mathematics

MDPI

Article

An Evolutionary Learning Whale Optimization Algorithm for Disassembly and Assembly Hybrid Line Balancing Problems

Xinshuo Cui [1], Qingbo Meng [1], Jiacun Wang [2], Xiwang Guo [1], Peisheng Liu [1,*], Liang Qi [3], Shujin Qin [4], Yingjun Ji [5] and Bin Hu [6,*]

[1] College of Artificial Intelligence and Software, Liaoning Petrochemical University, Fushun 113001, China; cuixinshuo@stu.lnpu.edu.cn (X.C.); mengqingbo@stu.lnpu.edu.cn (Q.M.); xguo1@njcu.edu (X.G.)
[2] Department of Computer Science and Software Engineering, Monmouth University, West Long Branch, NJ 07764, USA; jwang@monmouth.edu
[3] Department of Computer Science and Technology, Shandong University of Science and Technology, Qingdao 266590, China; qiliang@sdust.edu.cn
[4] College of Economics and Management, Shangqiu Normal University, Shangqiu 476000, China; qinshujin@sqnu.edu.cn
[5] Faculty of Information, Liaoning University, Shenyang 110036, China; jiyingjun@lnu.edu.cn
[6] Department of Computer Science and Technology, Kean University, Union, NJ 07083, USA
* Correspondence: liupeisheng@lnpu.edu.cn (P.L.); bhu@kean.edu (B.H.)

Abstract: In order to protect the environment, an increasing number of people are paying attention to the recycling and remanufacturing of EOL (End-of-Life) products. Furthermore, many companies aim to establish their own closed-loop supply chains, encouraging the integration of disassembly and assembly lines into a unified closed-loop production system. In this work, a hybrid production line that combines disassembly and assembly processes, incorporating human–machine collaboration, is designed based on the traditional disassembly line. A mathematical model is proposed to address the human–machine collaboration disassembly and assembly hybrid line balancing problem in this layout. To solve the model, an evolutionary learning-based whale optimization algorithm is developed. The experimental results show that the proposed algorithm is significantly faster than CPLEX, particularly for large-scale disassembly instances. Moreover, it outperforms CPLEX and other swarm intelligence algorithms in solving large-scale optimization problems while maintaining high solution quality.

Keywords: disassembly line balancing; disassembly sequence; sustainability; carbon savings; discrete whale optimization algorithm

MSC: 90B30

Academic Editor: Ioannis Tsoulos

Received: 23 November 2024
Revised: 7 January 2025
Accepted: 8 January 2025
Published: 14 January 2025

Citation: Cui, X.; Meng, Q.; Wang, J.; Guo, X.; Liu, P.; Qi, L.; Qin, S.; Ji, Y.; Hu, B. An Evolutionary Learning Whale Optimization Algorithm for Disassembly and Assembly Hybrid Line Balancing Problems. *Mathematics* **2025**, *13*, 256. https://doi.org/10.3390/math13020256

1. Introduction

With the development of the manufacturing industry, an increasing number of enterprises are aiming to establish their own closed-loop supply chains [1]. Consequently, the integration of traditional assembly and disassembly operations has become an inevitable trend. Disassembly refers to the careful process of taking apart End-of-Life (EOL) [2–5] products to retrieve valuable components and materials for recycling and reuse. While disassembly involves breaking down EOL products into parts for recycling and assembly focuses on combining parts into complete products, the two operations share similarities, such as the use of the same tools, parts, and operational skills, even though disassembly is not the exact opposite of assembly.

Recently, research on disassembly, scheduling, and multi-objective optimization has gained prominence in industrial engineering, driven by increasing production demands and advancements in automation. Guo et al. addressed the disassembly line balancing problem under stochastic operation times, proposing a multi-objective shuffled frog leaping algorithm to enhance efficiency and resource utilization [6]. In another study, they incorporated human fatigue indices as an optimization objective in U-shaped disassembly line balancing problems, employing advanced heuristic methods to improve solution efficiency [7]. Additionally, Guo et al. tackled multi-product disassembly challenges by introducing a stochastic hybrid discrete grey wolf optimizer to optimize sequencing and line balancing, demonstrating its effectiveness in complex disassembly scenarios [8]. These studies exemplify the feasibility and effectiveness of advanced algorithms in solving complex multi-objective disassembly line balancing problems. Fu et al. [9] reviewed the applications of ensemble meta-heuristics and reinforcement learning in manufacturing scheduling, emphasizing their potential to address complex production challenges with adaptive and intelligent optimization strategies. Zhang et al. [10] extended this work by developing a learning-driven multi-objective cooperative artificial bee colony algorithm to address distributed flexible job shop scheduling problems, incorporating preventive maintenance and transportation operations to enhance system robustness and efficiency. Furthermore, Fu et al. [11] applied a knowledge-based artificial bee colony algorithm to multi-objective home health care routing and scheduling, focusing on shared services and providing efficient solutions for resource allocation and task optimization in healthcare operations. Zhao et al. tackled scheduling and logistics optimization in batch manufacturing processes by developing meta-heuristic algorithms to manage temperature constraints and alternative thermal device configurations, thereby enhancing efficiency and production performance [12]. They also proposed an integrated optimization framework for multi-mobile-robot transport and production systems, combining robot scheduling with constraint planning to streamline resource coordination and system operations [13]. Moreover, Zhao et al. introduced a lexicographic dual-objective path finding algorithm for multi-agent systems, effectively balancing task priorities and navigation efficiency for improved multi-agent coordination in dynamic environments [14]. These studies underscore the versatility and effectiveness of meta-heuristic approaches in addressing complex manufacturing and logistics challenges, paving the way for future advancements in industrial automation and decision-making.

Ketzenberg et al. [15] were among the first to propose a framework for a hybrid production line that integrates assembly and disassembly lines, demonstrating its potential. Mete et al. [16] extended this framework by introducing common workstations capable of performing both disassembly and assembly tasks. They developed a mathematical model to minimize total costs and used a toy car as a test case to demonstrate that hybrid production lines can significantly reduce costs. The study also confirmed that meta-heuristic algorithms, such as ant colony optimization, remain effective in solving these problems. Subsequently, Zhang et al. [17] proposed a hybrid production line balancing optimization problem with objectives to minimize the number of workstations, idle time, and equipment usage, employing a multi-objective hybrid evolutionary search algorithm based on evolutionary simulated annealing to solve the problem. Guo et al. [18] addressed the impact of uncertainty in hybrid production line problems, proposing stochastic hybrid production line optimization with production cost and load balancing objectives, using a combined variable neighborhood search (VNS) and non-dominated sorting genetic algorithm II (NSGA-II) to verify the correctness of their model and the algorithm's effectiveness.

With the increasing integration of industrialization and information technology, the use of robots [19] has become increasingly popular. Human–robot collaboration [20–23]

effectively leverages human intelligence and robot speed, enabling robots to perform repetitive or simple disassembly tasks while allowing humans to focus on complex assembly and disassembly operations.

The whale optimization algorithm (WOA) [24], a swarm intelligence optimization algorithm inspired by the hunting behavior of humpback whales, is characterized by a small number of parameters and fast convergence. It has been applied to various problems, including pipe network design [25], the traveling salesman problem [26], image processing [27], and knapsack problems [28]. Given its success in these domains, this work explores its application to the human–machine collaboration disassembly and assembly hybrid line balance problem (DALHBP).

To address existing research gaps, this work proposes an evolutionary learning whale optimization algorithm (ELWOA) to solve the DALHBP. The main contributions of this work are as follows:

- A novel layout scheme for a parallel disassembly line running in reverse for the same product is proposed to enhance the utilization of disassembled parts at each workstation.
- A new mathematical model for DALHBP is developed, incorporating human–machine collaborative disassembly, with defined decision variables, objective functions, and constraints.
- The ELWOA is introduced to solve large-scale DALHBP cases, incorporating evolutionary learning strategies to enhance global and local search capabilities.
- The mathematical model is first solved using CPLEX to verify its correctness. The robustness and efficiency of ELWOA are then demonstrated through numerical experiments, and its superiority is confirmed by comparing it with other algorithms.

The rest of this paper is organized as follows. Section 2 introduces the DALHBP and its mathematical formulation. Section 3 presents the ELWOA for solving the problem. Section 4 provides experimental studies. Finally, Section 5 concludes the work and discusses future research directions.

2. Human–Machine Collaboration Disassembly and Assembly Hybrid Line Balance Problem

2.1. Problem Description

A DALHBP aims to rationally assign disassembly and assembly tasks to workstations. The objective of DALHBP is to maximize profit while adhering to the priority relationship constraints of disassembly tasks, the priority relationship constraints of assembly tasks, and the cycle time constraints of workstations. This section describes the DALHBP using an example of a flashlight, with its disassembled structure illustrated in Figure 1. The figure shows the parts that can be obtained when the flashlight is completely disassembled. The number of each part is used in the model and algorithm to represent the corresponding part.

