Multi-Factory Remanufacturing Process Optimization with Discrete Battle Royale Optimizer

Ziyan Zhao and Liangbo Zhou

***Abstract*—In the context of advancing transportation and communication infrastructures, companies are increasingly establishing factories across the globe to support their international expansion strategies. They achieve information sharing through networks, forming a distributed multi-factory working environment. This work introduces and addresses a multi-factory remanufacturing process optimization problem by considering hybrid disassembly line balancing and multi-skilled worker allocation. It has three phases: disassembly factory selection, disassembly task scheduling, and manufacturing factory selection. A linear programming model is established with the objective of optimizing profitability. This work proposes to use a discrete Battle Royale optimizer to solve the problem with a newly proposed encoding structure. The experimental results are benchmarked against CPLEX, confirming the validity and efficiency of the proposed method. The results of its comparisons with genetic algorithm, discrete migratory bird optimizer, fruit fly optimizer, and dingo optimizer further validate its superiority over its peers.**

***Key Words*—Multi-factory remanufacturing process optimization Scheduling, hybrid disassembly line balancing, battle royale optimization algorithm, multi-skilled workers**

I. Introduction

With the continuous development of science and technology and the growing demand for diverse and individualized markets, conventional centralized approaches to disassembly planning and scheduling are often inadequate in adapting to rapidly fluctuating market requirements [1], [2], [3]. In this environment, besides the possible plan failure and increased response cost, a classical single-factory mode may face closure due to some single-point failures [4]. Consequently, to effectively address regional market demands, multinational corporations frequently deploy a distributed network of factories across various geographical locations [5], [6]. With the increase in the number of factories, it has become an urgent problem to effectively schedule them to maximize the total profit.

Manuscript received August 5, 2025; revised August 12 and August 28, 2025; accepted October 7, 2025. This article was recommended for publication by Associate Editor Shujin Qin upon evaluation of the reviewers' comments.

This work was supported in part by NSFC under Grant Nos. 61573089, 62073069 and 51405075, and in part by the Natural Science Foundation of Shandong Prov-ince under Grant ZR2019BF004.

Z. Zhao is with the College of Information Science and Engineering, Northeastern University, Shenyang 110819, China (e-mail: zhaoziyan@mail.neu.edu.cn).

L. Zhou is with the College of Information and Control Engineering, Liaoning Petrochemical University, Fushun 113001, China (e-mail: 1316550824@qq.com).

Corresponding author: Ziyan Zhao

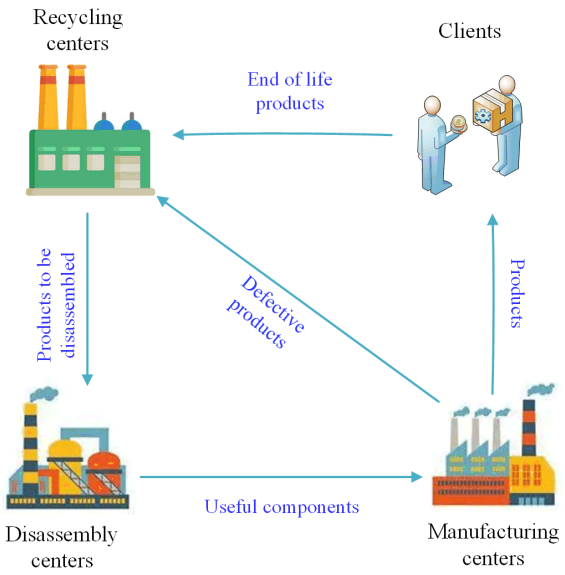


Fig. 1. Multi-factory remanufacturing process optimization.

There has been growing interest in the multi-factory production scheduling problem (MPSP) within distributed manufacturing environments in recent literature [7]. However, there is limited research on a multi-factory remanufacturing process optimization problem (MRPOP). MRPOP refers to the coordination and optimization of multiple factories' operations in the remanufacturing field, aiming to maximize the overall profit of a remanufacturing system by performing resource allocation, production planning, and logistics transportation. Its structure is shown in Fig. 1. Its scheduling research can be broadly divided into two classes: multi-factory homogeneous scheduling and multi-factory heterogeneous scheduling [8]. In practical operations, due to the varying environments in different factories and the differences in production and disassembly capacities, solving a multi-factory heterogeneous scheduling problem is required [9], [10]. Some progress has been made in solving it [11]. To tackle the challenges of MPSP, various metaheuristic approaches have been developed. For instance, Ziaee [12] introduced a fast metaheuristic built on a construction process for rapid generation of high-quality schedules. Similarly, Chang et al. [13] designed a hybrid genetic algorithm complemented by a novel encoding mechanism to overcome invalid job assignments in complex flexible job shop scheduling [14]. In a similar vein, Jia et al. [15] advanced an improved genetic algorithm capable of handling both traditional and distributed scheduling paradigms.

Due to the rapid development of electronic information tech-

nology, the replacement speed of electromechanical products such as televisions, refrigerators, and computers is increasing, leading to a strong need to handle end-of-life (EOL) products [16], [17], [18], [19], [20], [21]. Some scholars propose a disassembly line balancing problem (DLBP) [22]. They answer how to determine the number of active workstations and plan disassembly tasks to optimize such performance as disassembly profit. Many have conducted in-depth studies of DLBP, but mostly focused on linear layouts [23] or U-shaped ones [24], [25]. Although they have different characteristics and applicable scenarios, there lacks any research that combines them. Therefore, this work does so by establishing a hybrid disassembly line. Such line can fully utilize factory space by allocating different types of products or tasks to the most suitable disassembly lines, thus maximizing space utilization. Additionally, the most appropriate type of disassembly lines for different disassembly phases can be selected to optimize a disassembly process. This allows disassembly lines to quickly adapt to market demand changes, thereby improving disassembly efficiency and competitiveness.

In modern disassembly systems, manpower plays a crucial role. As complex entities, workers usually possess multiple skills that enable them to perform various tasks in a disassembly process. Confronting the frequent changes experienced by disassembly systems, such as changes in customer demands or the occurrence of unexpected events, effectively allocating these multi-skilled workers to machines can be a challenging task [26]. Their proper assignment is of utmost importance. Research on line balancing with multi-skilled workers has seen diverse methodological contributions. Blum and Miralles [27] applied a beam search-based algorithm to DLBP, optimizing task and worker assignments to minimize system cycle time. Similarly, Zaman et al. [28] leveraged genetic and heuristic algorithms for the same problem, but with the objective of minimizing a weighted sum of cycle time and total idle time. The context of U-shaped assembly lines was explored by Oksuz et al. [29], who formulated a mixed-integer linear program to maximize efficiency and solved it using both an artificial bee colony algorithm and a genetic algorithm. In a dynamic setting, Belassiria et al. [30] investigated the assembly line rebalancing problem under uncertain demand. Their approach combined a mathematical model with a heuristic-embedded genetic algorithm to maximize production line efficiency. This work combines MRPOP with DLBP to propose a Multi-factory Remanufacturing-process-optimization Problem based on Multi-skilled-workers (MRPM) and discusses its specific application scenarios.

As MRPM is an NP-hard problem, heuristic algorithms becomes a preferred solution especially for those sizable ones. Some MRPM related studies are given next. The field has seen the application of diverse metaheuristics, ranging from the genetic algorithm by McGovern et al. [31] for DLBP, to more specialized approaches for sequence planning like the improved multi-objective ant colony algorithm by Feng et al. [32] and the multiverse optimizer considering operational faults by Fu et al. [33]. Further advancing the field, Guo et al. [34] solved a human-machine collaborative DLBP with stochastic times using a Pareto-improved shuffled frog-leaping algorithm. These studies provide us with inspiration and ideas to solve MRPM.

Battle Royale Optimizer (BRO) is an optimization algorithm based on the strategy of a battle royale game [35]. It achieves the iterative evolution of populations by simulating the search process of trying to defeat neighboring soldiers in the game. We choose it to solve MRPM because 1) it is easy to understand and simple to operate and implement, 2) it has high efficiency, robustness, and strong global search capability, and 3) it excels at balancing exploration and exploitation, leading to the efficient discovery of high-quality solutions.

The main contributions this work aims to make are:

1) A key contribution of this work is the novel integration of MRPOP and DLBP into a new problem termed MRPM. This integrated problem is formally defined through a mixed-integer program, with the objective set to maximize total disassembly profit.

2) This work proposes a discrete BRO (DBRO) to solve MRPW. In this algorithm, we design a novel encoding structure to represent solutions, and develop four novel soldier search strategies: sequential variation, task variation, workstation variation, and factory swap. These strategies enable individuals to better search for quality solutions.

3) This work presents extensive empirical findings to demonstrate the model's correctness and the algorithm's effectiveness. For the former, case tests are conducted by using IBM CPLEX to confirms. For the latter, we compare the optimisation results and running time of DBRO and CPLEX given different case size. We also compare the results of DBRO and such optimization algorithms as genetic algorithm (GA) [36], discrete migratory bird optimizer (DMBO) [37], dingo optimizer (DOA) [38], and fruit fly optimizer (FOA) [39], and well show that DBRO is more effective in solving MRPM.

