mathematics

MDPI

Article

An Improved Moth-Flame Algorithm for Human–Robot Collaborative Parallel Disassembly Line Balancing Problem

Qi Zhang [1], Bin Xu [1], Man Yao [2], Jiacun Wang [3], Xiwang Guo [4], Shujin Qin [5], Liang Qi [6,*] and Fayang Lu [4]

[1] College of Information Engineering, Shenyang University of Chemical Technology, Shenyang 110142, China; qizhang@syuct.edu.cn (Q.Z.); binxu3111@163.com (B.X.)
[2] School of Basic Medicine, He University, Shenyang 110163, China; yaoman@huh.edu.cn
[3] Department of Computer Science and Software Engineering, Monmouth University, West Long Branch, NJ 07764, USA; jwang@monmouth.edu
[4] College of Information and Control Engineering, Liaoning Petrochemical University, Fushun 113001, China; xguo1@njcu.edu (X.G.); lfy6965@163.com (F.L.)
[5] College of Economics and Management, Shangqiu Normal University, Shangqiu 476000, China; qinshujin@sqnu.edu.cn
[6] Department of Computer Science and Technology, Shandong University of Science and Technology, Qingdao 266590, China
* Correspondence: qiliang@sdust.edu.cn

Abstract: In the context of sustainable development strategies, the recycling of discarded products has become increasingly important with the development of electronic technology. Choosing the human–robot collaborative disassembly mode is the key to optimizing the disassembly process and ensuring maximum efficiency and benefits. To solve the problem of human–robot cooperative parallel dismantling line balance, a mixed integer programming model is established and verified by CPLEX. An improved Moth-Flame Optimization (IMFO) algorithm is proposed to speed up convergence and optimize the disassembly process of various products. The effectiveness of IMFO is evaluated through multiple cases and compared with other heuristics. The results of these comparisons can provide insight into whether IMFO is the most appropriate algorithm for the problem presented.

Keywords: human–robot collaborative disassembly; parallel disassembly line; improved moth-flame algorithm

MSC: 90-08

Citation: Zhang, Q.; Xu, B.; Yao, M.; Wang, J.; Guo, X.; Qin, S.; Qi, L.; Lu, F. An Improved Moth-Flame Algorithm for Human–Robot Collaborative Parallel Disassembly Line Balancing Problem. *Mathematics* **2024**, *12*, 816. https://doi.org/10.3390/math12060816

Academic Editor: António Lopes

Received: 23 January 2024
Revised: 28 February 2024
Accepted: 6 March 2024
Published: 11 March 2024

1. Introduction

The rapid pace of product development in society, particularly in the realm of electronic products, has percipated a surge in end-of-life products. E-waste stands out as the fastest growing type of solid waste globally. However, these discarded products often harbor valuable subassemblies that can be recycled to generate fresh value. Disassembly emerges as a pivotal stage in the recycling process. The disassembly line balancing problem (DLBP) is characterized by the task of assigning disassembly tasks to workstations while adhering to various constraints [1–4]. Furthermore, there exists a spectrum of options for configuring the layout of the disassembly line, including a straight disassembly line [5], a U-shaped disassembly line [6], a bilateral disassembly line [7], and a parallel disassembly line [8,9].

As depicted in Figure 1, a parallel disassembly line pertains to the arrangement of two disassembly lines in running parallel on either side of the workstation. This configuration enables the separation of different types of products onto distinct disassembly lines, thereby averting the mixing and mishandling of subassemblies. Furthermore, given a fixed conveyor speed, the throughput of the parallel disassembly line, expressed as the number of products disassembled per unit time, is double that of the linear disassembly line. It is imperative to consider which subassemblies are suitable for disassembly and which

disassembly method yields the most cost-effective outcomes. The primary disassembly methods include worker disassembly, robot disassembly, and human–robot collaborative disassembly [10], with the latter increasingly emerging as a viable option. Robots can excel in handling repetitive and dangerous tasks, while humans are adept at handling more intricate and delicate operations.

Chen et al. [11] addressed the multi-objective sequential correlation robot disassembly line balancing problem. It incorporates sequence-related time increments employing a specific robot workstation allocation method. Subsequently, the resulting viable disassembly sequence must be assigned to ordered disassembly workstations. The research demonstrates that employing human–robot disassembly enhances the efficiency of the dismantling line. Xu et al. [12] introduced the problem of parallel disassembly line balancing involving human–robot collaboration and developed a multi-objective mixed integer programming model. The optimization objectives encompass the determination of the number of workstations, achieving balanced idle time, optimizing the total count of disassemblers, and minimizing disassembly costs. The efficacy of human–robot disassembly is further substantiated through comparative analysis with alternative algorithms. Huang [13] presents a novel experimental robotic disassembly cell comprising two collaborative robots and a human disassembler. Robots and disassemblers can collaboratively operate within a shared workspace, executing separate, parallel, or joint disassembly tasks safely. Intelligent optimization algorithms exhibit characteristics such as global optimization performance and broad applicability, rendering them suitable for parallel processing. These algorithms are generally capable of identifying or approximating optimal solutions over time. The utilization of intelligent optimization algorithms to solve DLBPs has been extensively documented.

Guo et al. [14] employed the grey wolf optimizer algorithm to address multi-objective disassembly sequencing problems and production line balance planning problems in their research. Given that DLBP often has NP-hard problems, Zhao et al. [15] proposed a method centered on an iterative greedy algorithm. Different neighborhood structures and the integral variable neighborhood descent method are used to procure an approximate optimal solution. Cui [16] proposed a quantum-inspired Moth-Flame optimizer featuring an enhanced local search strategy. The Wilcoxon rank-sum test and the Friedman test are utilized to evaluate the impact of Moth-Flame Optimization (MFO). Simulation results indicate that MFO outperforms other algorithms significantly in terms of accuracy, convergence speed, and stability. It is concluded that MFO can be effectively utilized for DLBP provided it enhances local search capabilities. Consequently, MFO is chosen as the solution algorithm for addressing the human–robot collaborative parallel disassembly line balancing problem. To the best of our knowledge, the MFO algorithm has not been previously applied to the disassembly line balancing problem.

End-of-life (EOL) products refer to products that have reached the end of their life cycle. For certain EOL products, profits can be realized through product disassembly and recycling [17,18]. Currently, there exists a wide array of EOL products with increasingly complex internal structures. Different subassemblies may exhibit differing degrees of complexity or hazards. Xu [12] acknowledged the presence of complex and hazardous subassemblies within the overall subassemblies. These components may necessitate distinct processing methodologies; however, a specific solution is not provided. On this basis, a new model of human–robot division of labor disassembly is established, wherein the subassemblies of each experimental case are classified. It is stipulated that special subassemblies require designated disassembler and specify the distribution method accordingly. This distribution method proves more conducive to multi-product disassembly.

This paper delves into parallel disassembly lines capable of processing multiple products at a time. An AND/OR figure is utilized to describe the disassembly priority and conflict relationships among any pair of disassembly tasks. In comparison with existing studies, the following two contributions are made:

1. A human–robot collaborative parallel disassembly line balancing problem (HCPDP) is proposed. This issue aims to efficiently resolve multi-product disassembly sequences while considering task distribution to maximize profit.
2. The Moth-Flame Optimization (MFO) algorithm is refined to enhance its local search capability. MFO alone is insufficient for solving the presented problem. The enhanced algorithm can effectively address this problem and achieve faster solutions and superior target values.