The priority constraints of the assembly tasks in this work are described using a priority relation graph G. Figure 2 shows the assembly task priority graph for the flashlight, where the numbers inside the circles represent task numbers. The directed arc between Task 1 and Task 2 indicates that Task 1 is the immediate predecessor of Task 2, meaning Task 2 can be assigned to the current workstation only after Task 1 has been assigned to the same or a preceding workstation. Tasks 1, 3, and 5 have no predecessor constraints, sharing the same priority. Furthermore, Task 4 can only proceed after both Tasks 2 and 3 are completed. This means the following steps must be executed: assembling the main housing spring and battery together (Task 2, forming the "back"), and assembling the bulb and head housing together (Task 3, forming the "front"). Only then can the "front" and

"back" be assembled together (Task 4). Similarly, Task 6 can only proceed after both Tasks 4 and 5 are completed. Detailed operational descriptions of the assembly tasks, along with their operating times, are provided in Table 1.

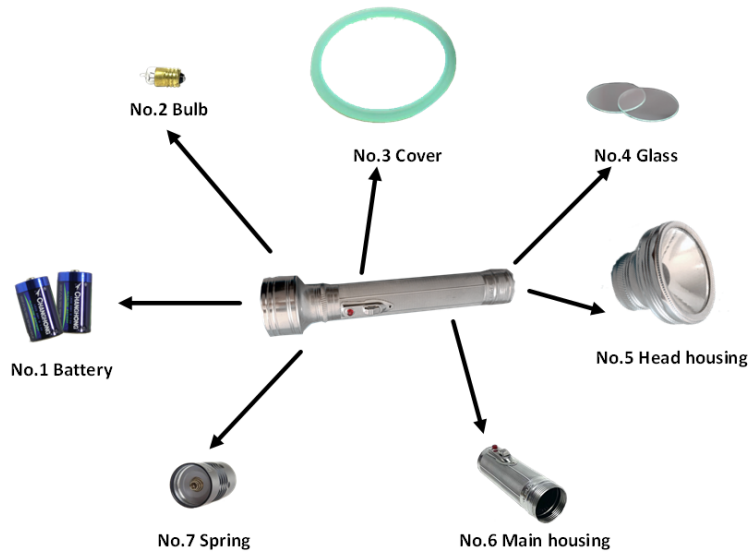

Figure 1. Flashlight disassembly structure graph.

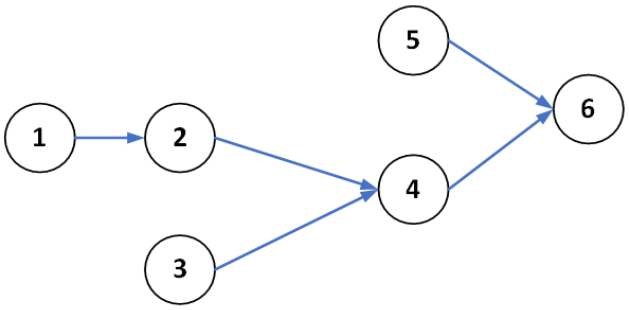

Figure 2. Assembly task precedence relationship graph of flashlight.

Table 1. Assembly task description and operation time.

Task Index	Description	Operation Time
1	Assemble the main housing and spring together	6
2	Assemble main housing spring and battery together	7
3	Assemble Bulb and Head housing together	3
4	Assemble front and back together	8
5	Assemble cover and glass	7
6	Assemble the cover glass and other parts together	3

The precedence relationship of assembly tasks can be represented by the immediate predecessor task matrix $IP^a = [p_{ij}]$, which defines the execution order between two assembly tasks, where i and j denote two distinct assembly tasks:

$$p_{ij} = \begin{cases} 1, & \text{if assembly task } j \text{ is the immediate predecessor task of task } i \\ 0, & \text{otherwise} \end{cases}$$

For example, Figure 2 can be transformed into the following immediate predecessor task matrix:

$$IP^a = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

The priority constraints for disassembly tasks in this work are described as a Transformed AND/OR Graph (TAOG) [29]. Figure 3 shows the TAOG for a flashlight. The graph consists of gray artificial nodes A and white normal nodes B. Each artificial node represents a component, while each normal node represents a disassembly task. The component represents the combination of multiple parts. The product is disassembled into different components through the disassembly task, and finally the parts are obtained. The direct connection between artificial and normal nodes forms an AND-type relationship, where all subsequent normal nodes of an artificial node must be selected and executed sequentially for disassembling the product. In contrast, arcs connecting artificial nodes and normal nodes form an OR-type relationship, meaning that only one of the subsequent normal nodes needs to be selected. To derive a feasible disassembly sequence, the selected normal node numbers are organized sequentially. A sample process is as follows: A_0 represents the product to be disassembled, with two tasks, B_1 and B_2, available. Since B_1 and B_2 have an OR-type relationship, either one can be chosen for execution. Assuming B_1 is selected, subassemblies A_1 and A_6 are obtained. A_1 and A_6 share an AND-type relationship, so both must be disassembled, but the order can vary. Assuming A_1 is chosen first, task B_3 is executed. After B_3 is completed, subassemblies A_3 and A_4 are obtained, along A_6. Among these, any one subassembly can be selected for further disassembly. Assuming A_3 is chosen, after task B_7 is executed, all its parts are disassembled, leaving A_4 and A_6 for further processing. This disassembly process continues sequentially until no subassemblies remain. Organizing the disassembly tasks in order produces a feasible disassembly sequence, such as (1, 3, 7, 6, 9, 10). The composition of components A_0–A_7 in the figure is shown in Table 2. The Included Parts column is the corresponding number in Figure 1.

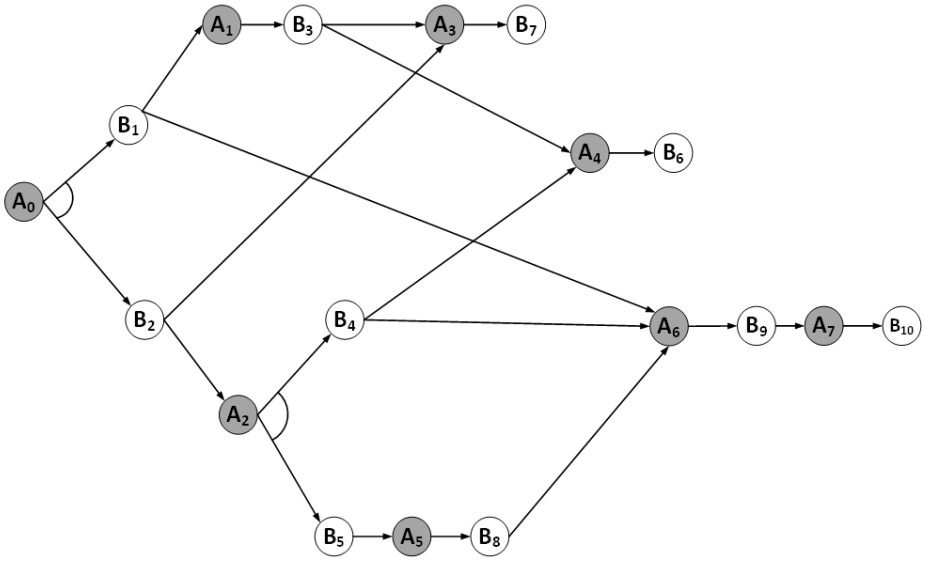

Figure 3. TAOG of flashlight.

Table 2. Component definitions for flashlight.

Component Index	Included Parts
0 (Entire product)	1, 2, 3, 4, 5, 6, 7
1	2, 3, 4, 5
2	1, 2, 5, 6, 7
3	3, 4
4	2, 5
5	1, 5, 6, 7
6	1, 6, 7
7	6, 7

Similarly, the TAOG can also be represented by an incidence matrix $D^d = [d_{lj}]$, which describes the relationship between normal nodes and artificial nodes. Here, l represents artificial nodes, and j represents normal nodes:

$$d_{lj} = \begin{cases} 1, \text{ if subassembly } l \text{ is obainted by task } j \\ -1, \text{if subassembly } l \text{ is disassembled by task } j \\ 0, \text{otherwise} \end{cases}$$

For example, Figure 3 can be transformed into the following incidence matrix:

$$D^d = \begin{bmatrix} -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Figure 4 shows the layout of a hybrid remanufacturing system featuring disassembly and assembly lines. The system consists of two parallel production lines that operate in opposite directions. One line, called the disassembly line, is used for disassembling EOL products and operates from left to right. The other, called the assembly line, is used for assembling new products and operates from right to left. This layout design reflects the consideration that disassembly and assembly processes can be viewed as inverse operations to some extent. The reverse layout offers advantages such as reducing material transport distances and improving part utilization rates.

Tasks involving the same parts in both assembly and disassembly are termed similar tasks. Table 3 provides examples of similar tasks for flashlights. Similar tasks can be assigned to shared workstations. For instance, workstations 1, 4, and 7 in Figure 4 are shared between the disassembly and assembly lines. Shared workstations enhance the flexibility of task allocation, reduce the total number of workstations required, and improve the overall efficiency of the disassembly and assembly lines.

