Hindawi
International Journal of Intelligent Systems
Volume 2023, Article ID 8847164, 47 pages
https://doi.org/10.1155/2023/8847164

Research Article

A New Pareto Discrete NSGAII Algorithm for Disassembly Line Balance Problem

ZhenYu Xu,[1] Yong Han,[1,2] ZhenXin Li,[1] YiXin Zou,[1] and YuWei Chen[1]

[1]*Faculty of Information Science and Engineering, Ocean University of China, No. 238, Songling Road, Qingdao 266100, China*
[2]*Laboratory for Regional Oceanography and Numerical Modeling, Qingdao National Laboratory for Marine Science and Technology, No. 1, Wenhai Road, Qingdao 266237, China*

Correspondence should be addressed to Yong Han; yonghan@ouc.edu.cn

Received 12 June 2023; Revised 7 November 2023; Accepted 27 November 2023; Published 18 December 2023

Academic Editor: Said El Kafhali

With the increasing variety and quantity of end-of-life (EOL) products, the traditional disassembly process has become inefficient. In response to this phenomenon, this article proposes a random multiproduct U-shaped mixed-flow incomplete disassembly line balancing problem (MUPDLBP). MUPDLBP introduces a mixed disassembly method for multiple products and incomplete disassembly method into the traditional DLBP, while considering the characteristics of U-shaped disassembly lines and the uncertainty of the disassembly process. First, mixed-flow disassembly can improve the efficiency of disassembly lines, reducing factory construction and maintenance costs. Second, by utilizing the characteristics of incomplete disassembly to reduce the number of dismantled components and the flexibility and efficiency of U-shaped disassembly lines in allocating disassembly tasks, further improvement in disassembly efficiency can be achieved. In addition, this paper also addresses the characteristics of EOL products with heavy weight and high rigidity. While retaining the basic settings of MUPDLBP, the stability of the assembly during the disassembly process is considered, and a new problem called MUPDLBP_S, which takes into account the disassembly stability, is further proposed. The corresponding mathematical model is provided. To obtain high-quality disassembly plans, a new and improved algorithm called INSGAII is proposed. The INSGAII algorithm uses the initialization method based on Monte Carlo tree simulation (MCTI) and the Group Global Crowd Degree Comparison (GCDC) operator to replace the initialization method and crowding distance comparison operator in the NSGAII algorithm, effectively improving the coverage of the initial population individuals in the entire solution space and the evenness and spread of the Pareto front. Finally, INSGAII's effectiveness has been affirmed by tackling both current disassembly line balancing problems and the proposed MUPDLBP and MUPDLBP_S. Importantly, INSGAII outshines six comparison algorithms with a top rank of 1 in the Friedman test, highlighting its superior performance.

1. Introduction

With the rapid development of technology, the speed of product updates is increasing, resulting in the generation of a large number of end-of-life (EOL) products. If EOL products are left unattended, it will bring significant economic and environmental pressures to human society. Therefore, it is necessary to recycle and reuse EOL products. Disassembly is one of the important steps in achieving the recycling and reuse of products. Disassembly lines are the optimal choice for large-scale disassembly, and achieving disassembly line balancing is the key to improve disassembly efficiency. Therefore, the disassembly line balancing problem (DLBP) has become a highly regarded issue among engineers and scholars.

On the one hand, reducing disassembly costs and improving disassembly efficiency are the key optimization goals of the DLBP. In earlier research, these goals were often achieved by setting certain optimization objectives, such as minimizing the number of workstations to reduce disassembly costs and improving the balance index to enhance disassembly efficiency [1, 2]. However, as technology advances, the products encountered during the disassembly process are becoming increasingly complex. The traditional approach of solely setting optimization goals to reduce disassembly costs and improve efficiency is becoming less

effective. Researchers have discovered that in addition to setting reasonable optimization goals, it is necessary to address the current complexity by changing the way disassembly lines are used, the layout of disassembly lines, and the disassembly methods. For example, in earlier studies, it was commonly believed that disassembly lines could only dismantle one type of end-of-life (EOL) product within a certain period of time [3–5]. However, with the increasing variety and quantity of industrial products, the traditional approach of using disassembly lines is no longer efficient. More and more scholars and engineers believe that mixed-flow disassembly lines, capable of handling multiple products, are urgently needed. In academia, Fang et al. [6] have pointed out that mixed-flow disassembly lines meet the disassembly needs of products and their similar variant products, reducing factory construction and maintenance costs. Guo et al. [7] have indicated that existing research on DLBP mostly focuses on a single product, which is not able to meet practical demands. Furthermore, they also mentioned that achieving balance in mixed-flow disassembly lines is highly challenging as it requires satisfying priority constraints for each product and optimizing predefined objectives to determine the disassembly scheme for each product. In the industry, Apple has developed a disassembly line called Daisy, which efficiently disassembles variants of 9 iPhone models [8], thus confirming the practicality of mixed-flow disassembly lines. Apart from the way the disassembly line is used, the layout of the disassembly line also affects the disassembly efficiency. Among various layout types, the U-shaped layout of a disassembly line has the characteristics of flexibility and efficiency, which can effectively enhance disassembly efficiency. For example, Li et al. [9] have pointed out that compared to a straight-line disassembly line, the U-shaped layout allows for more combinations of task assignments to different workstations, enabling the U-shaped disassembly line to effectively improve disassembly efficiency. Wang et al. [10] have also mentioned that workers on a U-shaped production line can work simultaneously on both sides of the production line, increasing production efficiency, reducing floor space, and significantly shortening product completion time. The aforementioned studies all demonstrate that the U-shaped layout is a highly competitive disassembly line layout method that can effectively improve disassembly efficiency. In addition, with the application of large crushers and sorting devices in the disassembly field, the traditional method of disassembling all components is gradually being abandoned by the industry. Instead, there is a growing focus on partial disassembly, which only requires the removal of parts with desired attributes and parts with hazardous attributes. Yin et al. [11] proposed a multimanned partial disassembly line balancing problem (MP-DLBP), where only parts with hazardous or desired attributes are required to be disassembled, while the remaining parts can be disassembled or not. Yin pointed out that partial disassembly can avoid unnecessary labor. Li et al. [12] introduced a profit-oriented U-shaped partial disassembly line balancing problem (PUPDLBP), which can greatly improve disassembly profitability. However, the PUPDLBP does not consider the requirement to dismantle parts with hazardous attributes. Wang et al. [13], in their 2019 research, analyzed the obtained disassembly schemes and concluded that as the number of disassembly tasks increases, the disassembly profit decreases, further highlighting the effectiveness of partial disassembly. Based on the above analysis, it can be seen that mixed-flow disassembly lines, U-shaped layout disassembly lines, and partial disassembly methods all contribute to improving disassembly efficiency and reducing disassembly costs. Therefore, in this study, the characteristics of mixed-flow disassembly lines, U-shaped layout disassembly lines, and partial disassembly methods were comprehensively considered in the design of the disassembly line.

On the other hand, apart from reducing disassembly costs and improving disassembly efficiency, it is also necessary to ensure that the disassembly scheme is as applicable as possible to the actual disassembly situation. This is to ensure that the obtained disassembly scheme can achieve a relatively ideal balance for the disassembly line. In the actual disassembly process, factors such as the degree of wear and tear of the obsolete products can introduce a certain level of randomness to the disassembly time of the components. If the randomness of the disassembly time is not taken into account, it will make it difficult for the disassembly line to achieve the desired balance. Therefore, some researchers have pointed out that considering the randomness of the disassembly time can better reflect the actual disassembly situation [14, 15]. In addition, when disassembling products with heavy weight and high rigidity characteristics, it is important to maintain the stability of the assembly to ensure that it does not spontaneously disintegrate or cause relative positional changes between internal components due to gravity or other disassembly operations during the disassembly process. This can effectively avoid injuries to disassembly personnel caused by the collapse of product components during the disassembly process. Therefore, it is necessary to consider the randomness of disassembly time and the stability of disassembly in the DLBP.

In summary, in the field of DLBP, it is beneficial to consider the characteristics of mixed-flow disassembly lines, U-shaped layout disassembly lines, and partial disassembly methods in order to improve disassembly efficiency and reduce disassembly costs. Therefore, in this research work, we first integrated the influences of multiproduct mixed-flow disassembly factors, U-shaped disassembly line factors, and partial disassembly methods on the DLBP. Second, we considered the uncertainties in the disassembly process and proposed a random multiobjective multiproduct U-shaped mixed-flow incomplete disassembly line balancing problem (MUPDLBP), which is specifically designed for the disassembly process of common household appliances. Furthermore, to make this research more valuable, we also took into account the stability of the assembly during the disassembly process, considering the characteristics of EOL products with heavy weight and high rigidity. This led to the proposal of a random multiobjective multiproduct U-shaped mixed-flow disassembly line balancing problem considering disassembly stability (MUPDLBP_S). Compared to existing research, this paper takes into account a more comprehensive set of factors, making it more applicable to the practical needs of current and future disassembly processes.

In the field of DLBP, it is also important to provide solution methods for the DLBP. The solution methods for the DLBP can be divided into exact solution methods and approximate solution methods. Although exact solution methods can obtain high-precision solutions, the DLBP is an NP-complete problem [16], which means that the difficulty of solving it increases exponentially with the problem size. Therefore, exact solution methods are not practical when dealing with large-scale DLBP. Metaheuristic algorithms are widely used approximate solution methods that strike a good balance between solution time and accuracy. They are widely applied in DLBP, such as ant colony algorithms [17], artificial bee colony algorithms [18], variable neighborhood search algorithms [19], and artificial fish swarm algorithms [20]. In the field of metaheuristic algorithms, NSGA-II is widely used in process optimization, workshop scheduling, and other areas due to its advantages of simple implementation and strong search capability and has achieved remarkable results [21–23]. Given the advantages of NSGA-II, some scholars have also used it to handle disassembly line balancing problems. For example, Yılmaz et al. [24] used the NSGA-II algorithm to solve disassembly line balancing problems with two different worker allocation strategies and achieved excellent results. Additionally, through Yılmaz et al.'s research, it was found that using the NSGA-II algorithm to solve DLBP requires a reasonable design of crossover and mutation operators. In their study, they designed the crossover and mutation processes reasonably and added a step of repairing genes based on the characteristics of their problems. Due to the complex precedence constraints between components of waste products, traditional crossover and mutation operators are difficult to handle U-shaped disassembly lines. In order to apply NSGA-II to U-shaped disassembly line balancing problems, we propose a new two-stage crossover (TSCO) and mutation operator (TSMO) to ensure that the newly generated solutions are feasible. Furthermore, in metaheuristic algorithms, the initial population is often generated through random initialization. In DLBP, due to the influence of complex precedence constraints between disassembly tasks, the probability of feasible disassembly sequences varies, resulting in insufficient coverage of the solution space by the randomly initialized individuals. To avoid this situation, we propose a new initialization method based on Monte Carlo tree simulation (MCTI). The crowding distance comparison operator in NSGA-II excessively protects boundary individuals, resulting in poor diversity and uniformity of the Pareto solution set when dealing with DLBP. In order to improve the diversity and uniformity of the Pareto solution set, we have designed a multilevel global crowding distance comparison (GCDC) operator, which greatly enhances the diversity and uniformity of the Pareto solution set.

The main contributions of this article are summarized as follows:

(1) We propose a random multiobjective multiproduct U-shaped mixed-flow incomplete disassembly line balancing problem (MUPDLBP) considering disassembly stability. In MUPDLBP, we integrate various factors that affect the disassembly process, including mixed-flow disassembly factors, U-shaped disassembly line factors, partial disassembly methods, and uncertainty during disassembly. Compared to the existing research study, this article considers more comprehensive factors that are more applicable to the actual requirements of current and future disassembly processes.

(2) While retaining the basic settings of MUPDLBP, we further consider the stability of assemblies during the disassembly process, especially for EOL products with large weight and rigidity. Therefore, we propose a random multiobjective multiproduct U-shaped mixed-flow disassembly line balancing problem considering disassembly stability (MUPDLBP_S).

(3) An initialization method based on Monte Carlo tree simulation (MCTI) and a Group Global Crowd Degree Comparison (GCDC) operator was proposed. By using MCTI and GCDC to replace the initialization method and crowding comparison operator in the NSGAII algorithm, respectively, the coverage of the entire solution space by the initial population and the evenness and spread of the Pareto frontier were effectively improved.

(4) The INSGAII algorithm's superiority has been proven through extensive experiments. When dealing with disassembly line balancing problems, managers can use the INSGAII algorithm to obtain high-quality disassembly solutions.

(5) In Section 5 of this paper, the effectiveness of U-shaped disassembly lines and mixed-flow disassembly lines in improving disassembly efficiency was analyzed and confirmed. It was also demonstrated that the U-shaped mixed disassembly line, which combines the characteristics of U-shaped and mixed-flow disassembly lines, is superior. Managers can consider designing disassembly lines as U-shaped mixed disassembly lines.

The rest of this article is organized as follows. Section 2 describes the proposed disassembly line balancing problem. Section 3 introduces the proposed INSGA-II algorithm. In Section 4, we present the experimental results and analysis. In Section 5, we analyze the obtained disassembly schemes and provide management suggestions. Finally, in Section 6, we summarize the article and discuss future research directions.

2. Problem Statement

In the present scenario, common discarded household appliances have the characteristics of diverse product types, small component quality, low rigidity of components, and a wide range of product lifespans with varying usage conditions. In order to further improve disassembly efficiency and reduce disassembly costs, we propose a random multiobjective multiproduct U-shaped mixed-flow incomplete disassembly line balancing problem (MUPDLBP) specifically for common discarded household appliances. MUPDLBP has the following characteristics:

(1) Mixed-flow disassembly: With the increasing variety of product types in common discarded household

appliances, the traditional approach of assigning one disassembly line to each product is becoming less economically efficient. As a result, mixed-flow disassembly lines have gained attention, leading to the problem of balancing mixed-flow disassembly lines. The main difference between the mixed-flow disassembly line balancing problem and the traditional single-product disassembly line balancing problem is that the former requires different types of discarded products to be disassembled on the same line while achieving line balancing, whereas the latter assigns one product to one disassembly line. Solving the mixed-flow disassembly line balancing problem is more challenging compared to the single-product disassembly line balancing problem, but it aligns better with future development trends [7].

(2) Incomplete disassembly: Due to the small component size and low rigidity of common discarded household appliances, there are a large number of components that can be crushed by crushers and automatically sorted. This makes partial disassembly methods feasible, and with the introduction of automated crushers and sorting devices, partial disassembly has gradually become the mainstream approach. In partial disassembly, only the removal of necessary and hazardous components is required, while the remaining components can be disassembled based on optimization objectives.

(3) Uncertainty in the disassembly process: The wide range of product lifespans and varying usage conditions of common household appliances make it difficult to determine the degree of product aging and the difficulty of disassembly. As a result, the disassembly time becomes uncertain due to these uncertain factors. Consider that the uncertainty in the disassembly process is essential to achieve an ideal balance in the disassembly line.

(4) U-shaped layout for disassembly lines: In a U-shaped disassembly line, tasks can be assigned simultaneously to the entrance and exit sides of the line, providing higher flexibility and production efficiency [9]. Therefore, U-shaped disassembly lines have gained increasing attention.

Unlike common discarded household appliances, EOL (end-of-life) products with large weight and rigidity have components that are difficult to be processed by crushers and automatically sorted due to their large weight and rigidity. Therefore, incomplete disassembly methods are usually not applicable to these EOL products. Additionally, due to the large weight and rigidity of the components, the collapse of components during disassembly can cause serious injuries to disassembly personnel. Therefore, considering the stability of the disassembly process is necessary when dealing with such products. Based on the characteristics of EOL products with large weight and rigidity, we consider the stability of assemblies in MUPDLBP and exclude certain disassembly methods, leading to the proposal of a random multiobjective multiproduct U-shaped mixed-flow incomplete disassembly line balancing problem considering disassembly stability (MUPDLBP_S).

2.1. List of Assumptions. To facilitate the mathematical description of the problem, this paper provides the following list of assumptions: (1) The disassembly time follows a normal distribution as a random variable. (2) The upper limit of the cycle time for the disassembly line is known. (3) The components of the product to be disassembled are intact and cannot be further divided. (4) There is an abundant quantity of the products, and interruptions in the production line are ignored. (5) The walking time of the personnel is disregarded. (6) The transportation time of the parts on the conveyor belt is negligible. (7) The impact of disassembly tools is ignored.

2.2. Notation Definition. The symbols used in this article are as follows:

k: index of workstation, $k \in \{1, 2, \ldots, K\}$, where K is the total number of workstations.

m: product index, $m \in \{1, 2, \ldots, N_{\text{ap}}\}$, where N_{ap} represents the number of disassembled products.

i, j: task index, $i, j \in \{1, 2, \ldots, N_m\}$, where N_m is the total number of disassembly tasks for product m.

N_{to}: sum of all product disassembly tasks.

t_{mi}: the disassembly time of the i th disassembly task of the m-type product.

μ_{mi}: the mean of the disassembly time of task i of the m-type product.

σ^2_{mi}: the standard deviation of the disassembly time of task i of the m-type product.

ϕ: the standard normal distribution function.

T_U: the upper limit of cycle time.

T_k: disassembly time of Workstation k.

α: the probability that the disassembly time of a station does not exceed T_U.

T_c: the actual cycle time of the disassembly line, which is the maximum of the disassembly time of all workstations.

$P_m(i)$: the immediately preceding task set of task i belonging to product m.

$S_m(i)$: the immediately subsequent task set of task i of product m.

$\text{SP}_m(i)$: the shadow immediately preceding task set of task i belonging to product m.

$\text{SSP}_m(i)$: the shadow immediately subsequent task set of task i of product m.

TP_m: the priority relationship matrix for product m.

STP_m: the shadow priority relationship matrix for product m.

$\mathbf{I}_m$: the disassembly interference matrix for product m.

$\mathbf{I}_{+xm}$: the disassembly interference matrix for product m in the $+x$ direction.

$\mathbf{I}_{+ym}$: the disassembly interference matrix for product m in the $+y$ direction.

$\mathbf{I}_{+zm}$: the disassembly interference matrix for product m in the $+z$ direction.

$\mathbf{I}_{-xm}$: the disassembly interference matrix for product m in the $-x$ direction.

$\mathbf{I}_{-ym}$: the disassembly interference matrix for product m in the $-y$ direction.

$\mathbf{I}_{-zm}$: the disassembly interference matrix for product m in the $-z$ direction.

The detailed information about TP_m, STP_m, $\mathbf{I}_{+xm}$, $\mathbf{I}_{+ym}$, $\mathbf{I}_{+zm}$, $\mathbf{I}_{-xm}$, $\mathbf{I}_{-ym}$, and $\mathbf{I}_{-zm}$ will be presented in Section 2.3.1.

d_{mi}: demand attribute, 1 if task i of product m has demand or remanufacturing value, 0 otherwise.

h_{mi}: hazardous attribute, 1 if task i of product m is harmful, 0 otherwise.

S: the disassembly sequence, where each element in S is a vector that contains the corresponding product number and disassembly task number. The lth element in S is denoted as S_l, and the corresponding product number and disassembly task number in S_l are denoted as S_{l0} and S_{l1}, respectively. The position of the elements in S represents the execution order of the disassembly tasks for the corresponding products. For example, a disassembly sequence S could be $\{[1,4],[3,1],\ldots,[m,i],\ldots,[1,10]\}$, where the element $[3,2]$ represents that the first disassembly task of product 3 is the second task to be executed among all tasks. Since this paper considers a U-shaped disassembly line layout, the disassembly sequence and encoding sequence are different. For more details about the difference between the disassembly sequence and encoding sequence, please refer to Section 3.1.1.

O_{mi}: the position of task i in product m in the disassembly sequence; for example, if task 2 in product 1 is in the third position in the disassembly sequence, then $O_{12} = 3$.

C^m: the connection matrix for product m, which reflects the interconnection relationship between components in product m. For more information about C^m, please refer to Section 2.4.5.

c_{ij}^m: c_{ij}^m reflects the connection relationship between component i and component j in product m. For more information about c_{ij}^m, refer to Section 2.4.5.

v_i^m: the stability of component i in product m during the disassembly process. For more information about v_i^m, refer to Section 2.4.5.

V_S: the stability of disassembly sequence S. For more information about V_S, refer to Section 2.4.5.

D_i^m: the set of feasible disassembly directions for component i in product m. For more information about D_i^m, refer to Section 2.4.6.

d_i^m: the disassembly direction of component i in product m, where $d_i^m \in \{+x,+y,+z,-x,-y,-z\}$.

N_{sdd}: the number of maximum subsequence of disassembly in the same disassembly direction (MSSDD). For more information about N_{sdd}, refer to Section 2.4.6.

s_k: if workstation k is opened, $s_k = 1$; otherwise, $s_k = 0$.

x_{mik}^a: if the i th task of the m th product is assigned to the entrance side of the k th workstation, $x_{\mathrm{mik}}^a = 1$; otherwise, $x_{\mathrm{mik}}^a = 0$.

x_{mik}^b: if the i th task of the m th product is assigned to the exit side of the k th workstation, then $x_{\mathrm{mik}}^b = 1$; otherwise, $x_{\mathrm{mik}}^b = 0$.

2.3. Problem Characteristics

2.3.1. Priority Relationship under U-Shaped Layout Conditions. When dealing with the disassembly line balancing problem (DLBP), the first thing to consider is the priority constraint relationship of the disassembly tasks. Only disassembly sequences that satisfy the correct priority constraints are usable. Disassembling without considering the priority constraints of the product may result in parts not being smoothly removed or damaged. Therefore, accurately expressing the priority constraint relationship between disassembled product parts/components is crucial. In the field of DLBP, priority relationship graphs are commonly used to describe the priority constraint relationship of disassembly tasks. Currently, the mainstream priority relationship graphs can be roughly divided into three types: task precedence diagram (TPD) [25], transform AND/OR graph (TAOG) [25], and part precedence diagram (PPD) [26]. Compared to TAOG, PPD does not require the product's components to have a complete structure and normal connection, making it easier to express large-scale products. Compared to TPD, PPD can fully retain the removal status information of the product. Therefore, in many ordinary DLBP cases, PPD is more practical than TAOG and TPD [27]. It is worth noting that although the disassembly interference matrix, which can effectively reflect the priority constraint relationship between disassembly tasks, is not widely used in the field of DLBP, it has the advantage of being able to generate automatically without the need for manual analysis and input of constraint information. With the three-dimensional model information of the assembly, the interference matrix can be quickly and automatically obtained. Considering the wide application of part precedence diagram (PPD) and the advantage of automatic generation of the disassembly interference matrix in the field of DLBP, this study describes the priority constraint relationship of disassembly tasks using PPD and the disassembly interference matrix, respectively. It is worth noting that due to the diverse types of ordinary household appliances and the large span of their usage time, disassembly companies are unable to establish three-dimensional models for all ordinary household appliances. Therefore, the interference matrix cannot exert its unique advantages in the process of disassembling ordinary household appliances.

Therefore, PPD is used to describe the priority constraint relationship between disassembly tasks in MUPDLBP, while the interference matrix is used to describe the priority constraint relationship between disassembly tasks in MUPDLBP_S.

PPD and the disassembly interference matrix both treat individual parts or components as disassembly tasks, so this paper will no longer make a strict distinction between disassembly tasks and parts/components, assuming that disassembly tasks are equivalent to parts/components.

Next, we will introduce how to use PPD and the interference matrix, respectively, to describe the priority constraint relationship between disassembly tasks.

(1) Disassembly Task Priority Relationship Based on PPD. Figure 1 shows the PPD of products with 8 disassembly tasks, 6 disassembly tasks, and 5 disassembly tasks, respectively. In Figure 1, the numbers represent disassembly tasks, which are also the part numbers of the product to be disassembled. The tasks connected by arrows have a priority constraint relationship. The task at the arrow's tail needs to be executed first before the task at the arrow's head can be continued. In other words, the task at the arrow's tail is the immediate predecessor task of the task at the arrow's head.

In the process of dealing with DLBP, it is necessary to convert PPD into a priority relationship matrix for computer reading. For example, the PPD of a product m with n disassembly tasks can be converted into a priority relationship matrix TP_m. Equation (1) shows the priority relationship matrix of product m. $p_{ij} = 1$ indicates that task i is an immediate predecessor of task j, i.e., task i belongs to the immediate predecessor task set of task j ($i \in P_m(j)$). $p_{ij} = 1$ also indicates that task j is an immediate successor of task i, i.e., task j belongs to the immediate successor task set of task i ($j \in S_m(i)$).

$$TP_m = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 1 & \cdots & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & \cdots & 1 \\ 0 & 0 & 0 & 0 & \cdots & 1 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & p_{ij} & \cdots & 1 \\ \vdots & \vdots & \vdots & \vdots & \cdots & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & \cdots & 0 \end{bmatrix}. \tag{1}$$

Only after all the tasks in the immediate predecessor task set of task j are completed, can task j be executed. Therefore, when using PPD, the priority constraint relationship that needs to be satisfied between disassembly tasks can be represented by the following equation:

$$O_{mi} > O_{mj}, \qquad i \in P_m(j). \tag{2}$$

(2) Disassembly Task Priority Relationship Based on Disassembly Interference Matrix. The disassembly interference matrix can visually represent the spatial geometric constraint relationship between components. It can describe the collision situation between a component and other components when it is disassembled in a certain direction in the product to be disassembled. Specifically, it describes the collision situation between the component and other components during the movement along the six orthogonal axes ($\pm x$, $\pm y$, $\pm z$) of the absolute coordinate system of the product to be disassembled. When the 3D model information of the product to be disassembled is available, the disassembly interference matrix can be automatically generated using automated analysis methods [28, 29], which contributes to the full automation of the DLBP solving process. Additionally, compared to PPD, the disassembly interference matrix can reflect possible disassembly directions of components, which help reduce the number of disassembly direction changes during the DLBP solving process and further improves disassembly efficiency.

The disassembly interference matrix of a product to be disassembled, which consists of n components, is an n-by-n matrix. The interference matrix of the product to be disassembled can be represented by the following equation:

$$\mathbf{I}_m = \begin{bmatrix} \left[I_{11}^{+xm}, I_{11}^{+ym}, I_{11}^{+zm}, I_{11}^{-xm}, I_{11}^{-ym}, I_{11}^{-zm}\right] & \left[I_{12}^{+xm}, I_{12}^{+ym}, I_{12}^{+zm}, I_{12}^{-xm}, I_{12}^{-ym}, I_{12}^{-zm}\right] & \cdots & \left[I_{1n}^{+xm}, I_{1n}^{+ym}, I_{1n}^{+zm}, I_{1n}^{-xm}, I_{1n}^{-ym}, I_{1n}^{-zm}\right] \\ \left[I_{21}^{+xm}, I_{21}^{+ym}, I_{21}^{+zm}, I_{21}^{-xm}, I_{21}^{-ym}, I_{21}^{-zm}\right] & \left[I_{22}^{+xm}, I_{22}^{+ym}, I_{22}^{+zm}, I_{22}^{-xm}, I_{22}^{-ym}, I_{22}^{-zm}\right] & \cdots & \left[I_{2n}^{+xm}, I_{2n}^{+ym}, I_{2n}^{+zm}, I_{2n}^{-xm}, I_{2n}^{-ym}, I_{2n}^{-zm}\right] \\ \vdots & \vdots & \vdots & \vdots \\ \left[I_{n1}^{+xm}, I_{n1}^{+ym}, I_{n1}^{+zm}, I_{n1}^{-xm}, I_{n1}^{-ym}, I_{n1}^{-zm}\right] & \left[I_{n2}^{+xm}, I_{n2}^{+ym}, I_{n2}^{+zm}, I_{n2}^{-xm}, I_{n2}^{-ym}, I_{n2}^{-zm}\right] & \cdots & \left[I_{nn}^{+xm}, I_{nn}^{+ym}, I_{nn}^{+zm}, I_{nn}^{-xm}, I_{nn}^{-ym}, I_{nn}^{-zm}\right] \end{bmatrix}. \tag{3}$$

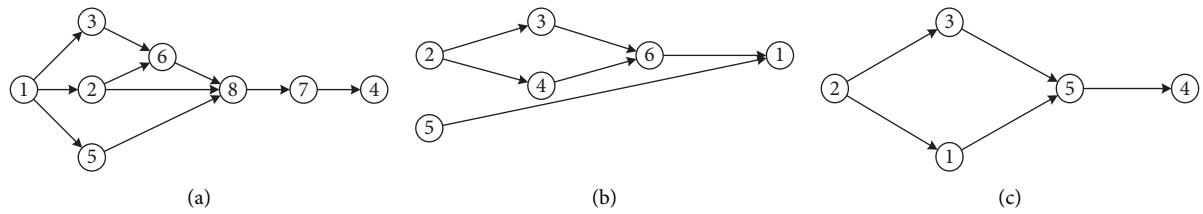


Figure 1: PPD of each product. (a) Product 1. (b) Product 2. (c) Product 3.

