Guo *et al.*
Chinese Journal of Mechanical Engineering (2022) 35:143
https://doi.org/10.1186/s10033-022-00812-2

Chinese Journal of Mechanical Engineering

ORIGINAL ARTICLE

Open Access

Neighborhood Modularization-based Artificial Bee Colony Algorithm for Disassembly Planning with Operation Attributes

Hongfei Guo[1], Linsheng Zhang[1], Yaping Ren[1*], Leilei Meng[2], Zhongwei Zhou[1] and Jianqing Li[1,3]

Abstract

The recycling and remanufacturing of end-of-life products are significant for environmental protection and resource conservation. Disassembly is an essential process of remanufacturing end-of-life products. Effective disassembly plans help improve disassembly efficiency and reduce disassembly costs. This paper studies a disassembly planning problem with operation attributes, in which an integrated decision of the disassembly sequence, disassembly directions, and disassembly tools are made. Besides, a mathematical model is formulated with the objective of minimizing the penalty cost caused by the changing of operation attributes. Then, a neighborhood modularization-based artificial bee colony algorithm is developed, which contains a modular optimized design. Finally, two case studies with different scales and complexities are used to verify the performance of the proposed approach, and experimental results show that the proposed algorithm outperforms the two existing methods within an acceptable computational time.

Keywords: End-of-life products, Disassembly planning, Artificial bee colony, Neighborhood modularization

1 Introduction

The awareness of environmental production and resource conservation has been widespread in recent decades, which has caused the government and society to pay attention to product recycling [1, 2]. Thus, many countries have introduced various policies to recover end-of-life (EOL) products effectively. Environmental pollution and resource shortage can be alleviated by using recycling technology to reuse EOL products [3]. Disassembly is an essential process in recovering EOL products [4, 5], which refers to the decomposition of a complete product into parts. In the process, we need to figure out the functions of product parts and the relationship between them.

Before implementing the disassembly process, determining the optimal disassembly sequences (disassembly scheme), which is the critical objective of disassembling planning (DP), helps improve the disassembly efficiency. The disassembly schemes based on the product structure and product design experience are favorable to the disassembly process's objective estimation. For the product to reach the end of its life cycle, DP can execute an effective benefit evaluation and realize the most profitable or least costly disassembly process [6]. The disassembly process can be divided into two disassembly types according to disassembly level, selective disassembly, and complete disassembly [7]. Selective disassembly, also named partial disassembly, pays much attention to the parts worth disassembling. The product is fully disassembled in completed disassembly to achieve the optimal objective, such as the maximum profit or minimum cost. Our work focuses on the studying of complete disassembly.

After years of work efforts, plenty of research has been conducted on disassembly plans. For example, Giri and Kanthababu [8] presented a novel method to produce complete disassembly sequences using a part interference matrix. The experiment studies verified the technique's

*Correspondence: renyp1@163.com

[1] GBA and B&R International Joint Research Center for Smart Logistics, School of Intelligent Systems Science and Engineering, Institute of Physical Internet, Jinan University (Zhuhai Campus), Zhuhai 519070, China
Full list of author information is available at the end of the article

effectiveness. Kheder et al. [9] proposed a genetic algorithm to determine a feasible disassembly sequence by considering several criteria. Hsu [10] developed a fuzzy knowledge-based disassembly planning system for disassembly process planning, which was found flexible and expandable. Zhang et al. [11] studied parallel disassembly sequence planning, and a solution method based on fuzzy-rough sets is proposed to reduce the time complexity. Subsequently, two cases verified the effectiveness of the method. ElSayed et al. [12] presented a genetic algorithm to address the optimal disassembly sequencing of EOL products, and a numerical example validated the proposed method. Tian et al. [13] focused on an uncertain disassembly environment. In their work, an original probability analysis method is proposed, illustrated by some numerical examples.

All the above research focuses on complete disassembly. Although the optimal/near-optimal disassembly solutions can be solved in these papers, they can only determine the disassembly sequences without considering the operation attributes, referred to as disassembly sequence planning (DSP). The selection of disassembly directions and tools also significantly affects the quality of the disassembly solution, which is out of the scope of a DSP. Thus far, a few studies consider the selection of disassembly directions and tools in the DP, such as Kongar and Gupta [14] and Tseng et al. [15, 16]. However, they do not develop a specific mathematical model to present the relationship between disassembly sequence, direction, and tool. Therefore, this paper presents a mathematical model of disassembly planning with operation attributes (DPOA), and the integrated decision on the disassembly sequence, directions, and tools is made.

The optimal disassembly solution often requires an exhaustive search because of the NP-complete characteristics of the DP problem [17]. It is computationally challenging to solve medium and large-scale instances by the solution methods based on graph theory. Many (meta) heuristics are widely proposed in the current work to determine a high-quality disassembly solution [18], for example, genetic algorithm [19, 20], particle swarm optimization [21, 22], Tabu search [23], and ant colony algorithm [24, 25]. Furthermore, the hybrid algorithms of multiple metaheuristics are also considered as solution methods for the disassembly problem [26, 27]. The artificial bee colony (ABC) algorithm is one of the most popular metaheuristics, the effectiveness of which is validated in solving disassembly problems [28–30] and other optimization problems [31, 32]. Moreover, ABC is an easy-to-understand technique that can provide fast and cost-effective solutions to issues that would otherwise cost plenty of time.

The solution complexity of DPOA is higher than a single DSP problem since the problem is an integrated decision problem and more disassembly information is considered. Therefore, a neighborhood modularization-based artificial bee colony (NM-ABC) algorithm is proposed, which contains a modular optimized design. Two module concept-based neighborhood structures are developed to improve the exploitation ability of NM-ABC in the employed bee phase and onlooker bee phase, respectively. They help guide the search in a better direction by extracting the subsequence with the best fitness value, making the algorithm quickly converge to a high-quality disassembly solution. After that, a global optimal solution-based learning strategy is adopted in the scout bee phase to generate a new disassembly solution. This strategy guarantees the quality of the disassembly solution while avoiding the proposed algorithm from falling into the local optimum. Moreover, to execute the NM-ABC in solving DPOA efficiently, the proposed algorithm's neighborhood structures and optimization operators are well developed according to the problem-specific characteristics.

The contributions of our work are as follows:

(1) This work presents a mathematical model of DPOA, in which an integrated decision of the disassembly sequence, disassembly directions, and disassembly tools is made.
(2) Module-based crossover (MBX) and module-based insertion (MBI) neighborhood structures are designed in NM-ABC to improve the exploitation ability.
(3) Experiment results show that NM-ABC can obtain a higher-quality disassembly solution than two existing algorithms in the same computational time.

The remaining structures of this paper are arranged as follows. Section 2 presents the DPOA problem and constructs its mathematical model. In Section 3, the proposed algorithm, NM-ABC, and its structural design are presented. The comparison experiments and computational results on two cases of different scales are presented in Section 4. Finally, Section 5 gives the concluding remarks and future scope.

2 Problem Statement

The process of disassembling EOL products contains many operation attributes, which affect the efficiency and cost of disassembly. To reduce the impact, the objective of DPOA is to minimize the penalty costs caused by changing the disassembly directions and tools. It is worth noting that DPOA is a complete disassembly that separates all parts from a product to improve the part recycling rate. Moreover, the mathematical model of DPOA is constructed on a series of assumptions listed as follows:

(1) Precedence relationships between tasks must be met.
(2) EOL products to be disassembled contain all parts with no deletions, modifications, or additions.
(3) All product parts are detachable.