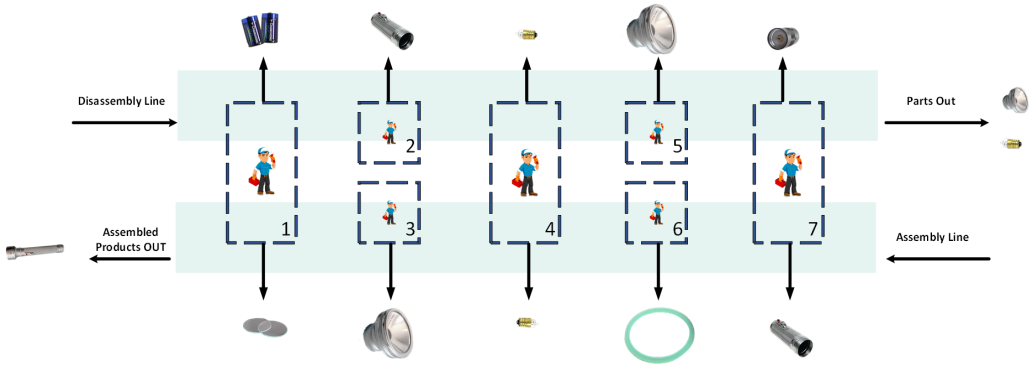

Figure 4. Layout of disassembly and assembly line. The number in the figure represents the workstation number.

Table 3. Similar tasks for flashlight.

Assembly Task Index	Similar Disassembly Task Index
1	10
2	9
3	6, 5, 8
4	4
5	7
6	2

Before establishing the DALHBP model, the following assumptions are made:

(1) The IP^a and D^d matrix of the products are known.

(2) Similarity tasks for each product are predefined based on their process characteristics.

(3) Since similarity tasks focus on parts, each product must be completely disassembled into all its components.

(4) Each running workstation is assigned at least one disassembly task.

(5) As parts disassembled on the disassembly line may not be directly applicable to the assembly line, they must be manually inspected. Consequently, it is assumed that all assembly tasks are performed manually, while robots are exclusively used on the disassembly line.

2.2. Mathematical Model

This section presents a mathematical model for the DALHBP problem. The notations and decision variables used in the model are defined as follows:

(1) Notations

I	Number of assembly tasks.
J	Number of disassembly tasks.
K	Number of workstations.
E	Number of subassemblies.
L	Sum of subassemblies and parts.
c	Cycle time of each workstation.
I	Set of all assembly tasks, I = 1, 2, . . . , I.
J	Set of all disassembly tasks, J = 1, 2, . . . , J.
E	Set of all subassemblies, E = 1,2,. . . ,E.
L	Set of all subassemblies and parts, L = 1, 2, . . . , L.
W	Set of workstations, W = 1, 2, . . . , W.
T_i^a	Time to execute the i-th assembly task.
T_j^H	Time for a worker to execute the j-th disassembly task.

T_j^R Time for a robot to execute the j-th disassembly task.
A_e Artificial node of TAOG.
B_j Normal node in TAOG.
$P(A_e)$ Set of immediate predecessors of A_e.
$S(A_e)$ Set of immediate successors of A_e.
ST Set of pairs $g = (i, j)$ of similar assembly task i and disassembly task j.
C_i^a Unit time cost of executing the i-th assembly task.
C_j^R Unit time cost of a robot execute the j-th disassembly task.
C_j^H Unit time cost of a worker execute the j-th disassembly task.
C_p Penalty cost for a pair of similar tasks not assigning to a same workstation.
C_k^W Operating cost of opening of the k-th workstation.
v_l The value of reusing the l-th subassembly of the product.
P^a Profit from assembly.

(2) Decision variables

$$Z_k = \begin{cases} 1 & \text{if workstation } k \text{ is opened} \\ 0 & \text{otherwise} \end{cases}$$

$$y_j^H = \begin{cases} 1 & \text{if the disassembly task } j \text{ is performed by} \\ & \text{a worker} \\ 0 & \text{otherwise} \end{cases}$$

$$y_j^R = \begin{cases} 1 & \text{if the disassembly task } j \text{ is performed by} \\ & \text{a robot} \\ 0 & \text{otherwise} \end{cases}$$

$$d_{jk}^R = \begin{cases} 1 & \text{if the disassembly task } j \text{ performed by a} \\ & \text{robot is assigned to workstation } k \\ 0 & \text{otherwise} \end{cases}$$

$$d_{jk}^H = \begin{cases} 1 & \text{if the disassembly task } j \text{ performed by a} \\ & \text{worker is assigned to workstation } k \\ 0 & \text{otherwise} \end{cases}$$

$$a_{ik} = \begin{cases} 1 & \text{if the assembly task } i \text{ is assigned to workstation } k \\ 0 & \text{otherwise} \end{cases}$$

$$C_j = \begin{cases} 1 & \text{if the disassembly task } j \text{ and its similar assembly} \\ & \text{task are not assigned to the same workstation} \\ 0 & \text{otherwise} \end{cases}$$

(3) Model formulation

Based on the notations and decision variables, the mathematical formulations for the objective and the constraints of the mathematical model are as follows:

$$\begin{aligned}\max f = {} & P^a + \sum_{j\in J}\sum_{l\in L} d_{lj} v_l \left(y_j^H + y_j^R\right) \\ & - \sum_{j\in J} T_j^H C_j^H y_j^H - \sum_{j\in J} T_j^R C_j^R y_j^R \\ & - \sum_{k\in W} C_k^W Z_k - \sum_{\forall g\in ST} C_p C_j\end{aligned} \tag{1}$$

$$\sum_{B_j\in S(A_e)} \left(y_j^H + y_j^R\right) = 1, \quad \forall j \in J,\, e = 0 \tag{2}$$

$$\sum_{B_j\in S(A_e)} \left(y_j^H + y_j^R\right) = \sum_{j:B_j\in P(A_e)} \left(y_j^H + y_j^R\right), \quad \forall e \in E,\, \forall j \in J \tag{3}$$

$$\sum_{k\in W} \left(d_{jk}^H + d_{jk}^R\right) = y_j^H + y_j^R, \quad \forall j \in J \tag{4}$$

$$\sum_{k\in W} d_{jk}^H = y_j^H, \quad \forall j \in J \tag{5}$$

$$\sum_{k\in K} d_{j,k}^R = y_j^R, \quad \forall j \in J \tag{6}$$

$$\sum_{k\in W} a_{ik} = 1, \quad \forall i \in I \tag{7}$$

$$\sum_{j:B_j\in P(A_e)} \sum_{k'=1}^{k} \left(d_{jk'}^R + d_{jk'}^H\right) \geq \sum_{j:B_j\in S(A_e)} \left(d_{jk}^R + d_{jk}^H\right), \quad \forall e \in L,\, \forall k \in W \tag{8}$$

$$\sum_{k\in W} ka_{ik} - \sum_{k\in W} ka_{fk} \geq 0, \quad \forall (i,f) \in G \tag{9}$$

$$\sum_{i\in I} T_i^a a_{ik} + \sum_{j\in J}\left(T_j^H d_{jk}^H\right) + \sum_{j\in J}\left(T_j^R d_{jk}^R\right) \leq cZ_k, \quad \forall k \in W \tag{10}$$

$$C_j \geq a_{ik} - (d_{jk}^H + d_{jk}^R) - M(1 - y_j^H - y_j^R), \quad \forall (i,j) \in ST,\, \forall k \in K \tag{11}$$

$$C_j \geq (d_{jk}^H + d_{jk}^R) - a_{ik} - M(1 - y_j^H - y_j^R), \quad \forall (i,j) \in ST, \forall k \in K \tag{12}$$

$$Z_k \in \{0,1\}, \quad \forall k \in W \tag{13}$$

$$y_j^H \in \{0,1\}, \quad \forall j \in J \tag{14}$$

$$y_j^R \in \{0,1\}, \quad \forall j \in J \tag{15}$$

$$d_{jk}^R \in \{0,1\}, \quad \forall j \in J, \forall k \in W \tag{16}$$

$$d_{jk}^H \in \{0,1\}, \quad \forall j \in J, \forall k \in W \tag{17}$$

$$a_{ik} \in \{0,1\}, \quad \forall i \in I, \forall k \in W \tag{18}$$

$$C_j \in \{0,1\}, \quad \forall j \in J \tag{19}$$

The objective function (1) aims to maximize profit. Constraints (2) and (3) ensure that in the TAOG, only one of the conflicting successors is selected, thereby establishing a correct disassembly sequence. Constraints (4) to (6) assign disassembly tasks to specific workstations. Constraint (7) ensures that each assembly task is assigned only once, while Constraint (8) states that for a subassembly, the task receiving the subassembly must be assigned to a workstation that precedes the workstation where the subassembly is disassembled. Constraint (9) maintains the precedence relationships between assembly tasks, and Constraint (10) ensures that the total runtime of both the assembly and disassembly tasks on each workstation does not exceed the cycle time. Constraints (11) and (12) verify whether similar tasks are assigned to the same workstation. Finally, Constraints (13) to (19) define the ranges and conditions for the decision variables.