The subsequent sections of this article are organized as follows. Section 2 delineates the problem. Section 3 introduces the proposed algorithm. Section 4 conducts numerical experiments on small-, medium-, and large-scale cases and analyzes the experimental results. Finally, Section 5 summarizes the findings and outlines future research directions.

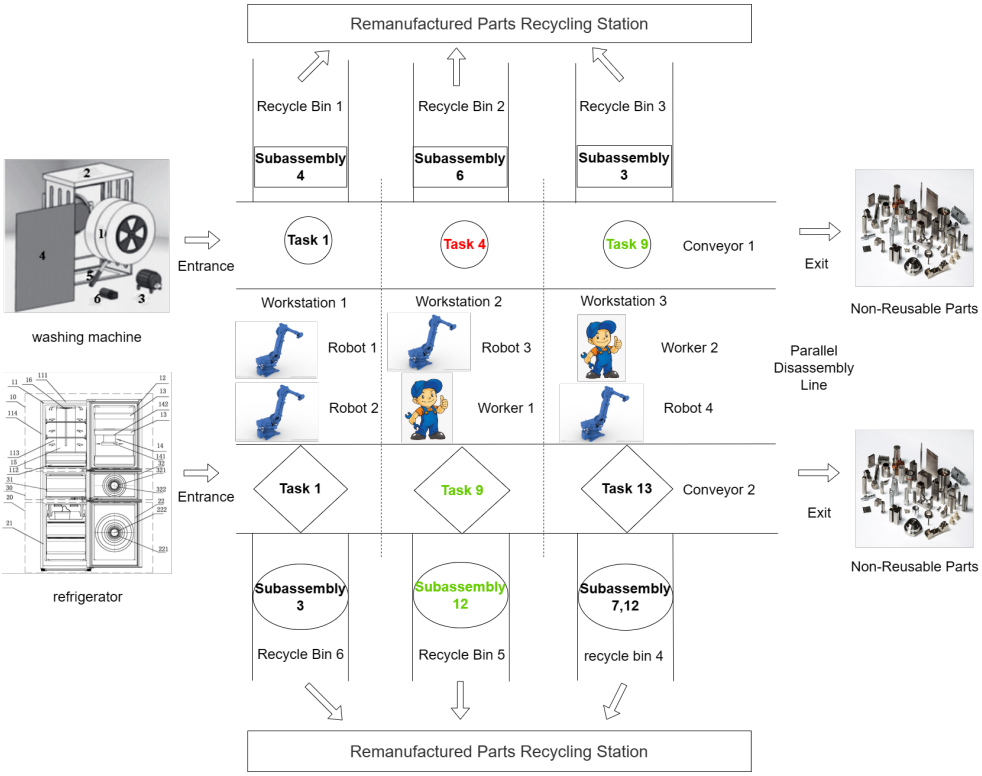

Figure 1. Schematic diagram of the parallel disassembly line.

2. Problem Statement and Formulation

To optimize the disassembly profit of the product, we first divide the disassembly tasks and components initially categorized based on their complexity and hazard level. Through the AND/OR graph of disassembling the product, we obtain disassembly sequences that adhere to priority and conflict relationships. By employing the refined algorithm, we obtain the optimal disassembly sequence which maximizes the profit for disassembling the product. The primary approach to address this problem is IMFO. IMFO discretizes the MFO algorithm, increases solution paths, and enhances the algorithm's global search capability.

2.1. Problem Description

There are many complex and hazardous subassemblies in EOL products. Complex subassemblies are small precision parts with high potential for reuse, such as circuit boards, capacitors, and chips. If a robot is used to remove them from the product, there is a risk of damage during the disassembly process. Therefore, it is mandatory that tasks involving these parts must be executed by workers. Frequent disassembly of hazardous subassemblies can also cause injury to workers; for example, batteries, radioactive metals, and saw blades, can pose safety hazards to workers. To mitigate these risks, it is mandatory

that tasks involving such parts be performed by robots to maximize worker safety [19]. The indices of complex and hazardous subassemblies and tasks for a washing machine are given in Table 1. The disassembler must be specified when performing these special tasks. Other common tasks may be assigned to either humans or robots. Considering the goal of maximizing profit, the allocation of routine tasks is contingent upon the disassembly costs for both humans and machines. Ordinary tasks are allocated to either humans or machines with lower disassembly costs wherever feasible.

Table 1. Special subassembly and task index of the washing machine.

Type	Complex Subassemblies	Complex Tassk	Hazardous Subassemblies	Hazardous Tasks
Index	3	2, 6, 9, 12	6	4, 7, 11

Given that a disassembly system is a discrete event system, various formal methods, such as Petri nets [20–22], timed Petri nets [23], or any other recognized techniques [24], can be employed for modeling and analyzing the disassembly processes. For simplicity, an AND/OR graph is used to represent task precedence relationships. The corresponding AND/OR graph for the washing is depicted in Figure 2. Similar cases are referenced for certain parameter values of the model [25]. To ensure that the final disassembly sequence aligns with requirements, the following assumptions are made about the model:

1. The disassembly AND/OR figure of each EOL product is known.
2. When each disassembly task is assigned to a workstation in the order in which it is disassembled, the priority matrix and the conflict matrix (both matrices are known) are satisfied.
3. Not all EOL products can be completely disassembled.
4. The cycle time of each workstation is known, and the processing time of each workstation does not exceed the cycle time.
5. The complex subassemblies and hazardous subassemblies of each EOL product are known.
6. The cost and energy consumption associated with workers and robots performing different disassembly tasks are known.
7. The value of each sub-subassembly is known.
8. Each workstation can be assigned a maximum of two disassemblers.

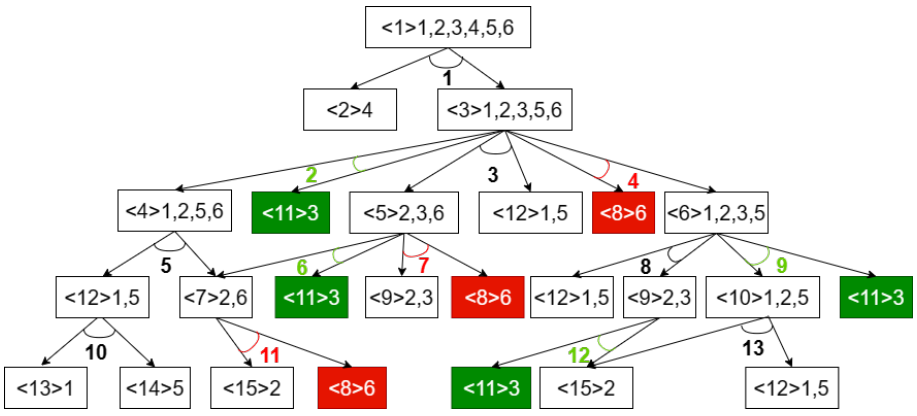

Figure 2. The AND/OR graph of a washing machine.

2.2. Description of an Example with an AND/OR Graph

As illustrated in Figure 2, an AND/OR graph is utilized to depict the precedence and conflict relationship in each EOL product disassembly task. The number enclosed in angle brackets within the rectangular box denotes the subassembly index, followed by the sequence index of parts. Arrows signify disassembly tasks, with the number positioned

between the two arrows indicating the task index. Complex subcomponents are represented by green rectangles, with associated tasks indicated by green numbers. Similarly, hazardous subassemblies are depicted by red rectangles, with corresponding hazardous tasks denoted by red numbers. Table 1 provides an overview of the specialized tasks and subassemblies associated with the washing machine.