2.1 Precedence Relationship Constraints

Due to engineering structure and production technology limitations, there is always a particular sequence for the connection of tasks, i.e., the precedence relationship among tasks. A precedence relationship graph in Kongar and Gupta's work [14] describes how to construct the DPOA in the following paragraphs.

As presented in Figure 1, the example product contains ten parts, divided into ten tasks. For this product, there is a one-to-one correspondence between parts and tasks. Each node i, $i = \{1, \ldots, N\}$, denotes a task of the product, where N represents the total number of tasks. Furthermore, some dummy tasks A_s, $s = \{1, \ldots, S\}$, are used to represent complex precedence relationships. It is worth noting that the disassembly of dummy tasks does not consume costs, and no additional part is recovered. Node A_0 indicates an initial point that is the product to be disassembled. For example, tasks 2 and 3 have arrows pointing to task A_1, so task A_1 can only be disassembled after tasks 2 and 3 are done. Similarly, since tasks 5 and 6 are pointed by the arrow from task 7, the two tasks can only be performed after task 7 is processed. Our work adopts the task precedence relationship matrix $\boldsymbol{P} = [p_{ij}]$ to indicate the priority of task execution [33]. The value of p_{ij} is determined by Eq. (1), and the task precedence relationship matrix of the product from Figure 1 is shown in Eq. (2).

$$p_{ij} = \begin{cases} 1 \text{ if task } i \text{ precede task } j, \\ 0 \text{ otherwise}, \end{cases} \tag{1}$$

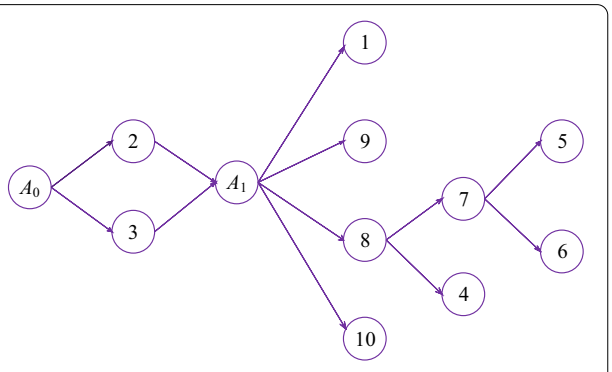

Figure 1 The task precedence relationships of one example product

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \end{array} \begin{array}{c} \begin{array}{cccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}. \tag{2}$$

2.2 The Impact of Disassembly Directions and Tools on Penalty Costs

The main notations used in our model are shown in Table 1.

For the disassembly direction, we use the same design as the work of Kongar and Gupta [14], including six directions: $\pm X$, $\pm Y$, $\pm Z$. During the disassembly process, for the disassembly direction remaining the same, there is no additional penalty cost; a penalty cost of 1 is given for a 90° change; if a 180° change is a need, the penalty cost is 2. The penalty cost matrix for the disassembly directions is shown in Table 2.

In the disassembly tool, an idea in line with practical conditions is proposed to match the change of disassembly tools. If the disassembly tool is not required to change, the additional penalty cost is 0; if a different tool is adopted, a penalty cost of 1 is given. Eq. (3) illustrates the penalty cost caused by the change in the disassembly tools. T'_i, $i = \{1, \ldots, N\}$ indicates the tool used to disassemble task i.

$$t_{ij} = \begin{cases} 0, \text{if } T_i^{'} = T_j^{'}, \\ 1, \text{if } T_i^{'} = T_j^{'}, \end{cases} \quad \forall i, j = 0, \ldots, N. \tag{3}$$

2.3 Mathematical Model

The example product from Figure 1 is adopted to help illustrate the mathematical model of DPOA. The disassembly information is shown in Table 3.

The total penalty costs for a disassembly sequence are related to changing the disassembly directions and tools. An example disassembly sequence {2, 3, 9, 8, 7, 1, 10, 5, 6, 4} is given to help illustrate the impact of disassembly directions and tools on penalty costs. For the first two tasks in the sequence, i.e., tasks 2 and 3, the disassembly direction of tasks 2 and 3 are the same; both are $+X$, therefore $d_{23} = 0$, $d_{12} = 0$. Further, we can get the general expression of C_d, as shown in Eq. (4):

Table 1 The main notations defined in our model

Notations	Descriptions
N	The total number of disassembly tasks for a product
i, j	Task indices, $i, j = 1, 2, \ldots, N$
F_seq	Set of feasible disassembly sequences
k	Position indices in the disassembly sequence (The positions in the disassembly sequence correspond to the tasks to be processed one by one)
d_{ij}	Penalty cost of direction change for performing tasks i and j
$d_{k,k+1}$	Penalty cost of direction change for performing two tasks at the kth and (k+1)th position
C_d	Penalty costs of direction change for the disassembly sequence
t_{ij}	Penalty cost of tool change for performing tasks i and j
$t_{k,k+1}$	Penalty cost of tool change for performing two tasks at the kth and (k+1)th position
C_t	Penalty costs of tool change for the disassembly sequence
K_i	Position of task i in the sequence
TC	The total penalty costs of performing the disassembly sequence
Decision variables:	
seq	Index for disassembly sequence
x_k	The task at the kth position

Table 2 The information on the penalty cost of disassembly direction changing

Parts *i/j*	$+X$	$-X$	$+Y$	$-Y$	$+Z$	$-Z$
$+X$	0	2	1	1	1	1
$-X$	2	0	1	1	1	1
$+Y$	1	1	0	2	1	1
$-Y$	1	1	2	0	1	1
$+Z$	1	1	1	1	0	2
$-Z$	1	1	1	1	2	0

Table 3 The disassembly information of the example product from Figure 1

Parts/tasks	1	2	3	4	5	6	7	8	9	10
Direction	$-Z$	$+X$	$+X$	$+Z$	$-Z$	$-Z$	$-Y$	$-X$	$-Y$	$+Y$
Tool	T_1	T_1	T_1	T_2	T_1	T_1	T_2	T_2	T_2	T_2

$$C_d = \sum_{k=1}^{N-1} d_{k,k+1}. \tag{4}$$

Similarly, for tasks 7 and 1 above positions 5 and 6 in the example sequence, the disassembly tool of task 7 is T_2; the disassembly tool of task 1 is T_1; the penalty cost of 1 is increased for the tool changed from T_2 to T_1, that is $t_{71} = 1$, $t_{56} = 1$. Therefore, the general expression of C_t can be represented by Eq. (5).

$$C_t = \sum_{k=1}^{N-1} t_{k,k+1}. \tag{5}$$

The total penalty costs are the summation of the penalty costs produced by changing the disassembly directions and tools, represented in the followings:

$$TC = C_d + C_t = \sum_{k=1}^{N-1} (d_{k,k+1} + t_{k,k+1}), \tag{6}$$

$$\text{Min } TC, \tag{7}$$

$$\text{s.t. } K_i < K_j, \forall p_{ij} = 1, \tag{8}$$

$$seq = \{x_1, x_2, ..., x_N\}, \tag{9}$$

$$seq \in F\_seq, \tag{10}$$

$$K_i, x_i \in \{1, ..., N\}, \forall i = 1, ..., N. \tag{11}$$

Eq. (7) indicates the objective function of our model, the purpose of which is to minimize the total penalty costs. Eqs. (8)–(11) denote the constraints of this model. Eq. (8) solves the task precedence relationships, indicating that tasks must be disassembled by priority. Eq. (9) shows the relationship between the two decision variables, which represents one disassembly sequence containing N positions. Eq. (10) indicates that the disassembly sequence is feasible. Eq. (11) defines that the value of K_i and x_i can take from 1 to N.