The matrix $\mathbf{I}_m$ has 6 components in each column, which correspond to the collision situations between components when they are disassembled along the $\pm x$, $\pm y$, $\pm z$ coordinate axes. Let $o \in \{\pm x, \pm y, \pm z\}$, the element I_{ij}^{o} represents the collision situation between component i and component j when component i is disassembled along the o direction. If $I_{ij}^{o} = 1$, it means that component i collides with component j when disassembled along the o direction. On the other hand, if $I_{ij}^{o} = 0$, it means that component i does not collide with component j when disassembled along the o direction.

Based on the above description, we can see that the interference matrix can be decomposed into disassembly interference matrices along the $+x, +y, +z, -x, -y, -z$ standard orthogonal axes: $\mathbf{I}_{om}$, where $o \in \{\pm x, \pm y, \pm z\}$. Each $\mathbf{I}_{om}$ can be represented by the following equations:

$$\mathbf{I}_{+xm} = \begin{bmatrix} I_{11}^{+xm} & I_{12}^{+xm} & \cdots & I_{1n}^{+xm} \\ I_{21}^{+xm} & I_{22}^{+xm} & \cdots & I_{2n}^{+xm} \\ \vdots & \vdots & \vdots & \vdots \\ I_{n1}^{+xm} & I_{n2}^{+xm} & \cdots & I_{nn}^{+xm} \end{bmatrix}, \tag{4}$$

$$\mathbf{I}_{+ym} = \begin{bmatrix} I_{11}^{+ym} & I_{12}^{+ym} & \cdots & I_{1n}^{+ym} \\ I_{21}^{+ym} & I_{22}^{+ym} & \cdots & I_{2n}^{+ym} \\ \vdots & \vdots & \vdots & \vdots \\ I_{n1}^{+ym} & I_{n2}^{+ym} & \cdots & I_{nn}^{+ym} \end{bmatrix}, \tag{5}$$

$$\mathbf{I}_{+zm} = \begin{bmatrix} I_{11}^{+zm} & I_{12}^{+zm} & \cdots & I_{1n}^{+zm} \\ I_{21}^{+zm} & I_{22}^{+zm} & \cdots & I_{2n}^{+zm} \\ \vdots & \vdots & \vdots & \vdots \\ I_{n1}^{+zm} & I_{n2}^{+zm} & \cdots & I_{nn}^{+zm} \end{bmatrix}, \tag{6}$$

$$\mathbf{I}_{-xm} = \begin{bmatrix} I_{11}^{-xm} & I_{12}^{-xm} & \cdots & I_{1n}^{-xm} \\ I_{21}^{-xm} & I_{22}^{-xm} & \cdots & I_{2n}^{-xm} \\ \vdots & \vdots & \vdots & \vdots \\ I_{n1}^{-xm} & I_{n2}^{-xm} & \cdots & I_{nn}^{-xm} \end{bmatrix}, \tag{7}$$

$$\mathbf{I}_{-ym} = \begin{bmatrix} I_{11}^{-ym} & I_{12}^{-ym} & \cdots & I_{1n}^{-ym} \\ I_{21}^{-ym} & I_{22}^{-ym} & \cdots & I_{2n}^{-ym} \\ \vdots & \vdots & \vdots & \vdots \\ I_{n1}^{-ym} & I_{n2}^{-ym} & \cdots & I_{nn}^{-ym} \end{bmatrix}, \tag{8}$$

$$\mathbf{I}_{-zm} = \begin{bmatrix} I_{11}^{-zm} & I_{12}^{-zm} & \cdots & I_{1n}^{-zm} \\ I_{21}^{-zm} & I_{22}^{-zm} & \cdots & I_{2n}^{-zm} \\ \vdots & \vdots & \vdots & \vdots \\ I_{n1}^{-zm} & I_{n2}^{-zm} & \cdots & I_{nn}^{-zm} \end{bmatrix}. \tag{9}$$

According to the definition of the disassembly matrix, component i can only be disassembled when at least one of the six matrices mentioned above has a row of all zeros. In addition, combined with the description of the precedence matrix TP_m in the previous text, it can be understood that the transpose of the matrix $\mathbf{I}_{om}$, denoted as $\mathbf{I}_{om}^{T}$, can be regarded as the precedence matrix corresponding to the PPD constraints that components disassembled along the o direction need to satisfy. In order to distinguish it from the precedence matrix TP_m, here $\mathbf{I}_{om}^{T}$ is referred to as the precedence matrix in the o direction, denoted as TP_m^o. The element in the i-th row and j-th column of TP_m^o is denoted as p_{ij}^{o}, where $p_{ij}^{o} \in \{0, 1\}$. If $p_{ij}^{o} = 1$, it means that when disassembling along the o direction, disassembling component i is a direct predecessor task of disassembling component j in the o direction, i.e., disassembling component i belongs to the direct predecessor task set of disassembling component j in the o direction ($i \in P_m^o(j)$), and disassembling component j belongs to the direct successor task set of disassembling component i in the o direction ($j \in S_m^o(i)$).

For a component j planned to be disassembled along the o direction, task j can only be executed after all of its immediate predecessor tasks in the o direction have been completed. Therefore, when using the interference matrix, the priority constraint relationships that need to be satisfied between disassembly tasks can be represented by the following equation:

$$O_{mi} > O_{mj}, \quad d_i^m = o \,\&\, j \in P_m^o(j). \tag{10}$$

2.3.2. Random Disassembly Time. In the current research study, the disassembly time is often seen as deterministic, but that is not how it actually works. On the one hand, the structure and condition of discarded products are uncertain, and there may be many deformed components. On the other hand, the disassembly process is influenced by unstable factors such as worker operations and working environment, which ultimately leads to uncertain disassembly time. In

order to characterize the uncertainty of disassembly time, in this research work, we define disassembly time as a random variable following a normal distribution. The disassembly time of any task at a workstation follows a normal distribution, and the sum of disassembly times for all tasks at the station also follows a normal distribution [30], as follows:

$$t_{mi} \sim N\left(\mu_{mi}, \sigma_{mi}^2\right) \Longrightarrow T_k = \sum_{m=1}^{N_{ap}} \sum_{i=1}^{N_m} \left(x_{\text{mik}}^a + x_{\text{mik}}^b\right) \bullet t_{mi} \sim N\left(\sum_{m=1}^{N_{ap}} \sum_{i=1}^{N_m} \left(x_{\text{mik}}^a + x_{\text{mik}}^b\right)\mu_{mi}, \sum_{m=1}^{N_{ap}} \sum_{i=1}^{N_m} \left(x_{\text{mik}}^a + x_{\text{mik}}^b\right)\sigma_{mi}^2\right). \tag{11}$$

2.3.3. Cycle Time Constraints. In order to complete disassembly tasks within a specified time, the pace of disassembly is controlled using cycle time on the production line. The task time at each workstation on the production line does not exceed the upper limit of the cycle time, known as the cycle time constraint. Since the disassembly time is a random variable, the working time at the workstation does not exceed the probabilistic form of the cycle time upper limit [31], as shown in the following equation:

$$\text{Prob}\left(T_k \le T_U\right) \ge \alpha. \tag{12}$$

Let $Z = T_k - \sum_{m=1}^{N_{ap}} \sum_{i=1}^{N_m} \left(x_{\text{mik}}^a + x_{\text{mik}}^b\right)\mu_{mi} / \sqrt{\sum_{m=1}^{N_{ap}} \sum_{i=1}^{N_m} \left(x_{\text{mik}}^a + x_{\text{mik}}^b\right)\sigma_{mi}^2}$, as indicated by (11), Z follows a standard normal distribution. Therefore, (12) can be further rewritten as follows:

$$\text{Prob}\left(Z \le \frac{T_U - \sum_{m=1}^{N_{ap}} \sum_{i=1}^{N_m} \left(x_{\text{mik}}^a + x_{\text{mik}}^b\right)\mu_{mi}}{\sqrt{\sum_{m=1}^{N_{ap}} \sum_{i=1}^{N_m} \left(x_{\text{mik}}^a + x_{\text{mik}}^b\right)\sigma_{mi}^2}}\right) \ge \alpha, \quad \forall k \in \{1, 2, \ldots, K\}. \tag{13}$$

Since Z follows a standard normal distribution, (13) can be further rewritten as (14). The meaning represented by (14) is that the k th workstation needs to satisfy the cycle time constraint.

$$\sum_{m=1}^{N_{ap}} \sum_{i=1}^{N_m} \left(x_{\text{mik}}^a + x_{\text{mik}}^b\right)\mu_{mi} + \phi^{-1}(\alpha) \sqrt{\sum_{m=1}^{N_{ap}} \sum_{i=1}^{N_m} \left(x_{\text{mik}}^a + x_{\text{mik}}^b\right)\sigma_{mi}^2} \le T_U, \quad \forall k \in \{1, 2, \ldots, K\}. \tag{14}$$

The working time of the k th workstation can be represented by the following equation:

$$T_k = \sum_{m=1}^{N_{ap}} \sum_{i=1}^{N_m} \left(x_{\text{mik}}^a + x_{\text{mik}}^b\right)\mu_{mi} + \phi^{-1}(\alpha) \sqrt{\sum_{m=1}^{N_{ap}} \sum_{i=1}^{N_m} \left(x_{\text{mik}}^a + x_{\text{mik}}^b\right)\sigma_{mi}^2}. \tag{15}$$

To reduce idle time at workstations, the actual cycle time of the production line is taken as the maximum working time of all workstations, expressed as follows:

$$T_c = \max\left(T_k\right), \quad k \in \{1, 2, \ldots, K\}. \tag{16}$$

2.3.4. Partial Disassembly. With the emergence of automated crushers and sorting devices, there is no longer a mandatory requirement to dismantle components that do not have hazardous or desired attributes. This disassembly method, which only requires the removal of components with hazardous and desired attributes, is called partial disassembly. In partial disassembly, the components with desired attributes must be removed because they are needed and cannot be crushed by automated crushers. The components with hazardous attributes must be removed because they need to be further processed to eliminate their harmfulness. Therefore, the constraints that partial disassembly needs to meet can be represented by the following equation:

$$\begin{cases} \sum_{k=1}^{K}\left(x_{\mathrm{mik}}^{a}+x_{\mathrm{mik}}^{b}\right)\le 1, & \forall m,i\in\{m,i\mid d_{mi}+h_{mi}=0\},\\ \sum_{k=1}^{K}\left(x_{\mathrm{mik}}^{a}+x_{\mathrm{mik}}^{b}\right)=1, & \forall m,i\in\{m,i\mid d_{mi}+h_{mi}\ge 1\}. \end{cases} \tag{17}$$

2.4. Evaluation Metrics

2.4.1. Number of Open Workstations (NS). Each workstation requires a certain amount of space, and having too many open workstations will occupy a large amount of factory space, thereby increasing the construction cost of the factory. In addition, the more workstations there are, the more workers and equipment are needed, which also incurs a significant cost. Therefore, it is necessary to consider the number of open workstations as an evaluation metric. The number of open workstations can be represented by the following equation:

$$NS=\sum_{k=1}^{K}s_k. \tag{18}$$

2.4.2. Workload Smoothness Indicator (SI). Improving the balance between workstations and reducing the differences in working time among them can be beneficial for enhancing operational efficiency and fairness in task allocation. The workload smoothness indicator effectively measures the differences in working time across different workstations. The smaller the differences in working time among workstations, the higher the value of the workload smoothness indicator. The workload smoothness indicator can be represented by the following equation [32]:

$$SI=-\frac{1}{lnK}\sum_{k=1}^{K}\left(\frac{T_c-T_k}{\sum_{k=1}^{K}(T_c-T_k)}\cdot\ln\frac{T_c-T_k}{\sum_{k=1}^{K}(T_c-T_k)}\right). \tag{19}$$

2.4.3. Hazard Index (HI). Many researchers have pointed out that there are a large number of hazardous components inside discarded products, and considering the impact of these components would have serious consequences. Bahubalendruni et al. [33] pointed out that handling end-of-life (EoL) waste electrical and electronic equipment (WEEE) without proper safety measures poses a serious threat to the environment and public health. Anil Kumar et al. [34] stated that improper disposal of waste materials such as plastic can lead to the release of toxic gases, resulting in incurable cancer. Gulivindala et al. [35] focused on the characteristics and strategic management of electronic waste in the healthcare sector after COVID-19. They highlighted that due to the pandemic, the healthcare industry has undergone significant changes, with an increase in the use of electronic devices for self-diagnosis and treatment through telemedicine. If these electronic devices are not properly handled after disposal, they can cause harm to both individuals and the environment. Ren et al. [36] also pointed out that removing hazardous substances is beneficial in reducing the impact of harmful components on the environment.

From the above research work, it can be concluded that there are always hazardous components in EOL products. During the disassembly process, the later these hazardous components are removed, the more likely they are to be damaged accidentally, leading to potential dangers. The Hazard Index (HI) can be represented as the sum of the disassembly order of all hazardous components. By minimizing the HI, the goal of removing hazardous components as soon as possible can be achieved. The HI can be represented by the following equation:

$$HI=\sum_{m=1}^{N_{ap}}\sum_{i=1}^{N_m}O_{mi}h_{mi}. \tag{20}$$

2.4.4. Number of Disassembled Parts (NDPs). With the emergence of automated crushers and sorting devices, there is no longer a mandatory requirement for manual disassembly of components that do not have hazardous or demand attributes. Under the condition of satisfying priority constraints, directly crushing and automating the sorting of nonmandatory dismantled components can effectively improve the disassembly efficiency. Therefore, it is necessary to use the quantity of disassembled components as an evaluation indicator. The number of disassembled parts (NDPs) can be represented by the following equation:

$$\mathrm{NDP}=\sum_{m=1}^{N_{ap}}\sum_{i=1}^{N_m}\left(x_{\mathrm{mik}}^{a}+x_{\mathrm{mik}}^{b}\right). \tag{21}$$

2.4.5. Disassembly Stability (DS). In their research, Kumar Gulivindala et al. [37] pointed out that the stability of a disassembly sequence refers to the ability of the remaining components in an assembly to resist automatic disintegration and positional changes under the influence of gravity and disassembly operations when dismantled according to a specific sequence. The stability of the disassembly sequence is considered during the planning process. A product with good stability during the disassembly process can effectively prevent damage to high-value components caused by tilting or rolling, ensuring the safety of workers.

To quantify the stability of the disassembly sequence, we refer to the ideas of Kumar Gulivindala et al. [37] and establish a connectivity matrix C^m for the product to be dismantled. Equation (22) shows the connectivity matrix for a product with n components to be dismantled.

$$C^m=\begin{bmatrix} c_{11}^m & c_{12}^m & \cdots & c_{1n}^m\\ c_{21}^m & c_{22}^m & \cdots & c_{2n}^m\\ \vdots & \vdots & \vdots & \vdots\\ c_{n1}^m & c_{n2}^m & \cdots & c_{nn}^m \end{bmatrix}_{n\times n}. \tag{22}$$

The connectivity relationship between component i and j of product m is represented by c_{ij}^m, which can be expressed by the following equation:

$$c_{ij}^{m} = \begin{cases} 2 \text{ component } i \text{ is stably connected to component } j, \\ 1 \text{ component } i \text{ is in contact with component } j, \\ 0 \text{ component } i \text{ is not connected to } j. \end{cases} \tag{23}$$

A stable connection indicates that the components are connected through fasteners or an interference fit, while a contact connection signifies that the components are in touch with each other but require an external force to maintain a stable connection. Because there is no connection between the components themselves, when $i = j$, $c_{ij} = 0$.

During the disassembly process according to disassembly sequence S, the stability of component i in product m is determined by how tightly it is connected to the remaining components in product m. Therefore, the stability of component i in product m can be represented by equation the following equation:

$$v_i^m = \sum_{l=O_{mi}}^{N_{to}} c_{i S_{l1}}^{S_{l0}}. \tag{24}$$

In (24), l is an index where S_{l0} and S_{l1} represent the product number and component number corresponding to the element at position l in the disassembly sequence S.

When disassembling according to a certain disassembly sequence S, the stability of all remaining components in the product needs to be considered. Therefore, in MUPDLBP, the stability of the disassembly sequence can be represented by the following equation:

$$V_S = \sum_{m=1}^{N_{ap}} \sum_{i=1}^{N_m} v_i^m. \tag{25}$$

The larger V_s is, the tighter the connections between internal components of each product are during the disassembly process according to sequence S. This indicates that the stability of disassembly sequence S is higher.

2.4.6. Number of Changes in Disassembly Direction (DDC). If there are frequent changes in the disassembly direction during the disassembly process, it will reduce the efficiency of disassembly. Therefore, it is necessary to minimize the number of changes in the disassembly direction during the disassembly process.

When determining the feasibility of disassembly using the interference matrix, a component may have multiple feasible disassembly directions. For example, if all elements in the i-th row of $\mathbf{I}_{+xm}$ and $\mathbf{I}_{-xm}$ are zero, the feasible disassembly directions for component i in product m are in the $+x$ and $-x$ directions. D_i^m is defined as the set of feasible disassembly directions for component i in product m and can be represented by the following equation:

$$D_i^m = \left\{ o \mid \sum_{j=1}^{N_m} I_{ij}^{om} = 0 \right\}, \quad o \in \{+x, +y, +z, -x, -y, -z\}. \tag{26}$$

It is worth noting that the removed component does not impose any constraints on other components. Therefore, in order to accurately calculate D_i^m for each task in a disassembly sequence, it is necessary to promptly remove the constraint information of the removed component on other components. For example, we will create a copy of $\mathbf{I}_{om}$, denoted as I_{ij}^{om}, and after calculating D_i^m, we will set all elements in the i-th column of this copy to 0.

To better explain DDC, let us first define the Maximum Subsequence of Disassembly in the Same Disassembly Direction (MSSDD). S_s represents a continuous subsequence of disassembly orders within the disassembly sequence S. If the intersection of the feasible disassembly direction sets for all disassembly tasks in S_s is not empty and adding a disassembly task to S_s would result in the intersection of the feasible disassembly direction sets for all disassembly tasks in S_s being empty, then we refer to S_s as a maximum subsequence of disassembly in the same disassembly direction within the disassembly sequence S.

Starting from the first disassembly task in the disassembly sequence, analyzing the sequence allows us to easily obtain all MSSDDs (the specific process of obtaining all MSSDDs can be done using dynamic programming or other algorithms, but due to space limitations, it is not described in detail here). These MSSDDs determine the disassembly direction and DDC for all disassembly tasks in the sequence. Obviously, the intersection of the feasible disassembly direction sets for each disassembly task in each MSSDD is the disassembly direction for those tasks (if there are multiple selectable disassembly directions in the intersection, randomly select any one as the disassembly direction for the corresponding tasks in the MSSDD). From this, we can determine the disassembly direction d_i^m for component i in product m. The number of MSSDDs minus 1 is the DDC.

$$\text{DDC} = N_{\text{sdd}} - 1. \tag{27}$$

In (28), N_{sdd} represents the number of MSSDDs.

2.5. Mathematical Model. As described at the beginning of Section 2, in this study, we propose a random multiobjective multiproduct U-shaped mixed-flow incomplete disassembly line balancing problem (MUPDLBP) for the disassembly process of ordinary waste household appliances. We also propose a random multiobjective multiproduct U-shaped mixed-flow disassembly line balancing problem considering disassembly stability (MUPDLBP_S) for EOL products with large weight and rigidity. Below are the corresponding mathematical models.

2.5.1. Mathematical Model for MUPDLBP. Based on the introductory description in Section 2, combined with the analysis in Sections 2.3 and 2.4, we can conclude that MUPDLBP is a combination optimization problem with multiple constraints and objectives. Moreover, the disassembly time for tasks is uncertain. The optimization objectives include the number of active workstations, the hazard index, the quantity of disassembled components, and

the maximization of the workload smoothness indicator, which can be represented by formula the following equation:

$$\min F_1 = \min(NS, -SI, HI, \text{NDP}). \tag{28}$$

The constraints are as given in equations (2), (14), and (17).

2.5.2. Mathematical Model for MUPDLBP_S. MUPDLBP_S is proposed for the disassembly process of EOL products with large weight and rigidity. Considering that the components in this type of EOL products are difficult to be crushed by a crusher, MUPDLBP_S does not adopt partial disassembly. Additionally, MUPDLBP_S uses an interference matrix to describe the priority constraint relationship between disassembly tasks, and the interference matrix contains information about the disassembly direction. Therefore, the number of changes in disassembly direction can be considered as an optimization objective. Hence, the optimization objective can be represented by the following equation:

$$\min F_2 = \min(NS, -SI, -DS, \text{DDC}). \tag{29}$$

The constraints are as given in equations (10), (14), and (17).

2.6. Example of Problem Description. MUPDLBP and MUPDLBP_S have certain similarities, especially in terms of disassembly line layout and constraint conditions, which makes it easy to confuse the two. To distinguish MUPDLBP and MUPDLBP_S more clearly, the following examples will be given for MUPDLBP and MUPDLBP_S problem, demonstrating their characteristics. It is worth noting that since the optimization objectives of both are very clear, the optimization objectives will not be reflected in the examples below.

2.6.1. Example of MUPDLBP. Figure 2 is an example of the MUPDLBP. From Figure 2, we can clearly see the characteristics of the MUPDLBP. There are three main constraints in the MUPDLBP, as shown in Figure 2. Based on the relevant data of tasks and these constraints, a feasible disassembly scheme can be obtained in the MUPDLBP. Additionally, Figure 2 also shows that the layout of the disassembly line in the MUPDLBP is U-shaped. A worker involved in the disassembly process can handle tasks both at the entrance and exit sides of the disassembly line, which increases flexibility and production efficiency. Furthermore, Figure 2 demonstrates that in the MUPDLBP, multiple products can be processed simultaneously on the same disassembly line. This production method, which allows for the mixed disassembly of multiple products, can effectively reduce the number of disassembly lines required and thus lower costs. Lastly, Figure 2 reveals that in the MUPDLBP, it is permissible for some components to not be disassembled and instead be directly sent to an automated crusher.

2.6.2. Example of MUPDLBP_S. Figure 3 is an example of the MUPDLBP_S problem. From Figure 3, we can see that MUPDLBP_S is very similar to MUPDLBP, but there are some differences between them. According to Figure 3, MUPDLBP_S has two main constraints, which is one less than MUPDLBP. The reason for this is that MUPDLBP_S is designed for end-of-life (EOL) products with heavy weight and high rigidity, which are difficult to be processed by automated crushers. Therefore, a complete disassembly mode is used for such EOL products in MUPDLBP_S, and there is no partial disassembly constraint. Additionally, MUPDLBP_S uses an interference matrix to describe the priority relationship constraint, which is different from MUPDLBP. Furthermore, in other aspects, MUPDLBP is basically the same as MUPDLBP_S.

3. The Proposed Method

Although MUPDLBP and MUPDLBP_S differ in the way they describe the priority constraint relationship between tasks and their optimization objectives, their essence is the same. Therefore, when solving MUPDLBP and MUPDLBP_S, the essence of the solution algorithm is the same. Only slight customization modifications need to be made to the rules of the encoding sequence and the decoding process based on the characteristics of PPD and interference matrix, as well as the characteristics of the optimization objectives. Hence, the solution algorithms for MUPDLBP and MUPDLBP_S will not be strictly distinguished below, but their differences will be introduced when necessary.

NSGAII has a strong global search capability, and its encoding method's discreteness can be well applied to discrete problems [38, 39]. However, NSGAII is difficult to directly and effectively deal with the problem of disassembly line balancing. In this regard, we have improved the initialization method, crossover, mutation operator, and crowding degree comparison operator in the NSGA II algorithm and proposed the INSGAII algorithm. The structure of INSGAII is shown in Figure 4

In Figure 4, the purple part is the different part from NSGA II, which is the part we improved. The blue part is the same part as NSGA II.

3.1. Encoding

3.1.1. Encoding Method. The MUPDLBP and MUPDLBP_S problems are oriented towards the disassembly process of multiple products, so the encoding needs to reflect both product information and disassembly task information. In this work, we adopted a two-dimensional gene array encoding method to encode the disassembly task sequence. A two-dimensional gene array encoding sequence can be represented as Q, where $Q = [[E_1, F_1]^T, [E_2, F_2]^T, \ldots, [E_{N_{to}}, F_{N_{to}}]^T]$, $E_1, E_2, \ldots, E_{N_{to}}$ represent the product numbers, and $F_1, F_2, \ldots, F_{N_{to}}$ represent the corresponding task numbers for each product. The second position of the disassembly sequence is $Q[2] = [E_2, F_2]^T$, where $Q[2][0] = E_2$ represents the product number E_2 and $Q[2][1] = F_2$ represents the task number F_2. To further

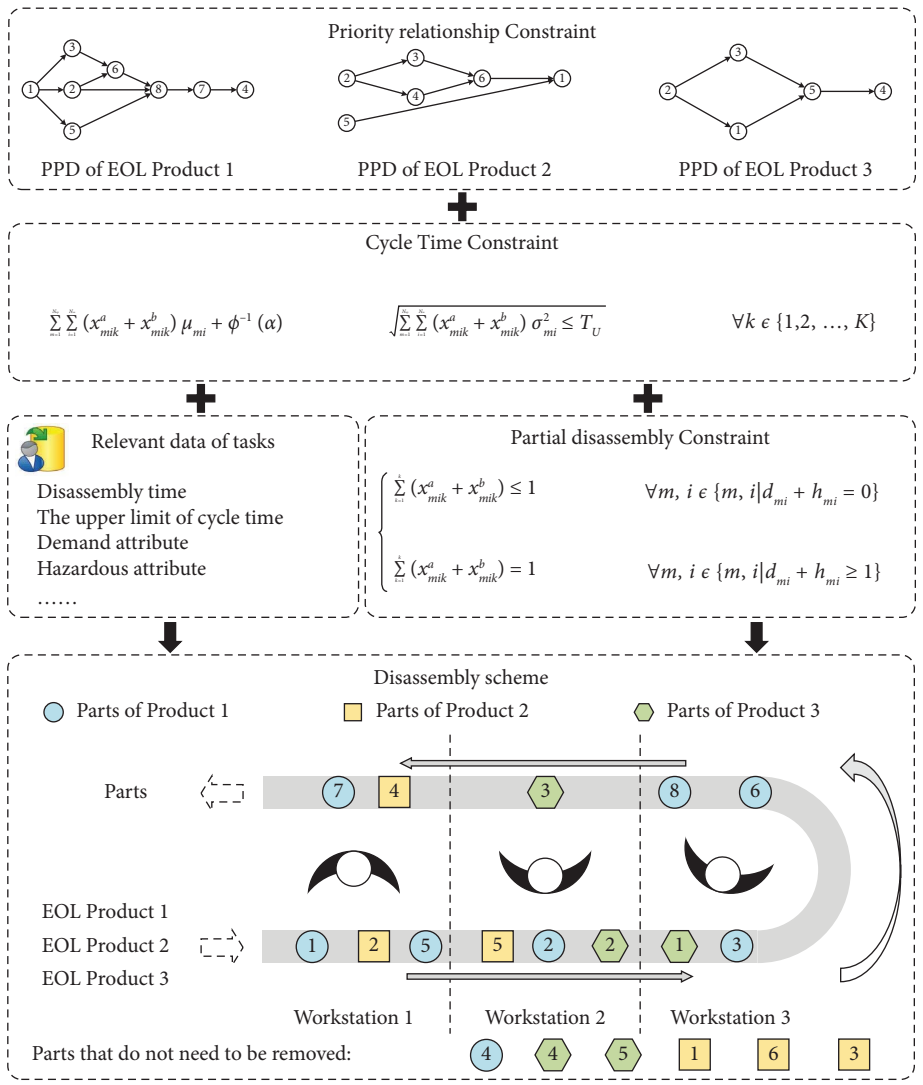


Figure 2: An example of the MUPDLBP problem.

distinguish whether a disassembly task is performed at the entrance or exit side of the U-shaped disassembly line, we added positive and negative signs to the task numbers. When $[E_m, F_i]^T$ represents a disassembly task being processed at the entrance side of the U-shaped disassembly line, $F_i > 0$; otherwise, $F_i < 0$.