As illustrated in Figure 2, a subassembly can have multiple disassembly ways, but it can only perform one disassembly task. For instance, Subassembly ⟨3⟩ 1, 2, 3, 5, 6 can obtain child Subassemblies ⟨4⟩ 1, 2, 5, 6 and ⟨11⟩ 3 via Task 2, or alternatively obtain child Subassemblies ⟨5⟩ 2, 3, 6 and ⟨12⟩ 1, 5 via Task 3. Task 2 and Task 3 cannot be performed in parallel, so Tasks 2 and 3 have an 'OR' relationship, which is the conflicted relationship between tasks. The 'AND' relationship is represented by the fact that every two child subassemblies can be dismantled by the parent component via a specified task. There is an 'AND' relationship between the two child subassemblies, for example, as shown in Figure 2, with child Subassembly ⟨12⟩ 1, 5 and child Subassembly ⟨7⟩ 2, 6. Moreover, a precedence relationship exists between tasks. For example, in Figure 2, there is a feasible disassembly sequence 1-3-7. Task 1 must be executed before Task 3 can be executed because Subassembly ⟨1⟩ 1, 2, 3, 4, 5, 6 can obtain child Subassemblies ⟨3⟩ 1, 2, 3, 5, 6 and ⟨2⟩ 4 via Task 1.

The precedence and conflict relationships ensure the rationality of the disassembly order of the parent components. The precedence relationship matrix, conflict relationship matrix, and relationship matrix for the washing machine are outlined. For instance, in Figure 3, the precedence relationship matrix of the washing machine is presented. Similarly, the precedence relationship, conflict relationship, and relationship matrices for other products can be derived from the AND/OR graph. The elements in matrices P, C, and R are defined below.

$$P = \begin{bmatrix} 0 & 0 & 1 & 1 & \dots & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & \dots & \vdots \\ 0 & 0 & 0 & 0 & \dots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 0 \end{bmatrix}$$

Figure 3. The precedence matrix of a washing machine.

$$P_{ij} = \begin{cases} 1, \text{if task } i \text{ is performed before task } j \\ 0, \text{otherwise} \end{cases}$$

$$C_{ij} = \begin{cases} 1, \text{if task } i \text{ is conflict with task } j \\ 0, \text{otherwise} \end{cases}$$

$$R_{ij} = \begin{cases} 1, \text{if subassembly } i \text{ is obtained by task } j \\ 0, \text{otherwise} \end{cases}$$

Notations used in the model to be presented are summarized as follows:
Indexes:
g Index of products, $g \in \{1, 2, \dots, G\}$.
h Index of humans, $h \in \{1, 2, \dots, H\}$.
i Index of tasks, $i \in \{1, 2, \dots, I\}$.
j Index of tasks, $j \in \{1, 2, \dots, J\}$.
k Index of disassemblers, $k \in \{1, 2, \dots, K\}$.
l Index of disassembly lines, $l \in \{1, 2\}$.
m Index of workstations, $m \in \{1, 2, \dots, M\}$.
w Index of subassemblies, $w \in \{1, 2, \dots, W\}$.
Parameters:
o_m The cycle time of each workstation m.

e_m Startup cost of workstation m.

N_l Total number of product disassembly tasks on disassembly line l.

d_{iw}^g A relationship between subassemblies and tasks of product g derived from the AND/OR graph.

p_{ij}^g A priority relationship derived from the AND/OR graph. If task i of product g takes precedence over task j of product g, p_{ij}^g=1; otherwise, p_{ij}^g=0.

q_{ij}^g A conflict relationship derived from the AND/OR graph. If task i of product g and task j of product g are in conflict, q_{ij}^g=1; otherwise, q_{ij}^g=0.

α_i^g Complex task. If task i of product g is a complex task, $\alpha_i^g = 1$; otherwise, $\alpha_i^g = 0$.

β_i^g Hazardous task. If task i of product g is a hazardous task, $\beta_i^g = 1$; otherwise, $\beta_i^g = 0$.

c_{ik}^g The cost when task i of product g is assigned to disassembler k.

t_{ik}^g The time when task i of product g is assigned to disassembler k.

v_w^g The reuse value of subassembly w in product g.

Decision variables:

$$x_{giml} = \begin{cases} 1, \text{if task } i \text{ of product } g \text{ is assigned to side } l \\ \text{ of workstation } m \text{ to be executed, } x_{giml} = 1; \\ 0 \text{ otherwise.} \end{cases}$$

$$y_{gimlk} = \begin{cases} 1, \text{if task } i \text{ of product } g \text{ is assigned to side } l \\ \text{ of workstation } m \text{ to be executed by} \\ \text{ disassembler } k, y_{gimlk} = 1; \\ 0 \text{ otherwise.} \end{cases}$$

$$u_m = \begin{cases} 1, \text{if the } m\text{-th workstation is started,} u_m = 1; \\ 0 \text{ otherwise.} \end{cases}$$

$$z_{mlk} = \begin{cases} 1, \text{if disassembler } k \text{ is assigned to side } l \text{ on} \\ \text{ workstation } m, \text{ then } z_{mlk} = 1; \\ 0 \text{ otherwise.} \end{cases}$$

2.3. Mathematical Model

Based on the listed notations, parameters and decision variables, the mathematical models of HCPDP are formulated as follows:

$$\begin{aligned} \max f = & \sum_{g=1}^{G} \sum_{i=1}^{I} \sum_{w=1}^{W} \sum_{m=1}^{M} \sum_{l=1}^{2} \sum_{k=1}^{K} d_{iw}^g v_w^g y_{gilmk} \\ & - \sum_{g=1}^{G} \sum_{i=1}^{I} \sum_{m=1}^{M} \sum_{l=1}^{2} \sum_{k=k}^{K} t_{gi}^k c_{gi}^k y_{gimlk} - \sum_{m=1}^{M} \mathrm{e_m} u_m \end{aligned} \tag{1}$$

$$\sum_{m=1}^{M} \sum_{l=1}^{2} \sum_{k=H+1}^{K} y_{gilmk} \alpha_i^g = 0, \forall g \in \{1, 2, \ldots, G\}, i \in \{1, 2, \ldots, I\}, \alpha_i^g = 1 \tag{2}$$

$$\sum_{m=1}^{M} \sum_{l=1}^{2} \sum_{k=1}^{H} y_{gilmk} \beta_i^g = 0, \forall g \in \{1, 2, \ldots, G\}, i \in \{1, 2, \ldots, I\}, \beta_i^g = 1 \tag{3}$$

$$\sum_{m=1}^{M} \sum_{l=1}^{2} \sum_{k=1}^{K} y_{gilmk} \leq 1, \forall g \in \{1, 2, \ldots, G\}, i \in \{1, 2, \ldots, I\} \tag{4}$$

$$\left.\begin{aligned}\sum_{k=1}^{K} z_{mlk} \le u_m, \forall m \in \{1,2,\ldots,M\}, l \in \{1,2\}\\ \sum_{l=1}^{2}\sum_{m=1}^{M} z_{mlk} \le 1, \forall k \in \{1,2,\ldots,K\}\end{aligned}\right\} \tag{5}$$