3 Solution Approach

As a swarm intelligence algorithm, the ABC algorithm was developed by Karaboga in 2005 [34]. The ABC includes four execution phases: population initialization phase, employed bee phase, onlooker bee phase, and scout bee phase [35, 36]. The random search method employed in the basic ABC is easy to fall into the local optimum when solving medium/large-scale instances. Therefore, a neighborhood modularization-based artificial bee colony algorithm (NM-ABC) is developed. The proposed algorithm combines a neighborhood structure based on the modular concept and the mechanism of ABCs. The design of NM-ABC considers the problem characteristics of DPOA to guarantee the feasibility of disassembly sequences. Furthermore, for a clear understanding of NM-ABC, the example product from Figure 1 is utilized to help understand the structures of the algorithm. Figure 2 demonstrates the flow chart of NM-ABC.

3.1 Food Source

The food sources where bees gather nectar represent the set of feasible solutions to the disassembly planning problem. Based on the characteristics of DPOA in the work, an integer vector encoding method is adopted, each element of which denotes a task for one product. A random integer vector indicates a food source from 1 to N. For example, a food source can be $seq_1 = \{2, 3, 10, 9, 8, 7, 5, 1, 6, 4\}$, which represents that the first one performed is task 2, and task 3 is processed at second, and so on.

Besides, the execution of tasks needs to meet certain priority relationships. An available task disassembly set U (the highest priority task) is developed in our work to facilitate the description of the solution encoding. After

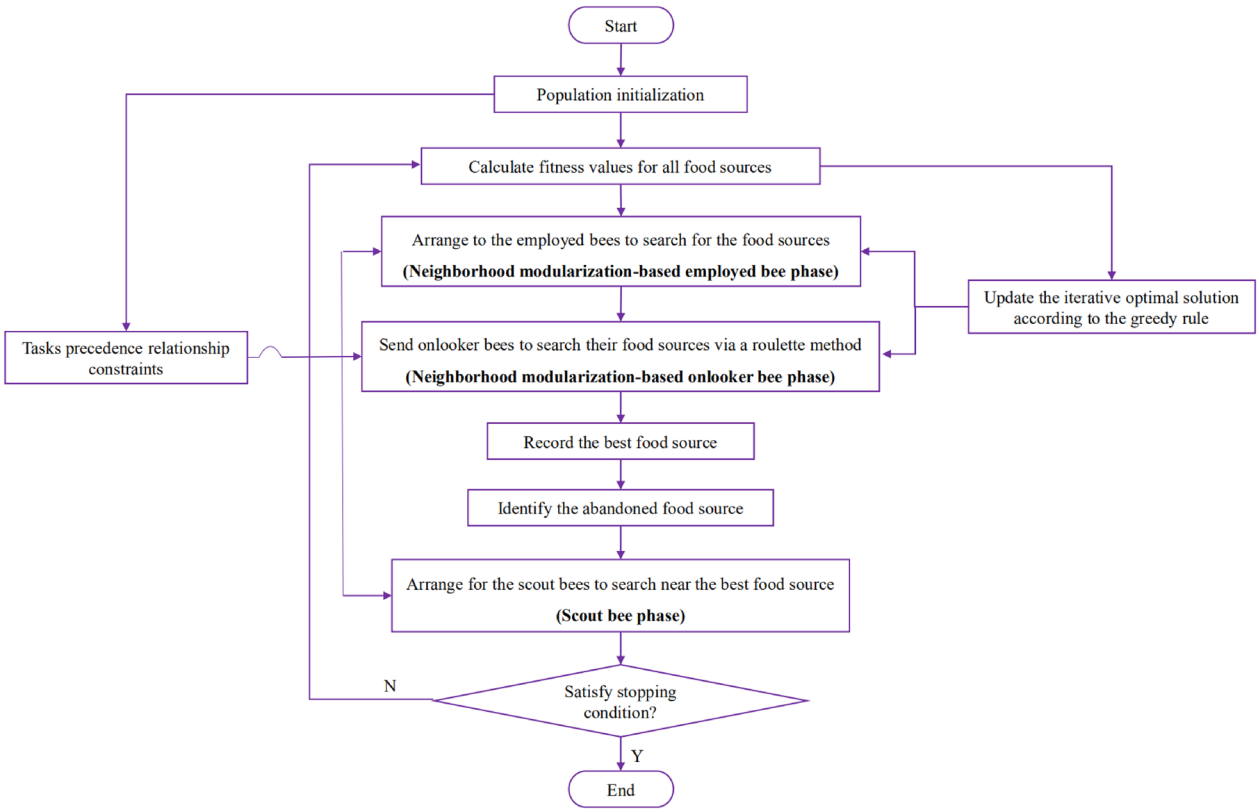

Figure 2 The procedure of NM-ABC

that, a feasible disassembly sequence *seq* is produced (see the steps below).

- **Step 1:** Start.
- **Step 2:** Determine the current set U of executable tasks according to the precedence relationships.
- **Step 3:** If U is null, a feasible disassembly sequence *seq* is generated, and skip to step 6; otherwise, go to the next step.
- **Step 4:** Select a task randomly from U and then put it into the current position of *seq*.
- **Step 5:** Update U, go back to step 3.
- **Step 6:** Stop the procedure.

3.2 Population Initialization Phase

In the basic ABC, three parameters should be considered, including the number of food sources (*NFS*), the number of iterations (*NI*), and the maximum number of trials (*Limit*) that is used to determine whether a food source is to be abandoned. There are only two bees with the same number in the colony in the initialization stage, which are the employed bees and onlooker bees respectively. Also, the number of employed bees is equal to the number of food sources since one employed bee is arranged to explore one food source. Furthermore, the initial populations are randomly produced according to the encoding rules, and the fitness value of each initial food source needs to be calculated. The fitness function in NM-ABC is the same as the objective function in our model, i.e., Eq. (7).

3.3 Neighborhood Modularization-based Employed Bee Phase

In the employed bee phase, a MBX neighborhood structure is designed to improve the exploitation ability of employed bees. Besides, the precedence relationship of the newly generated food source can be met by utilizing this neighborhood structure. Finally, the best food sources are preserved through the rules of greedy selection.

A schematic diagram in Figure 3 is used to illustrate the MBX in our algorithm. Suppose that the food source $seq_1 = \{2, 3, 10, 9, 8, 7, 5, 1, 6, 4\}$ is in the employed bee search phase. Another food source is required to perform MBX operation. Therefore, we randomly select a food source $seq_2 = \{3, 2, 1, 8, 4, 10, 7, 6, 9, 5\}$ from the remaining food sources. The steps below show how MBX extracts the module with optimal fitness (abbreviated as the optimal module) and produces a new sequence.