It is worth noting that the encoding sequence is not the same as the disassembly sequence. The disassembly sequence reflects the order of disassembly tasks. For example, the encoding sequence of a product with the number 1 is $Q = [[1,+1]^T, [1,-4]^T, [1,+5]^T, [1,+2]^T, [1,+3]^T, [1,-7]^T, [1,+6]^T, [1,+8]^T]$. One possible disassembly scheme corresponding to this encoding sequence is shown in Figure 5. From Figure 5, it can be seen that the disassembly sequence corresponding to the above encoding sequence is $S = [[1,1]^T, [1,5]^T, [1,2]^T, [1,3]^T, [1,6]^T, [1,8]^T, [1,7]^T, [1,4]^T]$. Clearly, the encoding sequence Q and the disassembly sequence S are different.

3.1.2. Constraints That the Encoding Sequence Needs to Satisfy. In order to ensure that the disassembly sequence corresponding to the encoding sequence satisfies the priority constraint relationship, the characteristics of PPD and interference matrix are utilized to describe the constraints that the encoding sequence needs to satisfy in MUPDLBP and MUPDLBP_S, respectively.

(1) When describing the precedence constraint between the disassembly tasks of product m using PPD.

In Section 2.3.1, the precedence relationship matrix of product m, denoted as TP_m, is introduced. The shadow precedence relationship matrix of product m is represented as STP_m, where $\mathrm{STP}_m = TP_m^T$. sp_{ij} is an element in STP_m, with $sp_{ij} \in \{0, 1\}$. If $sp_{ij} = 1$, it means that task i is a shadow predecessor task of task j ($i \in SP_m(j)$). Similarly, $sp_{ij} = 1$ also indicates that task j is a shadow successor task of task i

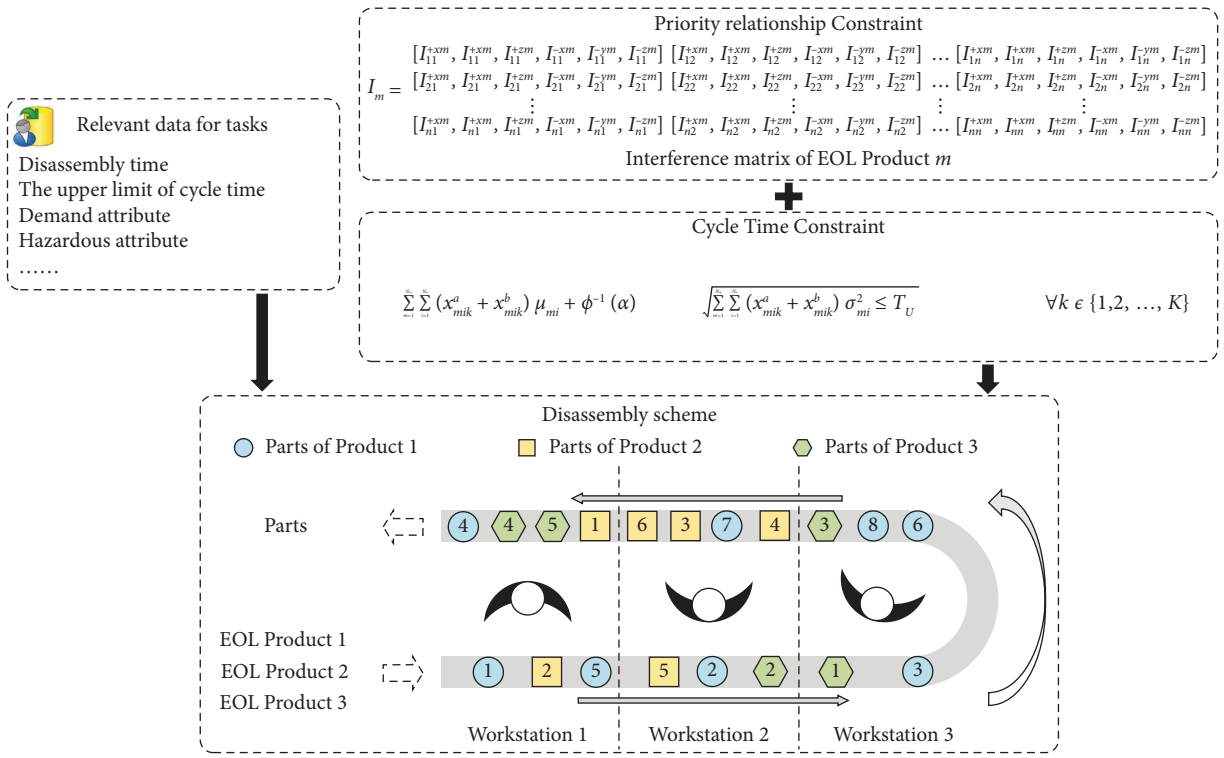


Figure 3: An example of the MUPDLBP_S problem.

($j \in SS_m(i)$). If task j is planned to be assigned to the entrance side of the U-shaped disassembly line, the following condition needs to be satisfied: all elements in the j th column of TP_m are equal to 0, i.e., the predecessor task set $P_m(j)$ of task j is empty. If task j is planned to be assigned to the exit side of the U-shaped disassembly line, the following condition needs to be satisfied: all elements in the j th column of STP_m are equal to 0, i.e., the shadow predecessor task set $SP_m(j)$ of task j is empty [40].

(2) When using the interference matrix to describe the precedence constraint relationship between the disassembly tasks of product m.

In Section 2.3.1, the precedence relationship matrix of product m in the o direction is introduced as TP^o_m. The shadow precedence relationship matrix of product m in the o direction is denoted as STP^o_m. $\mathrm{STP}^o_m = TP^o_m T$. sp^o_{ij} is an element in STP^o_m, where $sp^o_{ij} \in \{0, 1\}$. If $sp^o_{ij} = 1$, it means that when disassembling along the o direction, disassembly part i is a shadow predecessor task of disassembly part j. In other words, disassembly part i belongs to the shadow predecessor task set in the o direction of disassembly part j ($i \in SP^o_m(j)$), and disassembly part j belongs to the shadow successor task set in the o direction of disassembly part i ($j \in SS^o_m(i)$). For a part j that is planned to be assigned to the entrance side of the U-shaped disassembly line and disassembled along the o direction, the following condition needs to be satisfied: all elements in the j th column of TP^o_m are equal to 0; that is, the predecessor task set in the o direction of task j, $P^o_m(j)$, is empty. For a part j that is planned to be assigned to the exit side of the U-shaped disassembly line and disassembled along the o direction, the following condition needs to be satisfied: all elements in the j th column of STP^o_m are equal to 0, i.e., the shadow predecessor task set in the o direction of task j, $SP^o_m(j)$, is empty.

When task j is executed, all constraints related to task j should be deleted. Therefore, during the actual running process of the algorithm, copies of TP_m, STP_m, TP^o_m, and STP^o_m will be created. When task j is executed, the corresponding elements in the j th row and j th column of the respective copies will be deleted. This approach allows for real-time updating of constraint conditions, ensuring that the disassembly sequence corresponding to the encoding sequence satisfies the precedence constraint relationship.

3.2. Decoding. Decoding is the process of assigning tasks from the encoded sequence to workstations and calculating relevant evaluation indicators. On the one hand, in MUPDLBP, it is only required to remove parts that are necessary or hazardous, so the decoding program needs to have the ability to eliminate nonessential parts. On the other hand, since the calculation of Hazard Index (HI), disassembly stability (DS), and the number of changes in disassembly direction (DDC) requires the disassembly

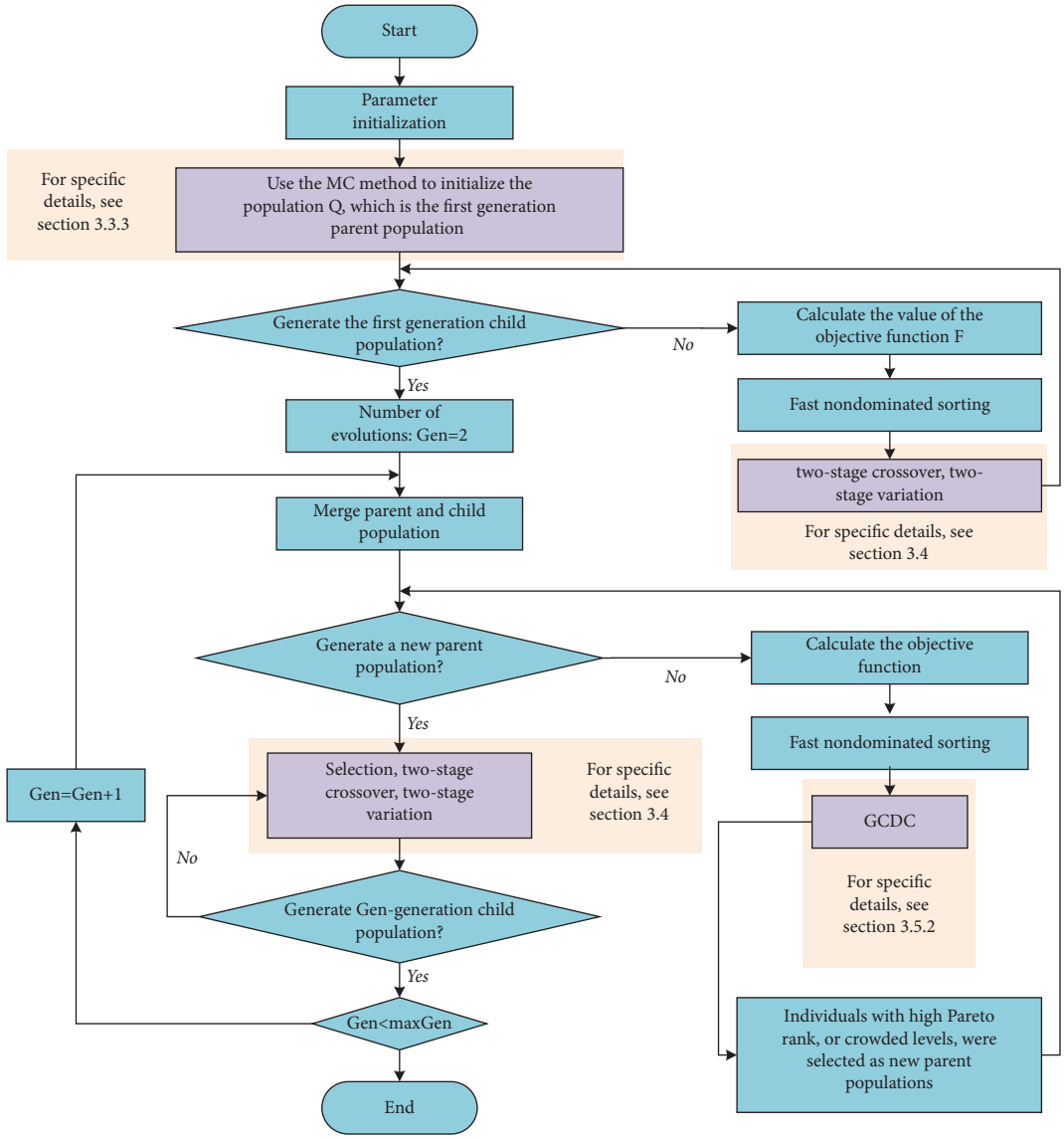


Figure 4: Main process of INSGA-II.

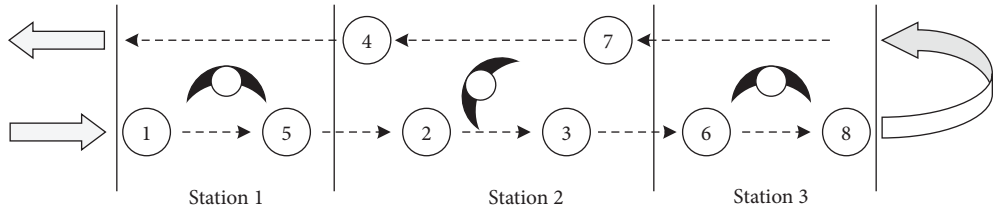


Figure 5: Illustration of the disassembly scheme.

sequence, the decoding program also needs to be able to convert the encoded sequence into a disassembly sequence. To meet the requirements of these two aspects, we have designed a three-stage decoding process. Since the decoding process is similar for both MUPDLBP and MUPDLBP_S problems, we do not strictly differentiate between them when describing the decoding process, but we will indicate the differences when necessary.

3.2.1. First-Stage Decoding. The main task of this decoding process is to remove unnecessary disassembly parts in the encoded sequence. It is worth noting that MUPDLBP_S is proposed for EOL products with heavy weight and high rigidity, which are difficult to be crushed by automated crushers. Therefore, MUPDLBP_S does not adopt partial disassembly, and this decoding process is not required for MUPDLBP_S. During the disassembly of discarded

household appliances, once parts with hazardous or required attributes are completely disassembled, the remaining parts that have not been disassembled can be directly crushed and sorted by automated crushers and sorting devices. Therefore, these parts do not need to be disassembled. It is necessary to analyze the relationship between the arrangement order of task codes in the encoded sequence and the execution order of tasks and then use this relationship to filter out non-essential disassembly parts, which can achieve the goal of the first-phase decoding process.

Based on Figure 5, it can be seen that in the U-shaped disassembly line, the disassembly tasks allocated to the exit side must wait for the tasks allocated to the entrance side to be executed before they can be executed. In other words, disassembly tasks with negative task numbers in the encoded sequence need to be executed after tasks with positive task numbers in the sequence are fully executed. In addition, based on Figure 5, it can also be observed that the execution order of tasks allocated to the entrance side is the same as the order in the encoded sequence, while the execution order of tasks allocated to the exit side is the reverse of the order in the encoded sequence. In other words, the execution order of disassembly tasks with positive task numbers in the encoded sequence is the same as the order in the sequence, while the execution order of disassembly tasks with negative task numbers in the encoded sequence is the reverse of the order in the sequence.

In conclusion, the following conclusions can be drawn: In the encoded sequence Q, if a task with a negative task number needs to be executed, all tasks with positive task numbers and all tasks with negative task numbers that appear after this task in the sequence need to be executed. In other words, unnecessary disassembly parts can be removed in the following way: find the position of the earliest occurrence of a task with a negative task number that must be executed in the encoded sequence. At this point, tasks with negative task numbers in the encoded sequence and tasks before it in the sequence do not need to be executed.

If a task with a positive task number needs to be executed, all tasks with positive task numbers that appear before this task in the sequence need to be executed. In other words, unnecessary disassembly parts can be removed in the following way: find the position of the last occurrence of a task with a positive task number that must be executed in the encoded sequence. At this point, tasks with negative task numbers in the encoded sequence or tasks with positive task numbers that appear after it do not need to be executed.

By combining the above conclusions, the goal of removing unnecessary disassembly parts can be achieved. The specific implementation process of the first-stage decoding is detailed in Algorithm 1.

3.2.2. Second-Stage Decoding. The main task of this decoding process is to allocate tasks from the encoded sequence Q to workstations. And calculate 3 evaluation indicators, including the number of workstations (NS), the number of disassembled parts (NDP), and the workload smoothness indicator (SI). The specific implementation process of the second-stage decoding is detailed in Algorithm 2.

3.2.3. Third-Stage Decoding. The main task of this decoding process is to convert the encoding sequence Q into a disassembly sequence. It also calculates three evaluation indicators, including the number of direction changes (DDC), stability of the disassembly sequence (DS), and the hazard index (HI). The specific implementation process of the third-stage decoding is detailed in Algorithm 3.

3.3. Initialization Method. In dealing with various DLBP, the commonly used method is to generate an initial population through random initialization. However, the initial population generated by random initialization has a low coverage of the solution space. This is because all DLBPs require disassembly sequences to satisfy various constraints. In order to improve the coverage of the solution space by the initial population, we propose the Monte Carlo Tree Simulation-based Initialization method (MCTI). To explain the necessity of proposing MCTI and the specific approach of MCTI, let us first introduce the approach of random initialization, the limitations of random initialization, and the specific approach of MCTI.

3.3.1. Approach of Random Initialization. Although there may be differences in the technical details of generating initial populations through random initialization for different DLBPs, the core idea remains the same. Specifically, when using random initialization, it is necessary to randomly select disassembly tasks that satisfy the constraint conditions and encode them until all tasks have been encoded [7, 10, 19]. The above process is actually a random search process for the Feasible Disassembly Sequence Tree (FDST) (detailed explanation in Section 3.3.2).

When dealing with MUPDLBP and MUPDLBP_S problems, the encoding of disassembly tasks for the same type of product needs to satisfy their respective constraint conditions. There is no constraint relationship between the disassembly tasks of different types of products. Therefore, when randomly initializing feasible encoding sequences, it is only necessary to randomly initialize the encoding sequences for individual products and then randomly combine the encoding sequences of each product. As mentioned in Section 3.1.2, the encoding constraint conditions for MUPDLBP and MUPDLBP_S are derived based on the characteristics of PPD and interference matrix. Therefore, the random initialization process is described based on the characteristics of PPD and interference matrix.

(a) When using PPD to describe the priority constraint relationship between the disassembly tasks of product m.

(1) The random initialization process of the encoding sequence for a single product: When randomly initializing the encoding sequence for product m, only tasks j with an empty set of

Input: Encoded sequence Q, the sum of disassembly tasks for all products N_{to}.
Output: The encoded sequence $\widetilde{Q}$ after removing nonessential disassembly parts.

(1) $x^a, x^b = [x^a_{111}, \ldots, x^a_{\text{mik}}, \ldots, x^a_{N_{ap}N_{to}K}], [x^b_{111}, \ldots, x^b_{\text{mik}}, \ldots, x^b_{N_{ap}N_{to}K}]$
(2) index = 0, i_{ex} = 0, flag = 0
(3) **for** index **to** N_{to} **do**
(4) $m, i = Q[\text{index}][0], \text{abs}(Q[\text{index}][1])$ //abs(∗) represents taking the absolute value.
//If a task with a negative number and must be executed is found, immediately exit the loop and record the position of that task in the encoded sequence.
(5) **if** $d_{mi} + h_{mi} \geq 1$ && $Q[\text{index}][1] < 0$ **then**
(6) i_{ex} = index
(7) flag = 1
(8) **Break**
//If a task with a positive number and must be executed is found, record the position of that task in the encoded sequence and continue the loop process.
(9) **else if** $d_{mi} + h_{mi} \geq 1$ && $Q[\text{index}][1] > 0$ **then**
(10) i_{ex} = index
(11) **end if**
(12) **end for**
(13) RS = [] //Used to store the positions of tasks that do not need to be executed in the encoded sequence.
//If there is a task with a negative number and must be executed, i_{ex} records the position of the earliest occurrence of such task in the encoded sequence. At this point, tasks with negative numbers in the encoded sequence and tasks before it in the sequence do not need to be executed.
(14) **if** flag == 1 **then**
(15) index = 0
(16) **for** index **to** i_{ex} **do**
(17) **if** $Q[\text{index}][1] < 0$ **then**
(18) RS. append (index)
(19) **end for**
//If there is no task with a negative task number that must be executed, i_{ex} records the position of the last occurrence of a task with a positive task number that must be executed in the encoded sequence. At this point, tasks with negative task numbers in the encoded sequence or tasks with positive task numbers that appear after it do not need to be executed.
(20) **else**
(21) index = 0
(22) **for** index **to** N_{to} **do**
(23) **if** $Q[\text{index}][1] < 0 \,\|\, \text{index} > i_{ex}$ **then**
(24) RS. append (index)
(25) **end for**
(26) **end if**
(27) $\widetilde{Q}$ = remove (Q, RS) //Remove unnecessary disassembly parts.
(28) **return** $\widetilde{Q}$

Algorithm 1: Pseudocode of the first-stage decoding.

Input: Disassembly sequence Q, the sum of disassembly tasks for all products N_{to}, and the upper limit of cycle time T_U.
Output: The allocation scheme of assigning disassembly tasks to workstations is x^a, x^b;
$x^a = [x^a_{111}, \ldots, x^a_{\text{mik}}, \ldots, x^a_{N_{ap}N_{to}K}], x^b = [x^b_{111}, \ldots, x^b_{\text{mik}}, \ldots, x^b_{N_{ap}N_{to}K}]$. The number of workstations NS, the number of disassembled parts NDP, and the workload smoothness indicator SI.

(1) $x^a, x^b = [x^a_{111}, \ldots, x^a_{\text{mik}}, \ldots, x^a_{N_{ap}N_{to}K}], [x^b_{111}, \ldots, x^b_{\text{mik}}, \ldots, x^b_{N_{ap}N_{to}K}]$
(2) $\widetilde{Q} \longleftarrow$ When dealing with the MUPDLBP, $\widetilde{Q}$ is obtained through the first stage decoding. When dealing with the MUPDLBP_S problem, $\widetilde{Q} = Q$.
(3) NDP = len $(\widetilde{Q})$ //Number of disassembled parts.
(4) index = 0, i_{ex} = 0, TL = [ ]
//Distribute the disassembly tasks to each workstation.
(5) **for** index **to** NDP **do**
(6) $m, i = Q[\text{index}][0], \text{abs}(Q[\text{index}][1])$
(7) **if** $Q[\text{index}][1] > 0$ **then**
(8) $x^a_{\text{mik}}, x^b_{\text{mik}} = 1, 0$
(9) **else**
(10) $x^a_{\text{mik}}, x^b_{\text{mik}} = 0, 1$

Algorithm 2: Continued.

(11) **end if**
(12) **if** $T_k + (x^a_{\text{mik}} + x^b_{\text{mik}}) \bullet \mu_{mi} + \phi^{-1}(\alpha) \bullet \sqrt{T_k + (x^a_{\text{mik}} + x^b_{\text{mik}}) \bullet \sigma^2_{mi}} > T_U$ **then**
(13) $k = k + 1$
(14) TL.append(T_k)
(15) $T_k = 0$
(16) **if** Q[index][1] > 0 **then**
(17) $x^a_{\text{mik}}, x^b_{\text{mik}} = 1, 0$
(18) **else**
(19) $x^a_{\text{mik}}, x^b_{\text{mik}} = 0, 1$
(20) **end if**
(21) **end if**
(22) $T_k = T_k + (x^a_{\text{mik}} + x^b_{\text{mik}}) \bullet \mu_{mi} + \phi^{-1}(\alpha) \bullet \sqrt{T_k + (x^a_{\text{mik}} + x^b_{\text{mik}}) \bullet \sigma^2_{mi}}$
(23) **end for**
(24) $T_c = \max(TL)$
(25) $NS = k$ //Number of workstations.
(26) $SI \longleftarrow$ Calculate the workload smoothness indicator by substituting TL, T_c, and NS into equation (19).
(27) **return** x^a, x^b, NDP, SI, NS

Algorithm 2: Pseudocode of the second-stage decoding.

Input: Disassemble sequence Q, the sum of all disassembly tasks for products N_{to}, and the upper time limit T_U.
Output: Change count of disassembly direction DDC, stability of disassembly sequence DS, hazard index HI, disassembly direction for each disassembly task.
(1) $\widetilde{Q} \longleftarrow$ When dealing with the MUPDLBP, $\widetilde{Q}$ is obtained through the first stage decoding. When dealing with the MUPDLBP_S problem, $\widetilde{Q} = Q$.
(2) NDP = len$(\widetilde{Q})$ //Number of disassembled parts.
(3) index $= 0, tlp = [], tln = []$, HI $= 0$
(4) **for** index **to** NDP **do**
(5) **if** $\widetilde{Q}$[index][1] > 0 **then**
(6) tlp. append$(\widetilde{Q}$[index]$)$
(7) **else**
(8) tln .append$(\widetilde{Q}$[index]$)$
(9) **end for**
(10) tln = reverse(tln) //Reverse tln
(11) S = concatenate(tlp, tln) //Combine tlp and tln to get the disassembly sequence
(12) DS, DDC $\longleftarrow$ Calculate disassembly stability and change count of disassembly direction separately according to (25) and (27). //Only execute this step when dealing with MUPDLBP_S problem.
(13) $HI \longleftarrow$ Calculate the hazard index according to (20). //Only execute this step when dealing with MUPDLBP.
(14) Disassembly direction for each disassembly task $\longleftarrow$ Obtain the disassembly direction for each disassembly task based on the description in Section 2.4.6.
(15) **return** DDC, DS, HI, disassembly direction for each disassembly task

Algorithm 3: Pseudocode of the third-stage decoding.

immediate predecessor tasks ($P_m(j) = \varnothing$) can be assigned to the entrance side of the U-shaped disassembly line for processing; only tasks j with an empty set of shadow immediate predecessor tasks ($SP_m(j) = \varnothing$) can be assigned to the exit side of the U-shaped disassembly line for processing. In other words, when encoding the disassembly tasks of product m, only tasks in the sets $\{j \mid p_{ij} = 0, \forall i \in [1, N_m], i \in Z, p_{ij} \in TP_m\}$ and $\{j \mid sp_{ij} = 0, \forall i \in [1, N_m], i \in Z, sp_{ij} \in \text{STP}_m\}$ can be encoded at the current position. Tasks in the set $\{j \mid p_{ij} = 0, \forall i \in [1, N_m], i \in Z, p_{ij} \in TP_m\}$ can be assigned to the entrance side of the U-shaped disassembly line for processing, which means they can be encoded as positive numbers; tasks in the set $\{j \mid sp_{ij} = 0, \forall i \in [1, N_m], i \in Z, sp_{ij} \in \text{STP}_m\}$ can be assigned to the exit side of the U-shaped disassembly line for processing, which means they can be encoded as negative numbers. To generate a feasible encoding sequence, randomly select task j from the set $\{j \mid p_{ij} = 0, \forall i \in [1, N_m], i \in Z, p_{ij} \in TP_m\}$ or the set $\{j \mid sp_{ij} = 0, \forall i \in [1, N_m], i \in Z, sp_{ij} \in \text{STP}_m\}$ and encode it as $[m, +j]$ or $[m, -j]$. Repeat the

above process until all disassembly tasks of product m have been executed, and the initialization encoding sequence for product m is generated.

(2) Generating a mixed product encoding sequence: After obtaining the encoding sequences for each product, it is possible to randomly combine the encoding tasks of different products to obtain an encoding sequence for the MUPDLBP while keeping the inherent encoding order of each product unchanged. For example, the encoding sequences for three types of scrap products, numbered 1, 2, and 3 in Figure 1, can be randomly combined to form a multiproduct mixed encoding sequence that satisfies the constraints of the MUPDLBP, as shown in Figure 6.

(b) When using the interference matrix to describe the priority constraint relationship between the disassembly tasks of product m.

(1) The random initialization process of the encoding sequence for a single product: When randomly initializing the encoding sequence for product m, a task j can only be assigned to the entrance side of the U-shaped disassembly line for processing if there exists $o \in \{+x, +y, +z, -x, -y, -z\}$ such that $P_m^o(j) = \varnothing$; a task j can only be assigned to the exit side of the U-shaped disassembly line for processing if there exists $o \in \{+x, +y, +z, -x, -y, -z\}$ such that $SP_m^o(j) = \varnothing$. In other words, when encoding the disassembly tasks of product m, only tasks in the sets $\{j \mid p_{ij}^o = 0, \forall i \in [1, N_m], i \in Z, \exists o \in \{+x, +y, +z, -x, -y, -z\}, p_{ij}^o \in TP_m^o\}$ and $\{j \mid sp_{ij}^o = 0, \forall i \in [1, N_m], i \in Z, \exists o \in \{+x, +y, +z, -x, -y, -z\}, sp_{ij}^o \in \mathrm{STP}_m^o\}$ can be encoded at the current position. Tasks in the set $\{j \mid p_{ij}^o = 0, \forall i \in [1, N_m], i \in Z, \exists o \in \{+x, +y, +z, -x, -y, -z\}, p_{ij}^o \in TP_m^o\}$ can be assigned to the entrance side of the U-shaped disassembly line for processing, which means they can be encoded as positive numbers; tasks in the set $\{j \mid sp_{ij}^o = 0, \forall i \in [1, N_m], i \in Z, \exists o \in \{+x, +y, +z, -x, -y, -z\}, sp_{ij}^o \in \mathrm{STP}_m^o\}$ can be assigned to the exit side of the U-shaped disassembly line for processing, which means they can be encoded as negative numbers. To generate a feasible encoding sequence, randomly select task j from the set $\{j \mid p_{ij}^o = 0, \forall i \in [1, N_m], i \in Z, \exists o \in \{+x, +y, +z, -x, -y, -z\}, p_{ij}^o \in TP_m^o\}$ or the set $\{j \mid sp_{ij}^o = 0, \forall i \in [1, N_m], i \in Z, \exists o \in \{+x, +y, +z, -x, -y, -z\}, sp_{ij}^o \in \mathrm{STP}_m^o\}$ and encode it as $[m, +j]$ or $[m, -j]$. Repeat the above process until all disassembly tasks of product m have been executed, and the initialization sequence for product m is generated.