$$\left.\begin{aligned}\sum_{m=1}^{M}\sum_{l=1}^{2} x_{gilm} \le 1, \forall g \in \{1,2,\ldots,G\}, i \in \{1,2,\ldots,I\}\\ \sum_{g=1}^{G}\sum_{i=1}^{I}\sum_{l=1}^{2} x_{gilm} \ge u_m, \forall m \in \{1,2,\ldots,M\}\end{aligned}\right\} \tag{6}$$

$$x_{gilm} \le u_m, \forall m \in \{1,2,\ldots,M\}, g \in \{1,2,\ldots,G\}, i \in \{1,2,\ldots,I\}, l \in \{1,2\} \tag{7}$$

$$u_{m-1} \ge u_m, m \in \{2,3.....M\} \tag{8}$$

$$y_{gilmk} \ge x_{gilm} + z_{mlk} - 1, \forall g \in \{1,2,\ldots,G\}, i \in \{1,2,\ldots,I\}, m \in \{1,2,\ldots,M\}, l \in \{1,2\}, k \in \{1,2,\ldots,K\} \tag{9}$$

$$y_{gilmk} \le x_{gilm}, \forall g \in \{1,2,\ldots,G\}, i \in \{1,2,\ldots,I\}, m \in \{1,2,\ldots,M\}, l \in \{1,2\}, k \in \{1,2,\ldots,K\} \tag{10}$$

$$y_{gilmk} \le z_{mlk}, \forall g \in \{1,2,\ldots,G\}, i \in \{1,2,\ldots,I\}, m \in \{1,2,\ldots,M\}, l \in \{1,2\}, k \in \{1,2,\ldots,K\} \tag{11}$$

$$\sum_{g=1}^{G}\sum_{i=1}^{I}\sum_{k=1}^{K} t_{ik}^{g} y_{gimlk} \le o_m, \forall m \in \{1,2,\ldots,M\}, l \in \{1,2\} \tag{12}$$

$$\left.\begin{aligned}\sum_{m=1}^{M}(m x_{giml}) - \sum_{m=1}^{M}(m x_{gjml}) - (\sum_{l=1}^{2} N_l)(1 - \sum_{m=1}^{M} x_{gjml}) \le 0, \forall g \in \{1,2,\ldots,G\},\\ i \in \{1,2,\ldots,I\}, j \in \{1,2,\ldots,J\}, l \in \{1,2\}, p_{ij}^{g} = 1\\ \sum_{m=1}^{M}(m x_{giml}) + (\sum_{l=1}^{2} N_l)(1 - \sum_{m=1}^{M} x_{gjml}) \ge 0, \forall g \in \{1,2,\ldots,G\}, i \in \{1,2,\ldots,I\},\\ j \in \{1,2,\ldots,J\}, l \in \{1,2\}, p_{ij}^{g} = 1\end{aligned}\right\} \tag{13}$$

$$\sum_{m=1}^{M}\sum_{l=1}^{2}(x_{gilm} + x_{gjml}) \le 1, \forall g \in \{1,2,\ldots,G\}, i \in \{1,2,\ldots,I\}, j \in \{1,2,\ldots,J\}, q_{ij}^{g} = 1 \tag{14}$$

$$x_{gilm}, y_{gilmk}, z_{mlk}, u_m, \in \{0,1\}, g \in \{1,2,\ldots,G\}, i,j \in \{1,2,\ldots,I\}, m \in \{1,2,\ldots,M\}, l \in \{1,2\}, k \in \{1,2,\ldots,K\} \tag{15}$$

Objective Function (1) represents the maximum expected profit to disassemble EOL products. Disassembly profit is calculated as total disassembly revenue minus total disassembly cost. The total disassembly cost consists of the expenses associated with workers and robots that perform the disassembly actions and the cost of workstations. Constraint (2) ensures that complex tasks can only be dismantled by workers. Constraint (3) ensures that hazardous tasks can only be disassembled by robots. Constraint (4) guarantees that each task can be executed only once. Constraint (5) ensures that each workstation must be operated by a different disassembler. The first formula of Constraint (5) specifies that only when the workstation is operational can employees be assigned to corresponding positions on each line. The second formula indicates that each employee can be assigned to a maximum of one workstation. Constraint (6) ensures that each task can only be as-

signed to one side of the workstation. The first formula of Constraint (6) implies that at most one task of a product is scheduled on one side of the workstation, and the second formula states that if the workstation is operational, at least one task is assigned. Constraint (7) mandates that each open workstation must be assigned at least one task. Constraint (8) ensures workstations are turned on one after another, disallowing idle workstations. Constraint (9) guarantees the performance of each assigned task. Constraint (10) asserts that tasks can only be executed if they are assigned to the workstation side. Constraint (11) ensures that tasks can only be executed if the disassembler is assigned to the workstation. Constraint (12) ensures that the processing time of each workstation cannot exceed the cycle time. Constraint (13) enforces a priority relationship for each disassembly product. Constraint (14) requires that the feasible disassembly sequence adheres to the conflict relation. Constraint (15) indicates the range of values for variables.

3. Proposed Algorithm

Seyedali Mirjalili [26], drawing inspiration from the laws of nature, proposes a new swarm intelligence optimization MFO algorithm. It is grounded in moth behavior, striking a good balance between exploration and development through a specialized mechanism called lateral directional navigation, thereby achieving globally optimized performance. MFO has strong parallel optimization ability and good overall characteristics. MFO excels at extensive exploration of search space and identifying the regions with a greater probability of global optima; however, it is susceptible to falling into the local optimal situation. Many researchers have made improvements to the local optimization ability of the algorithm.

In addressing the limitations of the standard algorithm—namely low optimization accuracy and vulnerability to local optimal solutions—Zhang proposes an enhanced algorithm based on sinusoidal mapping and Gaussian variation [27]. The incorporation of Gaussian variation to regulate the population of individuals improves population diversity, which is conducive to better searching potential areas. This modification also accelerates the search speed and increases the probability of escaping from the local optimal solution. The aforementioned improved algorithm, designed for continuous problems, may not be suitable for the disassembly line balance problem. Consequently, a modification of the continuous operator of the algorithm is necessary to solve HCPDP.

3.1. Introduction of Improved MFO

In MFO, each moth serves as a candidate solution, while in HCPDP, each moth represents a disassembly sequence. The process of the moth flying in space is regarded as the process of optimization. In the traditional MFO algorithm, the moth update formula guides the moth search through the information carried by the jth flame. However, this approach relies solely on the information obtained from one dimension that does not effectively utilize the information carried by other flames, resulting in the slow convergence of the algorithm in the late iteration period. To address this, the concept of the historical optimal flame mean as a factor influencing the moth's search behavior is introduced. It provides relevant information for individual moths so that the reference information of moth search behavior is no longer limited to the moths' own dimensions. The improved moth position updating formula is shown in Equation (16).

$$M_i = S(M_i, F_j) = D_i * e^{bt} * cos(2\pi t) + A_j \tag{16}$$

S is the logarithmic spiral function, M_i is the ith moth individual, F_j is the Jth flame, $D_i = |F_j - M_i|$ represents the distance between the ith moth individual and the Jth flame, b is a constant that determines the shape of the flight spiral, and t is the random number that controls the distance between the moth and the next flame. The value range is $[-1, 1]$. When $t = -1$, it indicates that the next flame is closest to the moth, and when $t = 1$, it implies that the next flame is farthest from the moth. In Equation (16), the historical optimal flame

average is introduced into the calculation of the moth flight target; see Equation (17) for calculating the mean value.

$$A_j = \frac{\sum_{j=1}^{n} F_j}{n} \tag{17}$$

In Equation (17), n represents the number of flames that surpass the flame of moth i, while A_j signifies the mean value of the superior flames relative to the historical optimal flame position of moth i among all moths, which equates to the mean value of the historical optimal flame. By leveraging the insights conveyed by each generation of optimal flames, the breadth of flame-related information accessible to moths can be expanded. This expansion guides the subsequent movements of moths and ensures the quality of the solutions obtained by moths during each search behavior, facilitating a swifter approach towards the optimal solution. Consequently, the algorithm's convergence speed is accelerated.