- **Step 1:** Randomly select a number from 2 to $N-2$ as the size of the module (*mod_size*). Therefore, a sequence can be divided into (N - *mod_size* + 1) modules.

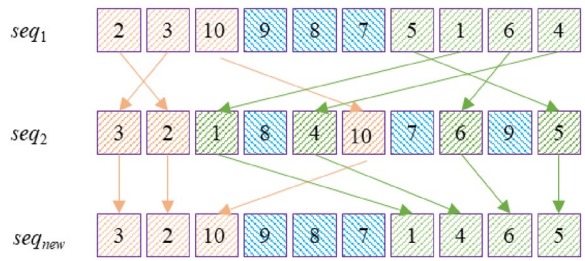

Figure 3 A schematic diagram of module-based crossover neighborhood structure

- **Step 2:** Calculate the fitness value of each module. Assume the selected number is 3, i.e., *mod_size* = 3. Hence, seq_1 has eight modules. The first module of seq_1 is {2, 3, 10}, and {3, 10, 9} is the second module, and so on.
- **Step 3:** Extract a module with the optimal fitness value. If there is more than one module with optimal fitness values, one module is randomly reserved from them. The lowest fitness value is 2. Hence, the module {9, 8, 7} is extracted.
- **Step 4:** The two subsequences before and after the optimal module need to perform crossover operations, i.e., {2, 3, 10} and {5, 1, 6, 4}. seq_2 maps these two subsequences, and two new subsequences {3, 2, 10} and {1, 4, 6, 5} are obtained, respectively.
- **Step 5:** A new sequence is constructed by MBX, $seq_{new} = \{3, 2, 10, 9, 8, 7, 1, 4, 6, 5\}$.

The optimal module satisfies the priority relationships, and seq_2 maps both the newly produced subsequences before and after the optimal module. Since these two sub-sequences also meet the priority relationships in seq_2, the freshly generated sequence by MBX must satisfy the priority relationship constraint.

3.4 Neighborhood Modularization-based Onlooker Bee Phase

In the onlooker bee search phase, each onlooker bee will select an employed bee to follow based on a specific probability. In our work, roulette is adopted to determine the probability. The food source with a better fitness value is more likely to be selected.

A module-based insert (MBI) neighborhood structure is designed in this stage to obtain food sources with a better fitness value. The greed rule is adopted to preserve the optimal food source.

A schematic diagram in Figure 4(a) is adopted to illustrate the MBI. Assuming there is a sequence *seq* = {2,

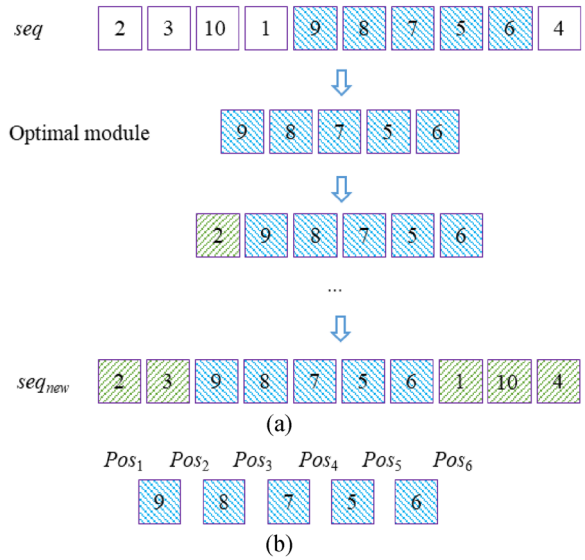

Figure 4 Module-based insert neighborhood structure: (**a**) a schematic diagram of MBI, (**b**) the set of positions in the optimal module

3, 10, 1, 9, 8, 7, 5, 6, 4} in the onlooker bee stage to perform MBI operation. After MBI operation, a new feasible sequence seq_{new} = {2, 3, 9, 8, 7, 5, 6, 1, 10, 4} is generated (see the steps below).

- **Step 1:** Randomly select a number from 2 to $N-2$ as the size of the module (*mod_size*). Therefore, a sequence can be divided into (N - *mod_size* + 1) modules.
- **Step 2:** Calculate the fitness value of each module. Supposing *mod_size* = 5, hence, seq_1 has 6 modules.
- **Step 3:** Extract a module with the optimal fitness value. If there is more than one module with optimal fitness values, one module is randomly reserved from them. The lowest fitness value is 4. Hence, the optimal module {9, 8, 7, 5, 6} is extracted.
- **Step 4:** For the remaining sequence segmentation {2, 3, 10, 1, 4}, they are inserted into the optimal module in turn under the precedence relationship constraint.
- **Step 5:** Determine the set of positions in the optimal module to insert task 2, as shown in Figure 4(b), and we find only that position 1 can be inserted. Therefore, task 2 is inserted in position 1. If more than one position can be inserted for a task, choose the position to insert that maximizes the fitness value of the food source.
- **Step 6:** According to step 5, insert all the tasks in the remaining sequence segmentation into the optimal module. Therefore, a new feasible sequence seq_{new} = {2, 3, 9, 8, 7, 5, 6, 1, 10, 4} is produced.

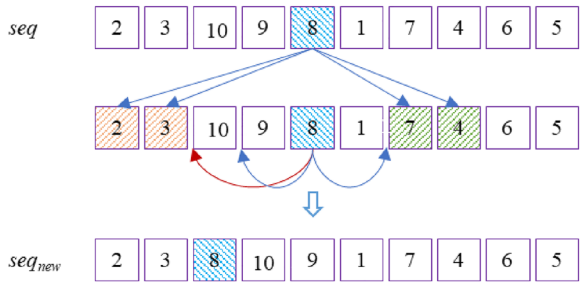

Figure 5 A schematic diagram of SPI

All the above steps meet the task precedence relationship. Hence, the newly produced sequence by MBI must be feasible.

3.5 Scout Bee Phase

To avoid the algorithm falling into a local optimum, a food source will be discarded when it has not been updated for a *Limit* consecutive generation. Therefore, a parameter *Trial* is adopted to record the number of food sources has not been updated. If a food source is not updated, *Trial* = *Trial* + 1; otherwise, *Trial* = 0. If *Trial* ≥ *Limit*, this food source will be abandoned, and a new food source will be generated by a single point insertion (SPI) operation [37] based on the global optimal solution learning strategy. It is more likely to obtain a better food source by searching near the current optimal food source [38]. Figure 5 denotes the process of SPI operation based on an example product from Figure 1.

As shown in Figure 5, it is assumed that the current optimal sequence to perform SPI in the scout bee phase is *seq* = {2, 3, 10, 9, 8, 1, 7, 4, 6, 5}. The steps below show how to generate a new sequence by performing SPI.

- **Step 1:** Randomly select a task as the insertion object, supposing task 8 is selected.
- **Step 2:** Find the immediately preceding {2, 3} and following tasks {7, 4} of task 8 that satisfy the precedence relationship constraints. Then, the insertable position of task 8 is further determined.
- **Step 3:** Randomly select a position and insert task 8. If task 8 is inserted into the position between tasks 3 and 10. Therefore, a new sequence seq_{new} = {2, 3, 8, 10, 9, 1, 7, 4, 6, 5} is produced.

The feasibility of the newly produced sequence also can be guaranteed since each step of SPI satisfies the precedence relationships.