(2) Generating the encoding sequence for mixed products: After obtaining the encoding sequences for each product, the encoding sequence for the MUPDLBP_S problem can be generated by randomly combining the encoding tasks of different products while maintaining the inherent encoding order of each product. Please refer to Figure 6 for this process.

3.3.2. Limitations of the Random Initialization Method. The essence of the random initialization process is to construct a feasible disassembly sequence tree (FDST) and then perform a random search from the root node of the FDST to the leaf nodes. During the random search in the FDST, the probability of selecting different branches of the FDST is the same. In the MUPDLBP, due to the priority constraints between disassembly tasks, the number of feasible disassembly sequences corresponding to different branches is different. For the sake of brevity, let us take product 3 in Figure 1 as an example to construct the FDST and explain the phenomenon in detail. It is worth noting that the limitations of the random initialization method are independent of the method used to describe the task priority constraints, so in this section, we do not strictly distinguish between PPD and interference matrix.

In Figure 7, it can be seen that there are 26 feasible disassembly sequences for product 3. Taking the first two elements from all feasible disassembly sequences, we can obtain 5 depth-2 incomplete disassembly sequences: $q_2^1 = [[2, 2]^T, [2, 1]^T]$, $q_2^2 = [[2, 2]^T, [2, 3]^T]$, $q_2^3 = [[2, 2]^T, [2, -4]^T]$, $q_2^4 = [[2, -4]^T, [2, 2]^T]$, and $q_2^5 = [[2, -4]^T, [2, -5]^T]$. Based on q_2^1, q_2^2, q_2^3, q_2^4, and q_2^5, we can continue to perform random searches on the feasible disassembly sequence tree to generate random feasible disassembly sequences. From Figure 7, it can be seen that by continuing to perform random searches on the feasible disassembly sequence tree based on q_2^1, q_2^2, q_2^3, q_2^4, and q_2^5, we can generate 4, 4, 6, 6, and 6 feasible disassembly sequences, respectively.

During the population initialization process, the focus is on how well the individuals in the population cover the entire search space. The greater the coverage, the more likely it is to efficiently discover regions where the optimal solution exists. Since the number of feasible disassembly sequences corresponding to different branches is different, if different branches are selected for searching with equal probabilities, it cannot fully explore branches with a larger number of feasible disassembly sequences, leading to a lower coverage of the search space by the initial population. In addition, selecting different branches for searching with equal probabilities is a waste of search resources for branches with fewer feasible disassembly sequences. A more reasonable approach is to allocate more search resources to branches with a larger number of feasible disassembly sequences. Clearly, this can help the initial population cover the entire search space more effectively.

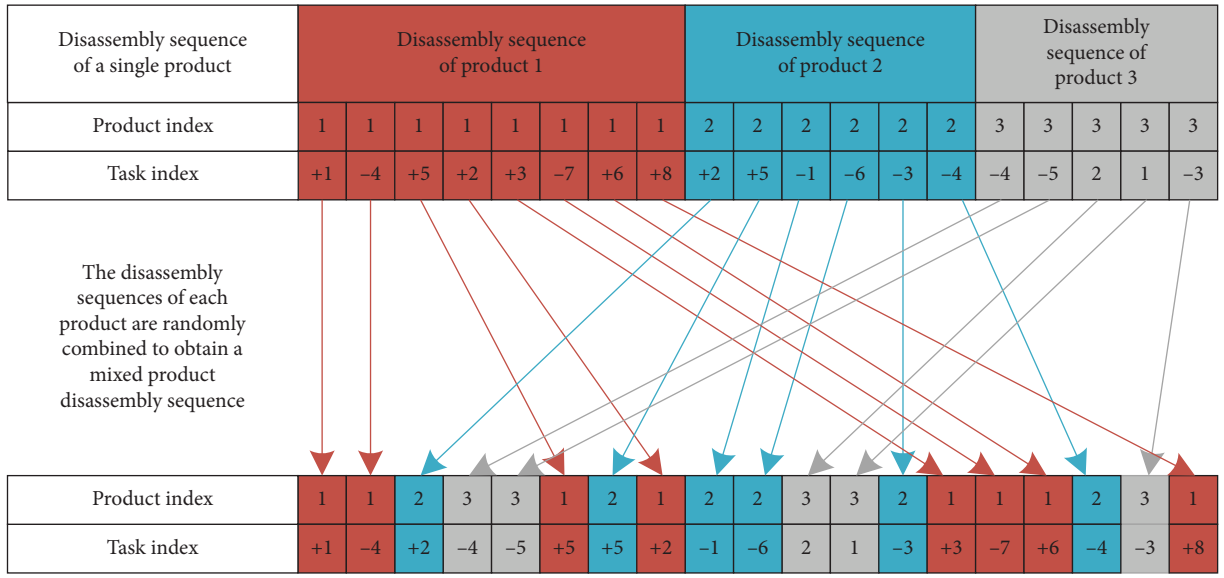


Figure 6: The generation process of a multiproduct mixed coding sequence.

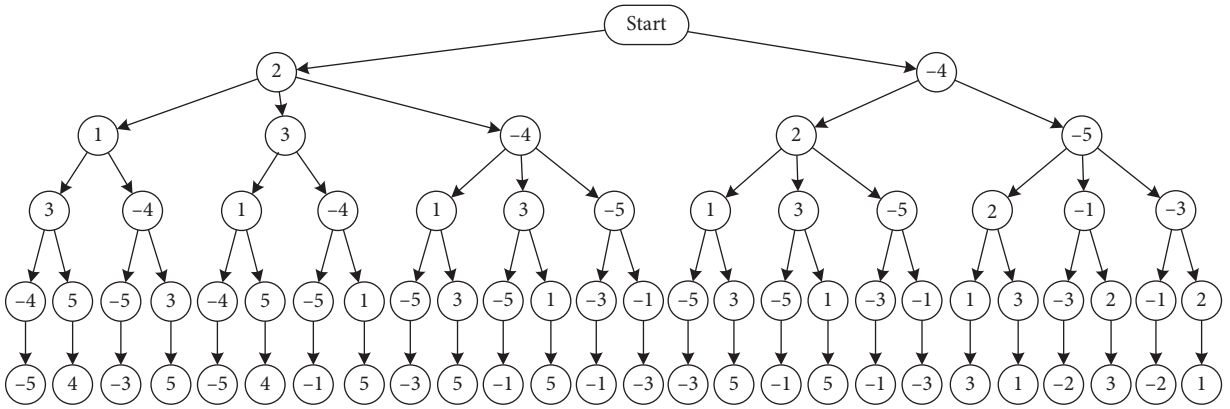


Figure 7: Feasible disassembly sequence tree.

3.3.3. Initialization Method Based on Monte Carlo Tree Simulation. Based on the analysis in Section 3.3.2, we can see that if we can obtain a complete FDST, we can allocate search resources effectively and generate more diverse individuals during the initialization process, thereby improving the quality of the initial population. However, as the number of disassembly tasks increases, the FDST becomes extremely large, making it difficult to construct a complete FDST. To address this, we propose an initialization method based on Monte Carlo Tree Simulation (MCTI). MCTI estimates the distribution of feasible disassembly sequences in the FDST using Monte Carlo methods and uses the estimation results to guide the generation of the initial population, ensuring better coverage of the entire search space. Whether using PPD or interference matrix to describe the priority constraints between disassembly tasks, the initialization process using MCTI is essentially the same. Therefore, in this section, we do not strictly distinguish between MUPDLBP and MUPDLBP_S, but we will indicate the differences between the two when necessary.

In MCTI, for product m, we first use a random initialization method to generate a certain number of simulated solutions qs_m. Then, we take the first D elements from each qs_m to obtain all incomplete disassembly sequences of depth D, qs'_{mD}. We remove duplicates from qs'_{mD} to obtain qs_{mD}, where $qs_{mD} = [q^1_{mD}, q^2_{mD}, \cdots, q^{\text{index}}_{mD}, \cdots]$. Next, we calculate the frequency fq^{index}_{mD} of q^{index}_{mD} in qs_{mD} and continue to generate $fq^{\text{index}}_{mD} \times N_{\text{pop}}$ feasible solutions based on q^{index}_{mD}, where N_{pop} is the population size. Finally, we merge all the feasible solutions of the products according to Figure 6 to obtain the initial feasible population pop_{init}, completing the initialization. The pseudocode for MCTI can be found in pseudocode Algorithm 4.

To demonstrate the superiority of MCTI, we independently ran 100 iterations using the random initialization method and the MCTI method. We recorded the number of nonrepeated individuals in each initial population. Figure 8 shows the number of nonrepeated individuals in the initial population when using the random initialization method and the MCTI initialization method.

Input: The number of types of products to be disassembled N_{ap}; $N = \{N_1, N_2, \cdots, N_m\}$; the set of priority constraint matrices for all products is TP, $TP = [TP_1, TP_2, \cdots TP_m]$; the set of shadow priority constraint matrices for all products is STP, $\text{STP} = [\text{STP}_1, \text{STP}_2, \cdots \text{STP}_m]$. If the interference matrix is used to describe the priority constraint relationships between tasks, the following inputs are required: the set of priority constraint matrices for different disassembly directions for all products, TP^o, $TP^o = [TP_1^o, TP_2^o, \cdots, TP_m^o]$; the set of shadow priority constraint matrices for different disassembly directions for all products, STP^o, $\text{STP}^o = [\text{STP}_1^o, \text{STP}_2^o, \cdots, \text{STP}_m^o]$; D: the depth of the feasible disassembly sequence tree; n_{pop}: the number of simulated solutions generated.
Output: Initial population pop_{init}

(1) $m = 0$, $QS = []$
(2) **for** m to N_{ap} **do**
(3) $qs_m \longleftarrow$ Product m is randomly initialized according to the method in Section 3.3.1, n_{pop} initial solutions are generated, and recorded
(4) $qs_{mD}' \longleftarrow$ Refer to the description in Section 3.3.2 to obtain all incomplete disassembly sequences at depth D in qs_m
(5) $qs_{mD} \longleftarrow$ deduplicates qs_{mD}'
(6) **for** index **to** len (qs_{mD}) **do**
(7) $q_{mD}^{\text{index}} \longleftarrow qs_{mD}$ [index]
(8) TP_m, STP_m, TP_m^o, $\text{STP}_m^o \longleftarrow$ Remove the constraint relationship as described in Section 3.1.2
(9) $fq_{mD}^{\text{index}} \longleftarrow$ calculate the frequency of occurrence of q_{mD}^{index} in qs_{mD}
(10) $QS_m \longleftarrow$ On the basis of q_{mD}^{index}, $fq_{mD}^{\text{index}} \times N_{\text{pop}}$ feasible solutions are randomly generated and recorded
(11) **end for**
(12) Store QS_m into the list QS
(13) **end for**
(14) $\text{pop}_{\text{init}} \longleftarrow$ Combine the initialization results QS of all products as shown in Figure 6
(15) **return** pop_{init}

Algorithm 4: Pseudocode of the initialization method based on Monte Carlo tree simulation (MCTI).

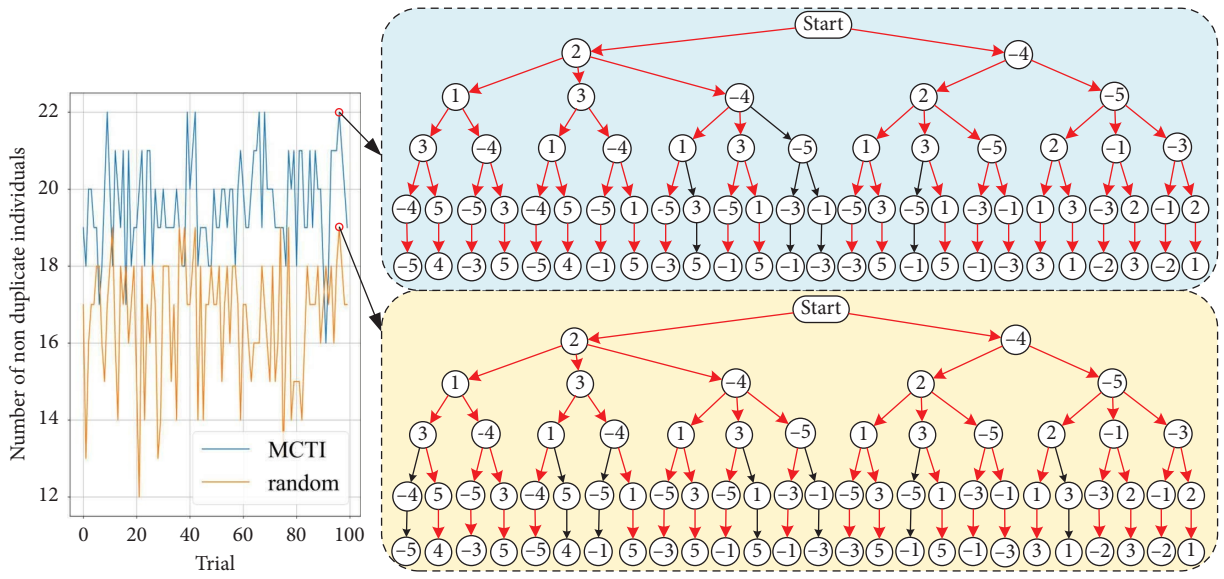


Figure 8: Comparison of the effect of the random initialization method and the MCTI initialization method.

3.4. Method for Generating New Solutions. In the NSGA-II algorithm, crossover and mutation operations are crucial for generating high-quality new solutions. Genetic algorithms use crossover and mutation operations to achieve global and local search capabilities. Since each disassembly task of the disassembled product is subject to various relationship constraints, the encoded sequence must satisfy various constraint conditions. The new encoded sequences generated through traditional crossover and mutation operations may not necessarily satisfy the constraint conditions, which is particularly evident in U-shaped disassembly lines.

In order to make NSGA-II more suitable for the U-shaped disassembly line balancing problem and better solve the MUPDLBP, it is necessary to ensure that the new encoded sequences generated after crossover and mutation satisfy the constraint conditions. To address the above issues, we have

designed a two-stage crossover operator (TSCO) and a two-stage mutation operator (TSMO). The following will provide a detailed introduction to the two-stage crossover and mutation operators.

3.4.1. Sign Change Detection Operator. In both the two-stage crossover operation and the two-stage mutation operation, a sign change detection operator is required. Therefore, let us first introduce the sign change detection operator. In the U-shaped disassembly line, there are the entrance side and the exit side. As mentioned in Section 3.1.1, we use positive and negative signs to differentiate the tasks assigned to the entrance side and the exit side. There are two situations that exist during the crossover and mutation process:

(1) Tasks originally assigned to the entrance side may be reassigned to the exit side, meaning that in the disassembly sequence, task i changes from $+i$ to $-i$

(2) Tasks originally assigned to the exit side may be reassigned to the entrance side, meaning that in the disassembly sequence, task i changes from $-i$ to $+i$

As mentioned in Section 3.1.2, the constraints for the entrance side and the exit side of the U-shaped disassembly line are different. Therefore, once the above two situations occur, it is necessary to further check whether task i (the disassembly task i of product m) can be transferred from the exit side of the U-shaped disassembly line to the entrance side (or from the entrance side to the exit side), which means whether the sign of task i in the disassembly sequence can change.

When using PPD to describe the priority constraint relationship between tasks, the sign of task i can only change if one of the following two conditions is met:

(1) Task i was originally processed on the entrance side of the U-shaped disassembly line, and the shadow predecessor task set of task i is empty ($SP_m(i) = \varnothing$), and then i can be reassigned to the exit side of the U-shaped disassembly line for processing.

(2) Task i was originally processed on the exit side of the U-shaped disassembly line, and the predecessor task set of task i is empty ($P_m(i) = \varnothing$), and then i can be reassigned to the entrance side of the U-shaped disassembly line for processing.

In MUPDLBP, when using PPD to describe the priority constraint relationship between tasks, the pseudocode for the sign change detection operator can be found in Algorithm 5.

In MUPDLBP_S, when using the interference matrix to describe the priority constraint relationship between tasks, the sign of task i can only change if one of the following two conditions is met:

(1) Task i was originally processed on the entrance side of the U-shaped disassembly line, and the disassembly direction is o. If $SP_m^o(j) = \varnothing$ holds true, then i can be reassigned to the exit side of the U-shaped disassembly line for processing.

(2) Task i was originally processed on the exit side of the U-shaped disassembly line, and the disassembly direction is o. If $P_m^o(j) = \varnothing$ holds true, then i can be reassigned to the entrance side of the U-shaped disassembly line for processing.

When using the interference matrix to describe the priority constraint relationship between tasks, the pseudocode for the sign change detection operator can be found in Algorithm 6.

3.4.2. Two-Stage Crossover Operation. The crossover operation is a key operation in the NSGA II algorithm. Traditional crossover operations often produce infeasible solutions, which reduces the efficiency of the algorithm. When the disassembly line is linear, the two-point mapping crossover proposed by Wang et al. [39] can ensure that the crossover result is still a feasible solution, enabling efficient search in the solution space. However, when faced with a U-shaped layout of the disassembly line, the two-point mapping crossover still cannot guarantee that the crossover result is a feasible solution. To address this, we propose a two-stage crossover operation based on the two-point mapping crossover to ensure that the crossover result is a feasible solution. Since there is no priority constraint relationship between disassembly tasks for different products, when designing the two-stage crossover operation, we only need to consider the mutual influence between disassembly tasks within the same product.

First-stage crossover

During the first stage crossover, we use a two-point mapping crossover. First, we randomly select two individuals Q_{crrent1} and Q_{crrent2} from the elite individuals as parents for the crossover. This gives us two offspring individuals, $Q_{\text{new1}}^{\text{step1}}$ and $Q_{\text{new2}}^{\text{step1}}$. We then randomly generate two integers, pos_1 and pos_2, as the crossover points ($\text{pos}_1, \text{pos}_2 \in [0, N_{to}]$ & $\text{pos}_1 < \text{pos}_2$). The task sequence of Q_{crrent1} outside the crossover points pos_1 and pos_2 remains unchanged and becomes part of $Q_{\text{new1}}^{\text{step1}}$. The task sequence between the crossover points pos_1 and pos_2 of Q_{crrent1} is mapped using Q_{crrent2}, and the mapping result becomes the sequence between pos_1 and pos_2 in $Q_{\text{new1}}^{\text{step1}}$. The same process is applied to obtain $Q_{\text{new2}}^{\text{step1}}$. Figure 9 illustrates the process of the first stage crossover.

Second-stage crossover

After completing the first stage crossover, the newly generated encoding sequences $Q_{\text{new1}}^{\text{step1}}$ and $Q_{\text{new2}}^{\text{step1}}$ may not be feasible sequences. This is due to the special nature of the U-shaped disassembly line. The U-shaped disassembly line is divided into the entrance side and the exit side, and the following two situations may occur after completing the first stage crossover.

(1) The tasks originally assigned to the entrance side may be reassigned to the exit side; that is, task i in the encoding sequence changes from $+i$ to $-i$

(2) The tasks originally assigned to the exit side may be reassigned to the entrance side; that is, task i in the encoding sequence changes from $-i$ to $+i$

Input: Product index m, Task index i
Output: Can task i of product m change the number, if it can output **True**, otherwise output **False.**
(1) canChange = **False**
(2) **if** $(i > 0 \,\&\&\, SP_m(i) = \varnothing) \,||\, (i < 0 \,\&\&\, P_m(i) = \varnothing)$ **then**
(3) canChange = **True**
(4) **return** canChange

Algorithm 5: Pseudocode of the sign change detection operator (MUPDLBP).

Input: Product index m, Task index i
Output: Can task i of product m change the number, if it can output **True**, otherwise output **False**.
(1) canChange = **False**
(2) **if** $(i > 0 \,\&\&\, d_i^m == k \,\&\&\, SP_m^o(j) = \varnothing) \,||\, (i < 0 \,\&\&\, d_i^m == k \,\&\&\, P_m^o(j) = \varnothing)$ **then**
(3) canChange = **True**
(4) **return** canChange

Algorithm 6: Pseudocode of the sign change detection operator (MUPDLBP_S).

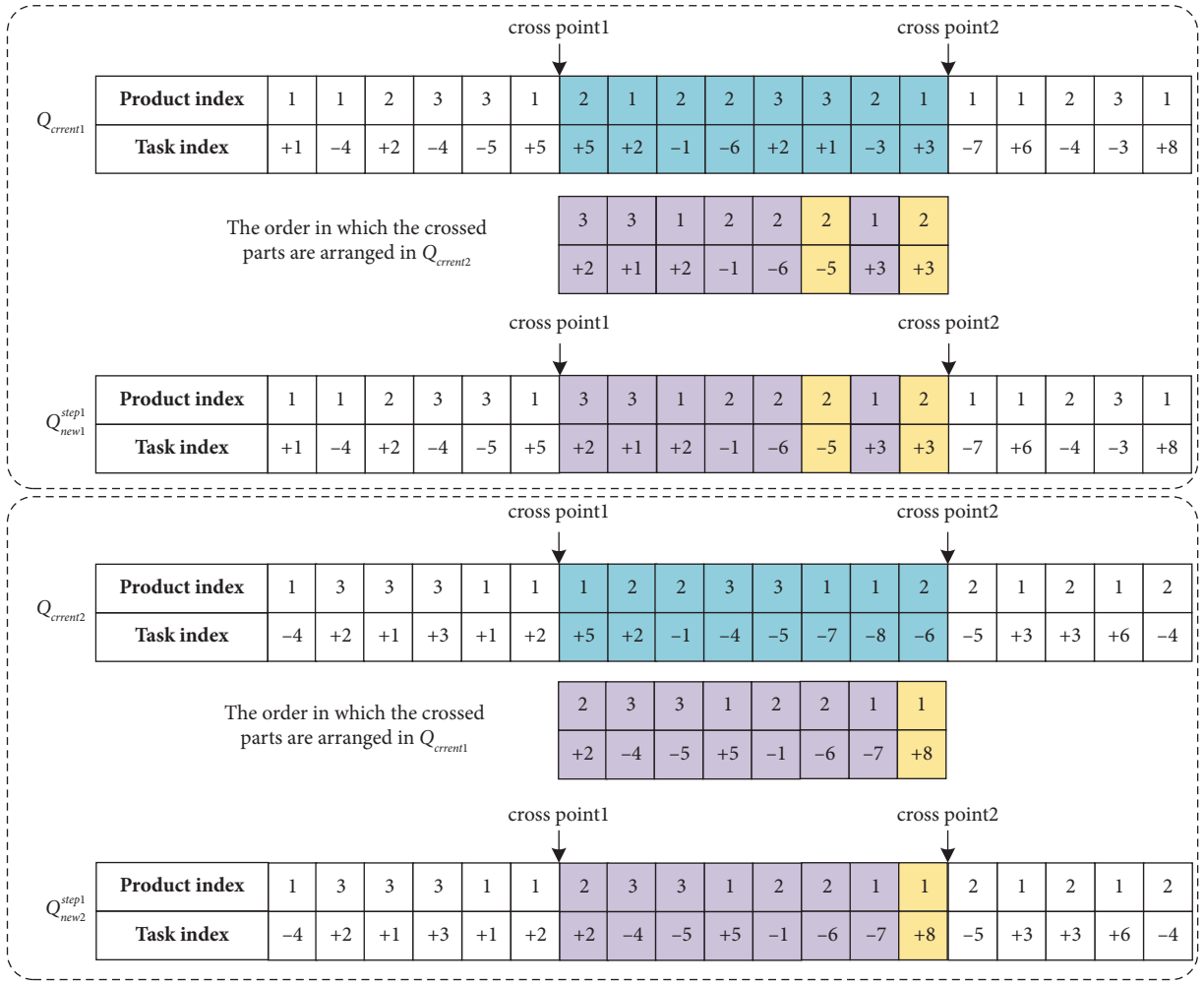


Figure 9: First-stage crossover.

The yellow background in Figure 9 shows these two situations. At this point, the sign-changing detection operator needs to be used to determine whether the tasks in the encoding sequence can change signs. For tasks that can change signs, further adjustments need to be made to their positions in the encoding sequence, and the following adjustment methods can be formulated according to Section 3.1.2:

(1) Suppose after the first stage crossover, task i needs to change from $-i$ to $+i$, that is, reassign task i to the entrance side of the U-shaped disassembly line. As explained in Section 3.1.2, if PPD is used to describe the precedence constraints between tasks, it is necessary to find the immediate predecessor of task i through TP_m and further find the position pos_{pre} of the latest immediate predecessor of task i in the encoding sequence. Then, insert task i after pos_{pre} and as close as possible to pos_{mi} (the position of task i for product m after the first stage crossover), at position $\text{pos}_{\text{insert}}$. If the interference matrix is used to describe the precedence constraints between tasks, it is necessary to find the immediate predecessor of task i in the o direction (o being the disassembly direction of task i) through TP_m^o and further find the position pos_{pre} of the latest immediate predecessor of task i in the o direction in the encoding sequence. Then, insert task i after pos_{pre} and as close as possible to pos_{mi} (the position of task i for product m after the first stage crossover), at position $\text{pos}_{\text{insert}}$.

(2) Suppose after the first stage crossover, task i needs to change from $+i$ to $-i$, that is, reassign task i to the exit side of the U-shaped disassembly line. As explained in Section 3.1.2, if PPD is used to describe the precedence constraints between tasks, it is necessary to find the shadow immediate predecessor of task i through STP_m and further find the position pos_{Spre} of the latest shadow immediate predecessor of task i in the disassembly sequence. Then, insert task i after pos_{Spre} and as close as possible to pos_{mi} (the position of task i for product m after the first stage crossover), at position $\text{pos}_{\text{insert}}$. If the interference matrix is used to describe the precedence constraints between tasks, it is necessary to find the shadow immediate predecessor of task i in the o direction (o being the disassembly direction of task i) through STP_m^o, and further find the position pos_{pre} of the latest o direction's shadow immediate predecessor of task i in the encoding sequence. Then, insert task i after pos_{pre} and as close as possible to pos_{mi} (the position of task i for product m after the first stage crossover), at position $\text{pos}_{\text{insert}}$.

In (1) and (2), the task i is chosen to be inserted at the position $\text{pos}_{\text{insert}}$, which is closest to pos_{mi}, in order to retain the parent information to the greatest extent.

Figure 10 shows the process of the second-stage crossover for $Q_{\text{new2}}^{\text{step1}}$.

The pseudocode for the two-stage crossover operation can be found in Algorithm 7.

3.4.3. Two-Stage Mutation Operator. In MUPDLBP and MUPDLBP_S, traditional mutation methods cannot guarantee that the mutated individuals are feasible solutions. In light of this, we propose a two-stage mutation strategy that takes into account the specific circumstances of MUPDLBP and MUPDLBP_S. Below, we will provide a detailed explanation of the two-stage mutation strategy.

In MUPDLBP, the layout of the disassembly line is U-shaped, with substations on the entry side and exit side. When assigning disassembly tasks to different substations, the performance (objective function value) of the entire disassembly sequence varies. Therefore, in the first stage of the mutation process, it is necessary to determine whether mutation task i (disassembly task i of product m) should be transferred from the exit side of the U-shaped disassembly line to the entry side (or from the entry side to the exit side). This means changing the sign of task i in the disassembly sequence. To determine whether task i can change its sign, a sign change detection operator is used. If the sign of task i can be changed, the sign of i is changed with a certain probability (probability is 0.5).

After completing the first-stage mutation, the second-stage mutation needs to be performed. According to the constraints that the encoding sequence needs to satisfy in Section 3.1.2, we have designed the second-stage mutation process:

(a) When using PPD to describe the precedence constraints between tasks:

If task i is processed on the entry side of the U-shaped disassembly line, simply randomly insert task i between the immediate preceding task and the immediate succeeding task. If task i is processed on the exit side of the U-shaped disassembly line, randomly insert task i between the shadow immediate preceding task and the shadow immediate succeeding task.