The rest of this paper is organized as follows. Section II describes MRPM and its mathematical model. Section III introduces the encoding and decoding scheme of MRPM, DBRO, and the soldier's search and battle processes. Section IV details the experimental results, while Section V concludes with a summary of the study and potential directions for future research.

II. Problem Description

A. Problem Statement

In recent years, the disassembly sector has undergone significant transformations driven by the forces of economic globalization. More and more disassembly companies are establishing factories in different locations, forming a multi-factory disassembly model to meet the product demands of different regions. Due to their different geographical locations, each factory may incur varying transportation cost and purchase price, even when transporting the same subassemblies. In addition, disassembly lines varies among disassembly factories, resulting in different disassembly time and cost even for the same type of products. The core challenge lies in coordinating the allocation of tasks and workstations within each disassembly facility alongside the strategic selection of both disassembly and manufacturing factories. Therefore,

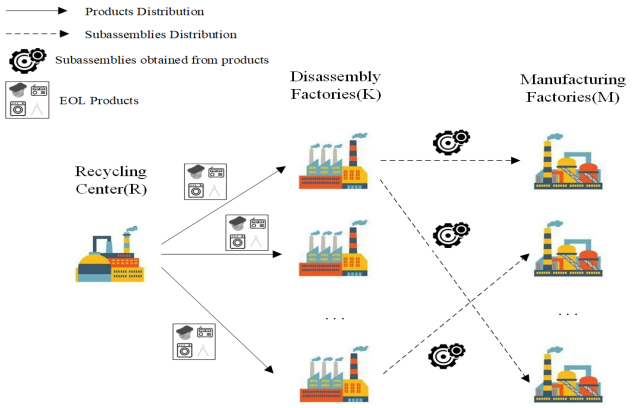


Fig. 2. The workflow of multi-factory remanufacturing process optimization

this work focuses on how to use the available resources effectively, allocate tasks from different customers to factories, and coordinate the plans of all factories.

This research is situated within an analytical framework of a large-scale disassembly enterprise that operates a network of geographically distributed factories. The enterprise has multiple factories in different regions, all capable of disassembling various products. In this context, each product can be assigned to different disassembly factories to complete its disassembly tasks, and the obtained subassemblies are transported to different manufacturing factories after the disassembly process. As shown in Fig. 2, MRPM is divided into three main phases.

1) Disassembly factory selection

This phase focuses on product assignment, which involves optimally assigning different EOL products to factories subject to various constraints.

2) Disassembly task scheduling

This phase encompasses three critical allocation decisions: disassembly line assignment, which entails selecting a specific line within a factory for each product; task assignment, which focuses on determining the optimal disassembly task sequence; and workstation assignment, which involves distributing the product's tasks across the various workstations on the chosen line.

3) Manufacturing factory selection

This phase orchestrates the distribution of subassemblies, where the allocation decision is dictated by cost variations arising from differing purchasing prices at manufacturing factories and the transportation expenses influenced by geographical distances between facilities.

The optimization objective of MRPM in this work is to maximize the total profit.

B. AND/OR Graph

There are precedence relationships among different disassembly tasks during a disassembly process of EOL products. The core challenge of the DLBP is to distribute disassembly tasks among a series of workstations while adhering to the predefined precedence relationships between these tasks. This task assignment process yields a disassembly sequence that meets all the constraints. For this purpose, the relationships among tasks are modeled using AND/OR graphs. It is noteworthy that alternative formal methods, including Petri nets [40][41], precedence graphs [42], or others [43], [44], can also be employed to model and analyze the disassembly process and its resource requirements.

Fig. 4 shows the schematic graph of a rigid caster [45]. It consists of nine subassemblies /components (both called subassemblies for conciseness in this paper) labeled as 1 to 9. Its AND/OR graph is shown in Fig. 3. An AND/OR graph is composed of nodes, which depict subassemblies and are denoted by integers within pointed brackets. The disassembly tasks are captured by directed edges connecting these related subassemblies. From this representation, it can be derived that the product comprises 25 subassemblies and 32 disassembly tasks.

To calculate the profits of subassemblies, three matrices are used to describe the relationship between disassembly tasks and subassemblies.

1) Incidence matrix

Let $D = [d_{pij}]$ denote the incidence matrix that links subassemblies and disassembly operations for each product p. The entry d_{pij} is defined as

$$d_{pij} = \begin{cases} 1, & \text{if task } j \text{ produces subassembly } i \text{ in product } p, \\ -1, & \text{if task } j \text{ removes subassembly } i \text{ in product } p, \\ 0, & \text{otherwise.} \end{cases}$$

2) Conflict matrix

Define the conflict indicator $R = [r_{pj_1j_2}]$, where

$$r_{pj_1j_2} = \begin{cases} 1, & \text{if tasks } j_1 \text{ and } j_2 \text{ for product } p \text{ cannot} \\ & \text{be processed simultaneously,} \\ 0, & \text{otherwise.} \end{cases}$$

3) Precedence matrix

Let $S = [s_{pj_1j_2}]$ represent task precedence, with

$$s_{pj_1j_2} = \begin{cases} 1, & \text{if task } j_1 \text{ must precede task } j_2 \text{ for product } p, \\ 0, & \text{otherwise.} \end{cases}$$

The following assumptions are made in this work:

1) Matrices D, R, and S are known.

2) Not all subassemblies need to be disassembled (called selective disassembly).

3) Each workstation is assigned one worker.

4) The disassembled products are infinitely supplied.

5) Each disassembly task requires at least one disassembly skill. The higher the skill level, the shorter the time required to complete the task.

6) The processing time at any workstation must not exceed the system cycle time.

7) Each activated workstation has at least one task.

C. Notations

Notation and symbols employed throughout this study are listed below for reference.

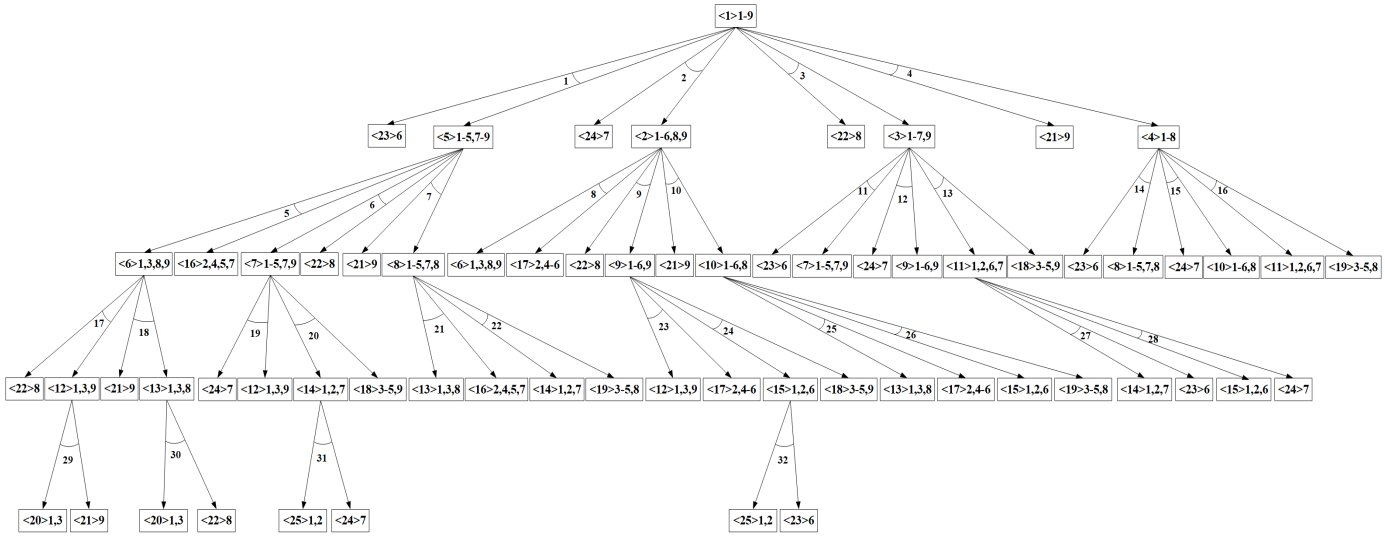


Fig. 3. The AND/OR graph of a rigid caster.

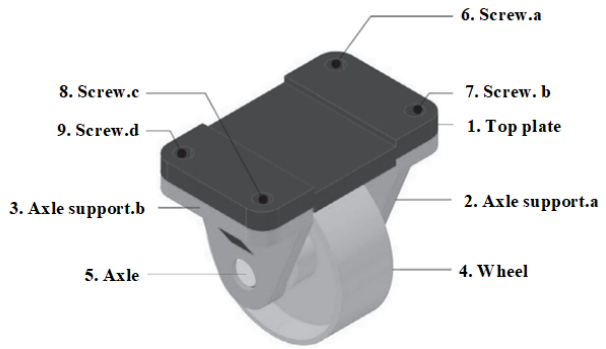


Fig. 4. A schematic of a rigid caster.

Sets:

$\mathbb{K}$ Set of disassembly factories, $\mathbb{K} = \{1, 2, ..., K\}$.