3.2. Improved MFO Algorithm from MFO with Crossover Probability

Based on the aforementioned characteristics, a flowchart of the enhanced algorithm is presented in Figure 4. Furthermore, improvements are made to the crossover section of the MFO. Given the algorithm's inherent continuity, it is not suitable for HCPDP, modifications are implemented to render it discrete. This adaptation involves the introduction of a random crossover probability. Notably, the cross-variation of the algorithm remains unchanged, preserving its iterative approach. Following the initial crossover operation, the algorithm incorporates the random crossover probability variable and repeats the crossover operation. The pseudocode outlining the improved algorithm is provided in Algorithm 1.

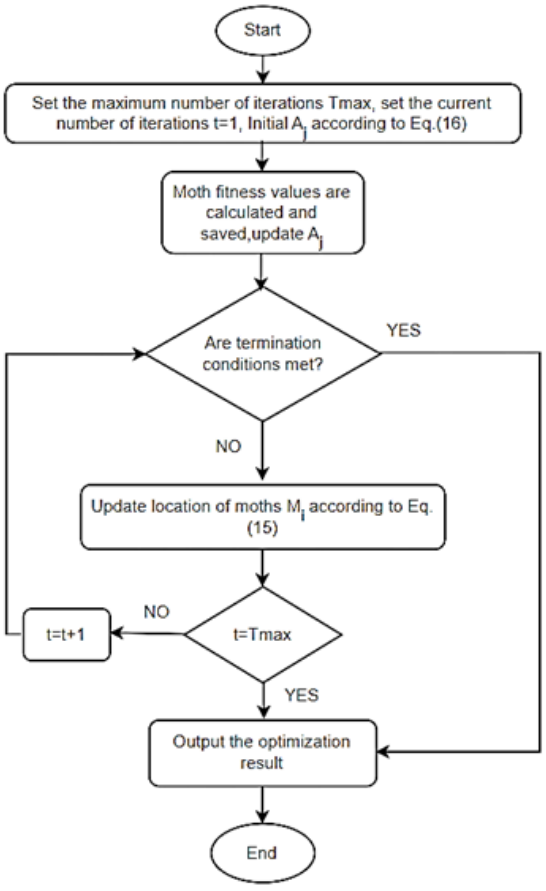

Figure 4. The flow chart of IMFO.

Algorithm 1 Improved Moth-Flame Optimization Algorithm

Input: case data, number of iterations, population size
Output: the best solutions

Initialize the moth population as solutions.
while (t $<$ maximum number of iterations) **do**
  for each individual **do**
    The objective function value is taken as the fitness of each individual.
    Calculate the fitness rank.
  end for
  for each individual **do**
    Select one best individual I and one random individual II from M.
    Choose a random number r between 0 and 2.
    if $r > 1$ **then**
      Execute the MFO interleaved operation.
      Get new individual III.
    else if $r < 1$ **then**
      Execute the MFO interleaved operation.
      Calculate and A_j and M_j then update A_j and M_j.
      Get new individual III.
    end if
    The best individual is selected from individual I, individual II, and the new individual III.
    Check if the optimal value is improved.
    Check if it is necessary to regenerate the population.
    Update the best solutions.
  end for
  $t = t + 1$
end while
return the best solutions.

As shown in Figure 5, an enhanced crossover process is demonstrated. Every individual in a population is a feasible disassembly sequence. To distinguish the tasks of different products, the task number of each product on the first conveyor belt is set to "+". The fitness of each individual is determined by the objective function value. Amalgamation of the two disassembly sequences results in the iterative generation of a new disassembly sequence treated as an individual within the algorithm. In the process of iteration and evolution of the algorithm, each iteration identifies the best individual based on the magnitude of the fitness function. Subsequently, these exceptional individuals are stored within the flame matrix for preservation. Within MFO, it is observed that the moth matrix functions to retain the individual population, while the flame matrix serves to preserve the optimal solution.

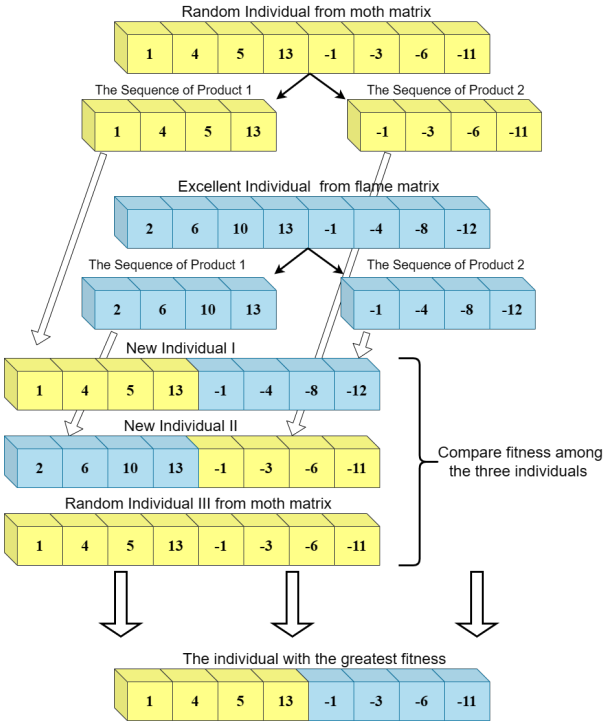

Figure 5. Cross−Process.

3.3. Encoding and Decoding

Within the moth population, individuals are ranked according to their fitness rank [28]. Then, the better random individual is selected as the first parent. The flame matrix which stores the local optimal solution is utilized to randomly select another individual from the flame matrix as the second parent. This crossover mode guarantees the production of more favorable individuals with each iteration. The MFO crossover and variation is retained and the code of the random crossover to increase the diversity of the population is added. The global searching ability of the algorithm is improved and the locally optimal solution is avoided [29,30].

The following steps are performed:

Step 1: The fitness rank of each individual in the population is calculated, and individuals are identified with the highest rank. An individual is randomly selected from this group to act as Parent 1.

Step 2: A random search is conducted in the flame matrix which stores the optimal solution in the iterative process, and then the individual is found as Parent 2. Equation (15) is updated.

Step 3: A_j is calculated and then the new individual is evaluated. The new disassembly sequence must meet the precedence relation matrix and the conflict relation matrix, and then the fitness must be calculated. The fitness value is compared with that of the parent in the moth population, and the parent is replaced if the fitness value is higher.