3. Proposed Algorithm

DALHBP is a discrete optimization problem characterized by high complexity and diversity, requiring the consideration of multiple constraints and objective functions. The whale optimization algorithm (WOA) simulates the feeding behavior of humpback whales in nature and is known for its fast global search capability. In the WOA, $\vec{X}^*(t)$ represents the position of the leading whale in the t-th generation, and $\vec{X}(t)$represents the position of the other individual from the whales in the t-th generation, where t indicates the current iteration. $\vec{A}$ and $\vec{C}$ are coefficient vectors, calculated as follows:

$$\vec{A} = 2\vec{a} \cdot \vec{r}_1 - \vec{a} \tag{20}$$

$$\vec{C} = 2 \cdot \vec{r}_2 \tag{21}$$

$$\vec{a} = 2 - (2t/M) \tag{22}$$

where $\vec{r}_1$ and $\vec{r}_2$ are random vectors in [0,1], $\vec{a}$ is linearly decreased from 2 to 0 over the course of iterations, and M is a maximum number of iterations. For a more detailed explanation of the algorithm, please refer to [24].

However, its effectiveness in solving discrete optimization problems, particularly those with multiple constraints, is limited. To address these limitations, this section introduces the ELWOA, an enhanced version of WOA that integrates crossover and mutation operations from genetic algorithms. These modifications simulate population update strategies, enhancing the ELWOA global and local search capabilities, improving exploration, convergence speed, and solution accuracy. Additionally, the designed crossover and mutation operators ensure that precedence relationships remain intact after the operations. Inspired by population retention strategies in evolutionary learning, an elite retention strategy is also incorporated to further enhance the ELWOA exploration capabilities. This strategy maintains a balance between global exploration and local exploitation. The flowchart of ELWOA is shown in Figure 5.

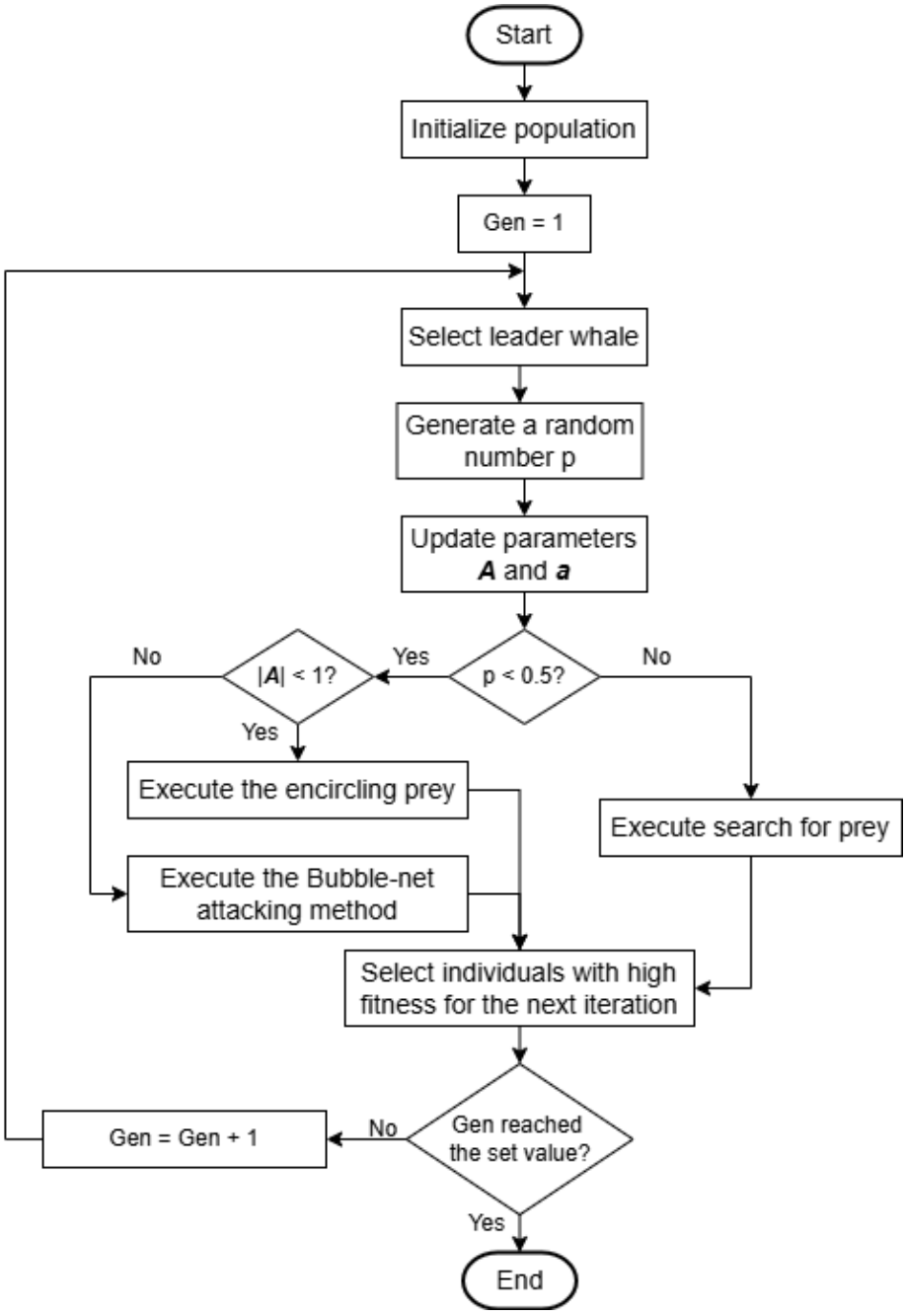

Figure 5. ELWOA flowchart.

3.1. Encoding

In order to solve DALHBP with the whale optimization algorithm, it is necessary to discretize the genes of an individual whale according to the features of DALHBP. In ELWOA, the encoding of each solution represents the genes of an individual whale, with the structure illustrated in Figure 6. A feasible solution s consists of three integer codes, i.e., $S = (s_1, s_2, s_3)$, where we have the following:

- s_1 represents the task number. A positive integer denotes a disassembly task, while a negative integer denotes an assembly task.
- s_2 represents the performer of the disassembly task: 0 indicates that the task is performed manually, and 1 indicates that it is performed by a robot. Assembly tasks are performed manually by default ("-" in the figure), so the performer is not distinguished.
- s_3 represents the workstation number to which each task is assigned.

Solution Sequence	−6	−5	1	3	7	6	−4	−3	−2	−1	9	10
Operator	–	–	1	0	1	0	–	–	–	–	1	0
Workstation	1	1	1	1	1	2	2	2	3	3	3	3

Figure 6. Solution encoding.

To generate a new solution, the process involves several steps: (1) Construct a disassembly sequence according to Algorithm 1. (2) Construct an assembly sequence according to Algorithm 2. (3) Merge the disassembly and assembly sequences randomly to form a solution sequence. (4) Assign appropriate operators to the tasks in the solution sequence. (5) Arrange suitable workstations for each task in the solution sequence using Algorithm 3. The overall process is illustrated in Figure 7.

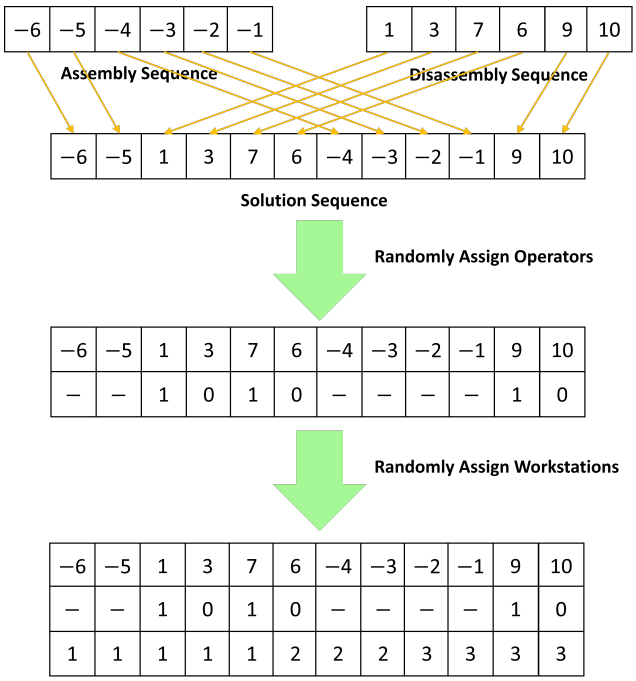

Figure 7. Solution generation process.