3.6 Pseudocode of NM-ABC

The pseudocode of NM-ABC is presented in Figure 6.

Algorithm 1: NM-ABC algorithm

Input: *NFS*, *NI*, *Limit*, *iter* = 1, and DP data.

Output: an optimal or near-optimal solution (seq_0).

(1) Start

(Population initialization phase)

(2) Initialize food sources seq_t ($t = 1, 2, \ldots, NFS$);

(3) Calculate the fitness value of each food source and record the optimal one (seq_0);

(4) While ($iter \le NFS$) Do

(Neighborhood modularization-based employed bee phase)

(5) For each employed bee

(6) Determine the optimal module for the current food source seq_t;

(7) MBX neighborhood structure is utilized to obtain seq_t' with a food source random from the remaining food sources;

(8) Calculate the fitness value $f(seq_t')$;

(9) If $f(seq_t') < f(seq_t)$, $seq_t = seq_t'$, $trial(seq_t) = 0$; else $trial(seq_t) = trial(seq_t)+1$;

(10) End for.

(Neighborhood modularization-based onlooker bee phase)

(11) For each onlooker bee

(12) Choose a food source seq_t via a roulette method to exploit, and determine the optimal module for seq_t;

(13) MBI neighborhood structure is used to acquire seq_t' with a food source random from the remaining food sources;

(14) Calculate the fitness value $f(seq_t')$;

(15) If $f(seq_t') < f(seq_t)$, $seq_t = seq_t'$, $trial(seq_t) = 0$; else $trial(seq_t) = trial(seq_t)+1$;

(16) End for.

(17) Update the current optimal food source seq_0;

(Scout bee phase)

(18) For each scout bee

(19) If $trial(seq_t) \ge Limit$

(20) SPI method is used to generate a new food source seq_t' on seq_0;

(21) End if.

(22) End for.

(23) $trial(seq_t) = 0$;

(24) Update fitness values of all food sources.

(25) $iter = iter + 1$;

(26) End while.

Figure 6 The pseudocode of the NM-ABC algorithm

4 Computational Examples and Analysis

In this section, two instances with different complexity are used to test the performance of the NM-ABC algorithm. The first case to be analyzed is the product from Figure 1, and a refrigerator is taken as the second case, which is more complex than the first one. Large-scale cases contain more operation attributes, the solving difficulty of which increases exponentially. The NM-ABC program is developed in MATLAB 9.8, and the experiments are run on a computer with AMD Ryzen 7 4800U CPU @ 1.80 GHz and 16 GB of RAM.

4.1 Case Study 1

The product case from Figure 1 contains a total of 10 tasks, and the disassembly information of case study 1 is presented in Table 3. Further, some parameters used in NM-ABC are set as follows: $NFS = 10$, $NI = 50$, $Limit = 5$. The experiments were executed five times to converge per experiment fully through the proposed algorithm, and Table 4 shows the computational results. The first and second columns denoted the number of experiments and gained the optimal or near-optimal disassembly sequence for each run. The third and fourth columns are the penalty costs of the corresponding disassembly sequence, i.e., the objective value and the execution time, respectively.

Table 4 Obtained optimal or near-optimal solutions for case study 1

Runs	The optimal or near-optimal solutions	Penalty costs	Running time (s)
1	{2, 3, 10, 8, 4, 7, 9, 1, 5, 6}	7	1.756
2	{2, 3, 10, 8, 4, 7, 9, 6, 5, 1}	7	1.679
3	{2, 3, 10, 8, 4, 7, 9, 5, 1, 6}	7	1.794
4	{2, 3, 10, 8, 4, 7, 9, 5, 6, 1}	7	2.146
5	{3, 2, 10, 8, 4, 7, 9, 6, 5, 1}	7	2.182

From Table 4, we find the same optimal objective value for each run (7 for 5 times), which shows the stability and effectiveness of NM-ABC. Also, the proposed approach is efficient since the execution time for each run is less

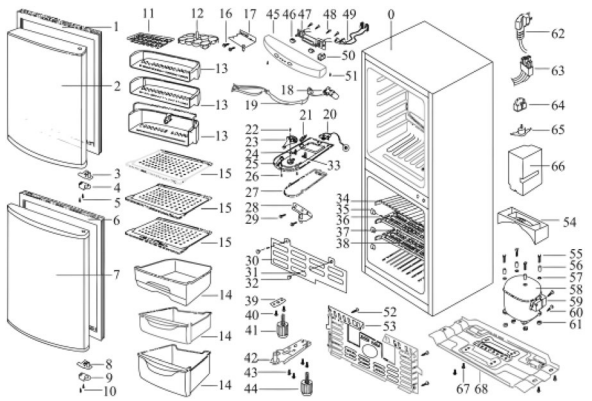

Figure 7 The 3D diagram for the refrigerator

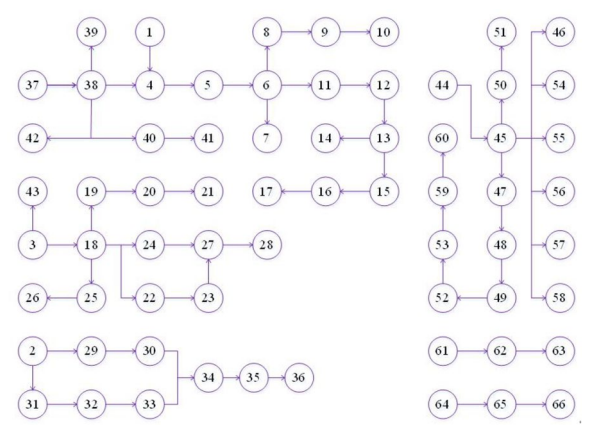

Figure 8 The precedence relationship of the refrigerator

than 2.2 s. Therefore, it can be concluded that NM-ABC is capable to solve the DPOA for case study 1.

4.2 Case Study 2

The experiments in case study 1 show the feasibility and efficiency of the NM-ABC algorithm. A larger-scale instance is utilized in this subsection to further validate the performance and the availability of the NM-ABC. Case study 2 contains more operation attributes, the solution complexity of which is higher than the first case. The refrigerator with 68 parts from Wang et al. [39] is taken as the second instance, and the parts can be divided into 66 tasks. Figures 7 and 8 indicate the 3D diagram for the refrigerator, and the task precedence diagram of the refrigerator product, respectively. Table 5 represents the disassembly data for each task of the instance product.

The parameters used in NM-ABC are set as follows: $NFS = 20$, $NI = 100$, $Limit = 20$. Similarly, case study 2 is also executed five times as in case study 1 to converge per trial fully through the proposed algorithm. Table 6 shows the computational results.

Since case study 2 contains too many tasks, the obtained optimal or near-optimal disassembly solution is not shown in Table 6. For the second instance, NM-ABC can generate a satisfactory disassembly sequence in a reasonable time from Table 6. In the five trials, the optimal or near-optimal value was 20 for four times, and the value for the other time was 21, which shows the effectiveness and stability of the algorithm. When the penalty costs is 20, one of the optimal disassembly sequence is {37, 38, 2, 31, 32, 29, 3, 18, 1, 22, 4, 19, 33, 5, 30, 6, 34, 11, 35, 40, 8, 36, 23, 24, 25, 39, 42, 41, 20, 21, 7, 9, 10, 43, 12, 13, 26, 15, 27, 28, 14, 16, 17, 64, 61, 65, 62, 44, 45, 46, 54, 56, 58, 55, 57, 47, 48, 49, 52, 53, 59, 50, 60, 51, 66, 63}. Moreover, the computational time of the instance is about 50 s, which is also acceptable. It takes more computational time than the first case since the second instance is more large-scale and complex. NM-ABC also performs well in solving case study 2 like the previous case.