(b) When using the interference matrix to describe the precedence constraints between tasks:

If task i is processed on the entry side of the U-shaped disassembly line, randomly insert task i between the immediate preceding task and the immediate succeeding task in the o direction. If task i is processed on the exit side of the U-shaped disassembly line, randomly insert task i between the shadow immediate preceding task and the shadow immediate succeeding task in the o direction. Here, o represents the disassembly direction of task i.

Please refer to Algorithm 8 for the pseudocode of the two-stage mutation.

3.5. Elite Preservation Strategy

3.5.1. Traditional Crowding Distance Calculation Method. The traditional crowding distance comparison operator was proposed by Deb et al. [41] in 2002. Although the use of traditional crowding distance comparison operators has been able to solve many practical problems [42–44], it still has many shortcomings. For example, Xiao et al. [45] pointed out that estimating the crowding degree between only two individuals within the neighborhood range of an individual cannot accurately reflect the structure and connections between individuals in the objective space. In addition, Kahraman et al. [46] pointed out that traditional crowding distance comparison operators cannot simultaneously consider the diversity of the decision space and the objective space, leading to the algorithm getting stuck in local optima. Therefore, although the traditional crowding distance comparison operator is very excellent, it still has some limitations, and it is very important to identify these limitations and provide solutions.

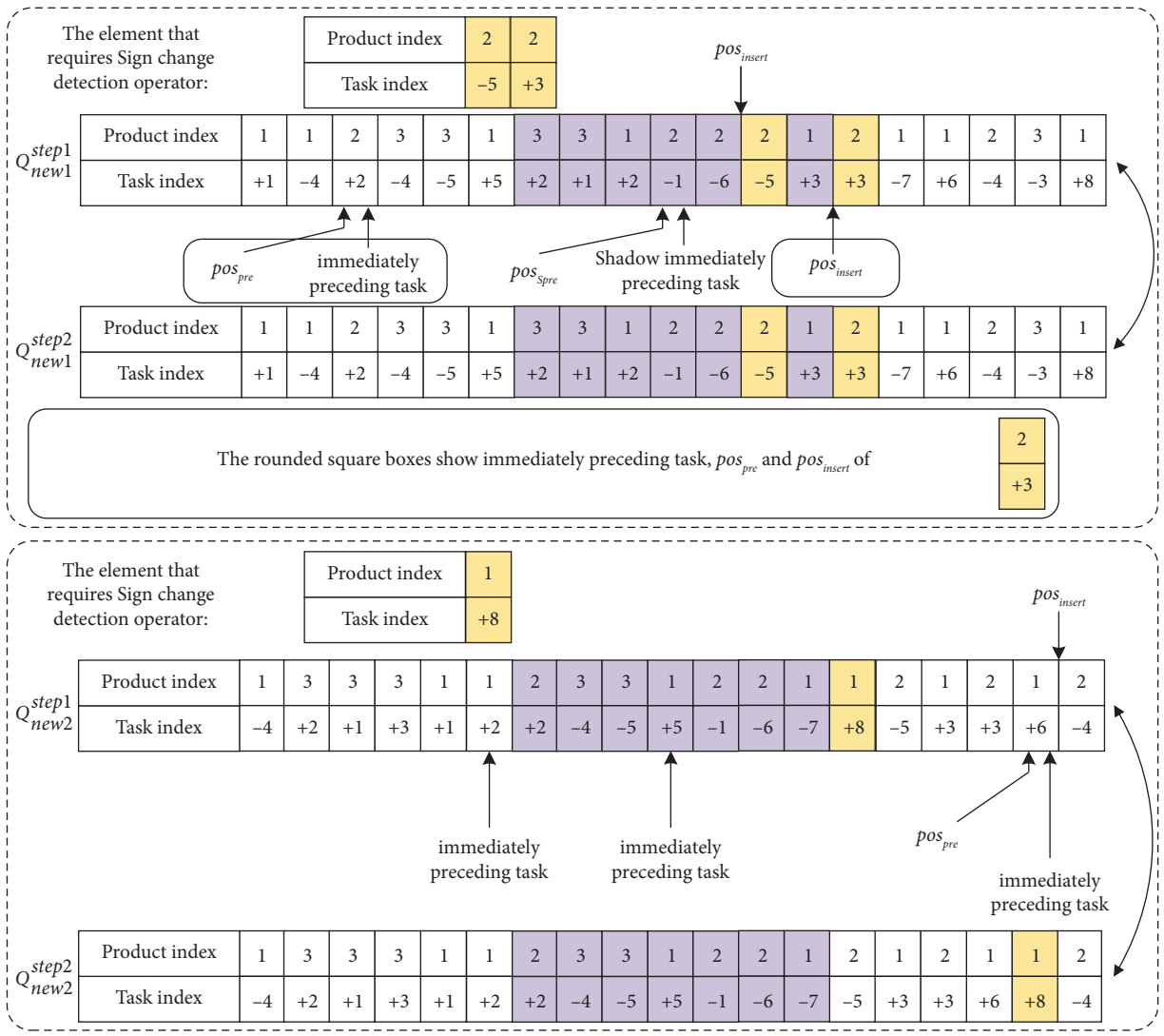


Figure 10: Second-stage crossover.

Input: The two parent individuals participating in the two-stage crossover are Q_{crrent1} and Q_{crrent2}. The total number of disassembly tasks for all products is N_{to}. Additionally, if PPD is used to describe the precedence constraints between tasks, the input requires the following: the set of precedence constraint matrices for all products, $TP = [TP_1, TP_2, \cdots TP_m]$; the set of shadow precedence constraint matrices for all products, $\text{STP} = [\text{STP}_1, \text{STP}_2, \cdots \text{STP}_m]$. If the interference matrix is used to describe the precedence constraints between tasks, the input requires the following: the set of precedence constraint matrices for all products in different disassembly directions, $TP^o = [TP^o_1, TP^o_2, \cdots, TP^o_m]$; the set of shadow precedence constraint matrices for all products in different disassembly directions, $\text{STP}^o = [\text{STP}^o_1, \text{STP}^o_2, \cdots, \text{STP}^o_m]$.

Output: After completing the two-stage crossover, the new disassembly sequences are $Q^{\text{step2}}_{\text{new1}}$ and $Q^{\text{step2}}_{\text{new2}}$.

(1) $a, b \longleftarrow \text{randint}(\text{low} = 0, \text{high} = N_{to}, 2)$ //Two random integers are generated within the range $[0, N_{to})$.
(2) $\text{pos}_1, \text{pos}_2 \longleftarrow \min(a, b), \max(a, b)$
(3) $\text{cross Piece}_1 \longleftarrow Q_{\text{crrent1}}[\text{pos}_1 : \text{pos}_2]$ //Sequence fragments between intersections in Q_{crrent1}
(4) $\text{crossPiece}_2 \longleftarrow Q_{\text{crrent2}}[\text{pos}_1 : \text{pos}_2]$ //Sequence fragments between intersections in Q_{crrent2}
(5) $\text{crossPiece}'_2 \longleftarrow$ Gets how the elements in crossPiece_1 are arranged in Q_{crrent2}
(6) $\text{crossPiece}'_1 \longleftarrow$ Gets how the elements in crossPiece_2 are arranged in Q_{crrent1}
(7) $Q_{\text{crrent1}}[\text{pos}_1 : \text{pos}_2] \longleftarrow \text{crossPiece}'_2$ //Phase 1 Crossover
(8) $Q_{\text{crrent2}}[\text{pos}_1 : \text{pos}_2] \longleftarrow \text{crossPiece}'_1$ //Phase 1 Crossover
(9) $Q^{\text{step1}}_{\text{new1}}, Q^{\text{step1}}_{\text{new2}} \longleftarrow Q_{\text{crrent1}}, Q_{\text{crrent2}}$ //Complete the Phase 1 Crossover
(10) $QS^{\text{step1}} \longleftarrow [Q^{\text{step1}}_{\text{new1}}, Q^{\text{step1}}_{\text{new2}}]$

Algorithm 7: Continued.

$QS^{step2} \longleftarrow [\,]$
for index_1 **to** $\text{len}(QS^{step1})$ **do**
  $Q^{step1} \longleftarrow QS^{step1}[\text{index}_1]$
  diffS $\longleftarrow$ Look for elements in Q^{step1} where the task sign has changed
  for index_2 **to** $\text{len}(\text{diffS})$ **do**
    $m, i \longleftarrow \text{diffS}[\text{index}_2][0], \text{diffS}[\text{index}_2][1]$
    $\text{canChange} \longleftarrow \text{signChangeDetect}(m, i)$ //Sign change detection operator
    if !canChange **then**
      $i \longleftarrow -1 \times i$
    if $i > 0$ **then**
      If PPD is used to describe the precedence constraints between tasks, determine the position of the immediate preceding task of task i in Q^{step1}, denoted as pos_{pre}, and determine the position of $\text{pos}_{\text{insert}}$, which is closest to pos_{mi} and after pos_{pre}. If the interference matrix is used to describe the precedence constraints between tasks, determine the position of the immediate preceding task of task i in Q^{step1} in the o direction, where o is the disassembly direction of task i. pos_{mi} is the position of task i of product m in Q^{step1}. Insert $\text{diffS}[\text{index}_2]$ at $\text{pos}_{\text{insert}}$ to get Q^{step2}.
      Store Q^{step2} in the list QS^{step2}.
    else
      If PPD is used to describe the precedence constraints between tasks, determine the position of the shadow immediate preceding task of task i in Q^{step1}, denoted as pos_{Spre}, and then insert task i at the position closest to pos_{mi} and after pos_{Spre}, denoted as $\text{pos}_{\text{insert}}$. If the interference matrix is used to describe the precedence constraints between tasks, determine the position of the shadow immediate preceding task of task i in Q^{step1} in the o direction, where o is the disassembly direction of task i. pos_{mi} is the position of task i of product m in Q^{step1}. Insert $\text{diffS}[\text{index}_2]$ at $\text{pos}_{\text{insert}}$ to complete the second-stage crossover.
      Store Q^{step2} in the list QS^{step2}.
  $Q^{step2}_{new1}, Q^{step2}_{new2} \longleftarrow QS^{step2}[0], QS^{step2}[1]$
return $Q^{step2}_{new1}, Q^{step2}_{new2}$

Algorithm 7: Pseudocode of the two-stage crossover operation.

Input: The individual Q participating in the two-stage mutation. The total number of disassembly tasks for all products N_{to}. The set of priority constraint matrices for all products TP, $TP = [TP_1, TP_2, \cdots TP_m]$; the set of shadow priority constraint matrices for all products STP, $\text{STP} = [\text{STP}_1, \text{STP}_2, \cdots \text{STP}_m]$. If the interference matrix is used to describe the precedence constraints between tasks, the input requires: the set of precedence constraint matrices for all products in different disassembly directions, $TP^o = [TP^o_1, TP^o_2, \cdots, TP^o_m]$; the set of shadow precedence constraint matrices for all products in different disassembly directions, $\text{STP}^o = [\text{STP}^o_1, \text{STP}^o_2, \cdots, \text{STP}^o_m]$. The mutation probability is p.
Output: The new encoding sequence MQ after completing the two-stage mutation
if $\text{randint}(1) < p$ **then** //Decide whether to mutate Q with a certain probability.
  $a = \text{randint}(\text{low} = 0, \text{high} = N_{to}, 1)$ //Generate a random integer as the mutation position.
  $m, i = Q[a][0], Q[a][1]$
  $\text{canChange} = \text{signChangeDetect}(m, i, TP, \text{STP})$ //Sign change detection operator
  $Q = \text{delete}(Q, a)$ //Remove elements that need to be mutated
  if canChange && $\text{random}(1) > 0.5$ **then**
    $i = -1 \times i$
  if $i > 0$ **then**
    $\text{pos}M_{\text{pre}} \longleftarrow$ Find the positions of all immediate predecessor tasks corresponding to the mutation element $Q[a]$ in Q (when describing the precedence constraint relationship between tasks using an interference matrix, find the positions of all immediate predecessor tasks in the o-direction corresponding to the mutation element $Q[a]$ in Q), and select the latest position from them.
    $\text{pos}M_{\text{suc}} \longleftarrow$ Find the positions of all immediate successor tasks corresponding to the mutation element $Q[a]$ in Q (when describing the precedence constraint relationship between tasks using an interference matrix, find the positions of all immediate successor tasks in the o-direction corresponding to the mutation element $Q[a]$ in Q), and select the earliest position from them.
    $\text{posM}_{\text{insert}} \longleftarrow$ Generate a random integer within the range posM_{pre} and posM_{suc}.
  else
    $\text{posM}_{\text{Spre}} \longleftarrow$ Find the positions of all shadow immediate predecessor tasks corresponding to the mutation element $Q[a]$ in Q (when describing the precedence constraint relationship between tasks using an interference matrix, find the positions of all shadow immediate predecessor tasks in the o-direction corresponding to the mutation element $Q[a]$ in Q), and select the latest position from them.

Algorithm 8: Continued.

(14) $\text{posM}_{\text{Ssuc}} \longleftarrow$ Find the positions of all shadow immediate successor tasks corresponding to the mutation element $Q[a]$ in Q (when describing the precedence constraint relationship between tasks using an interference matrix, find the positions of all shadow immediate successor tasks in the o-direction corresponding to the mutation element $Q[a]$ in Q), and select the earliest position from them.
(15) $\text{posM}_{\text{insert}} \longleftarrow$ Generate a random integer within the range $\text{posM}_{\text{Spre}}$ and $\text{posM}_{\text{Ssuc}}$.
(16) Insert the mutation element $Q[a]$ into the position $\text{posM}_{\text{insert}}$ in Q, resulting in MQ, and complete the two-stage mutation.
(17) **return** MQ

Algorithm 8: Pseudocode of the two-stage mutation operator.

According to the description by Deb et al., when calculating the crowding distance, individuals with the maximum and minimum objective function values are considered as boundary solutions, and the crowding distance of boundary solutions is set to infinity. Since it is necessary to prioritize the retention of individuals with larger crowding distances, the boundary solutions will always be retained. As the optimization objectives increase, more and more individuals become boundary individuals, and the traditional crowding distance comparison operator becomes increasingly ineffective. The case where the traditional crowding distance cannot work is illustrated using Figure 11 as an example.

From Figure 11, it can be observed that the points in the red plane ($f_1 = 0$) and the green plane ($f_3 = 0$) are boundary points. When using the traditional crowding calculation method, the congestion of the points in both the red and green planes will be set to infinity, making it necessary to prioritize their preservation. In reality, there are no other points around point A, making its crowding level optimal. Point A should be retained. However, since point A is not a boundary point, it has the lowest priority for preservation. When only a small number of individuals can be retained, it becomes challenging to preserve point A. When an extremely small number of individuals can be retained, it becomes difficult to decide whether to keep the points in the red plane or the points in the green plane. Additionally, there are fewer individuals around point B in the red plane, and preserving point B is more beneficial for improving the diversity and evenness of the Pareto front. However, the traditional crowding comparison method does not distinguish the crowding degree of point B, resulting in it not having a higher priority when retaining individuals.

3.5.2. Group Global Crowd Degree Comparison (GCDC). As shown in Figure 12, during each iteration's population update phase, individuals with a Pareto rank of F_i cannot all be preserved. Instead, N_n individuals need to be selected from them, along with individuals with a Pareto rank superior to F_i, to form the new parent population. To improve the evenness and spread of the Pareto frontier, we propose the Group global Crowd Degree Comparison (GCDC). Figure 12 illustrates the GCDC. In Figure 12, opt_o represents the set of individuals in the Pareto rank of F_i where the o-th objective function value achieves the optimal value, while opt_* represents the set of all individuals with a Pareto rank of F_i but without any objective function value achieving the optimal value.

As shown in Figure 12, GCDC first groups all individuals with a Pareto rank of F_i. After the grouping process, the crowding distance is calculated among individuals within each group, and uncrowded individuals are selected for preservation. The method for calculating the crowding distance is detailed in (30). The number of preserved individuals in each group is determined based on the proportion of the total number of individuals in each group to the total number of individuals with a Pareto rank of F_i. For example, the number of preserved individuals in the opt_o group can be represented as $(|\text{opt}_o|/|Pop_{F_i}|) \times N_n$, where $|\text{opt}_o|$ is the total number of individuals in the opt_o group, $|Pop_{F_i}|$ is the total number of individuals with a Pareto rank of F_i, and N_n is the total number of individuals that need to be preserved.

From the above description, it can be seen that GCDC avoids direct comparison between nonboundary solutions and boundary solutions by grouping the candidate individuals. This weakens the protection of boundary solutions, and thus GCDC can to some extent eliminate the shortcomings of traditional crowding distance calculation methods.

The pseudocode for GCDC can be found in Algorithm 9. $CR_o(*)$ in Algorithm 7 can be represented by (16)

$$CR_o(\text{pop}) = \sum_{\substack{\text{pop}\in\text{pop}_o,\\ \text{pop}'\in\text{pop}_o,\\ \text{pop}\neq\text{pop}'.}} \sum_{o=1}^{O} \left(f_o(\text{pop}) - f_o\left(\text{pop}'\right)\right)^2. \tag{30}$$

4. Experimental Testing and Case Studies

In this section, we first validate the performance of INSGAII by solving the existing disassembly line balancing problem. Specifically, we use INSGAII to solve the partial disassembly line balancing problem (PDLBP) and the U-shaped partially disruptive disassembly line balancing problem (UDPD). We compare the results with those obtained in related literature to verify the effectiveness of INSGAII. For more details, please refer to Section 4.1 and Section 4.2.

Furthermore, in Section 4.3, we also utilize classical multiobjective optimization algorithms and our proposed INSGA algorithm to solve MUPDLBP and MUPDLBP_S. We analyze the results and further demonstrate the effectiveness of our proposed INSGAII.

The programming language of the proposed algorithm is Python 3.7.1, and the testing platform is a PC equipped with

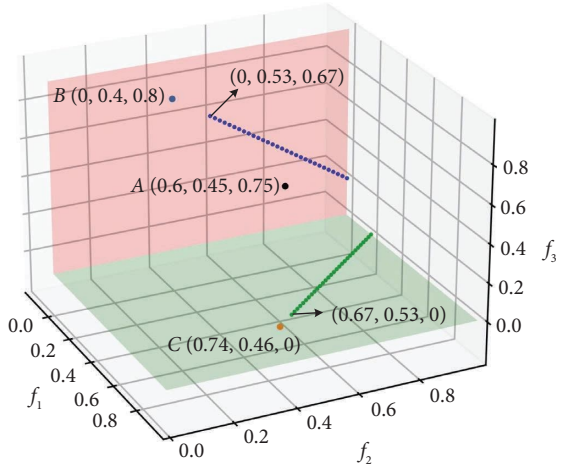


Figure 11: A 3-objective function of a Pareto frontier example.

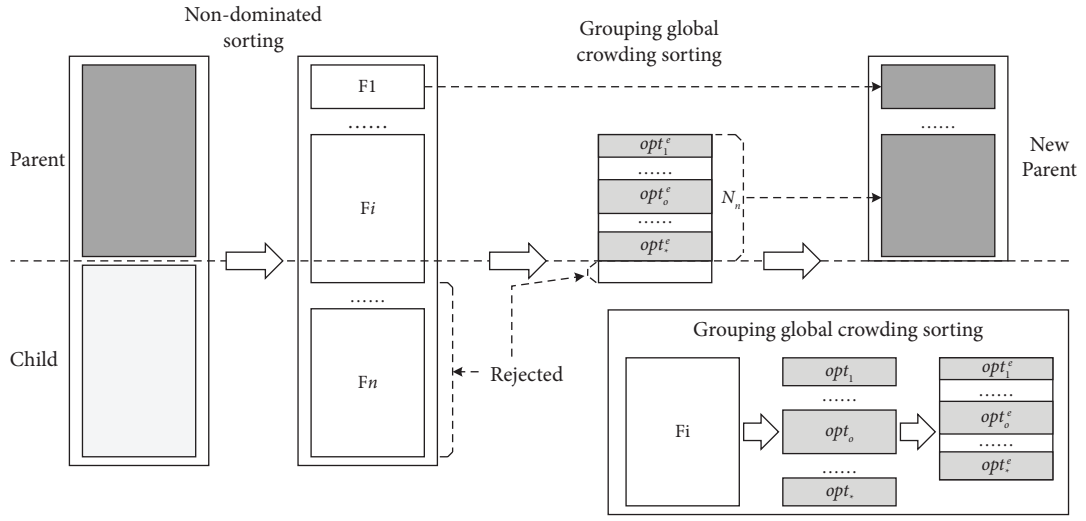


Figure 12: Group global crowd degree comparison.

Input: The number of objective functions O. Pareto level F_i individual set Pop_{F_i}. The objective function value F. F for all individuals is a two-dimensional array with a shape of $|\text{Pop}_{F_i}| \times O$.
Output: Elite individuals pop^e which are retained in Pop_{F_i}

(1) $o, \text{index}_1, \text{index}_2 = 0, 0, 0$
(2) $\text{pops}, \text{Num}_{\text{group}}, \text{pop}^e = [], [], []$
(3) **for** o **to** O **do**
(4) Take the value of the o th objective function of all individuals from F to form f_o.
(5) $\text{pop}_o = \{\text{pop} \mid f_o(\text{pop}) = \text{opt}(f_o), \text{pop} \in \text{Pop}_{F_i}, \text{pop} \notin \text{pops}\}$ //In the Pop_{F_i}, the individuals whose o th objective function f_o is the optimal value are selected
(6) Store pop_o into list pops
(7) Record the number $|\text{pop}_o|$ of individuals in pop_o into List $\text{Num}_{\text{group}}$
(8) $\text{pop}_* = \{\text{poppop} \in \text{Pop}_{F_i}, \text{pop} \notin \text{pops}\}$ //In Pop_{F_i}, select individuals whose objective function is not optimal
(9) Store pop_* into list pops
(10) Record the number $(|\text{pop}_*|)$ of individuals in pop_* into List $\text{Num}_{\text{group}}$
(11) **for** index_1 **to** O **do**
(12) $\text{pop}_o = \text{pops}[\text{index}_1]$
(13) CRS = [ ]
(14) **for** index_2 **to** $\text{len}(\text{pop}_o)$ **do**

Algorithm 9: Continued.

(15) pop = $pop_o[index_2]$
(16) Calculate the CR_o(pop) according to equation (30) and record the calculation result in the list CRS
(17) Sort the CRS and select $(Num_{group}[index_1]/|Pop_{F_i}|) \times N_n$ crowded individuals as elite individual $pop^e_{index_1}$
(18) Store $pop^e_{index_1}$ into list pop^e
(19) **return** pop^e

Algorithm 9: Pseudocode of the group global crowd degree comparison.

11th Gen Intel (R) Core (TM) i7-11800H @ 2.30 GHz and 2.30 GHz.

4.1. Solve PDLBP. In this section, we study the solving performance of the proposed INSGAII algorithm by solving a disassembly instance of a printer. The printer to be disassembled consists of 55 components, and specific data information about this printer instance can be found in the literature [47]. The optimization objectives include minimizing the number of workstations (F_1), balancing the idle time index (F_2), and partial disassembly cost (F_3). Zhu et al. [19] previously solved the PDLBP using this printer as an example and published their results. Now, we will compare the results obtained by the INSGAII algorithm with the results published by Zhu et al. The parameter settings for INSGAII are as follows: population size = 100, number of iterations = 150, MC simulation time = 100, and tree depth = 18. Table 1 presents the results obtained by MIPSO, goal-driven discrete cuckoo search (GDCS), hybrid group neighbourhood search (HGNS), and the proposed INSGAII algorithm. From Table 1, we can see that the MIPSO algorithm obtains 3 Pareto-optimal solutions, the GDCS algorithm obtains 9 Pareto-optimal solutions, the HGNS algorithm obtains 2 Pareto-optimal solutions, and the proposed INSGAII algorithm obtains 3 Pareto-optimal solutions. According to Table 1, the solution 1 obtained by INSGAII dominates all the solutions obtained by MIPSO, GDCS, and HGNS algorithms. Moreover, the solutions 2 and 3 obtained by INSGAII are not dominated by the solutions obtained by other algorithms. In conclusion, the quality of solutions obtained by the INSGAII algorithm is superior to that by the other three algorithms. This means that when dealing with the PDLBP, the INSGAII algorithm outperforms many known algorithms in the literature.

4.2. Solve UDPD. UDPD was proposed by Wang et al. [10]. In UDPD, the disassembly line layout is U-shaped, and the disassembly method for components is uncertain. Specifically, components with hazardous properties or remanufacturing value must be disassembled in the normal mode, while the remaining components can be disassembled in either the normal mode or the destructive mode. Additionally, the disassembly time in UDPD is also uncertain. Wang et al. expressed the uncertainty of disassembly time by assuming that it follows a normal distribution of random numbers. A real case of UDPD is a waste television with 27 components, and relevant data can be found in Wang et al.'s paper. Wang et al. used six optimization algorithms, including multiobjective flower pollination algorithm (MOFPA), nondominated sorting genetic Algorithm 3 (NSGA3), strength Pareto evolutionary Algorithm 2 shift-based density estimation (SPEA2 + SDE), preference-inspired coevolutionary algorithm with goals (PICEA-g), grid-based evolutionary algorithm (GrEA), and Hypervolume-based estimation (HypE), to solve the UDPD problem. In this section, we also use the proposed INSGAII algorithm to solve UDPD. Referring to Wang et al.'s report, the parameter settings for INSGAII are as follows: population size = 300, number of iterations = 500, MC simulation time = 100, and tree depth = 9. INSGAII is independently run 30 times, and the hypervolume (HV) obtained after each run is recorded. The recorded results are then compared with the results reported by Wang et al. Table 2 shows the average and standard deviation of each algorithm on the HV metric, along with a 95% confidence interval.

According to Wang et al.'s report, α represents the probability that the disassembly time at a workstation on the disassembly line does not exceed the cycle time and $\Phi^{-1}(\alpha)$ represents the inverse function value of probability α. From Table 2, it can be observed that in the majority of cases, the mean and standard deviation intervals of INSGAII are significantly better than the other six algorithms. This case demonstrates that the INSGAII algorithm is capable of effectively solving the UDPD problem, meaning that when dealing with UDPD, the INSGAII algorithm outperforms many known algorithms in the literature.

4.3. Solve MUPDLBP and MUPDLBP_S. In Sections 4.1 and 4.2, we verified the performance of the proposed INSGAII algorithm in solving the existing disassembly line balancing problem. This section will discuss the performance of the INSGAII algorithm in solving the MUPDLBP and MUPDLBP_S problems proposed in this paper. In order to verify the performance of the INSGAII algorithm in solving the MMUPDUT and MUPDLBP_S problems, we also introduced three classical multiobjective evolutionary algorithms and three recently proposed multiobjective optimization algorithms to handle the MMUPDUT and MUPDLBP_S problems. The classical multiobjective optimization algorithms include the following: NSGAII, hypervolume-based estimation (HypE), and cuckoo search (CS). The recently proposed multiobjective optimization algorithms include the following: MOAGED [48], IMOMRFO [49], and MO_Ring_PSO_SCD [50]. The above seven algorithms were independently run 30 times. The population size of the relevant algorithms was set to 300, and

Table 1: Solving results of each algorithm on PDLBP.