$\mathbb{M}$ Set of Manufacturing factories, $\mathbb{M} = \{1, 2, ..., M\}$.

$\mathbb{P}$ Set of products, $\mathbb{P} = \{1, 2, ..., P\}$.

$\mathbb{I}_p$ Set of all subassemblies in product p, $\mathbb{I}_p = \{1, 2, ..., I_p\}$.

$\mathbb{J}_p$ Set of all tasks in product p, $\mathbb{J}_p = \{1, 2, ..., J_p\}$.

$\mathbb{W}_k^L$ Set of linear workstations for the k-th factory, $\mathbb{W}_k^L = \{1, 2, ..., W_k^L\}$.

$\mathbb{W}_k^U$ Set of U-shaped workstations for the k-th factory, $\mathbb{W}_k^U = \{1, 2, ..., W_k^U\}$.

$\mathbb{E}$ Set of sides of U-shaped disassembly line workstation, $\mathbb{E} = \{1, 2\}$.

$\mathbb{N}$ Set of all additional skills, $\mathbb{N} = \{1, 2, ..., N\}$.

Indexes:

p Product index, $p \in \mathbb{P}$.

i Subassembly index, $i \in \mathbb{I}_p$.

j Disassembly task index, $j \in \mathbb{J}_p$.

e Index of U-shaped workstation side, $e \in \mathbb{E}$.

k Disassembly factory index. $k \in \mathbb{K}$.

m Manufacturing factory index. $m \in \mathbb{M}$.

n Skills index, $n \in \mathbb{N}$.

Parameters:

v_{mpi} The m-th manufacturing factory acquires the price of the i-th subassembly of the product p.

c_{kmpi}^T Transportation cost of the subassembly i of the product p from the k-th disassembly factory to the manufacturing m factory.

t_{kwpj} Disassembly time required by workers at the w-th workstation of the disassembly factory k to complete the task j of the product p.

c_{kpj}^d The unit time cost of executing the task j of the product p in the factory k.

c_k The unit time cost of activating the k-th factory.

c_{kw}^L Cost of activating the w-th linear workstation of the k-th disassembly factory.

c_{kw}^U Cost of activating the U-shaped workstation w of the disassembly factory k.

C Cost of worker skills training.

γ_{kwn} The matrix records the degree of mastery of the n-th skill by workers at the w-th workstation in the k-th factory in the initial state.

β_{pjn} The matrix depicts the disassembly relationship between the task j of the product p and the n-th skill.

Decision variables:

$$z_{pk} = \begin{cases} 1, & \text{If the product } p \text{ is assigned to the } k\text{-th disassembly factory;} \\ 0, & \text{otherwise.} \end{cases}$$

$$x^L_{pjkw} = \begin{cases} 1, & \text{If the task } j \text{ of the product } p \text{ is assigned} \\ & \text{for disassembly at the } w\text{-th linear workstation} \\ & \text{of the } k\text{-th disassembly factory;} \\ 0, & \text{otherwise.} \end{cases}$$

$$x^U_{pjkwe} = \begin{cases} 1, & \text{If the task } j \text{ of the product } p \text{ is assigned} \\ & \text{to the } e\text{-side of the U-shaped workstation } w \\ & \text{at the disassembly factory } k \text{ for disassembly;} \\ 0, & \text{otherwise.} \end{cases}$$

$$y^L_k = \begin{cases} 1, & \text{If the linear disassembly line of the } k\text{-th} \\ & \text{disassembly factory is activated;} \\ 0, & \text{otherwise.} \end{cases}$$

$$y^U_k = \begin{cases} 1, & \text{If the U-shaped disassembly line of the } k\text{-th} \\ & \text{disassembly factory is activated;} \\ 0, & \text{otherwise.} \end{cases}$$

$$u^L_{kw} = \begin{cases} 1, & \text{If the linear workstation } w \text{ of the } k\text{-th} \\ & \text{disassembly factory is activated;} \\ 0, & \text{otherwise.} \end{cases}$$

$$u^U_{kw} = \begin{cases} 1, & \text{If the U-shaped workstation } w \text{ of the } k\text{-th} \\ & \text{disassembly factory is activated;} \\ 0, & \text{otherwise.} \end{cases}$$

$$\alpha_{kmpi} = \begin{cases} 1, & \text{If the subassembly } i \text{ of the } p\text{-th product} \\ & \text{is transported from the disassembly } k \\ & \text{factory to the } m\text{-th manufacturing factory;} \\ 0, & \text{otherwise.} \end{cases}$$

$$\xi_{kwn} = \begin{cases} 1, & \text{If workers at the } w\text{-th workstation of the} \\ & k\text{-th factory utilize the } n\text{-th skill;} \\ 0, & \text{otherwise.} \end{cases}$$

T_k, Cycle time of the k-th disassembly factory.

D. Mathematical Model

$$\begin{aligned} \max f = & \left(\sum_{k\in\mathbb{K}} \sum_{m\in\mathbb{M}} \sum_{p\in\mathbb{P}} \sum_{i\in\mathbb{I}_p} \left(v_{mpi} - c^T_{kmpi} \right) \alpha_{kmpi} \right. \\ & - \sum_{k\in\mathbb{K}} \sum_{p\in\mathbb{P}} \sum_{j\in\mathbb{J}_p} \sum_{w\in\mathbb{W}^L_k} c^d_{kpj} t_{kwpj} x^L_{pjkw} \\ & - \sum_{k\in\mathbb{K}} \sum_{p\in\mathbb{P}} \sum_{j\in\mathbb{J}_p} \sum_{w\in\mathbb{W}^U_k} \sum_{e\in\mathbb{E}} c^d_{kpj} t_{kwpj} x^U_{pjkwe} \\ & - \sum_{k\in\mathbb{K}} c_k T_k - \sum_{k\in\mathbb{K}} \sum_{w\in\mathbb{W}^L_k} c^L_{kw} u^L_{kw} - \sum_{k\in\mathbb{K}} \sum_{w\in\mathbb{W}^U_k} c^U_{kw} u^U_{kw} \\ & \left. - C \sum_{k\in\mathbb{K}} \sum_{w\in\mathbb{W}^k} \sum_{n\in\mathbb{N}} \left(\xi_{kwn} - \gamma_{kwn} \right) \right) \end{aligned} \tag{1}$$

The objective (1) maximizes the profit from disassembling end-of-life (EOL) products. The first term denotes the net gain from subassemblies after subtracting transport costs; the second and third terms capture disassembly-related expenses; the fourth term is the fixed cost for opening a disassembly facility; the fifth and sixth terms correspond to the costs of activating the associated workstations; and the seventh term represents worker training costs. The model is subject to the following constraints.

$$\sum_{m\in\mathbb{M}} \alpha_{kmpi} \le \sum_{w\in\mathbb{W}^L_k} \sum_{j\in\mathbb{J}_p} d_{pij} x^L_{pjkw} + \sum_{w\in\mathbb{W}^U_k} \sum_{j\in\mathbb{J}_p} \sum_{e\in\mathbb{E}} d_{pij} x^U_{pjkwe} \tag{2}$$
$$\forall k \in \mathbb{K},\ \forall p \in \mathbb{P},\ \forall i \in \mathbb{I}_P \setminus \{1\}.$$

$$\sum_{k\in\mathbb{K}} z_{pk} = 1,\ \forall p \in \mathbb{P} \tag{3}$$

$$y^L_k + y^U_k \le 1,\ \forall k \in \mathbb{K} \tag{4}$$

$$z_{pk} \le y^L_k + y^U_k,\ \forall p \in \mathbb{P},\ \forall k \in \mathbb{K} \tag{5}$$

$$u^L_{kw} \le y^L_k,\ \forall w \in \mathbb{W}^S_k,\ \forall k \in \mathbb{K} \tag{6}$$

$$u^U_{kw} \le y^U_k,\ \forall w \in \mathbb{W}^U_k,\ \forall k \in \mathbb{K} \tag{7}$$

$$\sum_{w\in\mathbb{W}^L_k} x^L_{pjkw} + \sum_{w\in\mathbb{W}^U_k} \sum_{e\in\mathbb{E}} x^U_{pjkwe} \le z_{pk},\ \forall p \in \mathbb{P},\ \forall k \in \mathbb{K},\ \forall j \in \mathbb{J}_p \tag{8}$$

$$x^L_{pjkw} \le u^L_{kw},\ \forall p \in \mathbb{P},\ \forall j \in \mathbb{J}_p,\ \forall k \in \mathbb{K},\ \forall w \in \mathbb{W}^L_k \tag{9}$$

$$x^U_{pjkwe} \le u^U_{kw},\ \forall p \in \mathbb{P},\ \forall j \in \mathbb{J}_p,\ \forall k \in \mathbb{K},\ \forall w \in \mathbb{W}^U_k,\ \forall e \in \mathbb{E} \tag{10}$$

$$x^U_{pjkwe} \le \sum_{n\in\mathbb{N}} \beta_{pjn} \xi_{kwn},\ \forall p \in \mathbb{P},\ \forall j \in \mathbb{J}_p,\ \forall k \in \mathbb{K},\ \forall w \in \mathbb{W}^U_k,\ \forall e \in \mathbb{E} \tag{11}$$