In this part, a three-step formula is used to represent a disassembly sequence: $\pi = \pi_1$, π_2, π_3. $\pi_1 = O_1, O_2, O_3$ composed of integer strings is a disassembly task sequence; π_2 is composed of binary strings and used to determine whether the task is executed; $\pi_2 = X_1$, X_2, $X_3 \cdots X_j$; if $X_j = 1$, task j in π_1 is executed; if $X_j = 0$, it is not executed. $\pi_3 = Y_1, Y_2, Y_3 \cdots Y_j$ indicates whether the current task is performed by a human or a robot. If $Y_j = 1$, then task j in π_1 is performed by a human. If $Y_j = 0$, it means that task j in π_1 is performed by a robot.

In a decoding process, task sequence is allocated to each workstation, enabling the workstation to execute the disassembly sequence. Concurrently, for each station, the initial mission is commenced, and an assessment is performed to determine whether the task execution time surpasses the cycle time of the workstations. If the execution time does not

exceed the cycle time, the disassembly task is performed and the next task is added followed by an evaluation to ascertain whether the cumulative time exceeds the workstation's cycle time. The task is not assigned to the next workstation until the specified time is exceeded. As shown in Figure 6, a feasible coding diagram of the disassembly sequence is given. The disassembly sequence is 1-4-9-13, Ordinary Task 1 and Ordinary Task 9 are assigned to workers, Complex Task 4 is assigned to workers, and Hazardous Task 13 is assigned to robots.

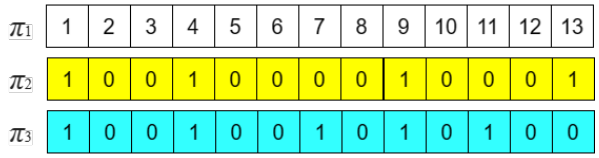

Figure 6. A coding example.

4. Experimental Studies

4.1. Test Instances

In the context of the parallel disassembly line, multiple products, including the washing machine [28], refrigerator [31], and copy machine [31], are employed to simulate multi-product disassembly tasks. Table 2 presents combinations of three EOL products, resulting in six distinct cases. The progression from Case I to Case VI reveals an increase in the number of both subassemblies and tasks, indicative of heightened difficulty and complexity in the disassembly process. Due to constraints stemming from incomplete disassembly and division disassembly, the number of workstations, workers, and robots along the parallel disassembly line varies across different combinations. As shown in Table 3, indexes of complex subassemblies and complex tasks of three EOL products are given, as well as indexes of hazardous subassemblies and hazardous tasks. Within the mathematical model, it is implied that complex tasks are to be performed exclusively by workers and hazardous tasks are to be performed by robots.

Table 2. Case description.

Case Num	Instance	Num of Tasks	Num of Subassemblies
I	Refrigerator and washing machine	35	40
II	Washing machine and copy machine	45	44
III	Refrigerator and copy machine	57	54
IV	Two copy machines and washing machine	52	59
V	Two refrigerators and washing machine	82	79
VI	Two copy machines and refrigerator	89	83

Table 3. Case index.

Index / Type / Case	Complex Subassemblies	Complex Task	Hazardous Subassemblies	Hazardous Task
Washing machine	3	2, 6, 9, 12	6	4, 7, 11
Copy machine	5, 6	1, 2, 17	26, 27	14, 22, 25, 27
Refrigerator	12	9, 11	21	2, 7

All of the algorithms and program tests are based on the implemented Intel IDEA 2021.2.2 "x64" and CPLEX and use the operating system Windows 10 AMD Ryzen 5 4600 h CPU (3.0 GHz/16 G RAM) run on a PC. Population sizes of 300, 400, 500, and 600 are used, respectively. The largest number of iterations is fixed 100 times, and each algorithm runs 20 times. The trial is conducted to compare the data objective value of IMFO with other algorithms.

4.2. Performance Metrics

Multi-objective parallel disassembly lines were simulated and experiments were conducted for each case. Optimal solutions were initially obtained using CPLEX. Subsequently, IMFO, Discrete Fruit Fly Optimization Algorithm (DFOA) [32], Elitist Evolution Strategy Algorithm (EA) [33], and Carnivorous Plant Algorithm (CPA) [34] were employed individually within the jMetal framework to solve the model. Finally, the optimal solutions obtained using the four algorithms were compared with those obtained using CPLEX.

The model studied is a single-objective model. CPLEX is used to to verify the correctness of the model. Six different cases were constructed for the three scenarios and evaluated using CPLEX. The solution results and disassembly sequences of CPLEX and IMFO algorithms are presented, respectively, in Tables 4 and 5. In these tables, hazardous tasks, denoted in red, necessitate robotic disassembly, while complex tasks, marked in green, require human intervention. CPLEX successfully resolved cases of medium and small scales but struggled with larger cases; for instance, Case VI exhibited extensive subassemblies and tasks, causing CPLEX to exceed its normal runtime. Conversely, IMFO consistently delivered accurate results across all case sizes, albeit with increased solving time corresponding to case complexity. Notably, as case complexity increased, IMFO significantly outperformed CPLEX in terms of computational efficiency.

Table 4. Results of solving the instances set on CPLEX.

Case Num	Disassembly Sequence	f	T
I	$1 \to 4 \to 8 \to 12 \to -14$	836.2	4.84 s
II	$2 \to 11 \to -14 \to 5 \to -18 \to 7 \to 8$	1306.3	4.798 s
III	$2 \to 4 \to -20 \to 12 \to -22 \to -26 \to 13$	1467.9	5.513 s
IV	$2 \to 5 \to 8 \to 9 \to 11 \to -25 \to -33 \to -35 \to -44$	1923.8	12.66 s
V	$1 \to 3 \to -20 \to -21 \to -25 \to 13 \to 12$	1411.0	10.737 s
VI	–	–	9000.00 s

The red numbers represent hazardous tasks, while the green numbers represent complex tasks.

Table 5. Results of solving the instances set based on IMFO.

Case Num	Disassembly Sequence	f	T
I	$1 \to 4 \to 8 \to 12 \to -14$	836.2	2.865 s
II	$2 \to 11 \to -14 \to 5 \to -18 \to 7 \to 8$	1306.3	2.679 s
III	$2 \to 4 \to -20 \to 12 \to -22 \to -26 \to 13$	1467.9	5.782 s
IV	$2 \to 5 \to 8 \to 9 \to 11 \to -25 \to -33 \to -35 \to -44$	1923.8	4.236 s
V	$1 \to 3 \to -20 \to -21 \to -25 \to 13 \to 12$	1411.0	6.456 s
VI	$2 \to -34 \to -66 \to 3 \to 12 \to -53 \to -82 \to 13 \to -44 \to -57$	2450.3	5.836 s

The red numbers represent hazardous tasks, while the green numbers represent complex tasks.

4.3. Instance Analysis

In the Jemetal framework, the HCPDP is addressed using IMFO, DFOA, EA, and CPA algorithms. Various population sizes, such as 300, 400, 500, and 600, are examined, and corresponding target values are provided. Additionally, the optimal target value attained at the maximum number of iterations is presented. Table 6 illustrates the results. It is evident that in Case I, the IMFO algorithm achieves the target value of 836.2, whereas DFOA only reaches 789.9. Similarly, in Case V, IMFO attains 1411 compared to DFOA's 1400. In Case VI, IMFO reaches 2450.7 while DFOA reaches only 2432.3.