Algorithm	Solution	Disassembly task allocation scheme	Parts not disassembled	F_1	F_2	F_3
MIPSO	1	{12, 8, 27, 9, 10, 1, 3, 4, 13, 28, 25, 15} ⟶ {11, 24, 41, 7, 45, 20, 35, 36, 39} ⟶ {21, 40, 5, 33, 48, 30, 34, 23, 22, 49, 2, 14, 16} ⟶ {6, 18, 47, 42, 19, 50, 51, 52, 26, 43, 53, 44, 17}	46, 29, 31, 32, 38, 37, 54, 55	4	0	5.4978
	2	{12, 2, 3, 8, 10, 1, 13, 9, 11, 28, 30, 15, 27} ⟶ {4, 45, 7, 41, 20, 35, 39, 36} ⟶ {21, 40, 5, 33, 34, 25, 48, 23, 6, 49, 24, 14} ⟶ {16, 22, 18, 42, 47, 19, 50, 51, 52, 26, 43, 46, 44, 17, 53}	29, 31, 32, 38, 37, 54, 55	4	2	5.4168
	3	{2, 8, 6, 12, 10, 27, 13, 9} ⟶ {11, 28, 25, 15, 35, 4, 41, 7, 45, 30, 1, 36} ⟶ {39, 21, 40, 5, 33, 23, 20, 34, 47} ⟶ {43, 49, 24, 14, 16, 22, 18, 48, 42, 19, 50, 51, 52, 53, 3, 46, 44, 17}	26, 29, 31, 32, 54, 38, 37, 55	4	18	5.3940
Goal-driven discrete cuckoo search (GDCS)	1	{8, 10, 12, 9, 15, 11, 28, 7, 21, 25, 1} ⟶ {39, 40, 24, 30, 35, 20, 49, 43, 48, 45, 3, 46} ⟶ {4, 14, 47, 52, 50, 16, 36, 23, 33, 51, 34, 17, 22, 53, 44}	31, 32, 27, 6, 38, 18, 2, 5, 54, 41, 55, 37, 13, 19, 42, 26, 29	3	0	3.9024
	2	{8, 10, 12, 9, 15, 11, 41, 28, 7, 14} ⟶ {21, 25, 39, 40, 24, 35, 1, 20, 49} ⟶ {43, 48, 45, 47, 52, 30, 50, 16, 36, 23, 33, 51, 34, 17, 22, 3, 53, 44}	31, 32, 27, 46, 6, 38, 18, 4, 2, 5, 54, 55, 37, 13, 19, 42, 26, 29	3	1	4.0379
	3	{8, 10, 12, 9, 15, 11, 28, 7, 21, 25} ⟶ {39, 40, 24, 35, 20, 49, 43, 48, 45, 30, 3} ⟶ {14, 47, 52, 50, 16, 2, 36, 18, 1, 23, 33, 51, 34, 17, 22, 53, 44}	31, 32, 27, 46, 6, 38, 4, 5, 54, 41, 55, 37, 13, 19, 42, 26, 29	3	1	3.8482
	4	{8, 10, 12, 9, 15, 11, 28, 7, 21, 25, 1} ⟶ {39, 40, 24, 30, 35, 20, 49, 43, 48, 45, 3} ⟶ {4, 14, 47, 52, 50, 16, 36, 23, 33, 51, 34, 17, 22, 53, 44}	31, 32, 27, 46, 6, 38, 18, 2, 5, 54, 41, 55, 37, 13, 19, 42, 26, 29	3	4	3.9024
	5	{8, 10, 12, 9, 15, 11, 41, 28, 7, 14} ⟶ {21, 25, 39, 40, 24, 35, 20, 49} ⟶ {43, 48, 45, 47, 52, 30, 50, 16, 36, 23, 33, 51, 34, 17, 22, 3, 53, 44}	31, 32, 27, 46, 6, 38, 18, 4, 2, 5, 54, 1, 55, 37, 13, 19, 42, 26, 29	3	10	4.0379
	6	{8, 10, 12, 9, 15, 11, 28, 7, 21, 25} ⟶ {39, 40, 24, 35, 20, 49, 43, 48, 45, 30, 3} ⟶ {14, 47, 52, 50, 16, 2, 36, 18, 23, 33, 51, 34, 17, 22, 53, 44}	31, 32, 27, 46, 6, 38, 4, 5, 54, 1, 41, 55, 37, 13, 19, 42, 26, 29	3	10	3.8482
	7	{8, 10, 12, 9, 15, 11, 28, 7, 21, 25} ⟶ {39, 40, 24, 30, 35, 20, 49, 43, 48, 45, 3} ⟶ {4, 14, 47, 52, 50, 16, 36, 23, 33, 51, 34, 17, 22, 53, 44}	31, 32, 27, 46, 6, 38, 18, 2, 5, 54, 1, 41, 55, 37, 13, 19, 42, 26, 29	3	13	3.9024
	8	{8, 10, 12, 9, 15, 11, 28, 7, 21, 25} ⟶ {39, 40, 24, 35, 20, 49, 43, 48, 45, 30, 3} ⟶ {14, 47, 52, 50, 16, 36, 18, 23, 33, 51, 34, 17, 22, 53, 44}	31, 32, 27, 46, 6, 38, 4, 2, 5, 54, 1, 41, 55, 37, 13, 19, 42, 26, 29	3	37	3.8482
	9	{8, 10, 12, 9, 15, 11, 28, 7, 21, 25} ⟶ {39, 40, 24, 35, 20, 49, 43, 48, 45, 30, 3} ⟶ {14, 47, 52, 50, 16, 36, 23, 33, 51, 34, 17, 22, 53, 44}	31, 32, 27, 46, 6, 38, 18, 4, 2, 5, 54, 1, 41, 55, 37, 13, 19, 42, 26, 29	3	197	3.8482
HGNS	1	{8, 10, 11, 9, 7, 15, 39, 12} ⟶ {35, 20, 23, 33, 36, 40, 14, 30, 24, 43, 49, 48, 47} ⟶ {16, 21, 25, 34, 50, 44, 52, 28, 22, 3, 17, 51, 53}	4, 54, 31, 45, 1, 18, 2, 27, 13, 19, 6, 55, 32, 41, 46, 42, 26, 5, 38, 37, 29	3	2	3.6585
	2	{8, 10, 9, 11, 25, 15, 12, 35, 7, 30, 23} ⟶ {39, 40, 3, 48, 49, 21, 24, 33, 43, 44, 36} ⟶ {20, 52, 47, 18, 14, 50, 51, 28, 16, 34, 53, 17, 22}	41, 2, 54, 4, 31, 45, 32, 38, 27, 1, 19, 42, 5, 46, 37, 13, 6, 26, 55, 29	3	0	3.7127

Table 1: Continued.

Algorithm	Solution	Disassembly task allocation scheme	Parts not disassembled	F_1	F_2	F_3
INSGAII	1	{2, 18, 12, 4, 8, 10, 9, 21, 11, 25, 19, 33, 24} ⟶ {20, 15, 14, 16, 35, 30, 28, 7, 26, 23, 36} ⟶ {39, 40, 52, 49, 47, 43, 48, 50, 51}	1, 3, 5, 6, 13, 17, 22, 27, 29, 31, 32, 34, 37, 38, 41, 42, 44, 45, 46, 53, 54	**3**	**0**	**3.5700**
	2	{12, 8, 9, 10, 11, 35, 33, 15, 14} ⟶ {21, 30, 25, 45, 1, 24, 16, 7, 28, 39} ⟶ {40, 52, 20, 49, 47, 43, 48, 50, 51, 53}	2, 3, 4, 5, 6, 13, 17, 18, 19, 22, 23, 26, 27, 29, 31, 32, 34, 36, 37, 38, 41, 42, 44, 46, 54	**3**	**1**	**3.4417**
	3	{8, 10, 2, 12, 11, 35, 9, 15, 14} ⟶ {21, 33, 16, 25, 30, 7, 24, 28, 39} ⟶ {40, 20, 49, 52, 47, 43, 48, 50, 51}	1, 3, 4, 5, 6, 13, 17, 18, 19, 22, 23, 26, 27, 29, 31, 32, 34, 36, 37, 38, 41, 42, 44, 45, 46, 53, 54	**3**	**4**	**3.3604**

The bold values indicate the better indicator values.

Table 2: The 95% confidence intervals of the mean and standard deviation of each algorithm.

$\Phi^{-1}(\alpha)$	Method	Hypervolume			
		Mean		Standard deviation	
1.2816	INSGAII	**281173.65**	**281343.16**	**188.63**	**318.37**
	MOFPA	280,909.98	281,114.87	218.50	368.82
	NSGA3	279,870.36	280,235.73	389.63	657.69
	SPEA2 + SDE	279,943.70	280,292.78	372.26	628.36
	PICEA-g	279,711.80	280,138.83	391.40	660.68
	GrEA	279,948.64	280,329.10	405.73	684.87
	GypE	279,903.75	280,257.88	377.65	637.46
1.6449	INSGAII	**273,566.20**	275,101.45	**1708.44**	**2883.47**
	MOFPA	273,331.13	**275,143.88**	1933.14	3263.10
	NSGA3	267,998.79	270,561.51	2732.91	4613.08
	SPEA2 + SDE	267,148.01	270,137.60	3188.12	5381.47
	PICEA-g	270,426.69	273,003.83	2748.29	4639.04
	GrEA	268,414.38	271,018.23	2776.77	4687.12
	GypE	268,344.77	271,253.35	3101.73	5235.64
1.9600	INSGAII	**266,653.48**	**266,874.35**	**254.12**	**428.90**
	MOFPA	266,463.49	266,712.48	265.52	448.20
	NSGA3	263,461.75	264,375.71	974.66	1645.19
	SPEA2 + SDE	260,328.90	263,407.00	3282.51	5540.80
	PICEA-g	261,581.86	263,583.00	2134.03	3602.19
	GrEA	263,125.32	264,516.36	1483.42	2503.98
	GypE	261,509.61	263,988.25	2643.23	4461.72

Bold indicates the better indicator values.

the algorithms were terminated when the running time of each algorithm reached 600 s. The crossover probability and mutation probability related to genetic operations were set to 0.95 and 0.85, respectively. HypE considers 10,000 points for hypervolume estimation. The discovery probability P_a of the CS algorithm is set to 0.3. The parameter settings of MOAGED, IMOMRFO, and MO_Ring_PSO_SCD can be found in the corresponding references.

4.3.1. MMUPDUT. In this section, we will study MMUPDUT by taking a mixed disassembly line that disassembles printers, CRT TVs, and refrigerators simultaneously as an example. For detailed data on printers, CRT TVs, and refrigerators, please refer to Appendix A in Supplementary Material. Since hypervolume can effectively evaluate the quality of the nondominated solution set without the need for a reference set, the evaluation results of the quality of the nondominated solution set are more objective. Therefore, we use hypervolume to evaluate the quality of the solutions obtained by each algorithm. The reference point for hypervolume is set as (15, 0, 5000, 108). Table 3 shows the average and standard deviation of each algorithm on the HV indicator. In Table 3, α represents the probability that the disassembly time at each workstation on the disassembly line does not exceed the cycle time, and Rank represents the ranking of the performance of each algorithm. It should be noted that Rank is based on the mean ranking of the hypervolume. The upper limit of cycle time is 800 s.

From Table 3, it can be seen that the average value of the hypervolume indicator for INSGA is significantly greater than the other 6 methods, and in most cases, the standard deviation of the INSGA II algorithm is smaller than the other comparative algorithms. This indicates that compared to other algorithms, INSGA performs better in handling the MMUPDUT problem.

To better demonstrate the performance of each algorithm in handling the MMUPDUT problem, we recorded the hypervolume values obtained by each method after each run and visualized them. Figure 13 shows the hypervolume values for each method after 30 runs. From Figure 13, it can be seen that in most cases, the hypervolume indicator of the INSGA II algorithm is superior to other algorithms.

Box plots can effectively reflect the distribution of data, and with the help of box plots, we can intuitively analyze the stability and performance boundaries of each algorithm. Figure 14 shows the box plots of hypervolume obtained by different methods. The box part of the box plot (which represents the range from the lower quartile to the upper quartile) represents the distribution of 50% of the data. Therefore, the width of the box can to some extent reflect the fluctuation of the 50% values. A narrower or flatter box indicates less data volatility and greater stability, while a wider box indicates greater data volatility and less stability. From Figure 14, it can be seen that in the vast majority of cases, the box of INSGA II is narrower, which indicates that the INSGA II algorithm has better stability. In addition, the box plot allows us to visually observe the maximum and minimum values of the hypervolume for each algorithm. From Figures 14(a) and 14(b), it can be seen that at $\alpha = 0.90$ and $\alpha = 0.95$, the maximum value of the INSGA II algorithm is significantly greater than the other algorithms, and at $\alpha = 0.97$, the maximum value of the INSGA II algorithm is close to the maximum value of the MO_Ring_PSO_SCD algorithm and significantly greater than the other algorithms. These observations indicate that in the vast majority

Table 3: The mean and standard deviation of each algorithm when dealing with the MMUPDUT problem.

α	Method	Hypervolume		Rank
		Mean	Standard deviation	
0.90	INSGAII	**919,812.35**	15,355.33	**1**
	NSGAII	767,504.70	15,097.99	5
	HypE	856,625.84	30,551.05	3
	CS	687,873.74	31,809.48	7
	MOAGDE	886,193.59	15,386.15	2
	IMOMRFO	798,563.82	23,452.99	4
	MO_Ring_PSO_SCD	737,817.12	16,815.72	6
0.95	INSGAII	**800,376.57**	9,489.22	**1**
	NSGAII	770,199.51	15,040.03	4
	HypE	771,556.97	14,953.31	3
	CS	732,432.18	7,976.97	7
	MOAGDE	780,823.04	15,366.97	2
	IMOMRFO	763,920.80	10,481.43	5
	MO_Ring_PSO_SCD	760,128.55	13,257.23	6
0.97	INSGAII	**781,720.45**	4,586.09	**1**
	NSGAII	747,615.16	11,663.35	4
	HypE	722,994.97	6,165.96	7
	CS	745,915.72	7,504.68	5
	MOAGDE	731,581.15	6,058.78	6
	IMOMRFO	775,960.93	9,112.78	3
	MO_Ring_PSO_SCD	778,308.22	8,613.82	2

In table 3, bold indicates the better indicator values.

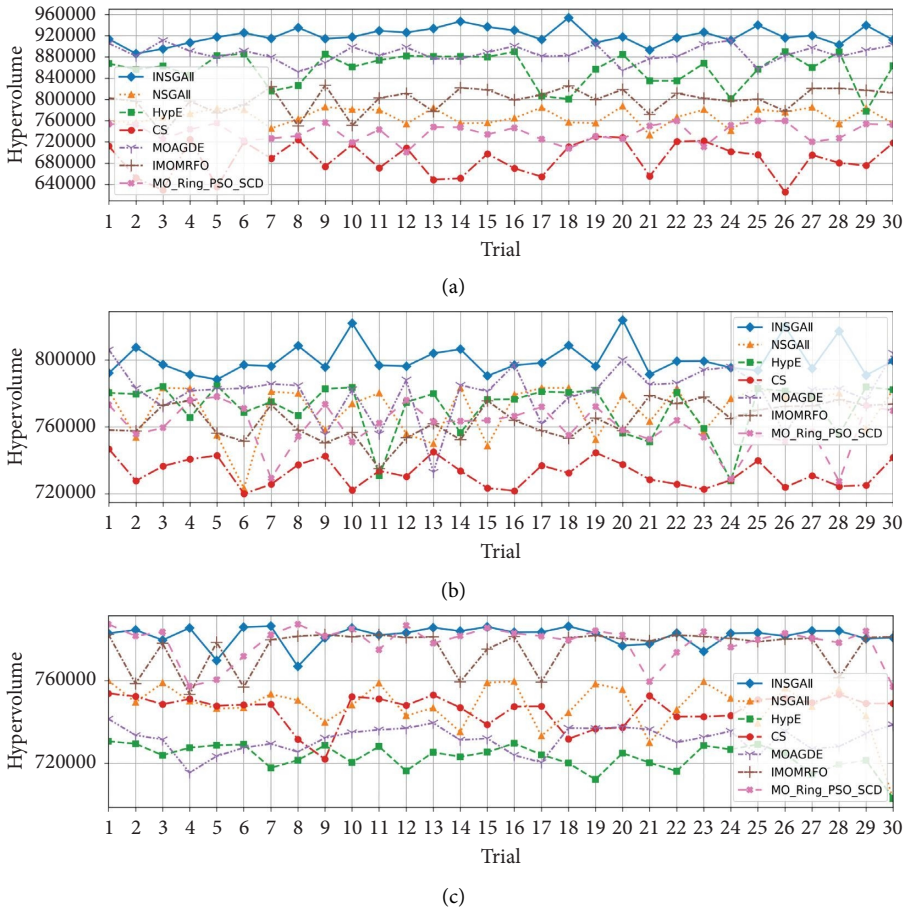


Figure 13: Hypervolume indicators for each algorithm solving the MMUPDUT problem. (a) Hypervolume indicator at $\alpha = 0.90$. (b) Hypervolume indicator at $\alpha = 0.95$. (c) Hypervolume indicator at $\alpha = 0.97$.

of cases, the upper performance limit of the INSGA II algorithm is better than that of other algorithms. Furthermore, from Figure 14, it can be seen that the minimum value of the hypervolume for the INSGA II algorithm is also greater than the minimum values of the other algorithms and even greater than the maximum values of some algorithms. This indicates that the lower performance limit of the INSGA II algorithm is also better than other algorithms. In conclusion, when handling the MMUPDUT problem, the performance of the INSGA II algorithm is superior to that of other algorithms.

4.3.2. MMUPDUT_S. In this section, we take a mixed disassembly line that simultaneously disassembles three EOL products: pumps, vises, and gearboxes, as an example to conduct research on MMUPDUT_S. For detailed data on pumps, vises, and gearboxes, please refer to Appendix B in Supplementary Material. In this section, we still use hypervolume to evaluate the quality of the solutions obtained by each algorithm. The reference point for hypervolume is set as (100, 0, 0, 100). Table 4 shows the average value and standard deviation of each algorithm on the HV indicator. In Table 4, α represents the probability that the disassembly time at each workstation on the disassembly line does not exceed the cycle time and Rank represents the ranking of algorithm performance. It should be noted that Rank is obtained based on the average value of hypervolume. The upper limit of cycle time is 100 s.

From Table 4, it can be observed that in all cases, the INSGA II algorithm obtains a higher average hypervolume than other algorithms, with a smaller variance. This indicates that compared to other algorithms, the INSGA II algorithm can consistently obtain high-quality disassembly solutions. This aligns with the performance of the algorithm in solving the MMUPDUT problem.

To provide a clearer visualization of the performance of each algorithm in handling the MMUPDUT_S problem, we recorded the hypervolume values obtained by each method after each run and visualized them. Figure 15 shows the hypervolume values for each method over 30 runs. From Figure 15, it can be seen that in most cases, the hypervolume indicator of the INSGA II algorithm is superior to that of other algorithms. This indicates that the higher average hypervolume value obtained by the INSGA II algorithm is not due to a random occurrence.

To better analyze the performance of each algorithm, we will use box plots to visually analyze the stability and performance bounds of each algorithm. Figure 16 shows the box plots of hypervolume obtained by different methods in solving the MMUPDUT_S problem. From Figure 16, it can be seen that in the vast majority of cases, the box of INSGA II is narrower, indicating that the INSGA II algorithm has better stability to some extent. Additionally, the box plots provide a visual representation of the maximum and minimum values of hypervolume for each algorithm. From Figures 16(a) and 16(b), it can be observed that at $\alpha = 0.90$ and $\alpha = 0.95$, the maximum value of the INSGA II algorithm is significantly higher than that of other algorithms. At $\alpha = 0.97$, the maximum value of the INSGA II algorithm is close to the maximum value of the NSGA II algorithm and significantly higher than that of other algorithms. However, it is worth noting that the data for the maximum hypervolume achieved by the NSGA II algorithm is considered an outlier. Therefore, compared to the NSGA II algorithm, the INSGA II algorithm still has a significant advantage. These observations indicate that in the vast majority of cases, the upper performance bound of the INSGA II algorithm is superior to that of other algorithms. Furthermore, from Figure 16, it can be seen that the minimum value of hypervolume for the INSGA II algorithm is also higher than the minimum values of other algorithms, indicating that the lower performance bound of the INSGA II algorithm is also superior to that of other algorithms. In conclusion, when dealing with the MMUPDUT_S problem, the performance of the INSGA II algorithm is superior to that of other algorithms.

4.3.3. Nonparametric Test. To ensure that the experimental results have statistical significance for comparison, two nonparametric test methods [51], including the Wilcoxon rank-sum test and the Friedman test, will be used to analyze the experimental results. The significance level is set to 0.05. These two tests serve different purposes. The former tests whether there is a significant difference between the INSGA II algorithm and other comparison algorithms, with the symbols "+", "=", and "−" indicating whether the performance of the comparison algorithms is better than, equal to, or worse than the INSGA II algorithm, respectively. The latter test is used at the algorithm level to provide an overall performance of the algorithm in terms of average rankings, where a smaller ranking indicates better algorithm performance.

(1) Wilcoxon Rank-Sum Test. Table 5 shows the results of the Wilcoxon rank-sum test for each algorithm in solving MMUPDUT and MMUPDUT_S. From Table 5, it can be seen that when solving each instance, INSGA II is significantly better than most algorithms. This indicates that the conclusion that INSGA II is superior to other algorithms is not due to random factors but has statistical significance.

(2) Friedman Test. Table 6 shows the results of the Friedman test. From Table 6, it can be seen that the p value is much smaller than 0.05, indicating significant performance differences among the algorithms. Additionally, from Table 6, it can be seen that the average rank of INSGA II is 1, which is the smallest among all algorithms. This indicates that the performance of INSGA II is the best. Therefore, the INSGA II algorithm has the best overall performance in solving MMUPDUT and MMUPDUT_S.

4.3.4. Sensitivity Analysis. The Monte Carlo Tree Simulation-based Initialization (MCTI) and Group Global Crowd Degree Comparison (GCDC) are crucial parts of the INSGA II algorithm. It is necessary to explore the impact of MCTI and GCDC on the performance of the INSGA II

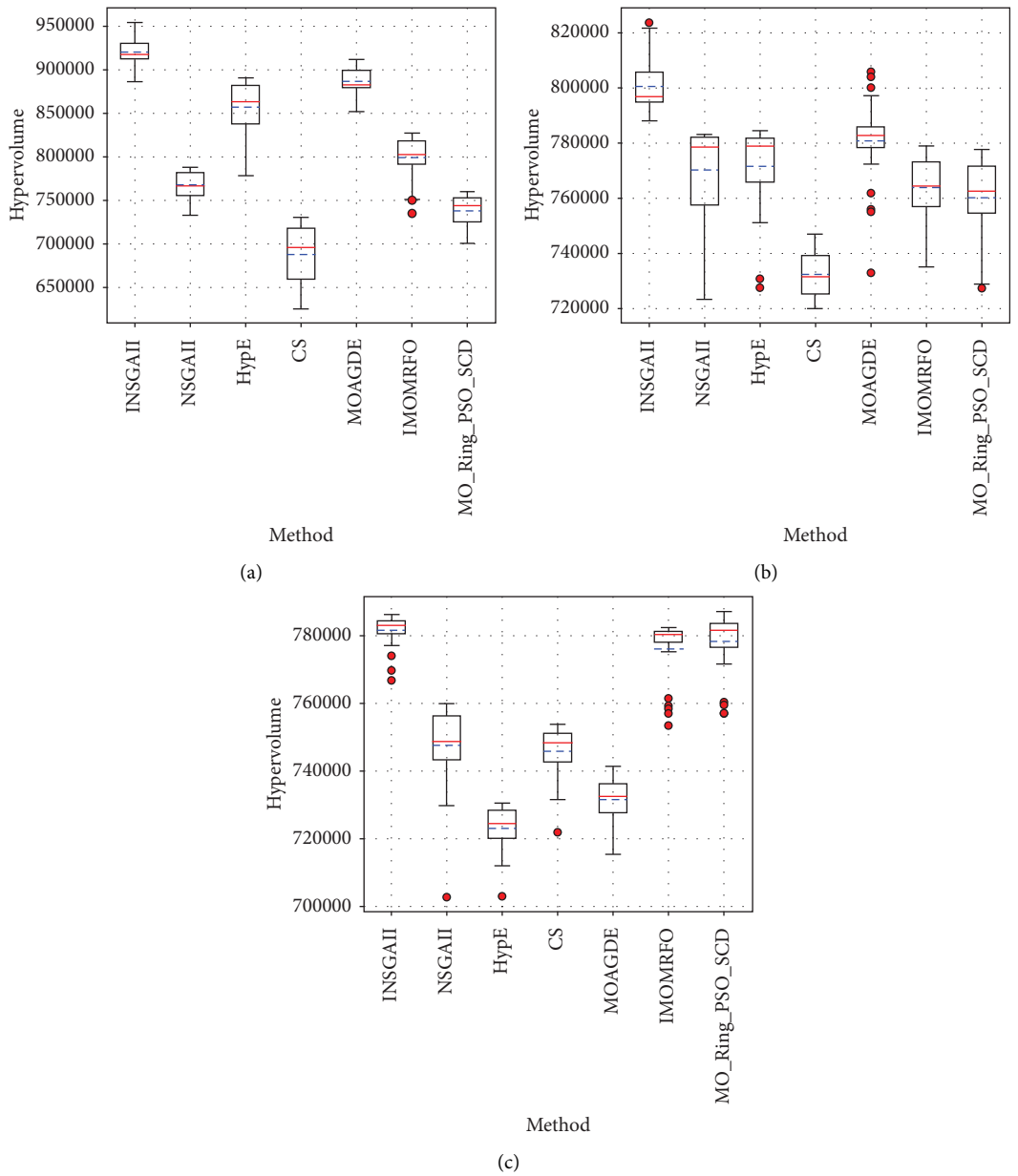


Figure 14: Box plot of hypervolume indicators for various algorithms in solving the MMUPDUT problem. (a) Box plot of hypervolume at $\alpha = 0.90$. (b) Box plot of hypervolume at $\alpha = 0.95$. (c) Box plot of hypervolume at $\alpha = 0.97$.

Table 4: The mean and standard deviation of each algorithm when dealing with the MMUPDUT_S problem.

α	Method	Hypervolume		Rank
		Mean	Standard deviation	
0.90	INSGAII	**1027904.45**	7831.97	**1**
	NSGAII	998766.63	16570.68	6
	HypE	1005785.53	13065.70	4
	CS	983370.36	13558.27	7
	MOAGDE	1009661.18	7146.24	2
	IMOMRFO	1005607.40	13058.97	5
	MO_Ring_PSO_SCD	1009357.54	11098.93	3

Table 4: Continued.

α	Method	Hypervolume		Rank
		Mean	Standard deviation	
0.95	INSGAII	**1015959.21**	8473.84	**1**
	NSGAII	988092.04	15686.76	5
	HypE	984777.32	11445.48	6
	CS	980268.16	15388.35	7
	MOAGDE	998536.12	13922.41	2
	IMOMRFO	994426.92	13033.31	4
	MO_Ring_PSO_SCD	996679.05	13503.60	3
0.97	INSGAII	**1014381.74**	7486.81	**1**
	NSGAII	999400.62	14074.86	4
	HypE	987743.40	14343.01	6
	CS	980177.78	11832.76	7
	MOAGDE	1003867.37	11750.23	2
	IMOMRFO	996541.55	15145.87	5
	MO_Ring_PSO_SCD	1003843.41	13800.82	3

In table 4, bold indicates the better indicator values.