$$x^L_{pjkw} \le \sum_{n\in\mathbb{N}} \beta_{pjn} \xi_{kwn},\ \forall p \in \mathbb{P},\ \forall j \in \mathbb{J}_p,\ \forall k \in \mathbb{K},\ \forall w \in \mathbb{W}^L_k \tag{12}$$

$$\xi_{kwn} \ge \gamma_{kwn},\ \forall k \in \mathbb{K},\ \forall w \in \mathbb{W},\ \forall n \in \mathbb{N} \tag{13}$$

$$\sum_{k\in\mathbb{K}} \left(\sum_{w\in\mathbb{W}^L_k} x^L_{pjkw} + \sum_{w\in\mathbb{W}^U_k} \sum_{e\in\mathbb{E}} x^U_{pjkwe} \right) \le 1,\ \forall p \in \mathbb{P},\ \forall j \in \mathbb{J}_p \tag{14}$$

$$\sum_{j\in\mathbb{J}_p} t_{kwpj} x^L_{pjkw} \le T_k,\ \forall k \in \mathbb{K},\ \forall w \in \mathbb{W}^L_k \tag{15}$$

$$\sum_{j\in\mathbb{J}_p} \sum_{e\in\mathbb{E}} t_{kwpj} x^U_{pjkwe} \le T_k,\ \forall k \in \mathbb{K},\ \forall w \in \mathbb{W}^U_k \tag{16}$$

$$\sum_{w\in\mathbb{W}^L_k} w \left(x^L_{pj_1kw} - x^L_{pj_2kw} \right) + W^L_k \left(\sum_{w\in\mathbb{W}^L_k} x^L_{pj_2kw} - 1 \right) \le 0 \tag{17}$$
$$\forall k \in \mathbb{K},\ \forall p \in \mathbb{P},\ \forall j_1, j_2 \in \mathbb{J}_p,\ s_{pj_1j_2} = 1$$

$$\sum_{w\in\mathbb{W}_k^U}\left(w\left(x_{pj_1kw1}^U-x_{pj_2kw1}^U\right)+\left(2W_k^U-w\right)\left(x_{pj_1kw2}^U-x_{pj_2kw2}^U\right)\right)$$
$$+2W_k^U\left(\sum_{w\in\mathbb{W}_k^U}\sum_{e\in\mathbb{E}}x_{pj_2kwe}^U-1\right)\le 0,\ \forall k\in\mathbb{K},\ \forall p\in\mathbb{P},$$
$$\forall j_1,j_2\in\mathbb{J}_p,\ s_{pj_1j_2}=1 \tag{18}$$

$$\sum_{w\in\mathbb{W}_k^L}x_{pj_2kw}^L\le\sum_{j_1\in\mathbb{J}_p}\sum_{w\in\mathbb{W}_k^L}s_{pj_1j_2}x_{pj_1kw}^L,\ \forall k\in\mathbb{K},\ \forall p\in\mathbb{P},$$
$$\forall j_2\in\mathbb{J}_p,\ d_{p1j_2}=0 \tag{19}$$

$$\sum_{w\in\mathbb{W}_k^U}\sum_{e\in\mathbb{E}}x_{pj_2kwe}^U\le\sum_{j_1\in\mathbb{J}_p}\sum_{w\in\mathbb{W}_k^U}\sum_{e\in\mathbb{E}}s_{pj_1j_2}x_{pj_1kwe}^U,\ \forall k\in\mathbb{K},$$
$$\forall p\in\mathbb{P},\ \forall j_2\in\mathbb{J}_p,\ d_{p1j_2}=0 \tag{20}$$

$$\sum_{w\in\mathbb{W}_k^L}\left(x_{pj_1kw}^L+x_{pj_2kw}^L\right)\le 1,\ \forall k\in\mathbb{K},\ \forall p\in\mathbb{P},$$
$$\forall j_1,j_2\in\mathbb{J}_p,\ r_{pj_1j_2}=1 \tag{21}$$

$$\sum_{w\in\mathbb{W}_k^U}\sum_{e\in\mathbb{E}}\left(x_{pj_1kwe}^U+x_{pj_2kwe}^U\right)\le 1,\ \forall k\in\mathbb{K},\ \forall p\in\mathbb{P},$$
$$\forall j_1,j_2\in\mathbb{J}_p,\ r_{pj_1j_2}=1 \tag{22}$$

$$z_{pk}\in\{0,1\},\ \forall p\in\mathbb{P},\ \forall k\in\mathbb{K} \tag{23}$$

$$x_{pj_1kw}^L\in\{0,1\},\ \forall p\in\mathbb{P},\ \forall j\in\mathbb{J}_p,\ \forall w\in\mathbb{W}_k^L,\ \forall k\in\mathbb{K} \tag{24}$$

$$x_{pj_1kwe}^U\in\{0,1\},\forall p\in\mathbb{P},\forall j\in\mathbb{J}_p,\ \forall w\in\mathbb{W}_k^U,\ \forall k\in\mathbb{K},$$
$$\forall e\in\mathbb{E} \tag{25}$$

$$y_k^L\in\{0,1\},\ \forall k\in\mathbb{K} \tag{26}$$

$$y_k^U\in\{0,1\},\ \forall k\in\mathbb{K} \tag{27}$$

$$u_{kw}^L\in\{0,1\},\ \forall k\in\mathbb{K},\ \forall w\in\mathbb{W}_k^L \tag{28}$$

$$u_{kw}^U\in\{0,1\},\ \forall k\in\mathbb{K},\ \forall w\in\mathbb{W}_k^U \tag{29}$$

$$\xi_{kwn}\in\{0,1\},\ \forall k\in\mathbb{K},\ \forall w\in\mathbb{W}^k,\ \forall n\in\mathbb{N} \tag{30}$$

$$T_k\in\mathbb{R}_+,\ \forall k\in\mathbb{K} \tag{31}$$

Constraint (2) ensures that subassemblies obtained by disassembling a product can be transported to only one manufacturing factory. (3) ensures that each product may be allocated to at most one disassembly facility. (4) ensures that only one type of disassembly line can be turned on at a disassembly factory. (5) ensures that each product can be assigned exclusively to a disassembly facility that has been activated. (6) and (7) ensure that the corresponding workstations are only used after the disassembly line has been activated at the disassembly factory. (8) ensures that a disassembly operation j for product p is permitted to be allocated only to workstations that are active in the disassembly plant to which product p has been assigned. (9) and (10) ensure that the disassembly task j of product p is assigned to the open workstation. (11) and (12) ensure that the skills of the workers at the workstation meet the skill constraints required for disassembly task j. (13) ensure that the skills acquired by all workers are not reduced. (14) specifies that every disassembly task is executed at most once for each product. Collectively, (15) and (16) guarantee that the total operational time at every workstation, across all disassembly lines, remains within the designated factory cycle time. Furthermore, (17) mandates that the assignment of tasks to a linear disassembly line adheres to the predefined precedence relationships.

(18) imposes a precedence constraint for task assignments on a U-shaped disassembly line. It stipulates that for task j_1 to precede j_2, the workstation index of j_2 must be greater than or equal to that of j_1 if both are on the inlet side, and less than or equal if both are on the outlet side, thereby ensuring the required execution sequence.

(19) and (20) ensure that the disassembly sequence of the product can begin from other tasks. (21) and (22) ensure that the assignment of disassembly tasks causes no conflict among them. (23)-(31) indicate the range of values of decision variables.

III. Proposed Algorithm

The Battle Royale Optimizer (BRO), introduced by Rahkar-Farshi in 2020, is a population-based metaheuristic algorithm. Its basic concept is inspired by the game PlayerUnknown's Battlegrounds (PUBG). In PUBG, there is a game mode called Deathmatch, where players aim to kill as many other players as possible until reaching the kill or time limit. Generally, battles occur on specific battlefield maps chosen by the players [46], [47], [48]. Such an optimization problem space is considered as a game map in BRO. The game starts by having players jump onto the map from an airplane. Similar to many other population-based optimization algorithms, the search agents in BRO are randomly initialized within the search space through uniform random initialization. Throughout the game, if other rivals kill a player, he respawns at a random zoon on the battlefield. The ultimate winner is the player who gets the most kills throughout the game. In this article, both soldiers and players represent individuals.

A. Discrete Battle Royale Optimizer (DBRO)

BRO is originally used to solve continuous optimization problems and needs its discrete version to solve MRPM that is a discrete optimization problem. We propose DBRO to do so.

In DBRO, the initial population is randomly distributed across the entire problem space. Each individual randomly searches for favorable positions and then attempts to inflict damage on its nearest target. Each soldier has a damage level with an initial value of zero. When a soldier gets injured, its damage level increases, and it immediately changes its current position to search for other favorable positions. If an

injured soldier can inflict damage on other soldiers in the next iteration, its damage level is reset to zero. If an individual soldier's damage metric exceeds a preset limit, that soldier is removed and reinitialized at a random feasible location with its damage reset to zero. This cycle continues until either a satisfactory solution is obtained or the iteration budget is exhausted. To avoid premature convergence, we employ a mutation-like restart: when the population's best objective value shows no improvement for a specified number of iterations, the population is reinitialized. The DBRO procedure is summarized in Algorithm 1.