4.3 Algorithm Performance Comparison

To further test the performance of the developed NM-ABC, taking the above two instances as examples, the basic artificial bee colony (ABC) algorithm [40] and the genetic algorithm (GA) [41] are used for comparison in this subsection. Also, in the other two algorithms, we will adopt the same encoding principle, population initialization, and objective function in each simulation with NM-ABC. To guarantee the rationality of the comparisons and gain sufficient data, for the different values of *PopSize* (*PopSize* = 2 *NFC*) and the computational time (*CPU* time), each of them is implemented ten times since the computational processes of the methods have probabilistic and randomized characteristics.

4.3.1 Comparison Results for Case Study 1

Table 7 shows the computational results and comparison for case study 1. The first four columns denote the indices of experiments, the computational time (*CPU* time), the population size (*PopSize*), and the approaches for each run, respectively. The last three columns are values of the best (f_{best}), worst (f_{worst}), and average (f_{mean}) fitness values solved by the three algorithms for different parameters. NM-ABC can gain a minimal penalty cost in each run, as shown in Table 4. The other two algorithms can also obtain the optimal fitness value but they are not stable enough. For f_{worst} and f_{mean}, the value solved by NM-ABC is better than the other two methods.

From Table 7, the stability of NM-ABC is superior to the other methods under the same computational time and parameter settings. Hence, the comparison results

Table 5 The tasks for disassembly data of the refrigerator

Tasks	Removed parts	Direction	Tool	Tasks	Removed parts	Direction	Tool
1	[1, #12, #13, #12, #13] from #2	$-X$	T_1	34	Cap-strew #32	$-X$	T_1
2	#14 from box #0	$-X$	T_1	35	Screws #31	$-X$	T_2
3	#15 from box #0	$-X$	T_1	36	Cover-pipe #30	$-X$	T_1
4	Screws #16	$-X$	T_2	37	Screws #51	$-X$	T_2
5	Hinge-up #17	$-X$	T_1	38	Top-table #45	$-X$	T_1
6	Door body #2	$-X$	T_1	39	Buttons #46	$-X$	T_1
7	Gasket door #1	$-X$	T_1	40	Screws #48	$-X$	T_2
8	Screws #5	$-X$	T_2	41	PCB-panel #47	$-X$	T_1
9	Stopper door #4	$-X$	T_1	42	PCB wire #49	$-X$	T_1
10	Stopper door #3	$-X$	T_1	43	Switch-door #50	$-X$	T_1
11	Screws #29	$-X$	T_2	44	Screws #52	$+X$	T_2
12	Hinge-mid #28	$-X$	T_1	45	Cover-comp #53	$+X$	T_1
13	Door body #7	$-X$	T_1	46	Water pans #54	$+X$	T_1
14	Gasket door #6	$-X$	T_1	47	Process pipeline from #58	$+X$	T_4
15	Screws #10	$-X$	T_2	48	Fix-earth #55	$+X$	T_5
16	Stopper door #9	$-X$	T_1	49	Sleeve #56, Shim #57	$+X$	T_1
17	Stopper door #8	$-X$	T_1	50	Fix-earth screw #60	$+X$	T_2
18	Lamp cover #27	$-X$	T_1	51	Cover-relay #59	$+X$	T_1
19	Screw #22	$-X$	T_2	52	Compressor #58	$+X$	T_1
20	Temperature control knob #23	$-X$	T_1	53	Gasket #61	$+X$	T_1
21	Gear #24	$-X$	T_1	54	Case-junction #66	$+X$	T_3
22	Screws #33	$-X$	T_2	55	Overload protector #65	$+X$	T_3
23	Thermostat #20	$-X$	T_1	56	PTC-relay #64	$+X$	T_3
24	Seasonal switch #21	$-X$	T_1	57	Wire-relay #63	$+X$	T_3
25	Wire #19	$-X$	T_1	58	Power-cord #62	$+X$	T_3
26	Lamp #18	$-X$	T_3	59	Screws #67	$+X$	T_2
27	Screws #26	$-X$	T_2	60	Compressor chassis #68	$+X$	T_1
28	Temperature control box #25	$-X$	T_1	61	Leg-front #44	$-Z$	T_1
29	Screws #36	$-X$	T_1	62	Screws #43	$-Z$	T_2
30	Roller-Tray #35	$-X$	T_1	63	Hinge-low #42	$-Z$	T_1
31	Stopper-EVAP #34	$-X$	T_1	64	Leg-front #41	$-Z$	T_1
32	Tray-EVAP #37	$-X$	T_1	65	Screws #40	$-Z$	T_2
33	Evaporator #38	$-X$	T_4	66	Reinforcement-leg #39	$-Z$	T_1

in Table 7 indicate the superiority of the proposed algorithm for the disassembly planning problem for case study 1.

Table 6 Obtained optimal or near-optimal solutions for case study 2

Runs	Penalty costs	Running time (s)
1	21	51.585
2	20	52.270
3	20	55.976
4	20	49.602
5	20	50.831

4.3.2 Comparison Results for Case Study 2

Table 8 shows the comparison results for case study 2. Since the product scale and complexity become larger, *CPU* time and *PopSize* increased to 70 s and 60. From Table 8, the minimal penalty cost value solved by NM-ABC in the three situations is equal to 20. The optimal fitness values solved by basic ABC and GA are 21 and 28, respectively. For $f_{best,}$ NM-ABC can obtain the best solution in each run, but the other two algorithms do not once. For f_{worst}, the solution value solved by NM-ABC also precedes the other two methods, and the basic ABC is inferior. The GA has the worst solution effect and seems not suitable for solving DPOA.

Table 7 Result comparison of the three algorithms for case study 1

No.	*CPU* time (s)	*PopSize*	Method	f_{best}	f_{worst}	f_{mean}
1	0.5	2	ABC	7	11	8.2
			GA	7	10	8.1
			NM-ABC	7	7	7
2	1	2	ABC	7	10	8.0
			GA	7	8	7.5
			NM-ABC	7	7	7
3	0.5	4	ABC	7	8	7.2
			GA	7	8	7.5
			NM-ABC	7	7	7

Table 8 Result comparison of the three algorithms for case study 2

No.	*CPU* time (s)	*PopSize*	Method	f_{best}	f_{worst}	f_{mean}
1	50	40	ABC	21	24	22.8
			GA	33	46	37.9
			NM-ABC	20	21	20.3
2	70	40	ABC	21	24	22.2
			GA	32	44	37.6
			NM-ABC	20	21	20.1
3	70	60	ABC	22	24	22.6
			GA	28	43	36.7
			NM-ABC	20	20	20

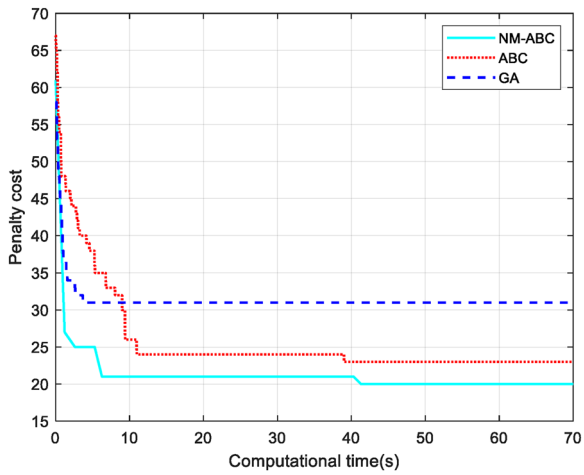

Figure 9 The convergence curve graph

Moreover, the values of f_{mean} solved by basic ABC and GA are inferior to NM-ABC.