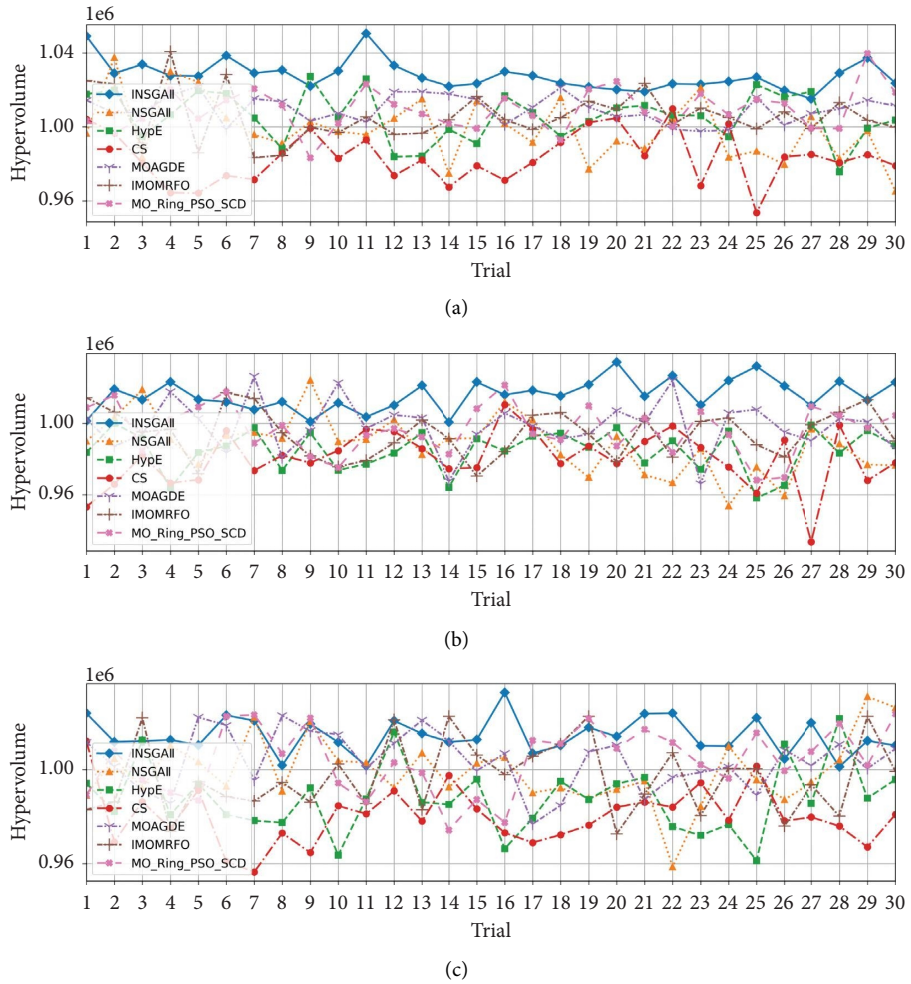


Figure 15: Hypervolume indicators for each algorithm solving the MMUPDUT_S problem. (a) Hypervolume indicator at $\alpha = 0.90$. (b) Hypervolume indicator at $\alpha = 0.95$. (c) Hypervolume indicator at $\alpha = 0.97$.

algorithm through experiments. In this section, we will use the INSGA II algorithm without MCTI (INSGA II without MCTI) and the INSGA II algorithm without GCDC (INSGA II without GCDC) to solve the MMUPDUT problem. We will compare and analyze the results with those obtained by the INSGA II and NSGA II algorithms. It should be noted

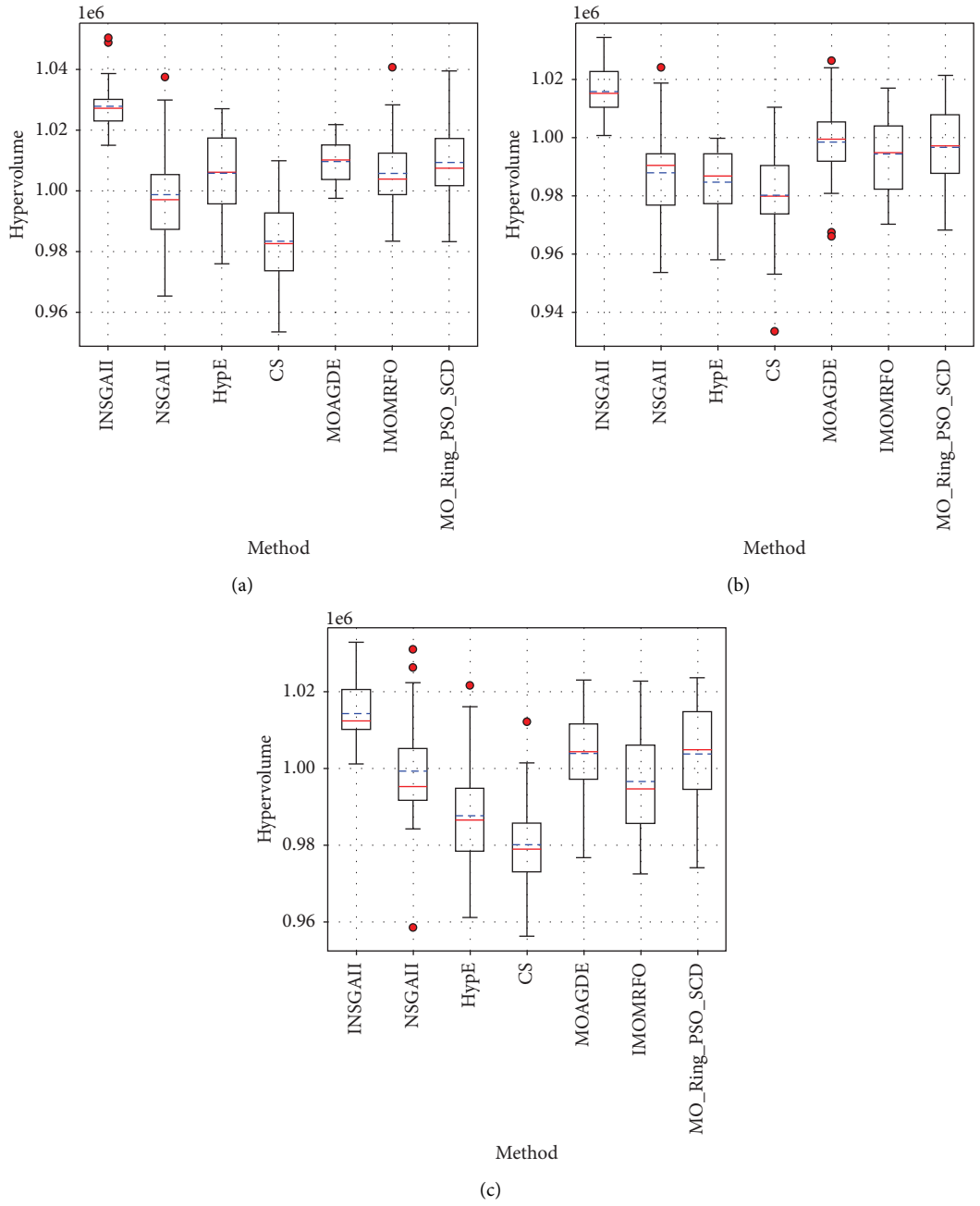


FIGURE 16: Box plot of hypervolume indicators for various algorithms in solving the MMUPDUT_S problem. (a) Box plot of hypervolume at $\alpha = 0.90$. (b) Box plot of hypervolume at $\alpha = 0.95$. (c) Box plot of hypervolume at $\alpha = 0.97$.

TABLE 5: Wilcoxon rank-sum test results.

Instances	α	NSGAII	HypE	CS	MOAGDE	IMOMRFO	MO_Ring_PSO_SCD
MMUPDUT	0.90	1.43×10^{-11} (–)	1.93×10^{-11} (–)	1.43×10^{-11} (–)	1.06×10^{-9} (–)	1.43×10^{-11} (–)	1.43×10^{-11} (–)
	0.95	1.43×10^{-11} (–)	1.43×10^{-11} (–)	1.43×10^{-11} (–)	1.05×10^{-7} (–)	1.43×10^{-11} (–)	1.43×10^{-11} (–)
	0.97	1.43×10^{-11} (–)	1.43×10^{-11} (–)	1.43×10^{-11} (–)	1.43×10^{-11} (–)	6.05×10^{-5} (–)	4.18×10^{-2} (=)
MMUPDUT_S	0.90	7.46×10^{-9} (–)	1.16×10^{-9} (–)	1.43×10^{-11} (–)	6.32×10^{-11} (–)	1.90×10^{-8} (–)	5.27×10^{-9} (–)
	0.95	5.75×10^{-9} (–)	1.43×10^{-11} (–)	3.18×10^{-11} (–)	8.96×10^{-7} (–)	3.40×10^{-8} (–)	8.99×10^{-8} (–)
	0.97	3.25×10^{-6} (–)	8.13×10^{-9} (–)	6.32×10^{-11} (–)	1.83×10^{-4} (–)	4.29×10^{-6} (–)	1.34×10^{-3} (–)

Table 6: Friedman test.

	INSGAII	NSGAII	HypE	CS	MOAGDE	IMOMRFO	MO_Ring_PSO_SCD
Mean of rank	1	4.67	4.83	6.67	2.67	4.33	3.83
p value				0.0003975425074978762			

that the INSGA II without MCTI uses a random initialization method instead of MCTI, and the INSGA II without GCDC uses the traditional crowding calculation method [41] instead of GCDC.

(1) Overall Performance Comparison of the Algorithms. According to Table 7, it can be seen that INSGA II has the best performance, and its performance decreases when MCTI or GCDC is removed. This indicates that both MCTI and GCDC play important roles in improving the performance of the INSGA II algorithm. Especially, the performance of the INSGA II algorithm decreases significantly when GCDC is removed. This suggests that compared to MCTI, GCDC plays a more crucial role in enhancing the performance of INSGA II. As mentioned in Section 3.5, GCDC is an elitism preservation strategy used by INSGA II to determine which individuals can be retained. The decrease in algorithm performance when the traditional crowding calculation method replaces GCDC indicates that GCDC can better select elite individuals.

(2) Algorithm Stability Analysis. In this section, we studied the success rate and search time of the algorithm in searching for feasible solutions and analyzed their stability. Specifically, following the approach described in the literature [48], we used the success rate (SR), mean fitness evaluation (MFE) number, and mean search time (MST) as evaluation metrics to assess the algorithms. SR reflects the success rate of finding feasible solutions, with a higher SR indicating a higher success rate and better performance of the corresponding algorithm. MFE reflects the mean fitness evaluation number of the algorithm when finding feasible solutions, and MST reflects the mean search time of the algorithm when finding feasible solutions. Smaller MFE and MST values indicate that the algorithm can find feasible solutions faster, indicating better performance. Please refer to the literature [48] for the specific calculation methods of SR, MFE, and MST. Additionally, it should be noted that according to the literature [48], we selected the top two algorithms with the best performance from Table 7 and assigned the average value of their hypervolume as the feasible solution for the corresponding problem. In this section, each algorithm was run independently for 30 times, with 500 iterations for each algorithm. The objective function was calculated once per iteration, and the algorithm stopped running when the evaluation count of the objective function reached 500. According to the results in Table 7, the feasible solutions for each case are shown in Table 8.

Table 9 shows the SR, MFE, and MST of each algorithm. According to the success rate shown in the SR column of Table 9, INSGA II performs more stably in finding feasible solutions, outperforming its competitors. This indicates that using both MCTI and GCDC together is beneficial for improving the stability of the algorithm. Additionally, INSGA II without MCTI ranks second in terms of SR performance, just behind INSGA II, while INSGA II without GCDC ranks third and significantly lags behind INSGA II and INSGA II without MCTI. This phenomenon suggests that both MCTI and GCDC contribute to improving the stability of the algorithm, but GCDC contributes more significantly. Furthermore, NSGA II struggles to find feasible solutions, indicating that it falls into local solution traps and converges prematurely in the search process. Considering that the data in the MFE and MST columns were calculated using only the averages of successful trials, INSGA II required fewer fitness evaluations, i.e., less effort, to find feasible solutions compared to INSGA II without MCTI and INSGA II without GCDC. Additionally, compared to INSGA II without GCDC, INSGA II without MCTI has a lower average value of MFE, indicating that GCDC can help the algorithm find feasible solutions more quickly. The data in the MST column further confirm this phenomenon.

In conclusion, both MCTI and GCDC proposed in this paper are beneficial for improving the performance of the algorithm. However, GCDC plays a more significant role in enhancing algorithm performance. Additionally, through the above analysis, it can be observed that the greatest help in improving algorithm performance comes from the combined use of MCTI and GCDC.

5. Analysis of Disassembly Schemes and Managerial Insights

5.1. Analysis of Disassembly Schemes. In this study, we propose two disassembly line balancing problems, including MUPDLBP and MUPDLBP_S, and provide corresponding constraint planning models. MUPDLBP and MUPDLBP_S have certain similarities. Let us start by analyzing the disassembly schemes based on the main similarities between MUPDLBP and MUPDLBP_S.

Both MUPDLBP and MUPDLBP_S aim to improve disassembly efficiency and reduce disassembly costs by integrating the advantages of mixed-model disassembly lines and U-shaped layout disassembly lines. We will analyze the obtained disassembly schemes to verify the promoting effect of mixed-model disassembly lines and U-shaped layout disassembly lines on improving disassembly efficiency and reducing disassembly costs.

First, let us analyze the benefits of mixed-model disassembly lines. The benefits of mixed-model disassembly lines are very intuitive. When multiple products need to be disassembled simultaneously, without using mixed-model disassembly lines, multiple disassembly lines need to be established, and necessary equipment needs to be allocated for each line, which incurs additional costs. However, using mixed-model disassembly lines can save these costs.

Table 7: The mean and standard deviation of each algorithm in sensitivity analysis.

α	Method	Hypervolume		Rank
		Mean	Standard deviation	
0.90	INSGAII	919812.35	15355.33	**1**
	NSGAII	767504.70	15097.99	4
	INSGAII without MCTI	804842.32	29946.15	2
	INSGAII without GCDC	771805.66	16790.75	3
0.95	INSGAII	800376.57	9489.22	**1**
	NSGAII	770199.51	15040.03	4
	INSGAII without MCTI	780126.27	9654.02	2
	INSGAII without GCDC	776265.37	13060.79	3
0.97	INSGAII	781720.45	4586.09	**1**
	NSGAII	747615.16	11663.35	4
	INSGAII without MCTI	756621.59	12475.30	2
	INSGAII without GCDC	751447.24	7164.87	3

In table 7, bold indicates the better indicator values.

Table 8: Feasible solutions.

MMUPDUT ($\alpha = 0.90$)	MMUPDUT ($\alpha = 0.95$)	MMUPDUT ($\alpha = 0.97$)
862,327.335	790,251.42	769,171.02

Next, let us analyze the benefits generated by the U-shaped layout disassembly lines. To better analyze the benefits generated by the disassembly process of U-shaped layout disassembly lines, we randomly selected multiple disassembly schemes and solved the corresponding disassembly schemes for linear layout disassembly lines and disassembly schemes without partial disassembly. After modifying the disassembly schemes, it was generally observed that the disassembly efficiency decreased or the number of workstations increased. Considering the similarity of these phenomena and due to space limitations, we randomly selected one typical example for analysis when $\alpha = 0.90$, $\alpha = 0.95$, and $\alpha = 0.97$. The maximum disassembly time in all workstations is taken as the actual cycle time T_c, and the workload rate of each workstation is denoted as Δ, $\Delta = T_k/T_c \times 100\%$.

Based on Figure 17, it can be seen that when using U-shaped layout disassembly lines and partial disassembly methods, the workload rates of each workstation are above 90%. When changing the U-shaped layout disassembly lines to linear layout disassembly lines, some workstations experience a significant decrease in workload rates. For example, in Figure 17(a), when changing from U-shaped layout to linear layout, the workload rate of some workstations is only 27.59%. Additionally, as shown in Figure 17(a), compared to linear layout disassembly lines, using U-shaped layout disassembly lines can reduce the number of workstations. From Figures 17(b) and 17(c), although using U-shaped layout disassembly lines may not necessarily reduce the number of activated workstations, the actual cycle time T_c is reduced, which effectively improves disassembly efficiency. In summary, this indicates that U-shaped layout disassembly lines are beneficial for improving disassembly efficiency, reducing the number of activated workstations, and reducing the investment cost of disassembly lines.

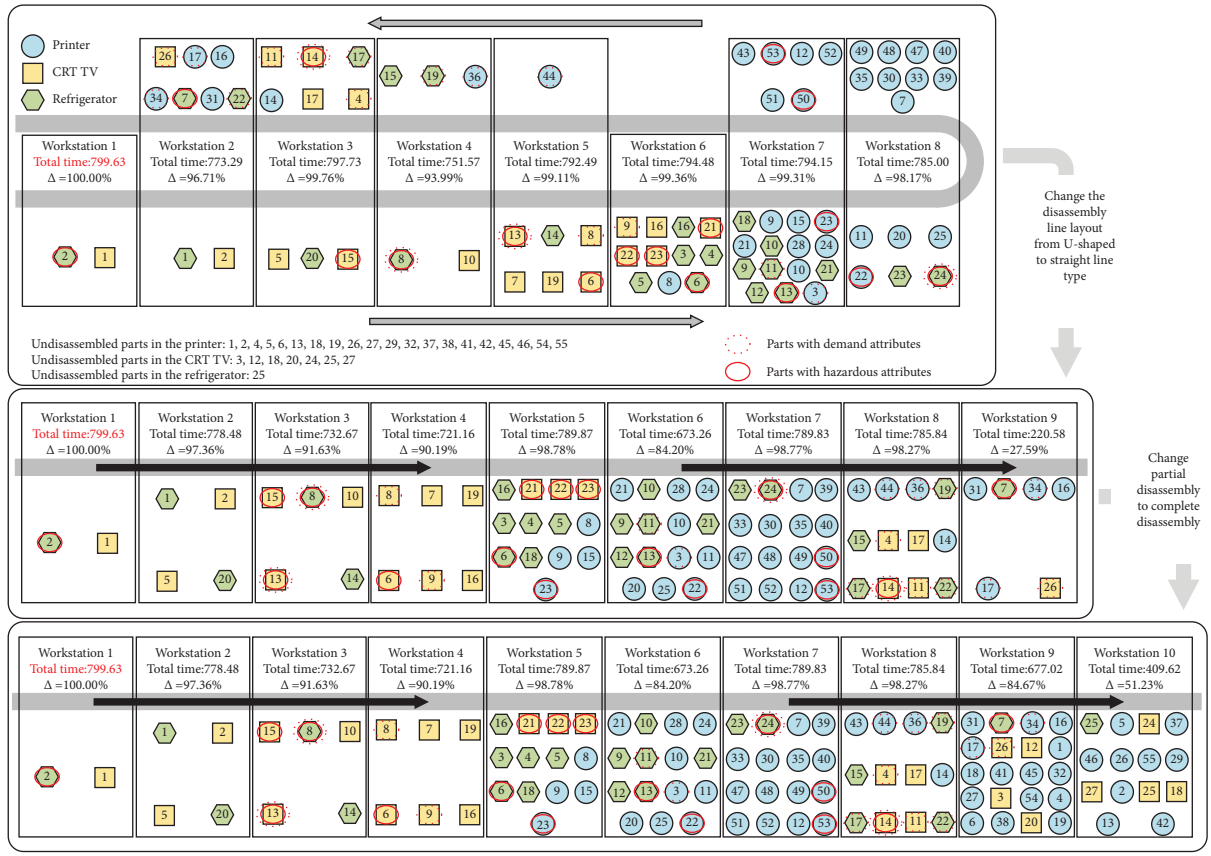
Based on Figure 18, when using U-shaped layout disassembly lines and partial disassembly methods, the workload rates of each workstation are above 78%. When changing the U-shaped layout disassembly lines to linear layout disassembly lines, some workstations experience a significant decrease in workload rates. For example, in Figure 18(a), the workload rates of each workstation are above 86%, but when changing the layout to linear layout, the lowest workload rate of a workstation is only 41.89%. Additionally, as shown in Figure 18, compared to linear layout disassembly lines, using U-shaped layout disassembly lines can effectively reduce the number of workstations.

MUPDLBP and MUPDLBP_S have some similarities and differences. Compared to MUPDLBP_S, the most significant difference in MUPDLBP is the use of partial disassembly methods. Figure 17 shows that using partial disassembly methods can effectively reduce the number of components that need to be disassembled. By reducing the number of components to be disassembled, it may also reduce the number of activated workstations. As shown in Figure 17, when using complete linear disassembly, the number of activated workstations is higher compared to using partial linear disassembly or U-shaped partial disassembly.

Compared to MUPDLBP, MUPDLBP_S has two major differences: (1) MUPDLBP_S minimizes the number of changes in disassembly direction as an optimization objective; (2) MUPDLBP_S considers the stability of the assembly during the disassembly process. From Figure 18, it can be seen that in the disassembly schemes obtained through MUPDLBP_S, the number of changes in disassembly direction along the conveyor belt does not exceed 8 times. This effectively reduces the number of turning operations on the assembly during the disassembly process, making it easier for the disassembly personnel to complete the task. As for the stability of the disassembly process,

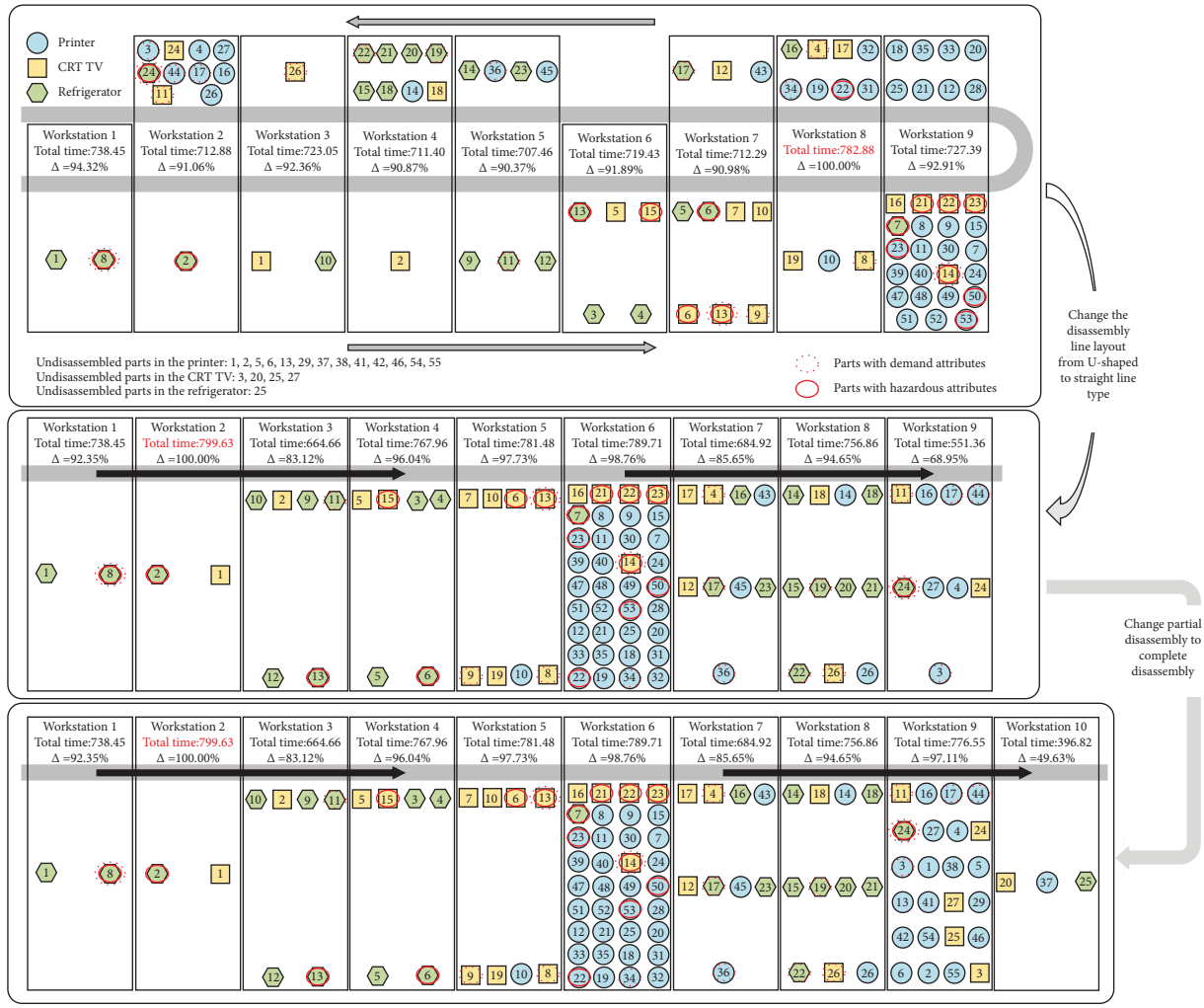
Table 9: SR, MFE, and MST performances of algorithms.

α	SR (%)				MFE				MST (s)			
	INSGAII	NSGAII	INSGAII without MCTI	INSGAII without GCDC	INSGAII	NSGAII	INSGAII without MCTI	INSGAII without GCDC	INSGAII	NSGAII	INSGAII without MCTI	INSGAII without GCDC
0.90	100.00	0.00	0.00	0.00	229.50	0.00	0.00	0.00	211.98	0.00	0.00	0.00
0.95	96.67	0.00	53.33	3.33	243.00	0.00	325.00	331.00	227.16	0.00	290.76	278.86
0.97	96.67	0.00	40.00	0.00	193.00	0.00	269.00	0.00	178.81	0.00	172.83	0.00
Mean	97.78	0.00	46.67	3.33	221.83	0.00	297.00	331.00	205.98	0.00	231.80	278.86



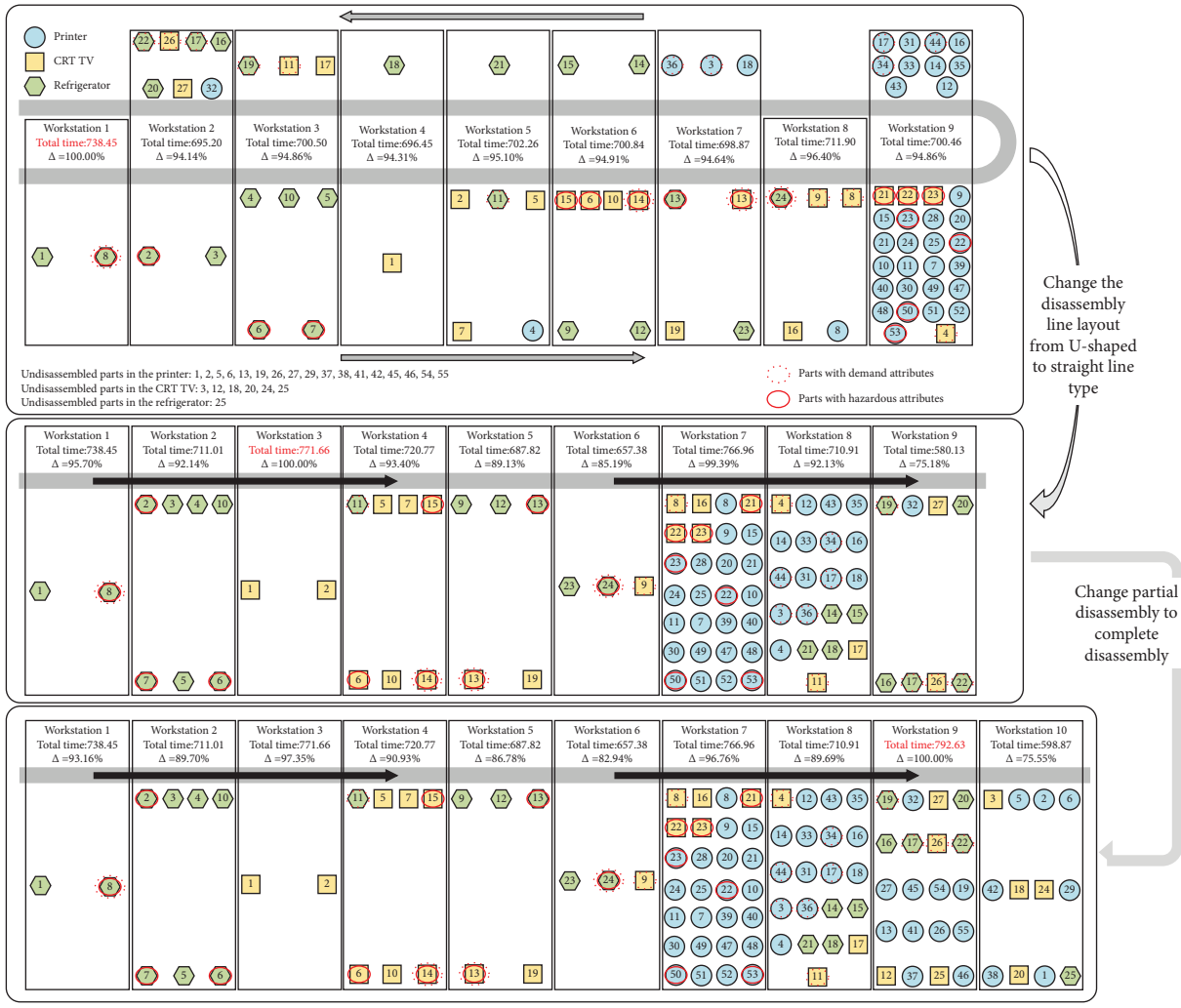
(a)

Figure 17: Continued.



(b)

Figure 17: Continued.