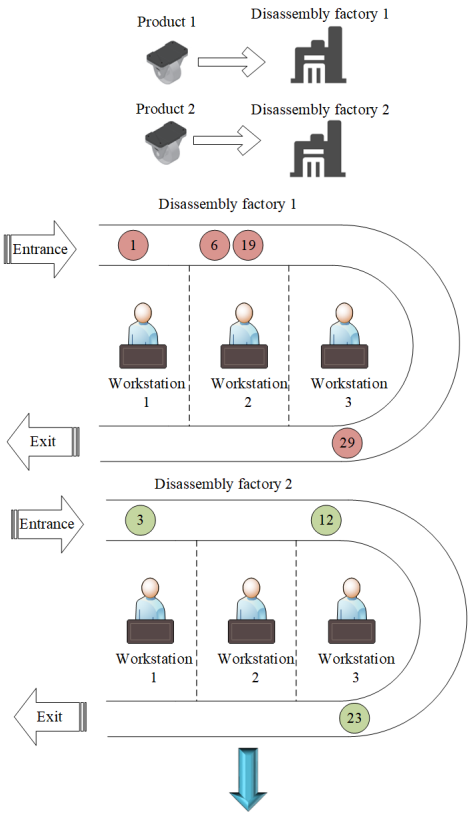
Algorithm 1 Discrete Battle Royale Optimizer

Input: maximum iterations Max, population size n
Output: the best solution X

Initializing the soldier population
x_{dam} = the damage level of each soldier.
r_{dam} = combat rate
Calculate the fitness of each soldier
while ($iter$<Max) **do**
  for each soldier **do**
    Execute the search process to reach a new location
    Generate random probability r
    if r>r_{dam} **then**
      Execute combat process
      x_{dam}=x_{dam}+1
    else
      Soldiers continue the search process to reach a new location
    end if
    if x_{dam}>Threshold **then**
      execute soldier rebirth process
    end if
  end for
  Update X and record the optimal objective value
  Decide whether to initialize the population
  $iter$=$iter$+1
end while
return X

B. Encoding and Decoding

Effective encoding strategies are crucial because they determine how information is transformed and encoded during transmission, thus better addressing problems. Based on the characteristics of MRPM, we expect the generated solution to correspond accurately to an allocation plan during an actual allocation process. In this problem, the selection of disassembly factories and their internal disassembly lines, allocation of workstations, the sequence of disassembly tasks performed on the workstations all affect the objective value of a solution. In order to better address the researched problem, we design a four-stage encoding strategy $\sigma(\sigma_1, \sigma_2, \sigma_3, \sigma_4)$ to represent the solutions of MRPM. The encoding framework is illustrated in Fig. 5. Specifically, σ_1 denotes the sequence of disassembly tasks, σ_2 specifies the index of the disassembly factory, σ_3 corresponds to the type of disassembly line within the factory, where the value of 1 represents a linear line and 2 refers to a U-shaped configuration, and σ_4 indicates the



Task sequence (σ_1)	1	6	19	29	3	12	23
Disassembly factory (σ_2)	1	1	1	1	2	2	2
Disassembly line (σ_3)	2	2	2	2	2	2	2
Workstation (σ_4)	1	2	2	3	1	3	3

Fig. 5. Schematic graph of encoding structure.

workstation number. As depicted in Fig. 5, product 1 is allocated to disassembly factory 1, whereas product 2 is assigned to factory 2. Both factories employ U-shaped disassembly lines. For product 1, task 1 is executed at workstation 1, tasks 6 and 19 are performed at workstation 2, and task 29 is handled at workstation 3. In the case of product 2, task 3 is processed at workstation 1, while tasks 12 and 23 are allocated to workstation 3.

During the decoding stage, the algorithm constructs a feasible disassembly sequence by considering both conflict constraints and precedence relations among tasks. Each task within the sequence remains in a state awaiting allocation. Taking the rigid caster as an example, the detailed decoding procedure is illustrated in Fig. 5.

Step 1: Assign the EOL products to disassembly factories.

Step 2: Select the disassembly lines that factories activate for EOL products.

Step 3: Assign disassembly tasks to workstations in order. If a disassembly task is assigned to a workstation and exceeds its cycle constraint, activate the next workstation and assign the task to it for execution.

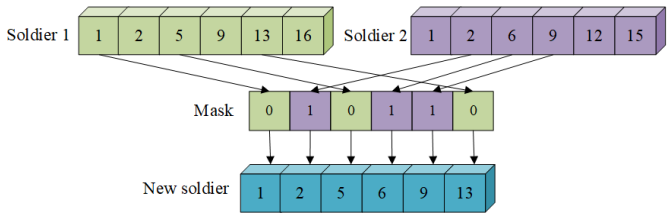


Fig. 6. Process of combat.

C. Initializing population

The initial population generally consists of multiple soldiers, each of which represents a set of feasible solution encodings. During the population initialization, a random-length disassembly task sequence is formed. Subsequently, a task sequence is made feasible by considering conflicts and precedence relationships among the disassembly tasks. Each task in a disassembly sequence is unallocated. Then, these EOL products are allocated to various factories for disassembly. The initialization process is given in the Supplementary File (S.F).

D. Combat and Search

Owing to the limitations of map size and available time, soldiers must search for and attack opponents to secure victory. During the search phase, soldiers may either enter combat or successfully evade encounters. In the combat phase, the algorithm employs a predefined combat rate. A random number is generated to determine the outcome: if the random value exceeds the combat rate, the soldier engages in battle; otherwise, the soldier evades combat and continues searching. The overall battle process is illustrated in Fig. 6.

Step 1: Select two ordinary soldiers.

Step 2: Randomly generate a set of binary masks, then evaluate the value of each mask from left to right. A value of 0 indicates that the disassembly task is obtained from Soldier 1, while a value of 1 indicates that the disassembly task is obtained from Soldier 2. If the acquired task already exists in the new individual, we need to skip the current task and obtain the next task from the selected soldier.

Step 3: Adjust the newly generated sequence to satisfy conflict and precedence constraints.

For soldier search, we design four actions to better search for the optimal solution, namely sequential variation, task variation, workstation variation, and factory swap. The soldier search process is given in the S.F.

Sequential variation: As illustrated in Fig. 7, task 13 is randomly chosen. According to the task precedence constraints, task 7 serves as its immediate predecessor, while task 17 is its immediate successor. Consequently, task 13 can be reinserted into any position between tasks 7 and 17.

Task variation: As shown in Fig. 8, task 10 is randomly selected and subsequently removed together with its following tasks. Referring to the AND/OR graph, the subassembly generated after completing task 10 is identified, and an alternative disassembly route for this subassembly is selected, producing tasks 8, 12, and 15.

Workstation variation: As presented in Fig. 9, a workstation is randomly chosen. Then, either the first or last task within that workstation is randomly determined. If the first task is chosen, it is transferred to the previous workstation; if the last task is chosen, it is moved to the next one.

Factory swap: As shown in Fig. 10, we randomly select the disassembly factory to which two products belong and swap them.

E. Rebirth Process

In DBRO, when a soldier's damage level surpasses a predefined threshold, the soldier is considered dead and respawns randomly within the feasible solution space, with the damage level reset to zero. The newly generated soldier is then merged with the existing ones to form an updated population. Afterward, the population is re-evaluated and ranked according to individual fitness values, and the top-performing individuals are retained for the next iteration.

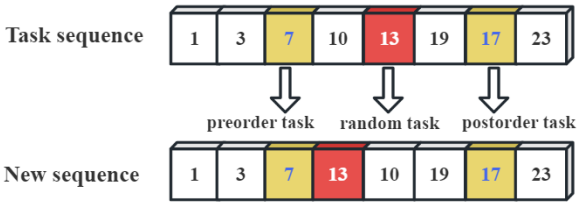


Fig. 7. Process of sequential variation.

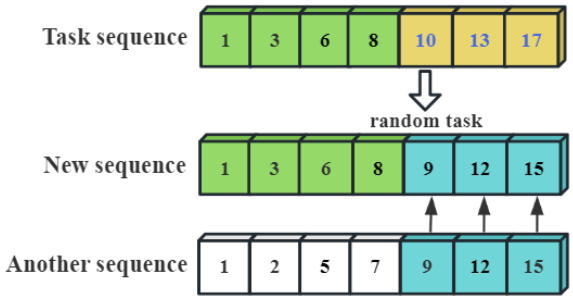


Fig. 8. Process of task variation.

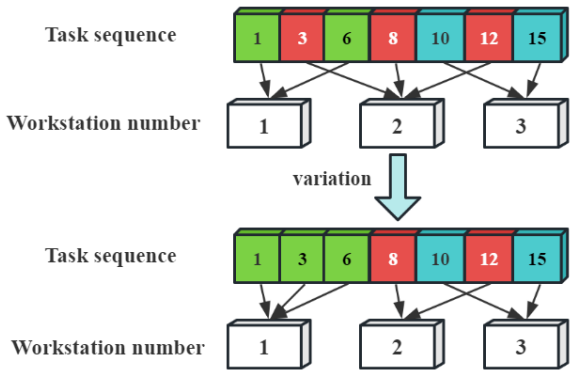


Fig. 9. Process of workstation variation.