Algorithm 1: Creating a disassembly sequence

Input: Subassembly stack, part stack, incidence matrix D
Output: Disassembly sequence

Initialize the set of tasks
Initialize the parts stack and put the product number into the parts stack
while Subassembly stack is not empty **do**
  Pop up the top element of the subassembly stack as the index of the subassembly to be disassembled
  for $j = 0$ to Length$(D[l])$ **do**
    if $D[l][j] == -1$ **then**
      Put task number j into the task set
    end if
  end for
  if task set is not empty **then**
    Randomly select a task from the task set and add it to the disassembly sequence
    Define a set of parts
    for $j = 0$ to Length$(D[l])$ **do**
      if $D[l][j] == 1$ **then**
        Put the part number into the part set
      end if
    end for
    Press all elements in the part set into the subassembly stack in random order
    if $p \geq 0.5$ **then**
      Disrupting subassembly stacks
    end if
  end if
end while

Algorithm 2: Create assembly sequence

Input: Total number of assembly tasks N_a, immediately predecessor task matrix IP
Output: Assembly Sequence

Randomly arrange $(1, N_a)$ to obtain a random sequence Sr;
for $i = 0$ to Length(Sr) **do**
  for $j = i + 1$ to Length(Sr) **do**
    if $IP[Sr[i] - 1][Sr[j] - 1] = 1$ **then**
      Swap$(Sr[i], Sr[j])$;
      $i \leftarrow i - 1$;
      break;
    end if
  end for
end for
Reverse assembly sequence and reverse numbering;

Algorithm 3: Assign workstations

Input: $S(s_1, s_2)$
Output: $S(s_1, s_2, s_3)$

```
Assign the first task to the first workstation;
while there are tasks that have not been assigned to workstations do
    if The current task has not exceeded the cycle after being assigned to the current workstation then
        Assign the current task to the current or next workstation;
    else
        Assign the current task to the next workstation;
    end if
end while
```

3.2. Decoding

For the decoding process, we propose a decoding scheme for the three-part encoding $S = (s_1, s_2, s_3)$ and analyze each segment layer by layer. And through s_1, s_2, and s_3, we can calculate the values required by the objective function. s_1 calculates the value of parts obtained from executing the task sequence and the associated task costs; s_2 determines the costs based on whether the disassembly sequence is performed by humans or robots; and s_3 evaluates the workstation activation costs and additional costs incurred when similar tasks are not assigned to the same workstation.

3.3. Search Process

The WOA optimizes the search process through three key strategies: searching for prey, encircling prey, and bubble-net attacking. In this work, the crossover and mutation methods from evolutionary learning are incorporated into the WOA to enhance its global and local search capabilities. To simulate the three predator–prey strategies of whales, three specialized operators are designed for the ELWOA. Since adjusting the solution sequence has been shown to significantly improve the objective value and computational efficiency, the three operators focus on refining the solution sequence.

3.3.1. Search for Prey

The primary purpose of random prey search in the WOA is to enable global exploration and prevent the algorithm from converging prematurely to local optima. In this work, a global crossover operator is designed to implement this strategy. The operator works as follows: two whales are randomly selected from the population. One whale retains the disassembly sequence from its solution, while the other retains the assembly sequence from its solution. These two sequences are then recombined to form a new solution sequence. Subsequently, operators and workstations are reassigned for the tasks in the new sequence. The detailed process of this operator is illustrated in Figure 8.

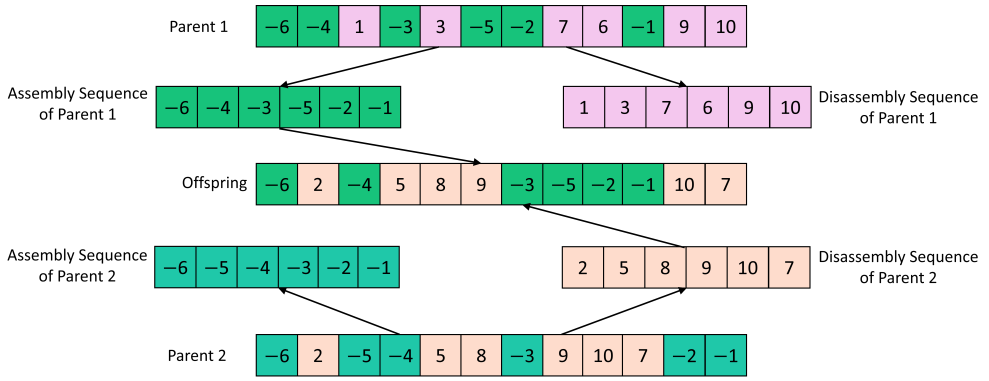

Figure 8. The process of encircling prey. The top and bottom of the figure are two parent solution sequences. Sequence 1 retains the disassembly sequence, and sequence 2 retains the assembly sequence. The two recombine to create a new child sequence.

3.3.2. Bubble-Net Attacking Method

This strategy in WOA primarily updates the position of each whale based on the position of the lead whale. By simulating the behavior of WOA and integrating the crossover operator from genetic algorithms in evolutionary learning, a discrete bubble-net attacking strategy is designed. The specific process is illustrated in Figure 9 and Algorithm 4.

Shrinking Encircling: The assembly sequence from the normal whale and the disassembly sequence from the lead whale are retained and recombined to form a new solution sequence. The operator assignment remains the same as the parent operator, while the workstation assignments are updated.

Spiral Updating Position: The disassembly sequence from the normal whale is preserved and crossed with the assembly sequence from the lead whale to generate a new disassembly sequence. The operator assignment remains the same as the parent operator, and the workstation assignments are updated.

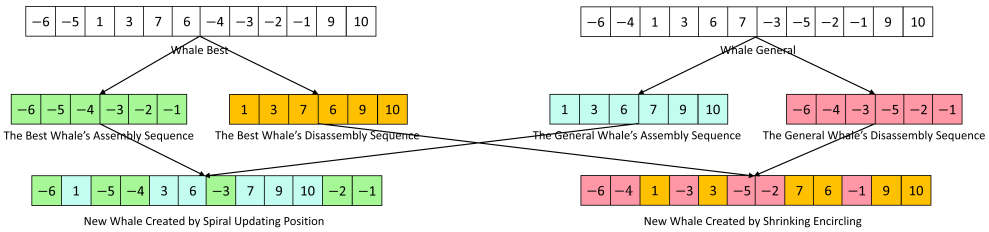

Figure 9. Bubble-net attacking method. In the figure, the left side is the leader whale, and the right side is the normal whale. The assembly sequence and disassembly sequence of the two whales are decomposed respectively, and then the assembly sequence of the leader whale and the disassembly sequence of the normal whale are combined, and the disassembly sequence of the leader whale and the assembly sequence of the normal whale are combined to obtain two new sequences.

Algorithm 4: Bubble-net attacking method

Input: Best sequence B, Normal sequence N

Output: New sequence 1, New sequence 2

1: Extract $B_{\text{assembly}}, B_{\text{disassembly}}, N_{\text{assembly}}, N_{\text{disassembly}}$
2: New sequence 1 $\leftarrow [N_{\text{disassembly}}, B_{\text{assembly}}]$
3: New sequence 2 $\leftarrow [B_{\text{disassembly}}, N_{\text{assembly}}]$
4: Retain operator assignments from N
5: Update workstation assignments
6: **return** New sequence 1, New sequence 2

3.3.3. The Process of Encircling Prey

The principle of surrounding prey in the WOA involves updating the positions of normal whales relative to the lead whale. In the ELWOA designed in this work, this principle is simulated by preserving certain superior genes from the lead whale and replacing corresponding genes in the normal whale. To achieve this, a mutation operator is designed. The strategy of the operator is as follows: Two random points are selected within the genes of a normal whale. The genes located at these points are checked against those in the lead whale. If they exist in the lead whale, they are sorted according to their sequence in the lead whale and then reinserted into the normal whale at their original positions. The detailed process of this operator is illustrated in Figure 10.

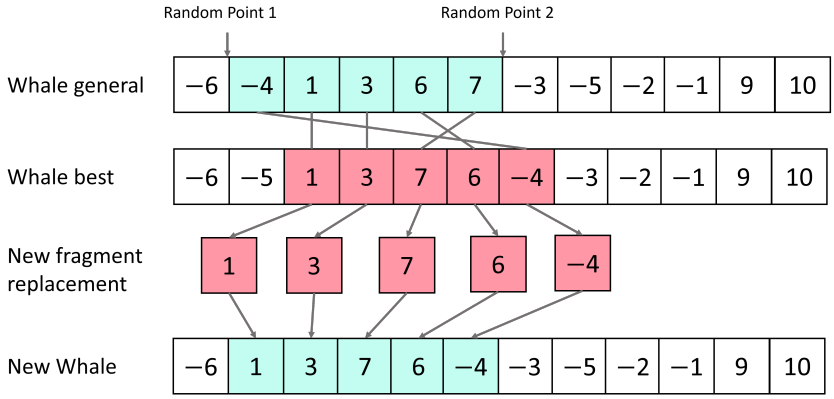

Figure 10. The process of encircling prey. Blue represents the corresponding random sequence fragments of the random normal whale and the mutated whale, and red represents the sequence fragments with the same elements as the lead whale.