As evident from the results presented in Table 6, the performance of algorithms varies due to the differing complexities of the cases. Cases I to IV, characterized by a relatively small number of subassemblies and tasks, are deemed small-scale with low complexity. IMFO outperforms DFOA and EA across small, medium, and large cases. Particularly for small-scale cases, DFOA and EA exhibit relatively inferior performance. Moreover,

Cases V and VI are large-scale with numerous subassemblies and tasks, resulting in longer program runtimes. Different algorithms demonstrate varying capabilities in handling large-scale cases. IMFO and EA display superior performance compared to DFOA in managing large-scale cases, although EA also exhibits the longest runtime among the algorithms evaluated.

The convergence of the algorithms can be assessed based on the number of iterations and the final target value, as depicted in Figure 7. A comparison is made between the number of iterations and the objective function values of the two algorithms. Notably, in the experiment conducted for Case V with a population size of 600, the DFOA demonstrates stronger optimization capabilities at the onset, particularly evident within the initial 10 iterations. However, beyond 12 iterations, the IMFO algorithm demonstrates superior optimization capabilities. Similarly, the EA algorithm exhibits better optimization abilities initially, albeit consuming the most time. The IMFO algorithm achieves convergence approximately 50 times, whereas the DFOA achieves convergence approximately 60 times. In conclusion, these results underscore the superiority of the IMFO algorithm in addressing HCPDP.

Table 6. Profit obtained using different algorithms.

Cases	Algorithm	Population Size			
		300	**400**	**500**	**600**
I	IMFO	708.3	759.5	789.9	836.2
	DFOA	667.2	735.0	765.9	789.9
	EA	688.6	733.0	746.7	759.5
	CPA	673.3	719.9	767.3	820.2
II	IMFO	1617.9	1658.9	1687.6	1706.3
	DFOA	1607.2	1622.9	1656.2	1688.9
	EA	1579.3	1600.3	1640.3	1669.7
	CPA	866.3	877.9	902.0	927.6
III	IMFO	1184.5	1267.9	1308.2	1467.9
	DFOA	1136.9	1195.4	1244.3	1288.6
	EA	1136.3	1344.2	1402.2	1447.8
	CPA	714.3	756.4	792.5	876.8
IV	IMFO	1821.4	1871.6	1888.9	1923.8
	DFOA	1809.5	1833.7	1853.0	1865.0
	EA	1827.3	1833.7	1855.4	1863.1
	CPA	1804.8	1810.5	1901.0	1916.4
V	IMFO	1406.0	1408.0	1411.0	1411.0
	DFOA	1399.0	1399.0	1400.0	1400.0
	EA	1402.3	1406.0	1410.1	1410.1
	CPA	1385.4	1388.0	1391.9	1392.3
VI	IMFO	2434.9	2443.0	2450.3	2450.3
	DFOA	2424.6	2427.3	2431.0	2432.3
	EA	2424.6	2417.3	2424.6	2441.0
	CPA	2426.3	2425.9	2433.7	2441.5

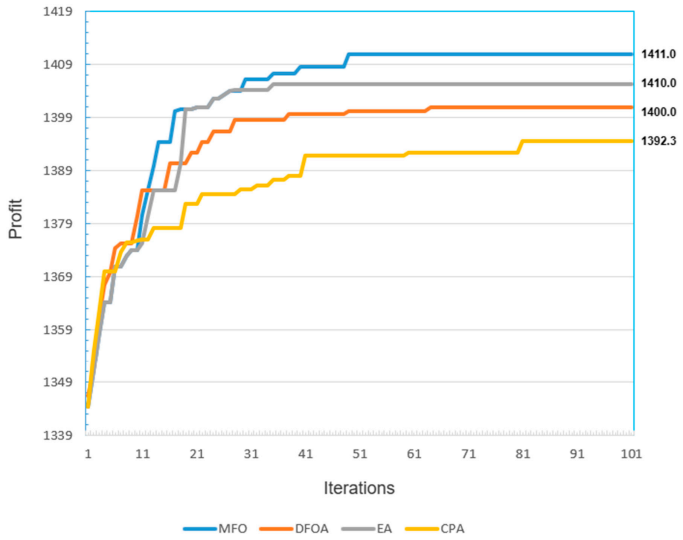

Figure 7. Algorithm convergence of Case V with population size 600.

5. Conclusions

This paper, in view of the parallel disassembly line, proposes a mixed integer programming mathematical model based on the human–robot disassembly joined division of subassemblies that allows the model to deal with more scenarios. An improved multi-objective optimization algorithm IMFO is proposed, which is adapted to HCPDP by changing the algorithm from continuous to discrete. Multiple cases are used, combined with their respective AND/OR graphs, to distinguish between complex and hazardous tasks. The results are compared with those of DFOA, EA, and CPA. The results show that IMFO is superior to DFOA, EA, and CPA in solving HCPDP. Because the disassembly time required for different disassembly sequences of the same product is not much different, and considering the disassembly time increases the complexity of the model, this article uses disassembly profit as the target value. The forthcoming endeavor involves the application of IMFO to alternative disassembly configurations, such as bilateral routes, while also delving into multi-objective optimization, encompassing factors such as disassembly time and profit. Additionally, it seeks to integrate recently proposed optimization methodologies.

Author Contributions: Writing—original draft preparation, Q.Z. and B.X.; Writing—review & editing, Q.Z., J.W. and L.Q.; Supervision, X.G. and S.Q.; Data curation, F.L.; Visualization, M.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported in part by NSFC under Grant Nos. 61573089 and 51405075; Liaoning Province Education Department Scientific Research Foundation of China under Grant No. JYTQN2023366; The Natural Science Foundation of Shandong Province under Grant ZR2019BF004.

Data Availability Statement: In accordance with MDPI's research data policy, we hereby declare that the data utilized and generated throughout the research process are thoroughly integrated into the paper, and the data presented in the paper have been adequately furnished and duly cited.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Zhao, Z.; Zhou, M.; Liu, S. Iterated Greedy Algorithms for Flow-Shop Scheduling Problems: A Tutorial. *IEEE Trans. Autom. Sci. Eng.* **2021**, *19*, 1941–1959. [CrossRef]
2. Zhao, Z.; Liu, S.; Zhou, M.; Abusorrah, A. Dual-Objective Mixed Integer Linear Program and Memetic Algorithm for an Industrial Group Scheduling Problem. *IEEE/CAA J. Autom. Sin.* **2021**, *8*, 1199–1209. [CrossRef]
3. McGovern, S.; Gupta, S. Combinatorial optimization methods for disassembly line balancing. *Proc.-Spie Int. Soc. Opt. Eng.* **2004**, *5583*, 53–66. [CrossRef]
4. Gungor, A.; Gupta, S.M. A solution approach to the disassembly line balancing problem in the presence of task failures. *Int. J. Prod. Res.* **2001**, *39*, 1427–1467. [CrossRef]