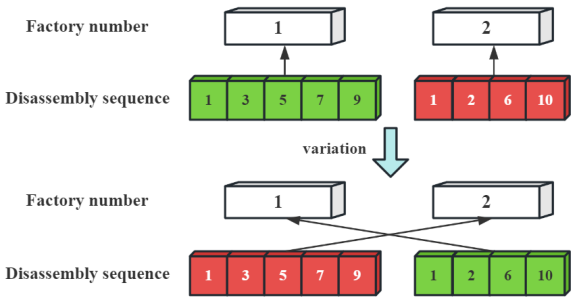


Fig. 10. Redistribution of disassembly factory.

IV. Experimental Studies

A. Experimental Cases and Parameter Settings

To verify the validity of the proposed model and evaluate the performance of the algorithm, IBM CPLEX is utilized to solve the experimental instances and obtain the optimal solutions. The same instances are also solved using DBRO for comparative analysis. All computations are executed on a computer equipped with an Intel(R) Core(TM) i5-8300H CPU (2.30 GHz) and 16.00 GB of RAM.

To ensure the comprehensiveness of the experimental analysis, four products of different sizes are selected, namely a washing machine [49], a compass [50], a rigid caster [45], and a radio [51]. We combine these products into different multiple product cases for testing. The scale information for the combined case and the parameter settings for the disassembly factories are given in the S.F. Table I shows detailed information about EOL products. Take the rigid caster as an example. It consists of 9 parts and involves 32 disassembly tasks. Disassembling it yields 25 subassemblies, with profits ranging from 53 to 75. The transportation cost falls between 6 and 20, while the disassembly cost per unit time ranges from 2 to 7. The completion time for the disassembly tasks ranges from 5 to 8.

B. Model Validation and Analysis

In our experimental setup, we employed IBM CPLEX with a maximum runtime limit of three hours to evaluate the test cases. The results are shown in Table II, where the column of factory number represents the disassembly factory assigned to the product, and w, c, r, and d, respectively stand for washing machine, compass, rigid caster, and radio. The column of disassembly line represents the type of disassembly line activated by the factory. Taking case 4 as an example, the washing machine is allocated to factory 1, and rigid caster 1 and rigid caster 2 are assigned to factory 3 for disassembly. Factories 1 and 3 activate a U-shaped disassembly line. The column of profit represents the objective value corresponding to this disassembly sequence, and the best bound indicates the range of values of feasible solutions. The column of gap represents the gap between the current solution and the best one. When solving a problem with CPLEX, the objective value gradually approaches the best bound. When the gap reaches 0, it indicates that the current solution is the optimal solution. The column of calculation time indicates the time required to solve. It can be seen that CPLEX is not very efficient in solving this problem due to the complexity of the model and the high dimensionality of the decision variables. For cases 1-4, CPLEX can find the optimal solution, but the required computation time grows very fast as the case size increases. For some multi-product cases, the optimal solution cannot be obtained within the set time of three hours. As the problem size increases, the gap between the current solution and the best bound becomes more pronounced.

DBRO is applied to the aforementioned cases, and the experimental results are summarized in Table III. As observed from Table III, DBRO achieves markedly shorter runtimes compared with CPLEX. Its runtime is mainly influenced by its time complexity, and the increase in the scale of the cases does not result in a substantial increase in time. From Tables IV and V, we can see that for cases 1-4, where CPLEX can find the optimal solution, DBRO can do so as well and has smaller running time. For cases 5-8, where CPLEX gives feasible solutions, the algorithm can find higher-quality solutions faster.

Table V compares the differences between DBRO and CPLEX regarding the optimal objective value and runtime. We can observe that for small to medium-sized instances, DBRO is significantly faster than CPLEX, and its solution accuracy is close to that of CPLEX. For large-scale instances or highly complex solutions, CPLEX may fail to reach the optimal solution within the allocated time, providing only feasible solutions. In such cases, DBRO demonstrates superior performance over CPLEX in both solution quality and computational efficiency.

C. Case Study

MRPM solutions are impacted by various factors, such as the prices charged by different manufacturing factories for the subassemblies, the costs required to complete the disassembly tasks in different disassembly factories, the costs of opening the factory workstations, and the transportation costs between factories, and so forth. Therefore, we select case 5 for our case study. Products w and c are allocated to factory 1, while product r is assigned to factory 3. The subassemblies obtained from disassembling product w, such as 2, 8, 14, and 15, are transported to manufacturing factory 1, while subassemblies 11 and 13 are transported to manufacturing factory 2. For product c, subassemblies 7, 13, and 16 obtained from disassembly are shipped to manufacturing factory 1, while subassemblies 14, 15, 17, and 18 are sent to manufacturing factory 2. Lastly, subassemblies 20, 22, and 24 obtained from disassembling product r are transported to manufacturing factory 1, and subassemblies 12, 21, and 23 are shipped to manufacturing factory 3. Disassembly factory 1 activates the U-shaped disassembly line and assigns tasks 1, 10, and 26 to workstation 1, tasks 2, 5, and 28 to workstation 2, tasks 11 and 21 to workstation 3, and tasks 18, 23, and 24 to workstation 5. Disassembly factory 3 activates a linear disassembly line, where tasks 1 and 6 are assigned to workstation 3 and tasks 19 and 29 to workstation 5.

The detailed assignment diagram for Case 5 is available in the S.F.

D. Verification of Algorithm With Different Scales of Cases

A population-based intelligence algorithm has a stochastic nature. To verify the superiority of DBRO, we select DMBO, FOA, DOA, and GA to conduct experimental case tests. For

TABLE I Product Parameters

Product	Num. of parts	Num. of task	Num of subassembly	Subassembly profit	Transport cost	Disassembly cost	Disassembly time
Washing machine	6	13	15	96 ~ 139	12 ~ 29	3 ~ 8	4 ~ 11
Compass	7	15	18	26 ~ 56	2 ~ 10	1 ~ 5	4 ~ 6
Rigid caster	9	32	25	53 ~ 75	6 ~ 20	2 ~ 7	5 ~ 8
Radio	10	30	29	69 ~ 98	7 ~ 23	2 ~ 6	5 ~ 7

The number of parts, tasks, and units for disassembling the product are all in units. The profit of parts, transportation costs, and dismantling costs are all in yuan. The unit of disassembly time is seconds.

TABLE II CPLEX Solutions

Case ID	Assignment of product to factory	Assignment of line type	Profit(Best Bound)	Gap	Calculation time
1	$\{r\} \rightarrow 3$	$< 3, L >$	228 (228)	0.00%	4.1s
2	$\{w_1, w_2\} \rightarrow 2$	$< 2, U >$	917 (917)	0.00%	7.5s
3	$\{c_1\} \rightarrow 1, \{c_2, r_1\} \rightarrow 3$	$< 1, L >, < 3, U >$	674 (674)	0.00%	562.5s
4	$\{w\} \rightarrow 1, \{r_1, r_2\} \rightarrow 3$	$< 1, U >, < 3, U >$	937 (937)	0.00%	9236.4s
5	$\{w,c,r\} \rightarrow 1$	$< 1, U >$	898 (985)	8.83%	10800.0s
6	$\{w,d\} \rightarrow 1, \{r\} \rightarrow 2$	$< 1, U >, < 2, L >$	1186 (1288)	7.92%	10800.0s
7	$\{w,c,r\} \rightarrow 1, \{d\} \rightarrow 3$	$< 1, U >, < 3, U >$	1413 (1598)	11.58%	10800.0s
8	$\{w,c\} \rightarrow 1, \{r_1, r_2, d\} \rightarrow 3$	$< 1, U >, < 3, U >$	1682 (1840)	9.41%	10800.0s

TABLE III DBRO Solutions

Case ID	Iterations / Population	Assignment of product to factory	Assignment of line type	Profit	Calculation time
1	200 / 100	$\{r\} \rightarrow 3$	$< 3, U >$	228	0.21s
2	200 / 100	$\{r, w_1, w_2\} \rightarrow 1$	$< 1, L >$	917	0.56s
3	300 / 150	$\{c_1, c_2\} \rightarrow 2, \{r\} \rightarrow 3$	$< 2, L >, < 3, U >$	674	1.55s
4	300 / 150	$\{w\} \rightarrow 1, \{r_1, r_2\} \rightarrow 2$	$< 1, U >, < 2, U >$	937	1.87s
5	400 / 200	$\{w,c\} \rightarrow 1, \{r\} \rightarrow 3$	$< 1, U >, < 3, L >$	922	3.14s
6	400 / 200	$\{w,d\} \rightarrow 1, \{r\} \rightarrow 3$	$< 1, U >, < 3, U >$	1214	3.65s
7	400 / 200	$\{w,c\} \rightarrow 1, \{r,d\} \rightarrow 2$	$< 1, U >, < 2, U >$	1484	3.38s
8	400 / 200	$\{w,c,r_1\} \rightarrow 2, \{r_2, d\} \rightarrow 3$	$< 2, U >, < 3, U >$	1703	3.73s