(c)

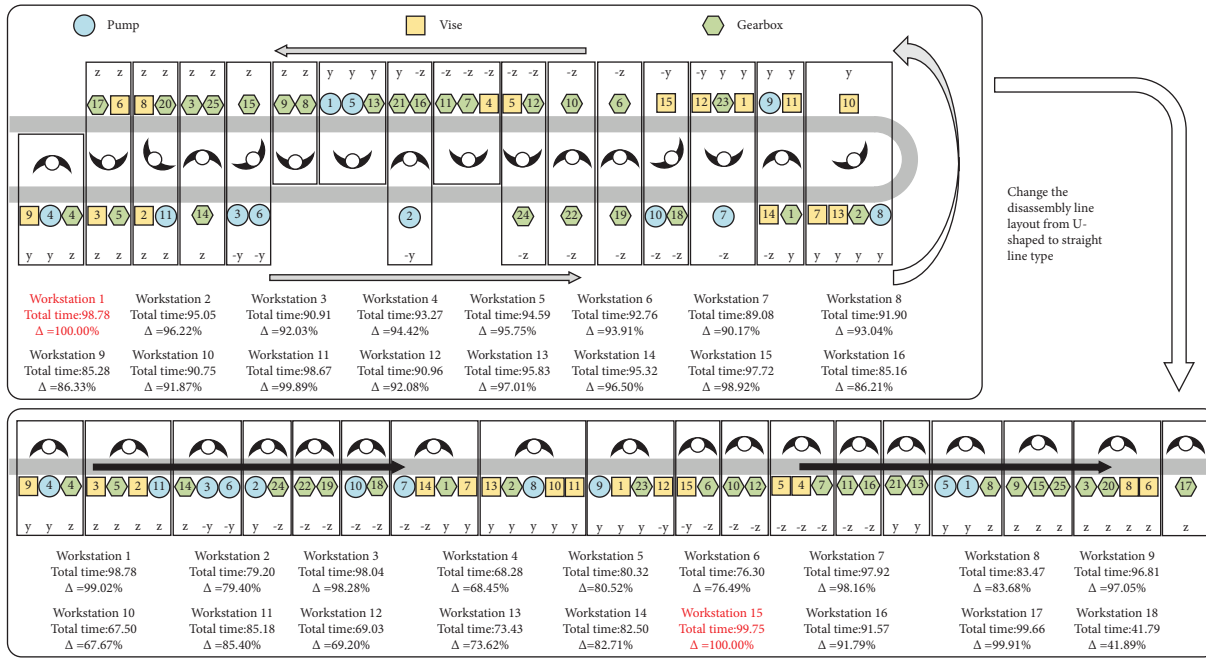
Figure 17: Disassembly scheme obtained when solving the MMUPDUT problem. (a) Disassembly scheme for $\alpha = 0.90$. (b) Disassembly plan for $\alpha = 0.95$. (c) Disassembly scheme for $\alpha = 0.97$.

although it is difficult to directly observe from the figures, combining the description in Section 2.4.5 and the disassembly scheme shown in Figure 18, it can be inferred that the assembly can maintain good stability during the disassembly process.

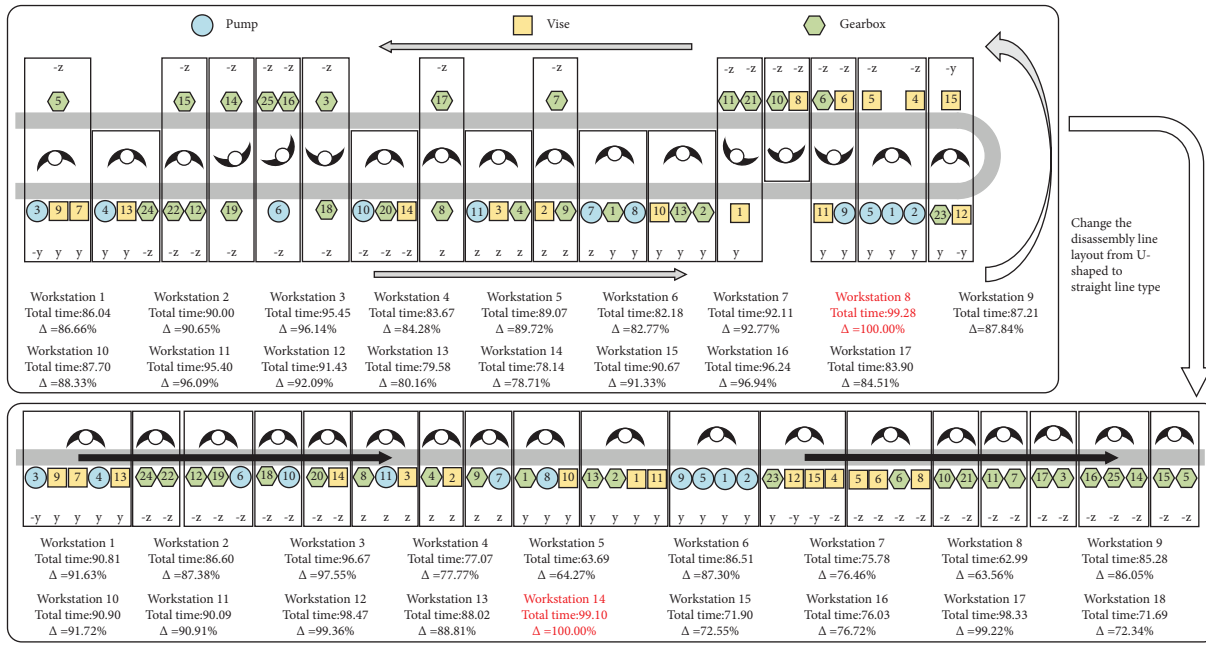
In conclusion, the proposed MUPDLBP and MUPDLBP_S in this study can obtain good disassembly schemes.

5.2. Managerial Insights. As environmental awareness increases, governments around the world have successively issued policies to promote the development of the remanufacturing industry, making product recycling and remanufacturing a critical focus for manufacturers. Disassembly is one of the most crucial aspects of the product recycling and remanufacturing process, and disassembly lines are the optimal choice for large-scale disassembly. Rational design of disassembly lines and achieving disassembly line balance can enhance disassembly efficiency. To improve disassembly efficiency and reduce input costs, this research comprehensively considers the features of mixed-flow disassembly lines and U-shaped layout disassembly lines, establishing a constraint planning model. As can be seen from the analysis in Section 5.1, this model can provide managers with more efficient and cost-effective disassembly solutions. Therefore, it is recommended that managers consider designing the disassembly line layout as U-shaped and use mixed-flow disassembly methods. Additionally, when the disassembly factory has automated crushers and sorting devices, they can consider partial disassembly, effectively reducing the number of components that need to be disassembled and the number of workstations that need to be opened.

In addition to considering the features of mixed-flow disassembly lines and U-shaped layout disassembly lines in the constraint planning model, this research also proposes two types of disassembly line balancing problems based on the characteristics of two different types of end-of-life (EOL) products and provides constraint planning models: (1) This

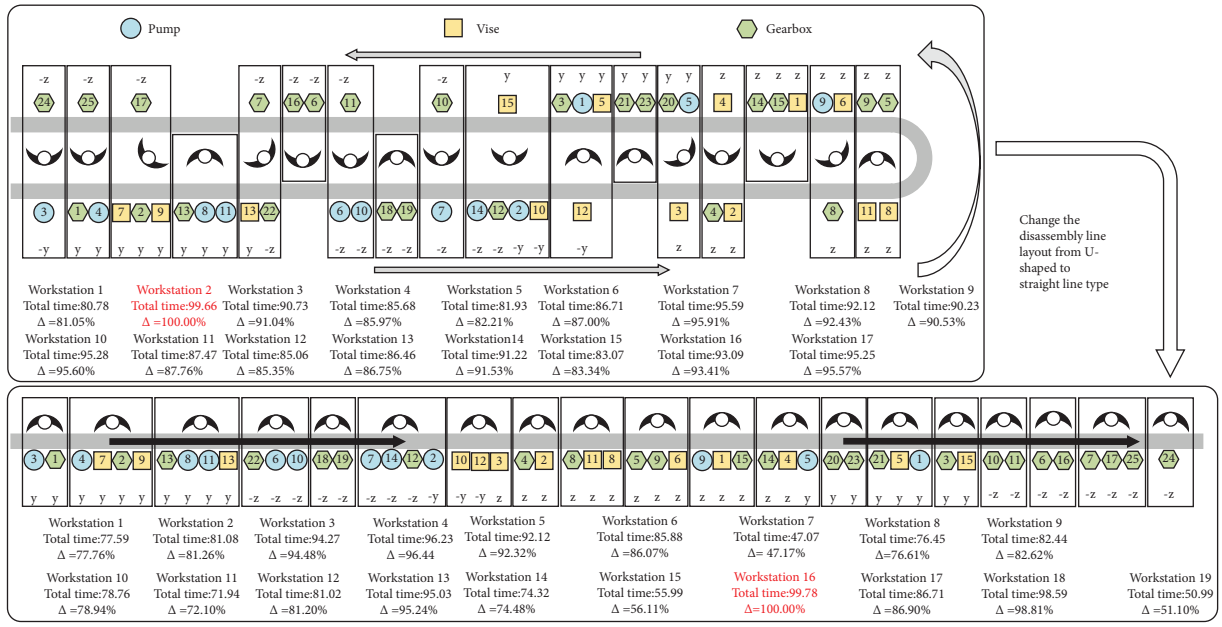


(a)



(b)

Figure 18: Continued.



(c)

Figure 18: Disassembly scheme obtained when solving the MUPDLBP_S problem. (a) Disassembly scheme for $\alpha = 0.90$. (b) Disassembly scheme for $\alpha = 0.95$. (c) Disassembly scheme for $\alpha = 0.97$.

research proposes a stochastic multiobjective multiproduct U-shaped mixed-flow incomplete disassembly line balancing problem (MUPDLBP) for waste household appliances. These appliances have small component quality, low component rigidity, a large span of product usage time, and significant differences in usage conditions. This model can effectively guide the disassembly of waste household appliances as it combines the benefits of mixed-flow disassembly lines and U-shaped layout disassembly lines and considers the uncertainty of the disassembly process. This not only provides efficient disassembly solutions but also ensures a more balanced state of the disassembly line. (2) This research also proposes a stochastic multiobjective multiproduct U-shaped mixed-flow disassembly line balancing problem (MUPDLBP_S) considering the stability of the assembly and excluding certain disassembly methods for EOL products with large weight and rigidity. This model can maintain good stability during the disassembly process, effectively avoiding product component collapse that could harm disassembly personnel and damage components.

In addition to establishing the constraint planning model, this research also proposes a metaheuristic algorithm called INSGAII. Compared to other algorithms, the INSGAII algorithm can provide managers with higher quality solutions.

Through this research, it can be seen that when designing disassembly lines, the comprehensive use of the characteristics of mixed-flow disassembly lines and U-shaped layout disassembly lines can effectively improve disassembly efficiency. When conditions permit, manufacturers can also reduce the number of components that need to be disassembled by introducing automated crushers and sorting devices, which is expected to further reduce the number of workstations that need to be opened. In summary, manufacturers can use the constraint planning model proposed in this research to guide the disassembly process. Specifically, manufacturers can use MUPDLBP to guide the disassembly process of waste household appliances or use MUPDLBP_S to guide the disassembly process of EOL products with large weight and rigidity. In addition, manufacturers can effectively handle the disassembly line balancing problem they face with the help of INSGAII.

6. Summary and Discussion

6.1. Summary. In this research, we proposed two disassembly line balancing problems, MUPDLBP and MUPDLBP_S, for two different types of end-of-life (EOL) products and formulated the corresponding constraint programming models. First, these two proposed problems and their corresponding constraint programming models have certain similarities. The most important commonality is that they both design the disassembly line as a U-shaped mixed-flow disassembly line. As analyzed in Section 5.1, this design method helps to improve disassembly efficiency and reduce disassembly costs. In addition to the above similarities, MUPDLBP and MUPDLBP_S also have certain similarities in setting optimization goals: (1) both aim to maximize the Smoothness Indicator (SI) of the workload; (2) both aim to minimize the number of workstations opened (NS); and (3) both consider the uncertainty of the disassembly process and handle this uncertainty by describing work time in a probabilistic manner. Among these three similarities, the first is to improve disassembly efficiency; the second is to reduce input costs (opening fewer workstations can also effectively reduce costs); the third is to reduce the

impact of random factors on the disassembly process and make the disassembly line as balanced as possible. Apart from the above similarities, there are also some differences between MUPDLBP and MUPDLBP_S. The main differences are as follows: (1) MUPDLBP adopts a partial disassembly method that only disassembles parts with hazardous and demand properties, while MUPDLBP_S adopts a complete disassembly method; (2) MUPDLBP_S considers the stability of the assembly. The reason for these differences is that they target different EOL products. Specifically, MUPDLBP is proposed for obsolete home appliances with small part quality, low part rigidity, and a large span of usage time and usage condition differences; while MUPDLBP_S is proposed for EOL products with large weight and high rigidity. On the one hand, because the parts of obsolete home appliances are small in quality and low in rigidity, they are easily broken, so MUPDLBP adopts a partial disassembly method that only disassembles parts with hazardous and demand properties. Using this partial disassembly method can effectively reduce the number of parts to be disassembled, which is conducive to reducing the number of workstations. It is worth noting that for EOL products with larger part rigidity, the above partial disassembly method cannot be used, as dealing with rigid objects will damage the automatic shredder, so MUPDLBP_S can only use the complete disassembly method. On the other hand, when disassembling EOL products with large weight and high rigidity, if the assembly becomes unstable, it is likely to cause the parts to collapse, thereby causing harm to the disassembly personnel, so it is necessary to consider the stability of the assembly in MUPDLBP_S. Analysis in Section 5 shows that the disassembly schemes obtained through MUPDLBP and MUPDLBP_S are effective.

Considering the difficulty of solving MUPDLBP, we further propose a new algorithm called INSGAII, which is an improved version based on the NSGAII algorithm. The main improvements include the Monte Carlo Tree Simulation-based Initialization (MCTI) method and the Group-based Global Crowding Distance Comparison (GCDC) operator. MCTI is proposed to address the problem of the traditional initialization method's inability to cover the entire search space well. It estimates the distribution of feasible disassembly sequences in FDST using Monte Carlo methods and guides the generation of the initial population based on the estimation results, resulting in a better coverage of the entire search space by the initial population. GCDC is proposed to solve the issue of poor diversity and uniformity of the Pareto solution set when dealing with DLBP caused by the crowding distance comparison operator of NSGAII overly protecting boundary individuals. GCDC groups the candidate individuals for processing, avoiding direct comparison between nonboundary and boundary solutions, weakening the protection of boundary solutions. GCDC can to some extent eliminate the shortcomings of traditional crowding distance calculation methods. To validate the performance of INSGAII, we used it to solve traditional DLBP, recently proposed DLBP, as well as MUPDLBP and MUPDLBP_S proposed in this paper. By comparing the results with other algorithms, we have demonstrated that the performance of the INSGAII algorithm is superior to that of many known algorithms. In addition, we also conducted ablation experiments, and the results show that both MCTI and GCDC are effective. They have both played a positive role in improving the algorithm's performance.

6.2. Discussion. The following will discuss future research directions. First, from the algorithmic perspective, further improvements can be made to the INSGAII algorithm. On the one hand, as analyzed in Section 4.3.4, compared to MCTI, GCDC contributes more to enhancing algorithm performance. On the other hand, as known from Section 3.3, MCTI needs to estimate the distribution of feasible disassembly sequences in FDST using the Monte Carlo method, which requires generating a certain number of disassembly sequences through random initialization, leading to additional time consumption and parameters. Considering these two aspects, although MCTI helps improve algorithm performance, there is still room for improvement, so future research can focus on improving MCTI and proposing more effective initialization methods. Second, from the perspective of disassembly line design, future research can focus on the layout of the disassembly line and disassembly methods to further improve disassembly efficiency. In this study, although the disassembly efficiency was improved to a certain extent by combining the characteristics of the U-shaped layout disassembly line and mixed-flow disassembly line, more unexplored disassembly line layouts and disassembly methods can be explored in the future to bring gains to the disassembly process.

Data Availability

The data used in this study can be found in the Supplementary Materials.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Authors' Contributions

ZhenYu Xu conceptualized the study, proposed the methodology, provided software, validated the study, investigated the study, wrote the original draft, and reviewed and edited the manuscript. Yong Han conceptualized the study, performed data curation and formal analysis, provided resources, supervised the study, was responsible for project administration, and reviewed and edited the manuscript. ZhenXin Li, YiXin Zou, and YuWei Chen reviewed and edited the manuscript.

Acknowledgments

This work was supported by the National Key Research and Development Program of China (grant no. 2022YFB3305803).

Supplementary Materials

This research includes supplementary material which contains Appendix A and Appendix B. Appendix A contains detailed data about printers, CRT TVs, and refrigerators; Appendix B contains detailed data about pumps, vises, and gearboxes. It should also be noted that the data about printers, CRT TVs, and refrigerators were used in Section 4.3.1, and the data about pumps, vises, and gearboxes were used in Section 4.3.2. Additionally, appropriate explanations are provided at the relevant places in Sections 4.3.1 and 4.3.2. (*Supplementary Materials*)

References

[1] E. B. Edis, "Constraint programming approaches to disassembly line balancing problem with sequencing decisions," *Computers & Operations Research*, vol. 126, Article ID 105111, 2021.

[2] T. Wu, Z. Zhang, Y. Zeng, Y. Zhang, L. Guo, and J. Liu, "Techno-economic and environmental benefits-oriented human–robot collaborative disassembly line balancing optimization in remanufacturing," *Robotics and Computer-Integrated Manufacturing*, vol. 86, Article ID 102650, 2024.

[3] X. Guo, T. Wei, J. Wang, S. Liu, S. Qin, and L. Qi, "Multi-objective U-shaped disassembly line balancing problem considering human fatigue index and an efficient solution," *IEEE Transactions on Computational Social Systems*, vol. 10, no. 4, pp. 2061–2073, 2023.

[4] S. M. McGovern and S. M. Gupta, "Ant colony optimization for disassembly sequencing with multiple objectives," *The International Journal of Advanced Manufacturing Technology*, vol. 30, no. 5-6, pp. 481–496, 2006.

[5] J. Wang, X. Guo, J. Wang, S. Qin, L. Qi, and Y. Tang, "Discrete migratory bird optimizer for disassembly line balancing problem considering tool deterioration," in *Proceedings of the 2022 IEEE International Conference on Networking, Sensing and Control (ICNSC)*, pp. 15–18, Shanghai, China, December 2022.

[6] Y. Fang, Q. Liu, M. Li, Y. Laili, and D. T. Pham, "Evolutionary many-objective optimization for mixed-model disassembly line balancing with multi-robotic workstations," *European Journal of Operational Research*, vol. 276, no. 1, pp. 160–174, 2019.

[7] X. Guo, Z. Zhang, L. Qi, S. Liu, Y. Tang, and Z. Zhao, "Stochastic hybrid discrete grey wolf optimizer for multi-objective disassembly sequencing and line balancing planning in disassembling multiple products," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 3, pp. 1744–1756, 2022.

[8] N. O. Ndubisi, A. Nygaard, G. Chunwe N, and N. Gibson Chunwe, "Managing sustainability tensions in global supply chains: specific investments in closed-loop technology vs 'blood metals," *Production Planning & Control*, vol. 31, no. 11-12, pp. 1005–1013, 2020.

[9] 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, Article ID 106056, 2019.

[10] K. Wang, L. Gao, and X. Li, "A multi-objective algorithm for U-shaped disassembly line balancing with partial destructive mode," *Neural Computing & Applications*, vol. 32, no. 16, pp. 12715–12736, 2020.

[11] T. Yin, Z. Zhang, T. Wu, Y. Zeng, Y. Zhang, and J. Liu, "Multimanned partial disassembly line balancing optimization considering end-of-life states of products and skill differences of workers," *Journal of Manufacturing Systems*, vol. 66, pp. 107–126, 2023.

[12] Z. Li and M. N. Janardhanan, "Modelling and solving profit-oriented U-shaped partial disassembly line balancing problem," *Expert Systems with Applications*, vol. 183, Article ID 115431, 2021.

[13] K. Wang, X. Li, and L. Gao, "A multi-objective discrete flower pollination algorithm for stochastic two-sided partial disassembly line balancing problem," *Computers & Industrial Engineering*, vol. 130, pp. 634–649, 2019.

[14] M. L. Bentaha, O. Battaïa, and A. Dolgui, "L-Shaped algorithm for stochastic disassembly line balancing problem," *IFAC Proceedings Volumes*, vol. 46, no. 9, pp. 407–411, 2013.

[15] K. Wang, X. Li, L. Gao, and A. Garg, "Partial disassembly line balancing for energy consumption and profit under uncertainty," *Robotics and Computer-Integrated Manufacturing*, vol. 59, pp. 235–251, 2019.

[16] 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.

[17] Z. A. Çil, S. Mete, and F. Serin, "Robotic disassembly line balancing problem: a mathematical model and ant colony optimization approach," *Applied Mathematical Modelling*, vol. 86, pp. 335–348, 2020.

[18] C. Xu, H. Wei, X. W. Guo, S. X. Liu, L. Qi, and Z. Y. 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.

[19] L. Zhu, Z. Zhang, and C. Guan, "Multi-objective partial parallel disassembly line balancing problem using hybrid group neighbourhood search algorithm," *Journal of Manufacturing Systems*, vol. 56, pp. 252–269, 2020.

[20] Z. Zhang, K. Wang, L. Zhu, and Y. Wang, "A Pareto improved artificial fish swarm algorithm for solving a multi-objective fuzzy disassembly line balancing problem," *Expert Systems with Applications*, vol. 86, pp. 165–176, 2017.

[21] Y. J. Wang, G. G. Wang, F. M. Tian, D. W. Gong, and W. Pedrycz, "Solving energy-efficient fuzzy hybrid flow-shop scheduling problem at a variable machine speed using an extended NSGA-II," *Engineering Applications of Artificial Intelligence*, vol. 121, Article ID 105977, 2023.

[22] Z. Xu, X. Ning, Z. Yu, Y. Ma, Z. Zhao, and B. Zhao, "Design optimization of a shell-and-tube heat exchanger with disc-and-doughnut baffles for aero-engine using one hybrid method of NSGA II and MOPSO," *Case Studies in Thermal Engineering*, vol. 41, Article ID 102644, 2023.

[23] H. Yan and X. Zhang, "A self-adaptive local diversity-preserving NSGA-II for the multidisciplinary design optimization of a miniaturized supersonic rocket," *Aerospace Science and Technology*, vol. 136, Article ID 108230, 2023.

[24] Ö F. Yılmaz, B. Yazıcı, and B. Yazıcı, "Tactical level strategies for multi-objective disassembly line balancing problem with multi-manned stations: an optimization model and solution approaches," *Annals of Operations Research*, vol. 319, no. 2, pp. 1793–1843, 2022.

[25] A. Scholl and C. Becker, "State-of-the-art exact and heuristic solution procedures for simple assembly line balancing," *European Journal of Operational Research*, vol. 168, no. 3, pp. 666–693, 2006.

[26] A. Güngör and S. M. Gupta, "Disassembly line in product recovery," *International Journal of Production Research*, vol. 40, no. 11, pp. 2569–2589, 2002.

[27] T. Yin, Z. Zhang, Y. Zhang, T. Wu, and W. Liang, "Mixed-integer programming model and hybrid driving algorithm for multi-product partial disassembly line balancing problem with multi-robot workstations," *Robotics and Computer-Integrated Manufacturing*, vol. 73, Article ID 102251, 2022.

[28] G. A. Kumar, M. Bahubalendruni, V. S. S. Vara Prasad, D. Ashok, and K. Sankaranarayanasamy, "A novel Geometric feasibility method to perform assembly sequence planning through oblique orientations," *Engineering Science and Technology, an International Journal*, vol. 26, Article ID 100994, 2022.

[29] V. S. S. V. Prasad, M. Hymavathi, C. S. P. Rao, and M. A. Bahubalendruni, "A novel computative strategic planning projections algorithm (CSPPA) to generate oblique directional interference matrix for different applications in computer-aided design," *Computers in Industry*, vol. 141, Article ID 103703, 2022.

[30] S. Agrawal and M. K. Tiwari, "A collaborative ant colony algorithm to stochastic mixed-model U-shaped disassembly line balancing and sequencing problem," *International Journal of Production Research*, vol. 46, no. 6, pp. 1405–1429, 2008.

[31] M. L. Bentaha, O. Battaïa, and A. Dolgui, "An exact solution approach for disassembly line balancing problem under uncertainty of the task processing times," *International Journal of Production Research*, vol. 53, no. 6, pp. 1807–1818, 2015.

[32] K. Wang, X. Li, and L. Gao, "Modeling and optimization of multi-objective partial disassembly line balancing problem considering hazard and profit," *Journal of Cleaner Production*, vol. 211, pp. 115–133, 2019.

[33] M. V. A. R. Bahubalendruni and V. P. Varupala, "Disassembly sequence planning for safe disposal of end-of-life waste electric and electronic equipment," *National Academy Science Letters*, vol. 44, no. 3, pp. 243–247, 2021.

[34] G. Anil Kumar, M. V. A. R. Bahubalendruni, V. S. S. Prasad, and K. Sankaranarayanasamy, "A multi-layered disassembly sequence planning method to support decision making in de-manufacturing," *Sādhanā*, vol. 46, no. 2, p. 102, 2021.

[35] A. K. Gulivindala, M. V. A. R. Bahubalendruni, P. Madhu Balan, and M. Eswaran, "Mechanical disassembly sequence planning for end-of-life products to maximize recyclability," *Sādhanā*, vol. 48, no. 3, p. 109, 2023.

[36] Y. Ren, H. 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*, vol. 18, no. 2, pp. 791–803, 2021.

[37] A. Kumar Gulivindala, M. V A Raju Bahubalendruni, R. Chandrasekar, E. Ahmed, M. Haider Abidi, and A. Al-Ahmari, "Automated disassembly sequence prediction for industry 4.0 using enhanced genetic algorithm," *Computers, Materials and Continua*, vol. 69, no. 2, pp. 2531–2548, 2021.

[38] I. Kucukkoc, "Balancing of two-sided disassembly lines: problem definition, MILP model and genetic algorithm approach," *Computers & Operations Research*, vol. 124, Article ID 105064, 2020.

[39] K. Wang, X. Li, L. Gao, P. Li, and S. M. Gupta, "A genetic simulated annealing algorithm for parallel partial disassembly line balancing problem," *Applied Soft Computing*, vol. 107, Article ID 107404, 2021.

[40] Q. Tang, B. Lin, X. He, L. Zhang, C. Zhang, and X. Cao, "Balancing optimization of U-shaped assembly lines based on stochastic chance constrained programming," *Jisuanji Jicheng Zhizao Xitong/Computer Integrated Manufacturing Systems, CIMS*, vol. 22, no. 4, pp. 955–964, 2016.

[41] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[42] Y. Liu, Q. Tang, X. Tian, and S. Yang, "A novel offline programming approach of robot welding for multi-pipe intersection structures based on NSGA-II and measured 3D point-clouds," *Robotics and Computer-Integrated Manufacturing*, vol. 83, Article ID 102549, 2023.

[43] J. Tong, Y. Li, J. Liu et al., "Experiment analysis and computational optimization of the Atkinson cycle gasoline engine through NSGA II algorithm using machine learning," *Energy Conversion and Management*, vol. 238, Article ID 113871, 2021.

[44] Z. Zhang, J. R. Yan, X. Lu, T. Zhang, and H. Wang, "Optimization of porosity and surface roughness of CMT-P wire arc additive manufacturing of AA2024 using response surface methodology and NSGA-II," *Journal of Materials Research and Technology*, vol. 24, pp. 6923–6941, 2023.

[45] J. Xiao, X. J. Bi, and K. J. Wang, "Research of global ranking based many-objective optimization," *Ruan Jian Xue Bao/Journal of Software*, vol. 26, no. 7, pp. 1574–1583, 2015.

[46] H. T. Kahraman, M. Akbel, S. Duman, M. Kati, H. H. Sayan, and H. H. Sayan, "Unified space approach-based Dynamic Switched Crowding (DSC): a new method for designing Pareto-based multi/many-objective algorithms," *Swarm and Evolutionary Computation*, vol. 75, Article ID 101196, 2022.

[47] L. Li, Z. Zhang, L. Zhu, and B. Zou, "Modeling and optimizing for multi-objective partial disassembly line balancing problem," *Journal of Mechanical Engineering*, vol. 54, no. 3, pp. 125–136, 2018.

[48] S. Duman, M. Akbel, and H. T. Kahraman, "Development of the multi-objective adaptive guided differential evolution and optimization of the MO-ACOPF for wind/PV/tidal energy sources," *Applied Soft Computing*, vol. 112, Article ID 107814, 2021.

[49] H. T. Kahraman, M. Akbel, S. Duman, and S. Duman, "Optimization of optimal power flow problem using multi-objective manta ray foraging optimizer," *Applied Soft Computing*, vol. 116, Article ID 108334, 2022.

[50] C. Yue, B. Qu, and J. Liang, "A multiobjective particle swarm optimizer using ring topology for solving multimodal multiobjective problems," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 5, pp. 805–817, 2018.

[51] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, pp. 3–18, 2011.