Due to the larger scale and complexity of case study 2, the experimental results can better represent the solution performance of these three algorithms. Further, when *CPU* time = 70 s, *PopSize* = 60, the convergence curve graph of the three algorithms by executing an experiment on case 2 is obtained, as shown in Figure 9, representing the penalty cost value tendency. Through the curve graph, it is evident that NM-ABC performs better in convergence and effectiveness.

Based on the above description, in both cases, the three methods can be sorted from their performance levels as NM-ABC > ABC > GA. Moreover, as the value of *CPU* time and *PopSize* increase, the proposed algorithm can obtain higher-quality solutions. One easy conclusion is that NM-ABC can efficiently solve our disassembly planning problem.

5 Conclusions

This work presents a mathematical model of complete disassembly planning with operation attributes. It aims to determine a disassembly task sequence with minimal penalty cost caused by the change of disassembly directions and tools. Moreover, a neighborhood modularization-based artificial bee colony algorithm (NM-ABC) is developed to solve the proposed disassembly planning problem. The neighborhood modularization design helps

guide the search in a better direction by extracting the module with the best fitness value. Two different module concept-based neighborhood structures and a global optimal solution-based learning strategy are employed in NM-ABC. Finally, taking two cases of different scales and complexity as experiment cases and compared with two well-known algorithms, the computational results show that NM-ABC can efficiently solve our disassembly planning problem.

Although the performance of the proposed approach has been proven to be superior, there are still some limitations. First, more operation attributes, such as disassembly time and classes of operations, need to be considered in the disassembly process to support the disassembly practice. Second, a disassembly uncertainty needs to be considered in the disassembly problem of EOL products. Finally, developing some more advanced and efficient neighborhood structures or metaheuristic algorithms is an interesting research direction in the future disassembly field.

Acknowledgments
Not applicable.

Author contributions
HG was in charge of the whole trial; LZ wrote the manuscript; YR, LM, and JL put forward some suggestions on the thinking of the paper; ZZ gave some advice on the arrangement of the structure diagram and table in the paper. All authors read and approved the final manuscript.

Authors' information
Hongfei Guo, born in 1980, is currently an associate professor at *School of Intelligent Systems Science and Engineering, Institute of Physical Internet, Jinan University (Zhuhai Campus), Zhuhai, 519070, China*. He received his Ph.D. degree from *Mongolia Industrial University, China*, in 2021. His current research interests include the industrial internet of things, digital twins, and intelligent manufacturing.
Linsheng Zhang, born in 1997, is currently a master candidate at *School of Management, Jinan University, (Guangzhou Campus), China*. He received his bachelor's degree from *Yangtze University, China*, in 2019. His main research interests include intelligent manufacturing and algorithm optimization.
Yaping Ren, born in 1995, is currently an associate professor at *School of Intelligent Systems Science and Engineering, Institute of Physical Internet, Jinan University (Zhuhai Campus), China*. He received his Ph.D. degree from *Huazhong University of Science and Technology, China*, in 2019. His research mainly focuses on industrial engineering, disassembly planning, transportation planning, decision making, and optimization methods.
Leilei Meng, born in 1991, is currently a lecturer at *School of Computer Science, Liaocheng University, China*. He received his Ph.D. degree from *Huazhong University of Science and Technology, China*, in 2019. His research mainly focuses on modeling, optimization of scheduling problems, tool wear prediction, and sustainable manufacturing.
Zhongwei Zhou, born in 1996, is currently a master candidate at *School of Electronic Information and Automation, Tianjin University of Science & Technology (Hexi Campus), China*. He received his bachelor's degree from *Anhui Jianzhu University, China*, in 2018. His research mainly focuses on optimized decision.
Jianqing Li, is currently a professor at *Faculty of Information Technology, Macau University of Science and Technology, Macau 999078, China*. He received his Ph.D. degree from *Beijing University of Posts and Telecommunications, China*, in 1999. His research interests include wireless networks, fiber sensors, and the Internet of Things.

Funding
Supported by National Natural Science Foundation of China (Grant Nos. 52205526, 52205529), Basic and Applied Basic Research Project of the Guangzhou Basic Research Program of China (Grant No. 202201010284), National Foreign Expert Project of the Ministry of Science and Technology of China (Grant No. G2021199026L), National Key Research and Development Program of China (Grant Nos. 2021YFB3301701, 2021YFB3301702), Guangdong Provincial Graduate Education Innovation Program of China (Grant No. 82620516), Guangzhou Municipal Innovation Leading Team Project of China (Grant No. 201909010006), Guangdong Provincial "Quality Engineering" Construction Project of China (Grant No. 210308), Guangdong Provincial Basic and Applied Basic Research Foundation of China (Grant No. 2019A1515110399), Fundamental Research Funds for the Central Universities of China (Grant No. 21620360).

Competing interests
The authors declare no competing financial interests.

Author Details
[1]GBA and B&R International Joint Research Center for Smart Logistics, School of Intelligent Systems Science and Engineering, Institute of Physical Internet, Jinan University (Zhuhai Campus), Zhuhai 519070, China. [2]School of Computer Science, Liaocheng University, Liaocheng 252059, China. [3]Faculty of Information Technology, Macau University of Science and Technology, Macau 999078, China.

Received: 24 December 2021 Revised: 31 May 2022 Accepted: 21 October 2022
Published online: 03 December 2022

References
[1] H F Guo, R Zhang, Y X Zhu, et al. Sustainable quality control mechanism of heavy truck production process for plant-wide production process. *International Journal of Production Research*, 2020, 58(24): 7548-7564.
[2] G D Tian, G Yuan, A Aleksandrov, et al. Recycling of spent Lithium-ion batteries: A comprehensive review for identification of main challenges and future research trends. *Sustainable Energy Technologies and Assessments*, 2022, 53: 102447.
[3] L M Qiu, X J Liu, S Y Zhang, et al. Disassemblability modeling technology of configurable product based on disassembly constraint relation weighted design structure matrix (DSM). *Chinese Journal of Mechanical Engineering*, 2014, 27(3): 511-519.
[4] Y P Ren, H Y Jin, F Zhao, et al. A multiobjective disassembly planning for value recovery and energy conservation from end-of-life products. *IEEE Transactions on Automation Science and Engineering*, 2021, 18(2):1791-803.
[5] Y P Ren, C Y Zhang, F Zhao, et al. An MCDM-based multiobjective general variable neighborhood search approach for disassembly line balancing problem. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2020, 50(10): 3770-3783.
[6] S Ghandi, E Masehian. Review and taxonomies of assembly and disassembly path planning problems and approaches. *Computer-Aided Design*, 2015, 67: 58-86.
[7] S Smith, P Y Hung. A novel selective parallel disassembly planning method for green design. *Journal of Engineering Design*, 2015, 26(10-12): 283-301.
[8] R Giri, M Kanthababu. Generating complete disassembly sequences by utilising two-dimensional views. *International Journal of Production Research*, 2015, 53(17): 5118-5138.
[9] M Kheder, M Trigui, N Aifaoui. Disassembly sequence planning based on a genetic algorithm. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 2015, 229(12): 2281-2290.
[10] H P Hsu. A fuzzy knowledge-based disassembly process planning system based on fuzzy attributed and timed predicate/transition net. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2016, 47(8): 1800-1813.
[11] X F Zhang, G Yu, Z Y Hu, et al. Parallel disassembly sequence planning for complex products based on fuzzy-rough sets. *The International Journal of Advanced Manufacturing Technology*, 2014, 72(1): 231-239.