TABLE IV Algorithm Performance Comparison

Case ID	Optimal					Hit rate					Calculation time(s)				
	DBRO	DMBO	FOA	DOA	GA	DBRO	DMBO	FOA	DOA	GA	DBRO	DMBO	FOA	DOA	GA
1	228	228	228	228	228	100%	100%	100%	90%	95%	0.21	0.27	0.34	0.16	0.11
2	917	917	917	917	917	60%	55%	60%	45%	35%	0.56	1.29	1.74	0.58	0.32
3	674	674	674	674	674	55%	30%	25%	20%	35%	1.55	2.38	2.89	1.34	0.87
4	937	935	935	933	933	35%	30%	25%	20%	40%	1.87	2.51	2.92	1.54	0.87
5	922	918	920	915	892	35%	30%	25%	15%	20%	3.24	4.67	5.21	2.86	1.75
6	1214	1210	1197	1203	1181	45%	40%	45%	20%	25%	3.65	4.22	6.34	3.24	1.98
7	1484	1475	1479	1468	1471	30%	25%	30%	25%	10%	3.38	4.71	5.95	2.72	1.89
8	1703	1703	1693	1682	1687	35%	15%	25%	15%	20%	3.73	4.78	6.38	3.53	1.96

each algorithm and case, we conduct 20 independent experiments, record the best objective value for each generation, and calculate the average of these objective values. Based on the averages, we plot the iteration curves for each algorithm. The iterator curves for case 5-8 are given in the S.F. Combining the iteration curves for the four cases, it can be observed that DBRO exhibits faster convergence and higher solution quality than its four peers.

To further evaluate the stability of these algorithms, the best and worst solutions obtained from 20 independent runs of each algorithm are recorded, with the results presented in the Supplementary Files. It can be seen that DBRO achieves a higher upper bound compared to the other algorithms. Moreover, its minimum and average values are also higher, demonstrating that it maintains strong stability across these scenarios.

Table IV presents the optimal solutions, hit rates, and average calculation times achieved by each algorithm. The hit rate is calculated as the proportion of times each algorithm obtains the optimal solution in 20 experiments. From the results in Table IV, it can be observed that for small-scale situations, such as case 1, DBRO consistently achieves the optimal solution and has a faster solving speed than CPLEX. As the case scale increases, the algorithm's stability gradually

TABLE V Comparison of Results Between CPLEX and DBRO

Case ID	Profit			Calculation time		
	CPLEX	DBRO	Increased by	CPLEX	DBRO	Reduced by
1	228	228	0.00%	4.1s	0.21s	94.88%
2	917	917	0.00%	7.5s	0.56s	92.53%
3	674	674	0.00%	562.5s	1.55s	99.72%
4	937	937	0.00%	9236.4s	1.87s	99.98%
5	898	922	2.60%	10800.0s	3.24s	99.97%
6	1186	1214	2.31%	10800.0s	3.65s	99.97%
7	1413	1484	4.78%	10800.0s	3.38s	99.97%
8	1682	1703	1.23%	10800.0s	3.73s	99.97%

decreases, meaning that it cannot consistently obtain the same optimal value in repeated experiments. The hit rates of DBRO and DMBO are better than the other three algorithms, while the running speed follows the order: GA > DOA > DBRO > DMBO > FOA. In summary, DBRO outperforms the other four algorithms regarding solution quality. Regarding running efficiency, DBRO is superior to DMBO and FOA, while there is not much difference with GA and DOA. Therefore, DBRO is more suitable for solving MRPM.

V. Conclusion

The optimization of multi-factory remanufacturing processes is a prominent research topic within supply chain management. In this study, we introduce the MRPM problem for the first time and formulate it as a mixed-integer program with the objective of maximizing profit. To solve this problem, DBRO is proposed, featuring a novel encoding scheme to represent candidate solutions. Moreover, four soldier search strategies are integrated into DBRO to enhance solution exploration and avoid entrapment in local optima. IBM CPLEX is employed to solve the model and verify its validity. The performance of DBRO is benchmarked against four other intelligent optimization algorithms, showing superior efficiency in addressing this problem.

Our future plans involve 1) incorporating other factors into the problem, e.g., a human worker's physical exertion and fatigue index; and 2) exploring other intelligent optimization algorithms and reinforcement learning to cope with the concerned problems [52].

References

[1] Q. Zhang, Y. Xing, C. Zhang, X. Sun, B. Hu, and A. Das, "Column generation algorithms for two-dimensional cutting problem with surface defects," *International Journal of Artificial Intelligence and Green Manufacturing*, vol. 1, no. 2, pp. 23–35, June 2025.

[2] L. Zhou, H. Zhu, and B. Akbari, "Multi-objective optimization of multi-factory remanufacturing process considering worker fatigue," *International Journal of Artificial Intelligence and Green Manufacturing*, vol. 1, no. 2, pp. 36–50, June 2025.

[3] H. Zhang, D. Pham, and Q. Kang, "Improved fruit fly algorithm for multi-objective disassembly line balancing problem considering learning effect," *International Journal of Artificial Intelligence and Green Manufacturing*, vol. 1, no. 2, pp. 51–62, June 2025.

[4] C. Xu, H. Wei, X. Guo, S. Liu, L. Qi, and Z. Zhao, "Human-robot collaboration multi-objective disassembly line balancing subject to task failure via multi-objective artificial bee colony algorithm," *IFAC-PapersOnLine*, vol. 53, no. 5, pp. 1–6, 2020.

[5] H. K. Chan, S. H. Chung, and T. M. Chan, "Combining genetic approach and integer programming to solve multi-facility economic lot-scheduling problem," *Journal of Intelligent Manufacturing*, vol. 23, pp. 2397–2405, 2012.

[6] F. Zhao, R. Ma, and L. Wang, "A self-learning discrete jaya algorithm for multiobjective energy-efficient distributed no-idle flow-shop scheduling problem in heterogeneous factory system," *IEEE Transactions on Cybernetics*, vol. 52, no. 12, pp. 12 675–12 686, 2021.

[7] J. Behnamian and S. Fatemi Ghomi, "A survey of multi-factory scheduling," *Journal of Intelligent Manufacturing*, vol. 27, pp. 231–249, 2016.

[8] J. Behnamian and S. F. Ghomi, "The heterogeneous multi-factory production network scheduling with adaptive communication policy and parallel machine," *Information Sciences*, vol. 219, pp. 181–196, 2013.

[9] A. M. Khedr and W. Osamy, "Minimum perimeter coverage of query regions in a heterogeneous wireless sensor network," *Information Sciences*, vol. 181, no. 15, pp. 3130–3142, 2011.

[10] X. Zhang, W. K. Cheung, and C. Li, "Learning latent variable models from distributed and abstracted data," *Information Sciences*, vol. 181, no. 14, pp. 2964–2988, 2011.

[11] Z. Zhang, X. Guo, M. Zhou, S. Liu, and L. Qi, "Multi-objective discrete grey wolf optimizer for solving stochastic multi-objective disassembly sequencing and line balancing problem," in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2020, pp. 682–687.

[12] M. Ziaee, "A heuristic algorithm for the distributed and flexible job-shop scheduling problem," *The Journal of Supercomputing*, vol. 67, pp. 69–83, 2014.

[13] H.-C. Chang and T.-K. Liu, "Optimisation of distributed manufacturing flexible job shop scheduling by using hybrid genetic algorithms," *Journal of Intelligent Manufacturing*, vol. 28, pp. 1973–1986, 2017.

[14] L. Qi, M. Li, X. Guo, and W. Luan, "Multi-objective optimization for robotaxi dispatch with safety-carpooling mode in pandemic era," *IEEE Transactions on Intelligent Transportation Systems*, vol. 26, no. 1, pp. 878–891, 2024.

[15] H. Jia, A. Y. Nee, J. Y. Fuh, and Y. Zhang, "A modified genetic algorithm for distributed scheduling problems," *Journal of Intelligent Manufacturing*, vol. 14, pp. 351–362, 2003.

[16] X. Guo, Z. Bi, J. Wang, S. Qin, S. Liu, and L. Qi, "Reinforcement learning for disassembly system optimization problems: A survey," *International Journal of Network Dynamics and Intelligence*, pp. 1–14, 2023.

[17] S. Qin, J. Li, J. Wang, X. Guo, S. Liu, and L. Qi, "A salp swarm algorithm for parallel disassembly line balancing considering workers with government benefits," *IEEE Transactions on Computational Social Systems*, 2023.

[18] S. Dai, Z. Zhang, W. Wang, C. Li, J. Parron, and E. Herrera, "Human-robot collaborative disassembly profit maximization via improved grey wolf optimizer," *International Journal of Artificial Intelligence and Green Manufacturing*, vol. 1, no. 2, pp. 12–22, 2025.

[19] X. Cui, X. Guo, M. Zhou, J. Wang, S. Qin, and L. Qi, "A discrete whale optimization algorithm for disassembly line balancing with carbon emission constraint," *IEEE Robotics and Automation Letters*, 2023.

