vol. 43, no. 9, pp. 1807–1829, May 2005.
[23] W. Liu, J. Zhang, M. Jin, S. Liu, X. Chang, N. Xie, and Y. Wang, "Key indices of the remanufacturing industry in China using a combined method of grey incidence analysis and grey clustering," *J. Cleaner Prod.*, vol. 168, pp. 1348–1357, Dec. 2017.
[24] A. Lova, P. Tormos, M. Cervantes, and F. Barber, "An efficient hybrid genetic algorithm for scheduling projects with resource constraints and multiple execution modes," *Int. J. Prod. Econ.*, vol. 117, no. 2, pp. 302–316, Feb. 2009.
[25] A. Raihanian Mashhadi and S. Behdad, "Optimal sorting policies in remanufacturing systems: Application of product life-cycle data in quality grading and end-of-use recovery," *J. Manuf. Syst.*, vol. 43, pp. 15–24, Apr. 2017.
[26] R. Pellerin and A. Gharbi, "Production control of hybrid repair and remanufacturing systems under general conditions," *J. Qual. Maintenance Eng.*, vol. 15, no. 4, pp. 383–396, Sep. 2009.
[27] M. Sakawa and T. Mori, "An efficient genetic algorithm for job-shop scheduling problems with fuzzy processing time and fuzzy duedate," *Comput. Ind. Eng.*, vol. 36, no. 2, pp. 325–341, Apr. 1999.

[28] J. Shi, W. Zhang, S. Zhang, W. Wang, J. Lin, and R. Feng, "A new environment-aware scheduling method for remanufacturing system with non-dedicated reprocessing lines using improved flower pollination algorithm," *J. Manuf. Syst.*, vol. 57, pp. 94–108, Oct. 2020.
[29] A. Singh, "An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem," *Appl. Soft Comput.*, vol. 9, no. 2, pp. 625–631, 2009.
[30] G. Tian, Y. Ren, Y. Feng, M. Zhou, H. Zhang, and J. Tan, "Modeling and planning for dual-objective selective disassembly using AND/OR graph and discrete artificial bee colony," *IEEE Trans. Ind. Informat.*, vol. 15, no. 4, pp. 2456–2468, Apr. 2019.
[31] Z. Wang and T. Y. Liu, "A novel multi-objective scheduling method for energy based unrelated parallel machines with auxiliary resource constraints," *IEEE Access*, vol. 7, pp. 168688–168699, 2019.
[32] Z. Wang, J. Zhang, and S. Yang, "An improved particle swarm optimization algorithm for dynamic job shop scheduling problems with random job arrivals," *Swarm Evol. Comput.*, vol. 51, Dec. 2019, Art. no. 100594.
[33] H. Xia, X. Li, and L. Gao, "A hybrid genetic algorithm with variable neighborhood search for dynamic integrated process planning and scheduling," *Comput. Ind. Eng.*, vol. 102, pp. 99–112, Dec. 2016.
[34] W. Xia and Z. Wu, "An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems," *Comput. Ind. Eng.*, vol. 48, no. 2, pp. 409–425, Mar. 2005.
[35] M. H. Yin, X. T. Li, and J. P. Zhou, "An efficient job shop scheduling algorithm based on artificial bee colony," *Sci. Res. Essays*, vol. 6, no. 12, pp. 2578–2596, 2011.
[36] J.-M. Yu and D.-H. Lee, "Scheduling algorithms for job-shop-type remanufacturing systems with component matching requirement," *Comput. Ind. Eng.*, vol. 120, pp. 266–278, Jun. 2018.
[37] J. M. Yu and D. H. Lee, "Solution algorithms to minimise the total family tardiness for job shop scheduling with job families," *Eur. J. Ind. Eng.*, vol. 12, no. 1, pp. 1–23, 2018.
[38] J.-M. Yu, J.-S. Kim, and D.-H. Lee, "Scheduling algorithms to minimise the total family flow time for job shops with job families," *Int. J. Prod. Res.*, vol. 49, no. 22, pp. 6885–6903, Nov. 2011.
[39] C. Zhang, Y. Rao, and P. Li, "An effective hybrid genetic algorithm for the job shop scheduling problem," *Int. J. Adv. Manuf. Technol.*, vol. 39, nos. 9–10, pp. 965–974, Nov. 2008.
[40] J. Zhang, W. Wang, and X. Xu, "A hybrid discrete particle swarm optimization for dual-resource constrained job shop scheduling with resource flexibility," *J. Intell. Manuf.*, vol. 28, no. 8, pp. 1961–1972, 2017.
[41] L. Zhang, L. Gao, and X. Li, "A hybrid genetic algorithm and Tabu search for a multi-objective dynamic job shop scheduling problem," *Int. J. Prod. Res.*, vol. 51, no. 12, pp. 3516–3531, Jun. 2013.
[42] R. Zhang and C. Wu, "An artificial bee colony algorithm for the job shop scheduling problem with random processing times," *Entropy*, vol. 13, no. 9, pp. 1708–1729, 2011.
[43] S. Zhang, S. Xu, X. Huang, W. Zhang, and M. Chen, "Networked correlation-aware manufacturing service supply chain optimization using an extended artificial bee colony algorithm," *Appl. Soft Comput.*, vol. 76, pp. 121–139, Mar. 2019.
[44] J. Zhou, Q. Deng, and T. Li, "Optimal acquisition and remanufacturing policies considering the effect of quality uncertainty on carbon emissions," *J. Cleaner Prod.*, vol. 186, pp. 180–190, Jun. 2018.
[45] L. Zhu, P. Li, G. Shen, and Z. Liu, "A novel service composition algorithm for cloud-based manufacturing environment," *IEEE Access*, vol. 8, pp. 39148–39164, 2020.
[46] W.-Q. Zou, Q.-K. Pan, and M. F. Tasgetiren, "An effective discrete artificial bee colony algorithm for scheduling an automatic-guided-vehicle in a linear manufacturing workshop," *IEEE Access*, vol. 8, pp. 35063–35076, 2020.

XIANGQI LIU received the Ph.D. degree in mechanical engineering from Zhejiang Sci-Tech University, China, in November 2017. She was with the School of Information Engineering, Hangzhou Dianzi University, Hangzhou, China. She is currently a full-time Lecturer with Hangzhou Dianzi University Information Engineering School, China. She has published about ten articles in journals in the recent years, covering mechanical engineering and robot vision.

JIE CHEN is currently pursuing the M.S. degree with the School of Information Management and Artificial Intelligence, Zhejiang University of Finance and Economics, Hangzhou, China. Her current research interest includes remanufacturing scheduling.

XIAOLING HUANG is currently a Lecturer with the Library, Zhejiang University of Finance and Economics, Hangzhou, China. She has published four articles in international journals in the recent three years, covering a wide range of data mining, artificial intelligence, and supply chain optimization.

SHANSHAN GUO is currently a Lecturer with the Library, Zhejiang University of Finance and Economics, Hangzhou, China. She has published six articles in international journals in the recent five years, covering a wide range of data mining, business intelligence, and manufacturing automation.

SHUAI ZHANG received the Ph.D. degree in mechanical engineering from Zhejiang University, China, in March 2005. He is currently a full-time Professor with the School of Information Management and Artificial Intelligence, Zhejiang University of Finance and Economics, Hangzhou, China. He has published more than 50 articles in international journals in the recent ten years, covering intelligent manufacturing, artificial intelligence, and data mining.

MENGJIAO CHEN is currently pursuing the bachelor's degree with the School of Information Management and Artificial Intelligence, Zhejiang University of Finance and Economics, Hangzhou, China.