[12] A ElSayed, E Kongar, S M Gupta. A genetic algorithm approach to end-of-life disassembly sequencing for robotic disassembly. *Proceedings of the 2010 Northeast Decision Sciences Institute Conference*, Alexandria, USA, March 26–28, 2010: 402–408.
[13] G D Tian, M C Zhou, J W Chu, et al. Probability evaluation models of product disassembly cost subject to random removal time and different removal labor cost. *IEEE Transactions on Automation Science and Engineering*, 2012, 9(2): 288-295.
[14] E Kongar, S M Gupta. Disassembly sequencing using genetic algorithm. *The International Journal of Advanced Manufacturing Technology*, 2006, 30(5): 497-506.
[15] H E Tseng, C C Chang, S C Lee, et al. Hybrid bidirectional ant colony optimization (hybrid BACO): an algorithm for disassembly sequence planning. *Engineering Applications of Artificial Intelligence*, 2019, 83: 45-56.
[16] H E Tseng, C C Chang, S C Lee, et al. A block-based genetic algorithm for disassembly sequence planning. *Expert Systems with Applications*, 2018, 96: 492-505.
[17] K Xia, L Gao, W D Li, et al. Disassembly sequence planning using a simplified teaching-learning-based optimization algorithm. *Sustainable Manufacturing and Remanufacturing Management. Springer, Cham*, 2019: 319-343.
[18] Y P Ren, C Y Zhang, F Zhao, et al. An asynchronous parallel disassembly planning based on genetic algorithm. *European Journal of Operational Research*, 2018, 269: 647-660.
[19] H Wang, D Xiang, G H Duan. A genetic algorithm for product disassembly sequence planning. *Neurocomputing*, 2008, 71(13-15): 2720-2726.
[20] J L Rickli, J A Camelio. Multi-objective partial disassembly optimization based on sequence feasibility. *Journal of Manufacturing Systems*, 2013, 32(1): 281-293.
[21] W C Yeh. Simplified swarm optimization in disassembly sequencing problems with learning effects. *Computers & Operations Research*, 2012, 39(9): 2168-2177.
[22] L Zhong, Y C Sun, O E Gabriel, et al. Disassembly sequence planning for maintenance based on metaheuristic method. *Aircraft Engineering and Aerospace Technology*, 2011, 83(3): 138-145.
[23] M Alshibli, Sayed A El, E Kongar, et al. Disassembly sequencing using tabu search. *Journal of Intelligent & Robotic Systems*, 2016, 82(1): 69-79.
[24] J F Wang, J H Liu, S Q Li, et al. Intelligent selective disassembly using the ant colony algorithm. *Ai Edam*, 2003, 17(4): 325-333.
[25] S M McGovern, S M Gupta. Ant colony optimization for disassembly sequencing with multiple objectives. *The International Journal of Advanced Manufacturing Technology*, 2006, 30(5): 481-496.
[26] C B Kalayci, O Polat, S M Gupta. A hybrid genetic algorithm for sequence-dependent disassembly line balancing problem. *Annals of Operations Research*, 2016, 242(2): 321-354.
[27] C B Kalayci, A Hancilar, A Gungor, et al. Multi-objective fuzzy disassembly line balancing using a hybrid discrete artificial bee colony algorithm. *Journal of Manufacturing Systems*, 2015, 37: 672-682.
[28] C B Kalayci, S M Gupta. Artificial bee colony algorithm for solving sequence-dependent disassembly line balancing problem. *Expert Systems with Applications*, 2013, 40(18): 7231-7241.
[29] Y P Ren, G D Tian, F Zhao, et al. Selective cooperative disassembly planning based on multi-objective discrete artificial bee colony algorithm. *Engineering Applications of Artificial Intelligence*, 2017, 64: 415-431.
[30] G D Tian, Y P Ren, Y X Feng, et al. Modeling and planning for dual-objective selective disassembly using AND/OR graph and discrete artificial bee colony. *IEEE Transactions on Industrial Informatics*, 2018, 15(4): 2456-2468.
[31] Q J Guo, S Fan, R F Xu, et al. Spindle thermal error optimization modeling of a five-axis machine tool. *Chinese Journal of Mechanical Engineering*, 2017, 30(3): 746-753.
[32] H Y Sang, L Gao, Q Pan. Discrete artificial bee colony algorithm for lot-streaming flowshop with total flowtime minimization. *Chinese journal of mechanical engineering*, 2012, 25(5): 990-1000.
[33] L X Zhu, Z Q Zhang, Y Wang, et al. On the end-of-life state oriented multi-objective disassembly line balancing problem. *Journal of Intelligent Manufacturing*, 2020, 31(6): 1403-1428.
[34] D Karaboga, B Basturk. On the performance of artificial bee colony (ABC) algorithm. *Applied Soft Computing*, 2008, 8(1): 687-697.
[35] L Z Cui, G H Li, X Z Wang, et al. A ranking-based adaptive artificial bee colony algorithm for global numerical optimization. *Information Sciences*, 2017, 417: 169-185.
[36] J J Zhou, X F Yao, Y Z Lin, et al. An adaptive multi-population differential artificial bee colony algorithm for many-objective service composition in cloud manufacturing. *Information Sciences*, 2018, 456: 50-82.
[37] K P Wang, X Y Li, L Gao. Modeling and optimization of multi-objective partial disassembly line balancing problem considering hazard and profit. *Journal of Cleaner Production*, 2019, 211: 115-133.
[38] J Liu, S Wang. Balancing disassembly line in product recovery to promote the coordinated development of economy and environment. *Sustainability*, 2017, 9(2): 309.
[39] K P Wang, X Y Li, L Gao, et al. A discrete artificial bee colony algorithm for multiobjective disassembly line balancing of end-of-life products. *IEEE Transactions on Cybernetics*, 2021, 52(8): 7415-7426.
[40] Q Lu, Y P Ren, H Y Jin, et al. A hybrid metaheuristic algorithm for a profit-oriented and energy-efficient disassembly sequencing problem. *Robotics and Computer-Integrated Manufacturing*, 2020, 61: 101828.
[41] Y P Ren, D Y Yu, C Y Zhang, et al. An improved gravitational search algorithm for profit-oriented partial disassembly line balancing problem. *International Journal of Production Research*, 2017, 55(24): 7302-7316.