5. Gungor, A.; Gupta, S.M.; Pochampally, K.; Kamarthi, S.V. Complications in disassembly line balancing. *SPIE* **2001**, *4193*, 289–298. [CrossRef]
6. Lu, F.; Liu, P.; Qi, L.; Qin, S.; Xu, G.; Xu, Z. Multi-objective discrete strength pareto evolutionary algorithm II for multiple-product partial U-shaped disassembly line balancing problem. *J. Phys. Conf. Ser.* **2021**, *2024*, 012058. [CrossRef]
7. Liang, J.; Guo, S.; Du, B.; Li, Y.; Guo, J.; Yang, Z.; Pang, S. Minimizing energy consumption in multi-objective two-sided disassembly line balancing problem with complex execution constraints using dual-individual simulated annealing algorithm. *J. Clean. Prod.* **2021**, *284*, 125418. [CrossRef]
8. Hezer, S.; Kara, Y. A network-based shortest route model for parallel disassembly line balancing problem. *Int. J. Prod. Res.* **2015**, *53*, 1849–1865. [CrossRef]
9. Zhu, L.; Zhang, Z.; Guan, C. Multi-objective partial parallel disassembly line balancing problem using hybrid group neighbourhood search algorithm. *J. Manuf. Syst.* **2020**, *56*, 252–269. [CrossRef]
10. Lou, S.; Zhang, Y.; Tan, R.; Lv, C. A human-cyber-physical system enabled sequential disassembly planning approach for a human-robot collaboration cell in Industry 5.0. *Robot.-Comput. Manuf.* **2024**, *87*, 102706. [CrossRef]
11. Chen, Q.; Yao, B.; Pham, D.T. Sequence-Dependent Robotic Disassembly Line Balancing Problem Considering Disassembly Path. In Proceedings of the International Manufacturing Science and Engineering Conference, Online, 3 September 2020; Volume 2. [CrossRef]
12. Xu, P.; Zhang, Z.; Guan, C. Modeling and Hybrid Teaching Optimization Algorithm for Parallel Disassembly Line Balance Problem of Human-machine Co-station. In *Computer Integrated Manufacturing Systems*; Prentice-Hall, Inc.: Upper Saddle River, NJ, USA, 2022; pp. 1–17.
13. Huang, J.; Pham, D.T.; R, L. An experimental human-robot collaborative disassembly cell. *Comput. Ind. Eng.* **2021**, *155*, 20. [CrossRef]
14. Guo, X.; Zhang, Z.; Qi, L.; Liu, S.; Tang, Y.; Zhao, Z. Stochastic Hybrid Discrete Grey Wolf Optimizer for Multi-objective Disassembly Sequencing and Line Balancing Planning in Disassembling Multiple Products. *IEEE Trans. Autom. Sci. Eng.* **2021**, *19*, 1744–1756. [CrossRef]
15. Zhao, Z.; Liu, S.; Zhou, M.; You, D.; Guo, X. Heuristic Scheduling of Batch Production Processes Based on Petri Nets and Iterated Greedy Algorithms. *IEEE Trans. Autom. Sci. Eng.* **2022**, *19*, 251–261. [CrossRef]
16. Cui, X.; Luo, Q.; Zhou, Y.; Deng, W.; Yin, S. Quantum-Inspired Moth-Flame Optimizer With Enhanced Local Search Strategy for Cluster Analysis. *IEEE/CAA J. Autom. Sin.* **2022**, *5*, 794–806. [CrossRef] [PubMed]
17. Özceylan, E.; Kalayci, C.B.; Güngör, A.; Gupta, S.M. Disassembly line balancing problem: A review of the state of the art and future directions. *Int. J. Prod. Res.* **2019**, *57*, 4805–4827. [CrossRef]
18. Kim, H.J.; Xirouchakis, P. Capacitated disassembly scheduling with random demand. *Int. J. Prod. Res.* **2010**, *48*, 7177–7194. [CrossRef]
19. Xu, W.; Cui, J.; Liu, B.; Liu, J.; Yao, B.; Zhou, Z. Human-robot collaborative disassembly line balancing considering the safe strategy in remanufacturing. *J. Clean. Prod.* **2021**, *324*, 129158. [CrossRef]
20. Xiang, D.; Lin, S.; Wang, X.; Liu, G. Checking Missing-Data Errors in Cyber-Physical Systems Based on the Merged Process of Petri Nets. *IEEE Trans. Ind. Inform.* **2023**, *19*, 3047–3056. [CrossRef]
21. Xiang, D.; Zhao, F.; Liu, Y. DICER 2.0: A New Model Checker for Data-Flow Errors of Concurrent Software Systems. *Mathematics* **2021**, *9*, 966. [CrossRef]
22. Qi, L.; Su, Y.; Zhou, M.; Abusorrah, A. A State-Equation-Based Backward Approach to a Legal Firing Sequence Existence Problem in Petri Nets. *IEEE Trans. Syst. Man Cybern. Syst.* **2023**, *53*, 4968–4979. [CrossRef]
23. Zhou, J.; Wang, J.; Wang, J. A simulation engine for stochastic timed Petri nets and application to emergency healthcare systems. *IEEE/CAA J. Autom. Sin.* **2019**, *6*, 969–980. [CrossRef]
24. Wang, J.; Tepfenhart, W. *Formal Methods in Computer Science*; John Wiley & Sons: Hoboken, NJ, USA, 2019.
25. Qin, G.; Guo, X.; Zhou, M.; Liu, S.; Qi, L. Multi-Objective Discrete Migratory Bird Optimizer for Stochastic Disassembly Line Balancing Problem. In Proceedings of the 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Toronto, ON, Canada, 11–14 October 2020; pp. 420–425. [CrossRef]
26. Seyedali, M. Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowl.-Based Syst.* **2015**, *89*, 228–249.
27. Zhang, Y.; Wang, P.; Yang, H.; Cui, Q. Optimal dispatching of microgrid based on improved moth-flame optimization algorithm based on sine mapping and Gaussian mutation. *Syst. Sci. Control Eng.* **2022**, *10*, 115–125. [CrossRef]
28. Nowakowski, P. A novel, cost efficient identification method for disassembly planning of waste electrical and electronic equipment. *J. Clean. Prod.* **2018**, *172*, 2695–2707. [CrossRef]
29. Li, H.; Gao, K.; Duan, P.; Li, J.; Zhang, L. An Improved Artificial Bee Colony Algorithm With Q-Learning for Solving Permutation Flow-Shop Scheduling Problems. *IEEE Trans. Syst. Man Cybern. Syst.* **2023**, *53*, 2684–2693. [CrossRef]
30. Fu, Y.; Ma, X.; Gao, K.; Li, Z.; Dong, H. Multi-Objective Home Health Care Routing and Scheduling With Sharing Service via a Problem-Specific Knowledge-Based Artificial Bee Colony Algorithm. *IEEE Trans. Intell. Transp. Syst.* **2024**, *25*, 1706–1719. [CrossRef]
31. Fu, Y.; Ding, J.; Wang, H.; Wang, J. Two-objective stochastic flow-shop scheduling with deteriorating and learning effect in Industry 4.0-based manufacturing system. *Appl. Soft Comput.* **2018**, *68*, 847–855. [CrossRef]

32. Shao, Z.; Pi, D.; Shao, W. Hybrid enhanced discrete fruit fly optimization algorithm for scheduling blocking flow-shop in distributed environment. *Expert Syst. Appl.* **2020**, *145*, 113147. [CrossRef]
33. Dao, S.D.; Abhary, K.; Marian, R.M. Maximising Performance of Genetic Algorithm Solver in Matlab. *Eng. Lett.* **2016**, *24*, EL_24_1_11.
34. Ong, K.M.; Ong, P.; Sia, C.K. A carnivorous plant algorithm for solving global optimization problems. *Appl. Soft Comput.* **2020**, *33*, 106833. [CrossRef]