3.4. Elite Reservation

In the search phase of the WOA, each ordinary whale updates its position using the position update formula. However, this strategy can lead to premature convergence and getting trapped in the local optima. To address this, an elite population reservation strategy inspired by genetic algorithms in evolutionary learning is designed for the ELWOA. This strategy retains both offspring and parent populations after the search phase. The total population is then reduced to the initial population size using the elite retention strategy as detailed in Algorithm 5. This approach ensures that the optimal solution is preserved, as high-quality individuals (elite whales) are retained in each generation, enhancing the convergence and stability of the ELWOA. Additionally, since new whales are added to the population in each generation, this strategy also increases population diversity, thereby improving the exploration capability and adaptability of the ELWOA. By balancing global search and local search, this method maintains high optimization efficiency and solution quality.

Algorithm 5: Elite population retention strategy.

Input: Initial Population P, population size N, offspring population Q
Output: Updated population P^*
1: Merge P and Q into R, with a size of 2N for R;
2: Sorting R in descending order of its objective value yields R^*;
3: Keep $0.8N$ whales from the front of R^* and put them into P^*;
4: Keep $0.05N$ whales from the back of R^* and put them into P^*;
5: Randomly select $0.15N$ whales from the middle of R^* and put them into P^*.

4. Experimental Studies

In this section, we first use CPLEX to solve five EOL product examples to verify the correctness of the established DALHBP mathematical model. Subsequently, a series of experiments are conducted to determine the parameter settings that optimize the performance of the ELWOA. Following this, the ELWOA is employed to obtain near-optimal solutions for the DALHBP model, and the results are compared with those from CPLEX to demonstrate the advantages of the ELWOA in solving the optimization problem of disassembly and assembly lines. Finally, the ELWOA is compared with FOA, DOA, CS, AO, and other algorithms to further establish its superiority. The problem-solving process is implemented using the JMetal framework and IBM ILOG CPLEX optimization studio. All experiments are conducted in a Windows 11 environment on a system equipped with an Intel(R) Core(TM) i7-7700HQ CPU (2.80 GHz) and 16.00 GB of RAM.

4.1. Test Instances

We select five EOL products as test cases, including ballpoint pen [30], a radio [30], a flashlight [31], a washing machine [32] and a hammer drill [33]. The complete set of test cases is presented in Table 4.

Table 4. Case description.

Case Index	EOL Product	Number of Assembly Tasks	Number of Disassembly Tasks	Subassemblies and Parts
1	flashlight	6	10	15
2	ballpoint pen	5	13	15
3	washing machine	5	13	15
4	radio	8	30	29
5	hammer	21	46	63

4.2. Experimental Parameters

To determine the optimal parameter settings for the ELWOA, extensive experiments are conducted to evaluate the algorithm's performance under various parameter combinations. The ELWOA involves several parameters, including the initial population size, the number of iterations, the percentage of population retention, and others. If these parameters are not configured appropriately, the algorithm's effectiveness in solving large-scale cases may be significantly compromised. In this section, eight representative parameter groups are selected for analysis. The details of these experimental parameter groups are presented in Table 5.

Table 5. Experimental parameters.

Experiment Index	Initial Population Num	Iteration Times	High Fitness Whale Retention Percentage	Middle Fitness Whale Retention Percentage	Low Fitness Whale Retention Percentage
1	50	200	0.8	0.15	0.05
2	100	200	0.8	0.15	0.05
3	300	200	0.8	0.15	0.05
4	400	200	0.8	0.15	0.05
5	300	100	0.8	0.15	0.05
6	300	500	0.8	0.15	0.05
7	300	200	0.9	0.05	0.05
8	300	200	0.5	0.25	0.25

In this section, the hammer drill is used as an experimental case, and extensive experiments are conducted based on the parameter sets listed in Table 6. The experiments measure the algorithm's running time and record the optimal, worst, and average objective values. By comparing parameter combinations 1, 2, 3, and 4, it is observed that a larger initial population size leads to a higher number of optimal solutions but also increases the solution time when other parameters remain constant. Notably, when the initial population size is set to 300, the number of optimal solutions is comparable to that with a population size of 400. However, the average objective value is slightly higher, and the solution time is significantly reduced. Based on this finding, an initial population size of 300 is selected for subsequent experiments. Comparing parameter sets 3, 5, and 6, it is found that increasing the number of iterations improves the number of optimal solutions. However, beyond 200 iterations, such as at 500 iterations, the improvement becomes negligible. This can be attributed to the fast convergence of the ELWOA, as the algorithm approaches the optimal solution within approximately 200 generations. Further comparison of parameter sets 3, 7, and 8 reveals that the best performance of the elite retention strategy occurs when the percentage of retained highly adaptive whales is set to 0.8. At a retention percentage of 0.9, the population diversity decreases, leading to a higher risk of getting trapped in the local optima. Conversely, at 0.5, the retention of good genes is insufficient, limiting the solution space and hindering the algorithm's ability to reach higher-quality solutions. Based on these analyses, the following experimental parameters are selected for the ELWOA: an initial population size of 300, 200 iterations, a retention percentage of 0.8 for highly adaptive whales, 0.15 for medium adaptive whales, and 0.05 for low adaptive whales.

Table 6. Parameter experiment results.

Experiment Index	Experiment Times	Solution Time	Excellent Times	Best Objective	Worst Objective	Average Objective
1	500	1.6 s	123	6292	6135	6205.668
2	500	2.5 s	144	6292	6130	6218.74
3	500	8.4 s	407	6292	6242	6287.86
4	500	14.5 s	415	6292	6185	6280.09
5	500	6.9 s	336	6292	6185	6270.264
6	500	22.3 s	412	6292	6192	6282.85
7	500	7.8 s	127	6292	6142	6266.74
8	500	9.1 s	76	6292	6130	6154.624

4.3. Experimental Results

The test case set is solved using both CPLEX and the ELWOA, with the results presented in Tables 7 and 8, respectively. The running parameters for the ELWOA are configured based on the experimental findings from the previous section: an initial population size of 300, 200 iterations, and retention percentages of 0.8 for highly adaptive whales, 0.15 for medium-adaptive whales, and 0.05 for low-adaptive whales.

Table 7. Results of solving the instances set based on CPLEX.

Case Index	Solution	Human–Machine Distribution		Objective	Solution Time
		Execute by Human	Execute by Robot		
1	(1,3,-4,-6) → (7,9,-2,-5) → (6,10,-1,-3)	3,6,10	1,7,9	1461	1.02 s
2	(1,12,-4,-5) → (7,9,10,-1,-2,-3)	7,10,12	1,9	1480	1.62 s
3	(1,-5) → (-3,-4) → (4,9,13,-1) → (10,-2)	1	4,9,10,13	1433	2.79 s
4	(2,11,14,20,30,-4,-6,-7,-8) → (9,24,27,29,-1,-2,-3,-5)	9,29,30	2,11,14,20,2,4,27	5633	4.32 s
5	(1,2,4,7,11,-19,-20,-21) → (12,19,29,-12,-13,-16,-17,-18) → (21,36,41,-8,-10,-11) → (18,28,31,-6,-7,-14,-15) → (13,22,23,-4,-5) → (32,38,43,45,-1,-2,-3,-9)	4,7,21,41,23,38	1,2,11,12,19,29, 36,18,28,31,13, 22,32,43,45	6292	7200 s+

Table 8. Results of solving the instances set based on the ELWOA.

Case Index	Solution	Human–Machine Distribution		Objective	Solution Time
		Execute by Human	Execute by Robot		
1	(-6,-5,1,3,7) → (-4,-2,9) → (-1,10,-3,6)	3,10	1	1461	1.112 s
2	(1,12,-4,-5) → (7,9,10,-1,-2,-3)	7,10,12	1,9	1480	1.092 s
3	(1,-5) → (-3,-4) → (4,9,13,-1) → (10,-2)	1	4,9,10,13	1433	1.026 s
4	(-8,-7,2,11,-6,14,-4,20,30) → (-5,24,-3,27,-2,29,-1,9)	9,29,30	2,11,14,20,2,4,27	5633	2.423 s
5	(-21,1,-20,2,-19,4,-18,-17) → (-16,7,12,-8,21,-7,11,31) → (-6,-4,13,22) → (-3,19,18,32,-15,-2,38,-13,-1,43) → (-12,29,-11,36,-10,41) → (-9,45,-5,23,-14,28)	4,7,21,38,41,23	1,2,12,11,31,13, 22,19,18,32,43, 29,36,45,28	6292	4.978 s

From Table 7, the CPLEX solutions satisfy the priority relationship constraints for disassembly and assembly tasks, verifying the correctness of the DALHBP mathematical model. CPLEX demonstrates better results for small-scale cases (Case 1, Case 2, and Case 3), achieving optimal solutions in a relatively short time. However, for large-scale cases (Case 4 and Case 5), the solution speed decreases significantly, and in some instances, CPLEX fails to find a feasible solution within the specified time.