[20] X. Guo, T. Wei, J. Wang, S. Liu, S. Qin, and L. Qi, "Multiobjective u-shaped disassembly line balancing problem considering human fatigue index and an efficient solution," *IEEE Transactions on Computational Social Systems*, 2022.

[21] S. Dai, Y. Feng, Z. Zhang, X. Guo, S. Qin, Q. Kang, and Y. Liu, "Solving disassembly and assembly line balancing problem with robot direction switching," in *2025 37th Chinese Control and Decision Conference (CCDC)*, IEEE. Xiamen, China: IEEE, may 2025, pp. 896–901.

[22] Y. Feng, S. Dai, Z. Zhang, X. Guo, S. Qin, Q. Kang, and Y. Liu, "A disassembly and assembly line balancing problem via an improved double q-learning," in *2025 37th Chinese Control and Decision Conference (CCDC)*. IEEE, 2025, pp. 890–895.

[23] C. B. Kalayci, A. Hancilar, A. Gungor, and S. M. Gupta, "Multi-objective fuzzy disassembly line balancing using a hybrid discrete artificial bee colony algorithm," *Journal of Manufacturing Systems*, vol. 37, pp. 672–682, 2015, reverse Supply Chains.

[24] S. Qin, S. Zhang, J. Wang, S. Liu, X. Guo, and L. Qi, "Multi-objective multi-verse optimizer for multi-robotic u-shaped disassembly line balancing problems," *IEEE Transactions on Artificial Intelligence*, 2023.

[25] Z. Li, I. Kucukkoc, and Z. Zhang, "Iterated local search method and mathematical model for sequence-dependent u-shaped disassembly line balancing problem," *Computers Industrial Engineering*, vol. 137, p. 106056, 2019.

[26] F.-A. G. La Forme, V. B. Genoulaz, and J.-P. Campagne, "A framework to analyse collaborative performance," *Computers in Industry*, vol. 58, no. 7, pp. 687–697, 2007.
[27] C. Blum and C. Miralles, "On solving the assembly line worker assignment and balancing problem via beam search," *Computers & Operations Research*, vol. 38, no. 1, pp. 328–339, 2011.
[28] T. Zaman, S. K. Paul, and A. Azeem, "Sustainable operator assignment in an assembly line using genetic algorithm," *International Journal of Production Research*, vol. 50, no. 18, pp. 5077–5084, 2012.
[29] M. K. Oksuz, K. Buyukozkan, and S. I. Satoglu, "U-shaped assembly line worker assignment and balancing problem: A mathematical model and two meta-heuristics," *Computers & Industrial Engineering*, vol. 112, pp. 246–263, 2017.
[30] I. Belassiria, M. Mazouzi, S. ELfezazi, A. Cherrafi, and Z. ELMaskaoui, "An integrated model for assembly line re-balancing problem," *International Journal of Production Research*, vol. 56, no. 16, pp. 5324–5344, 2018.
[31] S. M. McGovern and S. M. Gupta, "A balancing method and genetic algorithm for disassembly line balancing," *European journal of operational research*, vol. 179, no. 3, pp. 692–708, 2007.
[32] Y. Feng, M. Zhou, G. Tian, Z. Li, Z. Zhang, Q. Zhang, and J. Tan, "Target disassembly sequencing and scheme evaluation for cnc machine tools using improved multiobjective ant colony algorithm and fuzzy integral," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 12, pp. 2438–2451, 2018.
[33] Y. Fu, M. Zhou, X. Guo, L. Qi, and K. Sedraoui, "Multiverse optimization algorithm for stochastic biobjective disassembly sequence planning subject to operation failures," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 2, pp. 1041–1051, 2021.
[34] X. Guo, C. Fan, M. Zhou, S. Liu, J. Wang, S. Qin, and Y. Tang, "Human–robot collaborative disassembly line balancing problem with stochastic operation time and a solution via multi-objective shuffled frog leaping algorithm," *IEEE Transactions on Automation Science and Engineering*, 2023.
[35] T. Rahkar Farshi, "Battle royale optimization algorithm," *Neural Computing and Applications*, vol. 33, no. 4, pp. 1139–1157, 2021.
[36] Z. Cao, C. Lin, M. Zhou, C. Zhou, and K. Sedraoui, "Two-stage genetic algorithm for scheduling stochastic unrelated parallel machines in a just-in-time manufacturing context," *IEEE Transactions on Automation Science and Engineering*, vol. 20, no. 2, pp. 936–949, 2022.
[37] E. Duman, M. Uysal, and A. F. Alkaya, "Migrating birds optimization: a new metaheuristic approach and its performance on quadratic assignment problem," *Information Sciences*, vol. 217, pp. 65–77, 2012.
[38] H. Peraza-Vázquez, A. F. Peña-Delgado, G. Echavarría-Castillo, A. B. Morales-Cepeda, J. Velasco-Álvarez, and F. Ruiz-Perez, "A bio-inspired method for engineering design optimization inspired by dingoes hunting strategies," *Mathematical Problems in Engineering*, vol. 2021, pp. 1–19, 2021.
[39] Y. Fu, M. Zhou, X. Guo, and L. Qi, "Stochastic multi-objective integrated disassembly-reprocessing-reassembly scheduling via fruit fly optimization algorithm," *Journal of Cleaner Production*, vol. 278, p. 123364, 2021.
[40] J. Wang, "Petri nets for dynamic event-driven system modeling." *Handbook of Dynamic System Modeling*, vol. 1, p. 24, 2007.
[41] ——, "Charging information collection modeling and analysis of GPRS networks," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 4, pp. 473–481, 2007.
[42] R. J. Riggs, O. Battaïa, and S. J. Hu, "Disassembly line balancing under high variety of end of life states using a joint precedence graph approach," *Journal of Manufacturing Systems*, vol. 37, pp. 638–648, 2015.
[43] J. Wang and D. Li, "Resource oriented workflow nets and workflow resource requirement analysis," *International Journal of Software Engineering and Knowledge Engineering*, vol. 23, no. 05, pp. 677–693, 2013.
[44] Z. Diri Kenger, Ç. Koç, and E. Özceylan, "Integrated disassembly line balancing and routing problem," *International Journal of Production Research*, vol. 58, no. 23, pp. 7250–7268, 2020.
[45] M. L. Bentaha, A. Dolgui, O. Battaïa, R. J. Riggs, and J. Hu, "Profit-oriented partial disassembly line design: dealing with hazardous parts and task processing times uncertainty," *International Journal of Production Research*, vol. 56, no. 24, pp. 7220–7242, 2018.
[46] T. Akan, S. Agahian, and R. Dehkharghani, "Binbro: Binary battle royale optimizer algorithm," *Expert Systems with Applications*, vol. 195, p. 116599, 2022.
[47] S. Akan and T. Akan, "Battle royale optimizer with a new movement strategy," in *Handbook of Nature-Inspired Optimization Algorithms: The State of the Art: Volume I: Solving Single Objective Bound-Constrained Real-Parameter Numerical Optimization Problems*. Springer, 2022, pp. 265–279.
[48] T. Akan, S. Agahian, and R. Dehkharghani, "Battle royale optimizer for solving binary optimization problems," *Software Impacts*, vol. 12, p. 100274, 2022.
[49] P. Nowakowski, "A novel, cost efficient identification method for disassembly planning of waste electrical and electronic equipment," *Journal of Cleaner Production*, vol. 172, pp. 2695–2707, 2018.
[50] M. L. Bentaha, O. Battaïa, and A. Dolgui, "A sample average approximation method for disassembly line balancing problem under uncertainty," *Computers & Operations Research*, vol. 51, pp. 111–122, 2014.
[51] Q. Lu, Y. Ren, H. Jin, L. Meng, L. Li, C. Zhang, and J. W. Sutherland, "A hybrid metaheuristic algorithm for a profit-oriented and energy-efficient disassembly sequencing problem," *Robotics and Computer-Integrated Manufacturing*, vol. 61, p. 101828, 2020.
[52] Y. Tian, G. Liu, J. Wang, and M. Zhou, "ASA-GNN: Adaptive sampling and aggregation-based graph neural network for transaction fraud detection," *IEEE Transactions on Computational Social Systems*, vol. 11, no. 3, pp. 3536–3549, 2024.

Ziyan Zhao (Member, IEEE) Received his B.s, M.s., and Ph.D. degrees in 2015, 2017, and 2021, respectively, from the College of Information Science and Engineering, Northeastern University, Shenyang, China, where he is currently working as an Assistant Professor. From October 2018 to October 2020, he studied as a visiting Ph.D. student in the Department of Electrical and Computer Engineering, New jersey Institute of Technology, Newark,NJ, UsA. His research focuses on intelligent manufacturing, intelligent optimization algorithm, industrial big data, and production planning and scheduling. Till now, he has published over 30 international journal and conference papers in the above areas. His research focuses on intelligent decision-making, intelligent warehousing, and production planning and scheduling. Till now, he has published over 30 papers in the above areas.

LiangBo Zhou Graduated from Jiangsu University of Science and Technology in 2022 with a bachelor's degree in Internet of Things Engineering. Graduated from Liaoning Petrochemical University in 2025 with a master's degree in Artificial Intelligence. Currently working in the Development and Reform Bureau of Congjiang County, Congjiang, China.