The ELWOA solution results, shown in Table 8, differ from those of CPLEX due to differences in encoding and decoding methods. Nevertheless, the ELWOA solutions also satisfy the priority relationship constraints for disassembly and assembly tasks. Additionally, the algorithm's solution time is very short, demonstrating the high efficiency of the ELWOA.

Table 9 compares the experimental results of the ELWOA and CPLEX. The findings reveal that the ELWOA performs comparably to CPLEX in small-scale cases, with both algorithms achieving optimal solutions. However, for large-scale cases, the ELWOA not only obtains optimal solutions but also significantly reduces the solution time. These results indicate that the ELWOA is an efficient and effective algorithm for solving the DALHBP.

Table 9. Comparison of the ELWOA and CPLEX results.

Case Index	ELWOA		CPLEX	
	Objective	Solution Time	Objective	Solution Time
1	1461	1.112 s	1461	1.02 s
2	1480	1.092 s	1480	1.62 s
3	1433	1.026 s	1433	2.79 s
4	5633	2.423 s	5633	196.47 s
5	6292	4.978 s	6292	7200 s+

4.4. Algorithm Performance Experiments

To assess the robustness and efficiency of the ELWOA, we compare it with several meta-heuristic algorithms proposed in recent years. The following four algorithms are selected for comparison: the Discrete Whale Optimization Algorithm (DWOA) [34], the Dingo Optimizer (DOA) [35], the Aquila Optimizer (AOA) [36], and the Discrete Fruit Fly Optimization Algorithm (DFOA) [37].

The performance of these algorithms is evaluated by testing their number of Excellent times and calculating their Excellent rates. The experimental results are presented in Table 10. The results show that all algorithms perform well in solving small-scale cases. However, for large-scale cases, the ELWOA achieves a significantly higher Excellent rate compared to the other algorithms, demonstrating its superior robustness in solving the DALHBP.

Table 10. Excellent rates for different cases.

Case Index	Objective	Runtimes	ELWOA		DWOA		DOA		AOA		FOA	
			Excellent Times	Excellent Ratio	Excellent Times	Excellent Ratio	Excellent Times	Excellent Ratio	Excellent Times	Excellent Ratio	Excellent Times	Excellent Ratio
1	1461	500	500	100%	500	100%	500	100%	500	100%	500	100%
2	1480	500	500	100%	500	100%	500	100%	500	100%	500	100%
3	1433	500	500	100%	500	100%	500	100%	220	44%	450	90%
4	5633	500	500	100%	500	100%	440	88%	380	76%	500	100%
5	6292	500	475	95%	300	60%	250	50%	240	48%	270	54%

Additionally, this section presents a comparison of the optimal objective value, worst objective value, average objective value, and solution time achieved by different algorithms across various test cases. The experimental results are summarized in Table 11. While all algorithms are capable of achieving the optimal value, the ELWOA outperforms the others in large-scale cases. Specifically, the ELWOA produces significantly better worst and average objective values compared to the other algorithms. Furthermore, the ELWOA demonstrates a substantially faster solution time for large-scale cases. These findings highlight the superior robustness and efficiency of the ELWOA in solving DALHBP.

Table 11. Optimization solutions for different algorithms.

Case Index		1	2	3	4	5
Experiment Times		500	500	500	500	500
Optimal	ELWOA	1461	1480	1433	5633	6292
	DWOA	1461	1480	1433	5633	6292
	DOA	1461	1480	1433	5633	6292
	AOA	1461	1480	1433	5633	6292
	FOA	1461	1480	1433	5633	6292
Worst	ELWOA	1461	1480	1433	5633	6285
	DWOA	1461	1480	1433	5633	6185
	DOA	1461	1480	1433	5583	6242
	AOA	1461	1480	1383	5583	6180
	FOA	1461	1480	1418	5633	6242
Average	ELWOA	1461	1480	1433	5633	6291.65
	DWOA	1461	1480	1433	5633	6233.74
	DOA	1461	1480	1433	5627	6269.58
	AOA	1461	1480	1411.9	5621	6267.5
	FOA	1461	1480	1431.5	5633	6271.3
Solution Time/s	ELWOA	1.1 s	0.5 s	0.6 s	1.5 s	6.3 s
	DWOA	0.6 s	0.5 s	0.4 s	1.1 s	3.2 s
	DOA	1.2 s	0.8 s	0.8 s	4.2 s	14.7 s
	AOA	0.9 s	0.7 s	0.8 s	2.3 s	15.7 s
	FOA	7.3 s	2.5 s	4.7 s	5.7 s	30.4 s

Finally, this section compares the convergence speeds and results of the selected algorithms. Figure 11 illustrates the convergence curves for each algorithm in solving the hammer drill case. The results indicate that the ELWOA converges the fastest and achieves a solution closer to the optimal compared to the other algorithms.

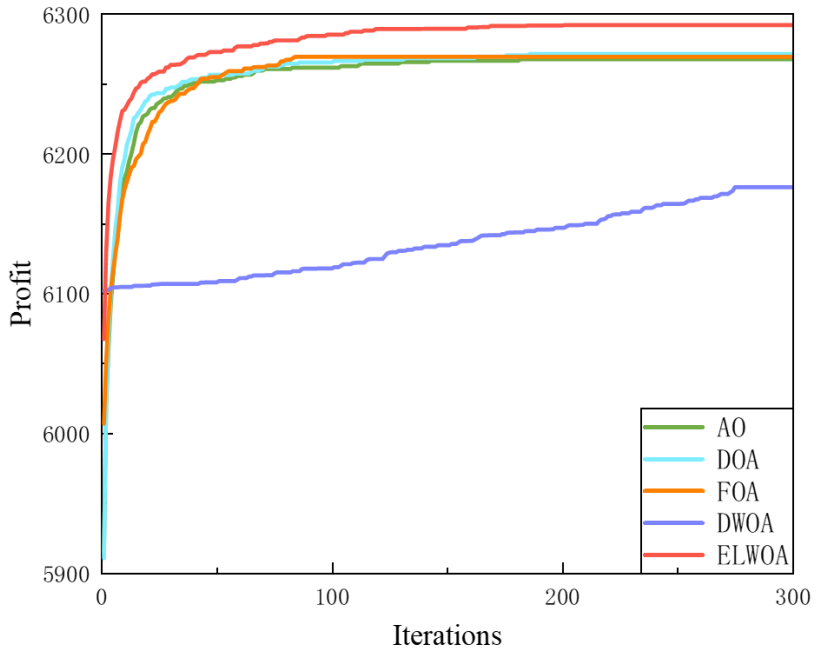

Figure 11. Convergence curves for different algorithms.

5. Conclusions

This work addresses the human–machine collaboration disassembly and assembly hybrid line balance problem (DALHBP) by proposing a mathematical model. The correctness of the model is verified using the IBM ILOG CPLEX solver, which is also employed to determine the optimal sequence for a set of test instances. To solve large-scale cases, an evolutionary learning-based whale optimization algorithm (ELWOA) is developed. The ELWOA incorporates crossover and mutation operations from genetic algorithms into the WOA framework, simulating population update strategies to achieve both global and local search. This enhances the diversity and exploratory capability of the algorithm while improving its convergence speed and accuracy. The designed crossover and mutation operators ensure that the original priority relationships are preserved after these operations. Additionally, an elite retention strategy, inspired by population retention strategies in genetic algorithms, is implemented to balance global and local search, thereby improving the exploration capability of the ELWOA. To demonstrate the superiority of the ELWOA in terms of objective function values and computational efficiency, its performance is compared with IBM ILOG CPLEX. The results reveal that the ELWOA achieves significantly faster computation times than CPLEX and produces consistent results for small-scale instances. For large-scale instances, the ELWOA outperforms CPLEX by obtaining better solutions, making it an efficient and robust algorithm for solving discrete optimization problems.

In our future work, we plan to address the variability in human–machine collaboration efficiency, which can significantly impact the performance of disassembly and assembly line balancing in real-world scenarios. By incorporating models that capture this variability, we aim to enhance the practical applicability and robustness of the proposed approach. Additionally, we plan to expand the current model by incorporating more comprehensive human factors, such as ergonomic factors, safety constraints, and task learning, to better capture the complexities of human–robot collaboration. This will move beyond the idealized assumptions in the current study and provide a more practical framework for real-world applications.

Author Contributions: Methodology, X.C.; Software, Q.M.; Formal analysis, J.W.; Investigation, X.G.; Resources, P.L. and Y.J.; Writing—original draft, X.C.; Writing—review & editing, L.Q., S.Q. and B.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by 'National Local Joint Engineering Laboratory for Optimization of Petrochemical Process Operation and Energy saving Technology' grant number LJ232410148002. and 'the Innovation Team Project of the Educational Department of Liaoning Province' grant number LJ222410148036.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

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

References

1. Yolmeh, A.; Saif, U. Closed-loop supply chain network design integrated with assembly and disassembly line balancing under uncertainty: An enhanced decomposition approach. *Int. J. Prod. Res.* **2021**, *59*, 2690–2707. [CrossRef]
2. Brennan, L.; Gupta, S.M.; Taleb, K.N. Operations planning issues in an assembly/disassembly environment. *Int. J. Oper. Prod. Manag.* **1994**, *14*, 57–67. [CrossRef]
3. Ilgin, M.A.; Akçay, H.; Araz, C. Disassembly line balancing using linear physical programming. *Int. J. Prod. Res.* **2017**, *55*, 6108–6119. [CrossRef]
4. Liu, M.; Liu, X.; Chu, F.; Zheng, F.; Chu, C. Robust disassembly line balancing with ambiguous task processing times. *Int. J. Prod. Res.* **2020**, *58*, 5806–5835. [CrossRef]
5. Wang, K.; Li, X.; Gao, L.; Li, P. Modeling and balancing for disassembly lines considering workers with different efficiencies. *IEEE Trans. Cybern.* **2021**, *52*, 11758–11771. [CrossRef]
6. Guo, X.; Fan, C.; Zhou, M.; Liu, S.; Wang, J.; Qin, S.; Tang, Y. Human–robot collaborative disassembly line balancing problem with stochastic operation time and a solution via multi-objective shuffled frog leaping algorithm. *IEEE Trans. Autom. Sci. Eng.* **2023**, *21*, 4448–4459. [CrossRef]
7. Guo, X.; Wei, T.; Wang, J.; Liu, S.; Qin, S.; Qi, L. Multiobjective U-shaped disassembly line balancing problem considering human fatigue index and an efficient solution. *IEEE Trans. Comput. Soc. Syst.* **2022**, *10*, 2061–2073. [CrossRef]
8. 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]
9. Fu, Y.; Wang, Y.; Gao, K.; Huang, M. Review on ensemble meta-heuristics and reinforcement learning for manufacturing scheduling problems. *Comput. Electr. Eng.* **2024**, *120*, 109780. [CrossRef]
10. Zhang, Z.; Fu, Y.; Gao, K.; Pan, Q.; Huang, M. A learning-driven multi-objective cooperative artificial bee colony algorithm for distributed flexible job shop scheduling problems with preventive maintenance and transportation operations. *Comput. Ind. Eng.* **2024**, *196*, 110484. [CrossRef]
11. 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.* **2023**, *25*, 1706–1719. [CrossRef]
12. Zhao, Z.; Bian, Z.; Liang, J.; Liu, S.; Zhou, M. Scheduling and logistics optimization for batch manufacturing processes with temperature constraints and alternative thermal devices. *IEEE Trans. Ind. Inform.* **2024**, *20*, 11930–11939. [CrossRef]
13. Zhao, Z.; Li, X.; Liu, S.; Zhou, M.; Yang, X. Multi-Mobile-Robot Transport and Production Integrated System Optimization. *IEEE Trans. Autom. Sci. Eng.* **2024**, 1–12. [CrossRef]
14. Zhao, Z.; Li, S.; Liu, S.; Zhou, M.; Li, X.; Yang, X. Lexicographic dual-objective path finding in multi-agent systems. *IEEE Trans. Autom. Sci. Eng.* **2024**, 1–11. [CrossRef]
15. Ketzenberg, M.E.; Souza, G.C.; Guide Jr, V.D.R. Mixed assembly and disassembly operations for remanufacturing. *Prod. Oper. Manag.* **2003**, *12*, 320–335. [CrossRef]
16. Mete, S.; Çil, Z.A.; Özceylan, E.; Ağpak, K.; Battaïa, O. An optimisation support for the design of hybrid production lines including assembly and disassembly tasks. *Int. J. Prod. Res.* **2018**, *56*, 7375–7389. [CrossRef]
17. Zhang, Z.; Zhu, L.; Chen, Y.; Guan, C. A multi-objective hybrid evolutionary search algorithm for parallel production line balancing problem including disassembly and assembly tasks. *Int. Trans. Oper. Res.* **2023**, *30*, 3508–3553. [CrossRef]
18. Guo, J.; Pu, Z.; Du, B.; Li, Y. Multi-objective optimisation of stochastic hybrid production line balancing including assembly and disassembly tasks. *Int. J. Prod. Res.* **2022**, *60*, 2884–2900. [CrossRef]
19. Murphy, R.R.; Nomura, T.; Billard, A.; Burke, J.L. Human–robot interaction. *IEEE Robot. Autom. Mag.* **2010**, *17*, 85–89. [CrossRef]

20. Maurtua, I.; Ibarguren, A.; Kildal, J.; Susperregi, L.; Sierra, B. Human–robot collaboration in industrial applications: Safety, interaction and trust. *Int. J. Adv. Robot. Syst.* **2017**, *14*, 1729881417716010. [CrossRef]
21. Umbrico, A.; Orlandini, A.; Cesta, A.; Faroni, M.; Beschi, M.; Pedrocchi, N.; Scala, A.; Tavormina, P.; Koukas, S.; Zalonis, A.; et al. Design of advanced human–robot collaborative cells for personalized human–robot collaborations. *Appl. Sci.* **2022**, *12*, 6839. [CrossRef]
22. Lee, R.K.J.; Zheng, H.; Lu, Y. Human-robot shared assembly taxonomy: A step toward seamless human-robot knowledge transfer. *Robot. Comput.-Integr. Manuf.* **2024**, *86*, 102686. [CrossRef]
23. Wang, L.; Gao, R.; Váncza, J.; Krüger, J.; Wang, X.V.; Makris, S.; Chryssolouris, G. Symbiotic human-robot collaborative assembly. *CIRP Ann.* **2019**, *68*, 701–726. [CrossRef]
24. Mirjalili, S.; Lewis, A. The whale optimization algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
25. Ezzeldin, R.M.; Djebedjian, B. Optimal design of water distribution networks using whale optimization algorithm. *Urban Water J.* **2020**, *17*, 14–22. [CrossRef]
26. Zhang, J.; Hong, L.; Liu, Q. An improved whale optimization algorithm for the traveling salesman problem. *Symmetry* **2020**, *13*, 48. [CrossRef]
27. Abd El Aziz, M.; Ewees, A.A.; Hassanien, A.E. Whale optimization algorithm and moth-flame optimization for multilevel thresholding image segmentation. *Expert Syst. Appl.* **2017**, *83*, 242–256. [CrossRef]
28. Li, Y.; He, Y.; Liu, X.; Guo, X.; Li, Z. A novel discrete whale optimization algorithm for solving knapsack problems. *Appl. Intell.* **2020**, *50*, 3350–3366. [CrossRef]
29. Koc, A.; Sabuncuoglu, I.; Erel, E. Two exact formulations for disassembly line balancing problems with task precedence diagram construction using an AND/OR graph. *IIE Trans.* **2009**, *41*, 866–881. [CrossRef]
30. Lu, Q.; Ren, Y.; Jin, H.; Meng, L.; Li, L.; Zhang, C.; Sutherland, J.W. A hybrid metaheuristic algorithm for a profit-oriented and energy-efficient disassembly sequencing problem. *Robot. Comput.-Integr. Manuf.* **2020**, *61*, 101828. [CrossRef]
31. Tang, Y. Disassembly modeling, planning, and application. *J. Manuf. Syst.* **2002**, *21*, 200–217. [CrossRef]
32. 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]
33. Pistolesi, F.; Lazzerini, B.; Dalle Mura, M.; Dini, G. EMOGA: A hybrid genetic algorithm with extremal optimization core for multiobjective disassembly line balancing. *IEEE Trans. Ind. Inform.* **2017**, *14*, 1089–1098. [CrossRef]
34. Cui, X.; Guo, X.; Zhou, M.; Wang, J.; Qin, S.; Qi, L. Discrete whale optimization algorithm for disassembly line balancing with carbon emission constraint. *IEEE Robot. Autom. Lett.* **2023**, *8*, 3055–3061. [CrossRef]
35. Bairwa, A.K.; Joshi, S.; Singh, D. Dingo optimizer: A nature-inspired metaheuristic approach for engineering problems. *Math. Probl. Eng.* **2021**, *2021*, 2571863. [CrossRef]
36. Abualigah, L.; Yousri, D.; Abd Elaziz, M.; Ewees, A.A.; Al-Qaness, M.A.; Gandomi, A.H. Aquila optimizer: A novel meta-heuristic optimization algorithm. *Comput. Ind. Eng.* **2021**, *157*, 107250. [CrossRef]
37. Wang, J.; Guo, X.; Zhou, M.; Wang, J.; Qin, S.; Qi, L. Discrete fruit fly optimization algorithm for disassembly line balancing problems by considering human worker's learning effect. In Proceedings of the 2022 Australian & New Zealand Control Conference (ANZCC), Gold Coast, Australia, 24–25 November 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 201–206.